Partial Disentanglement with Partially-Federated GANs (PaDPaF)

Abdulla Jasem Almansoori [1] **Samuel Horváth** [1] **Martin Takáč** [1]

Abstract

Federated learning has become a popular machine learning paradigm with many potential real-life applications, including recommendation systems, the Internet of Things (IoT), healthcare, and self-driving cars. Though most current applications focus on classification-based tasks, learning personalized generative models remains largely unexplored, and their benefits in the heterogeneous setting still need to be better understood. This work proposes a novel architecture combining global client-agnostic and local client-specific generative models. We show that using standard techniques for training federated models, our proposed model achieves privacy and personalization that is achieved by implicitly disentangling the globally-consistent representation (i.e. *content*) from the client-dependent variations (i.e. *style*). Using such decomposition, personalized models can generate locally unseen labels while preserving the given style of the client and can predict the labels for all clients with high accuracy by training a simple linear classifier on the global content features. Furthermore, disentanglement enables other essential applications, such as data anonymization, by sharing only content. Extensive experimental evaluation corroborates our findings, and we also provide partial theoretical justifications for the proposed approach.

1 Introduction

Federated learning (FL) (Konečný et al., 2016b;a) is a recently proposed machine learning setting where multiple clients collaboratively train a model while keeping the training data decentralized, i.e. local data are never transferred. FL has potential for the future of machine learning systems as many machine learning problems cannot be efficiently solved by a single machine/client for various reasons, including scarcity of data, computation, and memory, as well as the adaptivity across domains.

Most of the focus of federated learning has been on classification models, mainly for decision-based applications. These applications include a wide variety of problems such as next-word prediction, out-of-vocabulary word and emoji suggestion, risk detection in finance, and medical image analysis (Wang et al., 2021; Kairouz et al., 2019). However, based on recent works on representation learning and causality (Schölkopf et al., 2021; Wang and Jordan, 2021), we believe that to understand the causes, explain the process of making a decision, and generalize it to unseen domains of data, it is crucial, if not necessary, to learn a generative model of the data as well, such that it is sufficient to know the representation of the data to describe how it looks and make decisions from it. For example, an agent can describe what a digit looks like and infer its label based on this description. Therefore, by collaboratively learning a generative model and communicating with other agents with the same representation, the agents can potentially generalize to the unseen domains of other agents. In general, the potential of a group of specialized agents that communicate and plan efficiently among each other is almost always superior to that of a single monolith agent that can handle every task. These insights motivated us to propose the idea presented in our work.

We take a step towards this objective and introduce our core idea, which can be described as follows: *to learn generative models of heterogeneous data sources collaboratively and to learn a globally-consistent representation of the data that generalizes to other domains*. The representation should contain sufficient information to classify the data's content (e.g. label) with reasonable accuracy, and the generative model should be able to generate data containing this content under different domains (e.g. styles). Thus, we make no assumptions about the availability of labels per client and the intersection of the clients' data distributions.

A direct application of our model is to disentangle the latent factors that correspond to the content and learn a simple classifier on top of it. This approach tackles the problem of domain adaptation (Zhang et al., 2013), where, in our case,

[*]Equal contribution [1]Mohamed bin Zayed Univeristy of Artificial Intelligence, Abu Dhabi, UAE. Correspondence to: Abdulla Jasem Almansoori <abdulla.almansoori@mbzuai.ac.ae>.

Federated Learning Systems (FLSys) workshop, in Conjunction with the 6^{th} MLSys Conference, Miami, Florida, USA, 2023. Copyright 2023 by the author(s).

the domain is the client's local distribution. In this scenario, our model can learn a client-agnostic classifier based on the representation learned by the discriminator, which should generalize across all clients. We call the client-agnostic part of the representation–*the content*, and the private part–*the style*. Thus, we are concerned with the partial disentanglement of latent factors into content and style factors, inspired by (Kong et al., 2022). As for the generator part, its benefits mainly involve generating data samples from the client's distribution, either for data augmentation or for privacy reasons by sampling synthetic data. We can also actively remove private information or any client-identifying variations from the client's data by introducing a specialized reference client that trains only on standardized, publicly available data or data with no privacy concerns. After that, we can generate the same content of the private data with the reference client's style.

The generative model we focus on in this work is Generative Adversarial Networks (GANs) (Goodfellow et al., 2020). In particular, we use the style mapping network idea from StyleGAN (Karras et al., 2019). In the case of supervised data, we condition the GAN using a projection discriminator (Miyato and Koyama, 2018) with spectral normalization (Miyato et al., 2018) and a generator with conditional batch normalization (De Vries et al., 2017). We believe that designing more sophisticated architectures can improve our results and make them applicable to a wider variety of problems. In particular, using transformers (Vaswani et al., 2017) for natural language processing tasks and diffusion models (Ho et al., 2020), vision transformers (Dosovitskiy et al., 2021) for image generation would make an interesting direction for future work.

Contributions. Below, we summarize our contributions:

- We propose our framework for Partial Disentanglement with Partially-Federated Learning Using GANs. Only specific parts of the model are federated so that the representations learned by the GAN's discriminator are partially disentangled into content and style factors, where we are mainly concerned with the content factors, which can be used for inference. We enforce this disentanglement through a "contrastive" regularization term based on the recently proposed Barlow Twins (Zbontar et al., 2021).
- Our model can learn a globally-consistent representation of the content of the data (e.g. label) such that it classifies the labels with high accuracy. In the case of supervised data, our model can generalize to locally-unseen labels (within the client) with respect to classification and generation.
- Through extensive experimentation, we validate our claims and show how popular techniques from federated learning, GANs, and representation learning can seamlessly blend to create a more robust model for both classifications and generation of data in real-world scenarios.
- We make the code publicly available for reproducibility at `https://anonymous.4open.science/r/FedGAN-F629`.

Organization. Our manuscript is organized as follows. Section 2 talks about related work and the novelty of our approach. Section 3 describes notations and preliminary knowledge about the frameworks used in our model. Section 4 contains the main part of our paper and describes the model design and training algorithm in detail. Section 5 shows a detailed evaluation of the model's generative capabilities across different domains, robustness under limited label availability, and generalization capabilities based on its learned representation. Section 6 conclude the manuscript by providing exciting future directions and the paper's summary.

2 Related Work

Training GANs in the FL setting is not new. For example, (Fan and Liu, 2020) considers the options of averaging either the generator or the discriminator, but their training algorithm has no client-specific components or disentanglement. Similarly, FedGAN (Rasouli et al., 2020) also proposes a straightforward federated algorithm for training GANs and proves its convergence for distributed non-iid data sources. Another work exploring distributed GANs with non-iid data sources is (Yonetani et al., 2019), which is interesting because the weakest discriminator can help stabilize training. Still, in general, their framework does not align with ours.

The idea of splitting the model into two or more submodules where one focuses on personalization or local aggregation has been explored before under the name of Split Learning. Very few of these works consider a generative framework, which is the core consideration of our work. For example, FedPer (Arivazhagan et al., 2019) and ModFL (Liang et al., 2022) suggest splitting the model depth-wise to have a shared module at the beginning, which is trained with FL, and then a personalization module is fine-tuned by the client. Inversely, (Vepakomma et al., 2018) personalize early layers to provide better local data privacy and personalization. A more recent work (Pillutla et al., 2022) proposes the same idea of partial personalization based on domain expertise (e.g. in our case, setting style-related modules for personalization). Finally, HeteroFL (Diao et al., 2020) is another framework for addressing heterogeneous clients by assigning different levels of "locality" where the next level aggregates all parameters excluding ones from the previous levels, thus higher levels are subsets contained in the earlier levels. Finally, (Horváth et al., 2021) trains orderly smaller subsets of weights for devices with less resources. For a two-layer linear neural net and a target linear in the features, it recovers the singular value decomposition in the hidden layer. This is similar in spirit as we aim to decompose the parameters based on the way they are federated.

Regarding the privacy of federated GANs, (Augenstein et al., 2020) considers a setup where the generator can be on the server as it only needs the gradient from the discriminator on the generated data. However, they are mainly concerned with resolving the issue of debugging ML models on devices with non-inspectable data using GANs, which is a different goal from ours. Though, they mention that the generator can be completely deployed on the server, but the discriminator needs direct access to data. Indeed, the discriminator is the part with privacy concerns, which we will discuss.

Combining GANs with contrastive learning has also been explored before. ContraGAN (Kang and Park, 2020) is a conditional GAN model that uses a conditional contrastive loss. Our model is not necessarily conditional, and we use a contrastive loss as a regularizer and train GANs regularly. Another work (Yu et al., 2021) trains GANs with attention and a contrastive objective instead of the regular GAN objective, which is again different from our implementation. Perhaps the most relevant is ContraD (Jeong and Shin, 2021), which is a contrastive discriminator that learns a representation, from which parts are fed to different projections for calculating a GAN objective and a contrastive objective. Our work differs in many details, where the main similarity to prior work is the usage of contrastive learning on a discriminator representation.

3 Preliminaries

This section describes the notations we use in detail to remove ambiguity, especially since our work brings together ideas from different fields. We also describe the frameworks we employ in the design and training of our model, namely federated learning and generative models. In general, we follow prevalent machine learning notation. For example, parameters at time t will be denoted by $\theta(t) \in \mathbb{R}^d$, where d is the dimensionality of parameters. A set of indices from 1 to to M is denoted as $[M] = \{1, \cdots, M\}$. We also follow standard notation used in federated learning (Wang et al., 2021) and GANs (Miyato and Koyama, 2018).

3.1 Federated Learning

Federated learning is a framework for training massively distributed models, where a server typically orchestrates the training that happens locally on the models deployed on user devices (i.e. clients) (Kairouz et al., 2019). The most widely used algorithm for training models in such a setting is `FedAvg` (McMahan et al., 2016), which operates by running stochastic gradient descent on the client's models for a few steps and then aggregating the updates on the server. Another popular algorithm that accounts for data heterogeneity is `FedProx` (Li et al., 2018), which adds a proximal term to the objective that regularizes the local model to be close to the global model.

The aim of federated learning is to optimize the following objective

$$F(\theta) = \mathop{\mathbb{E}}_{i\sim\mathcal{P}}[F_i(\theta)], \quad F_i(\theta) = \mathop{\mathbb{E}}_{\xi\sim\mathcal{D}_i}[f_i(\theta;\xi)], \tag{1}$$

where $\theta \in \mathbb{R}^d$ is the parameter, $\mathcal{P}$ is the client distribution, i is the index of the sampled client, ξ is a random variable of the local distribution $\mathcal{D}_i$ of client i. Due to the nature of our framework, we also denote θ_i as the parameters at client i. We reserve the client index 0 for the server. The client distribution $\mathcal{P}$, for example, could be uniform or proportional to the size of the local dataset. The local distribution $\mathcal{D}_i$ could be a minibatch distribution on the local dataset $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{N_i}$, so that $\xi \sim \mathcal{D}_i$ is a subset of indices of a minibatch.

However, the notation for the distributions that we will follow is based on the generative model literature to avoid confusion. Namely, $q_i(\mathbf{x}, \mathbf{y}) = q(\mathbf{x}, \mathbf{y}|i)$ is the target (data) distribution for client i and $p_i(\mathbf{x}, \mathbf{y})$ is its generative model.

Private Modules. Our model makes use of private modules or parameters (Bui et al., 2019). Namely, for each client i, we split the parameters of its model θ_i into a *private* part θ_i^{pvt} and a *federated* part θ_i^{fed}, typically in a non-trivial, structured manner that is specific to the model design. The only difference between these two parts is that the federated parameters are aggregated normally as in `FedAvg`, whereas the private parameters are kept as is and outside the server's control. Thus, it could be thought of that θ_i^{fed} should be more biased towards a trajectory that is close to all clients, whereas θ_i^{pvt} should be free from this constraint and could potentially traverse different regions from the other clients to account for the domain shift effect, all while still optimizing the global objective (1).

3.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2020) are powerful generative models that are well-known for their high-fidelity image generation capabilities (Karras et al., 2019) among other applications. The framework consists of two competing models playing a minimax game. The competing models are called the *generator* and the *discriminator*. The generator's main objective is to generate samples so that they cannot be discriminated from real samples.

To put it formally, let the true distribution over the data be $q(\mathbf{x})$, the generative model $G : \mathcal{Z} \to \mathcal{X}$ and the discrim-

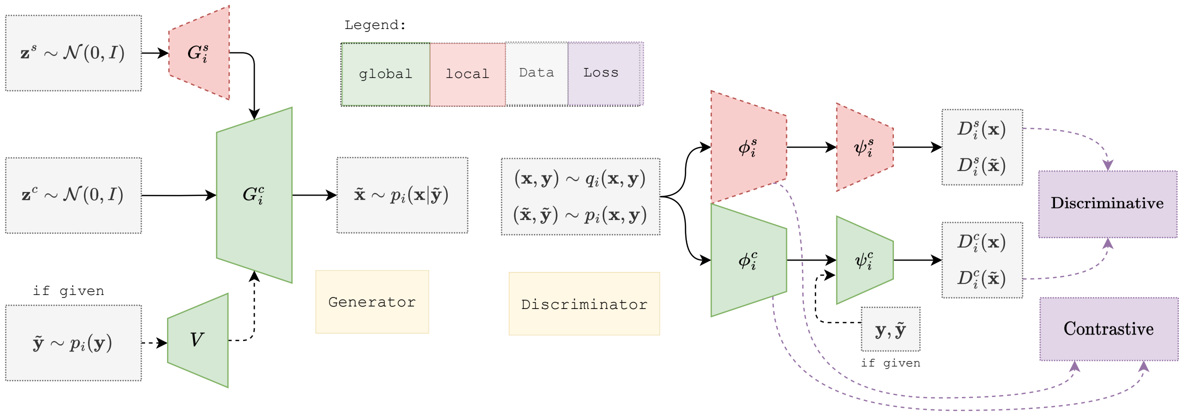

Legend:
global
local
Data
Loss
$\mathbf{z}^s \sim \mathcal{N}(0, I)$
G_i^s
$\mathbf{z}^c \sim \mathcal{N}(0, I)$
G_i^c
$\tilde{\mathbf{x}} \sim p_i(\mathbf{x}|\tilde{\mathbf{y}})$
if given
$\tilde{\mathbf{y}} \sim p_i(\mathbf{y})$
V
Generator
Discriminator
$(\mathbf{x}, \mathbf{y}) \sim q_i(\mathbf{x}, \mathbf{y})$
$(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \sim p_i(\mathbf{x}, \mathbf{y})$
ϕ_i^s
ψ_i^s
$D_i^s(\mathbf{x})$
$D_i^s(\tilde{\mathbf{x}})$
ϕ_i^c
ψ_i^c
$D_i^c(\mathbf{x})$
$D_i^c(\tilde{\mathbf{x}})$
$\mathbf{y}, \tilde{\mathbf{y}}$
if given
Discriminative
Contrastive

Figure 1. The overall structure of the GAN model. Federated modules are aggregated in each communication round, whereas private modules are not. The *discriminative* loss is just the GAN loss as described in Sec. 4.1, whereas the *contrastive* regularization term is explained in more detail in Sec. 4.3. We describe the model's components in more detail in the supplementary material.

inator $D : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} := \mathbb{R}^{\text{height}\times\text{width}\times\text{depth}}$ is the image space and $\mathcal{Z}$ is the latent space. Further, let $p(\mathbf{z})$ be the latent distribution, which is typically standard normal $\mathcal{N}(0, I)$. Thus, the GAN objective can be written as $\min_G \max_D V(D, G)$ where

$$V(D, G) = \mathop{\mathbb{E}}_{q(\mathbf{x})} \log(D(\mathbf{x})) + \mathop{\mathbb{E}}_{p(\mathbf{z})} \log(1 - D(G(\mathbf{z}))). \quad (2)$$

Conditional GANs. In general, we might want our model to make use of labels $\mathbf{y}$ in supervised data, so the space of data becomes $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y}$ can be $\{0, 1\}^{d_y}$ for a set of d_y classes, for example. In our case, the label is a one-hot vector indicating one class at a time, so we also have $\sum_j y_j = 1$. We can define a generative distribution over $\mathbf{x}$ in the previous equation as $p(\cdot|\mathbf{z})p(\mathbf{z}) = G(p(\mathbf{z}))$, so by adding labels we can get $p(\cdot, \mathbf{y}) = p(\cdot|\mathbf{z}, \mathbf{y})p(\mathbf{z})p(\mathbf{y}) := G(p(\mathbf{z}), \mathbf{y})$, where the prior $p(\mathbf{y})$ can be a uniform distribution on the class labels, for example. Then, we can rewrite the GAN objective above as follows

$$\begin{aligned} V(D, G) = &\mathop{\mathbb{E}}_{q(\mathbf{y})} \mathop{\mathbb{E}}_{q(\mathbf{x}|\mathbf{y})} \log(D(\mathbf{x}, \mathbf{y})) \\ &+ \mathop{\mathbb{E}}_{p(\mathbf{y})} \mathop{\mathbb{E}}_{p(\mathbf{x}|\mathbf{y})} \log(1 - D(\mathbf{x}, \mathbf{y})). \quad (3) \end{aligned}$$

The method we adopt for conditioning the GAN is based on (Miyato and Koyama, 2018). First, in the unconditional case, we can write $D(\mathbf{x}) = (\mathcal{A} \circ \psi \circ \phi)(\mathbf{x})$ a decomposition of the discriminator, where $\mathcal{A}$ is the activation (e.g. $\mathcal{A}$ is the sigmoid function in (3)), $\phi(\mathbf{x}) : \mathcal{X} \to \mathcal{R}$ returns the hidden features of $\mathbf{x}$, and $\psi : \mathcal{R} \to \mathbb{R}$, which is often a linear layer. Thus, $\psi(\phi(\mathbf{x}))$ can be thought of as the logit of $p(\mathbf{x})$, where logit = sigmoid^{-1}. In other words, $\psi(\phi(\mathbf{x}))$ is a real number that can be mapped to $[0, 1]$ with the sigmoid function so that it represents a probability. Thus, the choice $\mathcal{A}$ = sigmoid implies $D(\mathbf{x}) = p(\mathbf{x})$.

Then, the basic idea of adding a conditional variable $\mathbf{y}$ is to note that $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ and model $\log p(\mathbf{y}|\mathbf{x})$ as a simple log linear model

$$\log p(\mathbf{y} = y|\mathbf{x}) := (\boldsymbol{v}_y^p)^\intercal \phi(\mathbf{x}) - \log Z(\phi(\mathbf{x})), \quad (4)$$

where $Z(\phi(\mathbf{x})) := \sum_{y' \in \mathcal{Y}} \exp\left((\boldsymbol{v}_{y'}^p)^\intercal \phi(\mathbf{x})\right)$ is the partition function. The authors of (Miyato and Koyama, 2018) observed that the optimal solution of D has the form of the ratio of the two log-likelihoods $D^* = \mathcal{A}(\log q^*(\mathbf{x}, \mathbf{y}) / \log p^*(\mathbf{x}, \mathbf{y}))$ (Goodfellow et al., 2020), and given the assumption that D should follow the optimal form and the log linear model in (4), they reparameterized D by absorbing the terms depending on $\mathbf{y}|\mathbf{x}$ and $\mathbf{x}$ separately to obtain an expression of the form

$$D(\mathbf{x}, \mathbf{y}) = \mathcal{A}\left(\mathbf{y}^\intercal V \phi(\mathbf{x}) + \psi(\phi(\mathbf{x}))\right), \quad (5)$$

where V is the embedding matrix of $\mathbf{y}$.

The choice $\mathcal{A}$ = sigmoid activation is used in the vanilla GAN objective, and $\mathcal{A}$ = identity used for the Wasserstein metric (Gulrajani et al., 2017). We use the latter in our implementation.

3.3 Self-Supervised Learning

Self-supervised learning (SSL) is a modern paradigm of learning in which the supervision is implicit, either by learning a generative model of the data itself or by leveraging some invariance in the data (Liu et al., 2021). For example, we know that the label of an image is invariant to small translations and distortions. Such properties help for learning a low dimensional representation such that the labels, which are not used during training, are highly separable in this representation space, i.e. can be classified with high accuracy using a linear classifier. One well-known method is to use a contrastive objective (Le-Khac et al., 2020), which simply pulls similar data together and pushes different data away. Knowing which data are similar and which are not is the part where implicit supervision happens. For example, data augmentations should result in similar data because the label should be invariant to augmentations.

We describe the general form of a contrastive loss, which includes InfoNCE (Oord et al., 2018) and NT-Xent (Chen et al., 2020). Let ϕ be an encoder that takes data $\mathbf{x}$ and return its representation, and let $\mathbf{x}$ and $\mathbf{x}^+$ be two similar data. Then, given a similarity metric "sim", such as the dot product, and τ a temperature (i.e. sensitivity) hyperparameter, we have

$$\mathcal{L}_{\text{contrast}}(\mathbf{x}; \mathbf{x}^+) = -\log \frac{e^{\text{sim}(\phi(\mathbf{x}),\phi(\mathbf{x}^+))/\tau}}{\mathbb{E}_{\mathbf{x}^-}[e^{\text{sim}(\phi(\mathbf{x}),\phi(\mathbf{x}^-))/\tau}]}. \quad (6)$$

The similarity metric is often a dot product taken after embedding the representations in some metric space via a learnable projection. For example, the projection can be parameterized as a one hidden layer neural net with a non-linearity.

Recently, many methods have been proposed to tackle the problem of mining negative samples (for computing the expectation in the denominator) by momentum-contrasting with a moving-average encoder (He et al., 2020) or by using stop gradient operator (Chen and He, 2021). One technique of particular interest is Barlow Twins (Zbontar et al., 2021), which optimizes the empirical cross-correlation matrix of two similar representations to be as close to the identity matrix as possible, which should maximize correlation and minimizes redundancy, thus the name Barlow for his redundancy-reduction principle (Barlow, 2001). The empirical cross-correlation between two batches of projected representations $\mathbf{A} = g(\phi(\mathbf{x}))$ and $\mathbf{B} = g(\phi(\mathbf{x}^+))$ for some projection g is defined as

$$\mathcal{C}(\mathbf{A}, \mathbf{B})_{ij} = \frac{\sum_b \mathbf{A}_{b,i}\mathbf{B}_{b,j}}{\sqrt{\sum_b (\mathbf{A}_{b,i})^2}\sqrt{\sum_b (\mathbf{B}_{b,i})^2}}. \quad (7)$$

Now, the objective of Barlow Twins is

$$\mathcal{L}_{\text{BT}}(\phi(\mathbf{x}), \phi(\mathbf{x}^+)) = \textstyle\sum_i (1 - \mathcal{C}_{ii})^2 + \lambda_{BT} \sum_{i \neq j} \mathcal{C}_{ij}^2, \quad (8)$$

where we omit the dependence of $\mathcal{C}$ on $g(\phi(\mathbf{x}))$ and $g(\phi(\mathbf{x}^+))$ for clarity. The hyper-parameter λ_{BT} controls the sensitivity of the off-diagonal "redundancy" terms and is often much smaller than 1.

4 Partial Disentanglement with Partially-Federated GANs

In this section, we present a framework for **Pa**rtial **D**isentanglement with **Pa**rtially-**F**ederated (PaDPaF) GANs, which aims at recovering the client invariant representation (the content) by disentangling it from the client-specific representation (the style). A contrastive regularization term is used to force the discriminator to learn a representation of some factor invariant to the variations of other factors, i.e. a content representation invariant to style latent variations given the same content latent, and similarly for the style representation.

4.1 Federated Conditional GANs

Starting from the GAN framework in Sec. 3.2, we aim to optimize (3) in the federated setting with respect to the parameters of D_i and G_i per client i. We define the generator to be a style-based generator with the architecture in (Gulrajani et al., 2017), and a style vectorizer as in (Karras et al., 2019) for producing the conditioning variable of the conditional batch normalization layers (De Vries et al., 2017). The discriminator architecture is based on (He et al., 2016) with the addition of spectral normalization (Miyato et al., 2018).

The style vectorizer is set to be private (i.e. unique to each client), while the base parameters of the generator are federated (i.e. shared). Also, in addition to the normal—or henceforth "content"—discriminator, we introduce another "style" discriminator in parallel, which is assumed to be client-specific with private parameters. This split is done so that we reserve the domain-specific parameters to be private and solely optimized by the clients' optimizers, whereas the "content" parameters are federated and optimized collaboratively.

Namely, for each client i, we introduce $D_i^c(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the content discriminator, and $D_i^s(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ is the style discriminator. As for the generator, we let $G_i^c(\mathbf{z}^c, \mathbf{s}_i, \mathbf{y}) : \mathcal{Z}^c \times \mathcal{S}_i \times \mathcal{Y} \to \mathcal{X}$, and $G_i^s : \mathcal{Z}_i^s \to \mathcal{S}_i$ is the style vectorizer, where $\mathcal{Z}^c, \mathcal{Z}_i^s := \mathbb{R}^{d_\mathbf{z}}$, for example. For simplicity, we assume that D_i^c and D_i^s have the same architectures, but this is not necessary. Let $\phi_i^c : \mathcal{X} \to \mathbb{R}^{d_c}$ for some content representation dimension d_c, and for simplicity, assume that the style representation dimension is the same $d_s = d_c$. Finally, let the parameters of D_i^c, D_i^s, G_i^c, and G_i^s be $\theta_i^{c,D}$, $\theta_i^{s,D}$, $\theta_i^{c,G}$, and $\theta_i^{s,G}$, respectively. Then, we have $\theta_i^{\text{fed}} := [\theta_i^{c,D}; \theta_i^{c,G}]$ the federated parameters and $\theta_i^{\text{pvt}} := [\theta_i^{s,D}; \theta_i^{s,G}]$ the private parameters. We also introduce the shorthand $\theta_i^D = [\theta_i^{c,D}; \theta_i^{s,D}]$ and $\theta_i^G = [\theta_i^{c,G}; \theta_i^{s,G}]$. Now, we rewrite the client objective to follow the convention in (1)

$$f_i(\theta; \xi) = \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y} \in \xi} [\log(D_i^c(\mathbf{x}, \mathbf{y}) + \log(D_i^s(\mathbf{x}))] + \mathop{\mathbb{E}}_{\tilde{\mathbf{x}},\tilde{\mathbf{y}} \sim p_i(\cdot;\theta)} [\log(1 - D_i^c(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})) + \log(1 - D_i^s(\tilde{\mathbf{x}}))], \quad (9)$$

where $\xi \sim q_i(\cdot)$. We make the dependence on θ explicit for the generative model p_i since $p_i(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}; \theta) = G_i^c(\mathbf{z}^c, G_i^s(\mathbf{z}_i^s), \tilde{\mathbf{y}})$ where $\mathbf{z}^c$, $\mathbf{z}_i^s$ are standard Gaussian, i.e. $\mathbf{z}^c, \mathbf{z}_i^s \sim \mathcal{N}(0, I)$ and $\tilde{\mathbf{y}}$ is drawn randomly from $q_i(\mathbf{y})$. We make the dependence on θ implicit elsewhere, making the objective less convoluted.

4.2 Partially-Federated Conditional GANs

When the client's data distributions are not iid—which is clearly the case in practice—then we can have a confounding variable, which is the client's domain variable i. In our case, we assume that each client has its own domain and unique variations. In practice, especially when we have hundreds of millions of clients, meta-data can be used to cluster clients into similar domains. The implementation of this idea will be left for future work.

Before we proceed, we make a simplifying assumption about the content of $\mathbf{y}$.

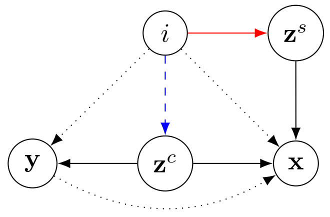
Assumption 1 (Independence of content and style) *Given an image $x \sim q(\mathbf{x})$, the label $\mathbf{y}$ is independent from i, i.e. $\mathbf{y}$ should contain no information about i given $\mathbf{x}$. Namely,*

$$q_i(\mathbf{y}|\mathbf{x}) = q(\mathbf{y}|\mathbf{x}). \tag{10}$$

It is worth mentioning that this assumption might not be entirely true in practice. Take, for example, the case with MNIST, where some client writes 1 and 7 very similar to each other, and another client also writes 1 in the same way but always writes a 7 with a slash, so it is easily distinguishable from a 1. Then, a sample of the digit 7 without a slash would be hard to distinguish from a 1 for the first client but easily recognizable as a 1 for the second client. Hence, given $\mathbf{x}$, there could be some information in the dependence $\mathbf{y}|\mathbf{x}$ that would allow us to discriminate the domain i from which $\mathbf{x}$ came from. However, we do not concern ourselves with such edge cases and make the simplifying assumption that the global representation of the content of $\mathbf{x}$ should be, in theory, independent from the local variations introduced by the client.

Given this assumption, we would like to construct our base generative model such that $p_i(\mathbf{y}|\mathbf{x}) := p(\mathbf{y}|\mathbf{x}, i)$ is invariant to i and $p_i(\mathbf{x}) = p(\mathbf{x}|i)$ generates personalized data from domain i. First, we let $p(i)$ the probability of sampling client i, which we assume to be uniform on the clients, and $p(\mathbf{y})$ the prior of the labels, which we also assume to be uniform on the labels. Note here that the label distribution for a given i could be disjoint or has a small overlap with different i's. This is particularly true in practice, as clients rarely have access to all possible labels for a specific problem. More importantly, it implies that $\mathbf{x}$ can depend non-trivially on i as well (e.g. through data augmentations, more details on the construction in the Appendix). However, we want our model to be agnostic to the client's influence on $\mathbf{y}|\mathbf{x}$ for a globally-consistent inference while being specific to its influence on $\mathbf{x}$ for a personalized generation.

As for $p_i(\mathbf{y}|\mathbf{x})$, we enforce the independence of $\mathbf{y}|\mathbf{x}$ from i via a split of private and federated parameters. This split should influence the mechanism of the generative latent variables $\mathbf{z}^c$ and $\mathbf{z}_i^s$ of $\mathbf{x}$, so that $\mathbf{z}^c$'s generative mechanism becomes increasingly independent of i as we run federated averaging on it, while $\mathbf{z}^s$'s generative mechanism is private and depends on i. The causal model we assume is shown in Fig. 2, and the idea of the independence of mechanisms is inspired by (Gresele et al., 2021).



$$\begin{tikzcd} & i \arrow[r] \arrow[d, dashed] \arrow[dl, dotted] \arrow[dr, dotted] & \mathbf{z}^s \arrow[d] \\ \mathbf{y} \arrow[rr, bend right, dotted] & \mathbf{z}^c \arrow[l] \arrow[r] & \mathbf{x} \end{tikzcd}$$

Figure 2. A causal model with the generative latent variables $\mathbf{z}^c$ and $\mathbf{z}^s$ of $\mathbf{x}$. The dotted arrows show the underlying causal model without latents. Both latents depend on the client i, and we assume that the content latent $\mathbf{z}^c$ generates the label $\mathbf{y}$ as well. The blue arrows drop when we $do(\mathbf{z}^c)$, and the red arrows drop when we $do(\mathbf{z}^s)$. The dashed arrows indicate the reduced influence of i over $\mathbf{z}^c$ as we run federated averaging. $\mathbf{z}^s$ and its mechanism, on the other hand, are specific to the client in our case.

We can write the full probability distribution as $p(\mathbf{y}|\mathbf{z}^c)p(\mathbf{x}|\mathbf{z}^c, \mathbf{z}^s)p(\mathbf{z}^c|i)p(\mathbf{z}^s|i)p(i)$. Assuming an optimal global solution for all $\theta_i^c \to \theta^{c,\star}$ exists as well as local optimal solutions $\theta_i^{s,\star}$, and assuming they are attainable with federated learning on (9), we have that

$$p(\mathbf{y}, \mathbf{z}^c, \mathbf{z}^s|\mathbf{x}, i) \overset{t\to\infty}{\longrightarrow} p(\mathbf{y}|\mathbf{z}^c)p(\mathbf{z}^c|\mathbf{x}; \theta^{c,\star})p(\mathbf{z}^s|\mathbf{x}, i; \theta_i^{s,\star}). \tag{11}$$

Thus, we can look at the problem of inferring $\mathbf{y}, \mathbf{z}^c, \mathbf{z}^s$ given $\mathbf{x}$ from client i as a problem of "inverting the data generating process" $p(\mathbf{z}^c|\mathbf{x}; \theta_i^c)$ and $p(\mathbf{z}^s|\mathbf{x}, i; \theta_i^s)$, and then classifying $\mathbf{y}$ from $\mathbf{z}^c$. We will next show how we can disentangle the content latent from the style latent using a contrastive regularizer on the discriminators, inspired by recent results in contrastive learning (Zimmermann et al., 2021).

4.3 Contrastive Regularization

The main motivation for partitioning our model into federated and private components is that the federated components should be biased towards a solution invariant to the client. This is a desirable property to have for predicting $\mathbf{y}$ given $\mathbf{x}$. However, from a generative perspective, we still want to learn $\mathbf{x}$ given $\mathbf{y}$ and i, so we are interested in learning a personalized generative model that can model the difference in changing the client i while keeping $\mathbf{x}$ and $\mathbf{y}$ fixed, and similarly change x while keeping i fixed. This type of intervention can help us understand how to represent these variations better.

Assume for now that we do not have $\mathbf{y}$. We want to learn how i influences the content in $\mathbf{x}$. The variable i might have variations that correlate with the content, but the variations that persist across multiple clients should be the ones that can potentially identify the content of the data. From another point of view, a datum with the same content might have a lot of variations within one client. Still, if these variations are useless for another client, we should not regard them as content-identifying variations. Thus, our proposed approach tries to maximize the correlation between the content representation of two generated images with the same content latent. This is also applied analogously to style. To achieve that, we use the Barlow Twins objective (8) as a regularizer and the generator as a model for simulating these interventions. Indeed, such interventional data are almost impossible to come by in nature without simulating the generative process.

First, let g be a projector from the representation space to the metric embedding space, and let us denote a shorthand for the embedding of the #-discriminator representation of a sample from the generator as $\Phi_i^{\#}(\mathbf{z}^c, \mathbf{z}^s; \mathbf{y}) = g_i\left(\phi_i^{\#}((\mathrm{sg}[G_i^c(\mathbf{z}^c, G_i^s(\mathbf{z}^s), \mathbf{y})])\right)$, where $\# \in \{c, s\}$ and sg is a stop gradient operator to simulate a sample from the generative model without passing gradients to it. Omitting $\mathbf{y}$ for clarity, the Barlow Twins regularizer will then be

$$\begin{aligned}\Gamma_i(\theta; \mathbf{z}^c, \mathbf{z}^s) = \underset{\tilde{\mathbf{z}}^c, \tilde{\mathbf{z}}^s}{\mathbb{E}} [&\mathcal{L}_{\text{BT}}\left(\Phi_i^c(\mathbf{z}^c, \mathbf{z}^s), \Phi_i^c(\mathbf{z}^c, \tilde{\mathbf{z}}^s)\right) \\ &+ \mathcal{L}_{\text{BT}}\left(\Phi_i^s(\mathbf{z}^c, \mathbf{z}^s), \Phi_i^s(\tilde{\mathbf{z}}^c, \mathbf{z}^s)\right)]. \end{aligned} \tag{12}$$

The intuition here is that the correlation of the content representation between two images generated by fixing the content latent and changing other variables should be exactly the same, and similarly for the style and other components if any. This is related to the intervention concept in causal inference (Pearl, 2009), and assuming the generator is optimal, we can simulate a real-world intervention and self-supervise the model by a contrastive regularization objective as in (12). If we let $\mathcal{L}_{\text{BT}}^{\#}(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{L}_{\text{BT}}\left(g_i(\phi_i^{\#}(\mathbf{x}_1)), g_i(\phi_i^{\#}(\mathbf{x}_2))\right)$ the consistency regularizer from the #-discriminator's perspective, then we can rewrite the objective in terms of interventions on the generator's latent variables as

$$\begin{aligned}\Gamma_i(\theta; \mathbf{z}^c, \mathbf{z}^s) =& \underset{\tilde{\mathbf{z}}^s}{\mathbb{E}} \underset{\substack{\mathbf{x}_1, \mathbf{x}_2 \sim \\ p(\cdot | \tilde{\mathbf{z}}^s, do(\mathbf{z}^c))}}{\mathbb{E}} \mathcal{L}_{\text{BT}}^c(\mathbf{x}_1, \mathbf{x}_2) \\ &+ \underset{\tilde{\mathbf{z}}^c}{\mathbb{E}} \underset{\substack{\mathbf{x}_1, \mathbf{x}_2 \sim \\ p(\cdot | \tilde{\mathbf{z}}^c, do(\mathbf{z}^s))}}{\mathbb{E}} \mathcal{L}_{\text{BT}}^s(\mathbf{x}_1, \mathbf{x}_2),\end{aligned} \tag{13}$$

Thus, we can write the regularized client objective as

$$F_i(\theta; \lambda) = \underset{\mathbf{x}, \mathbf{y}}{\mathbb{E}} [f_i(\theta; \mathbf{x}, \mathbf{y})] + \lambda \underset{\mathbf{z}^c, \mathbf{z}^s}{\mathbb{E}} \Gamma_i(\theta; \mathbf{z}^c, \mathbf{z}^s). \tag{14}$$

See Fig. 2 for a clearer understanding of the interventions.

Implementation details. The algorithm for training our model is a straightforward implementation of GAN training in a federated learning setting. We use `FedAvg` (McMahan et al., 2016) algorithm as a backbone for aggregating and performing the updates on the server. The main novelty stems from combining a GAN architecture that can be decomposed into federated and private components, as depicted in Fig. 1, with a contrastive regularizer (12) for the discriminator in the GAN objective. The training algorithm and other procedures are described in more detail in the Appendix.

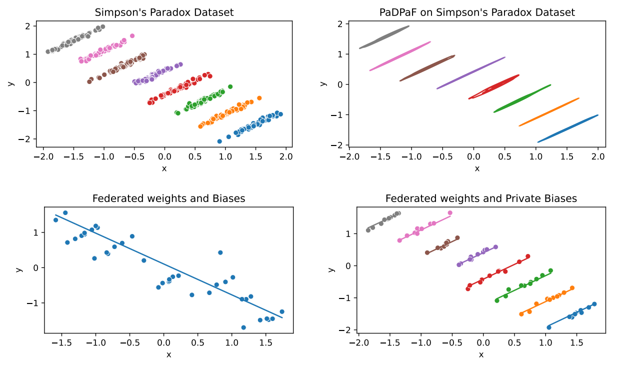


Figure 3. Our model can learn the right data generating process for this dataset (top right). Top right is the density plot of the learned PaDPaF generator. Federated averaging on a linear model will yield a linear regression solution at best as shown on the bottom left, whereas a linear model with a private bias has better generalization (per client) as the shown on the bottom right. Note that only a sample of the data points is shown for clarity, where the top left plot shows the full dataset. (color = client)

5 Experiments

We run three experiments to show the capabilities of our proposed model. First, we run a simplified version of our model on a simple linear regression problem with data generated following the Simpson's Paradox as shown in Fig. 3. Next, we run the main experiment on MNIST (LeCun et al., 1998). Finally, we show our model's performance on CelebA (Liu et al., 2015) to show its robustness and generalization. More experimental details can be found in the supplementary material. We generally use Adam (Kingma and Ba, 2014) for both the client's and the server's optimizer.

In our experiments, we show how our generator disentangles content from style visually. We also show how the generator can generate locally-unseen labels or attributes, i.e. generate samples from the client's distribution given labels or attributes that the client has never seen. Next, we show how the representation learned by the content discriminator contains sufficient information so that a simple classifier on it yields high accuracy on downstream tasks.

Linear Regression. We generate a federated dataset following Simpson's Paradox by fixing a weight w and changing

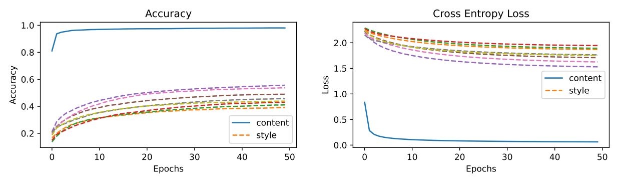


Figure 4. Accuracy and loss of a linear classifier on the content and style representations of an unconditional GAN on MNIST. The final accuracy is 98.1%.

the bias b across clients i, which would be $y = wx + b_i$. Further, for each client, we restrict x to regions such that the opposite trend is observed when looking at data from all clients (see Fig. 3).

Training a naive federated linear classifier leads to seemingly acceptable performance but learns the complete opposite trend as the negative trend is only due to the way of sampling $x|i$ and shifted bias. By federating the weight and keeping the bias private, we allow each client to learn a shared trend with their own biases. Then, training a simple linear router that outputs a mixture of each clients output, we can achieve significantly smaller error, generalize better, and keep the client's models private and unchanged.

We can train our proposed model on this dataset by simplifying the architecture and changing the generator as above (with federated weight and private bias). Namely, for all clients i, we generate a point $\mathbf{x} = w\mathbf{z}^c + b_i$, where $\mathbf{z}^c \sim \mathcal{N}(0, 1) \in \mathbb{R}$ and $\mathbf{x}, w, b_i \in \mathbb{R}^2$. We can also consider $\mathbf{x} \sim w^c\mathbf{z}^c + w_i^s\mathbf{z}^s + b^c + b_i^s$, where $\mathbf{z}^s \sim \mathcal{N}(0, 1)$. We show the performance of the latter model in Fig. 3.

MNIST. In our MNIST experiments, we generate a federated dataset by partitioning MNIST equally on all clients and then adding a different data augmentation for each client. This way, we can test whether our model can generate global instances unseen to the client based on its own data augmentation. The data augmentations in the figures are as follows (in order): 1) zoom in, 2) zoom out, 3) colour inversion, 4) Gaussian blur, 5) horizontal flip, 6) vertical flip, 7) rotation of at most 40 degrees, and 8) brightness jitter.

See Fig. 5 for samples from a conditional GAN. One interesting observation about the samples is that, since the number 7 was seen only in the horizontally flipped client, it retains this flipped direction across other clients as well. This is not a bug in our model but a feature. In general, learning flipping as a style is not incorporated in the design of our generator, and the same goes for rotations (see Fig. 5, lower left). However, the content was generally preserved through most other clients. We show performance on other more difficult augmentations in the Appendix.
Finally, Fig. 4 shows that the content representation contains sufficient information such that a linear classifier can achieve of 98.1% accuracy on MNIST. It is also clear that the style representations do not contain as much information about the content. This empirically confirms that our method can capture the most from $\mathbf{z}^c$ and disentangle it from $\mathbf{z}^s$.

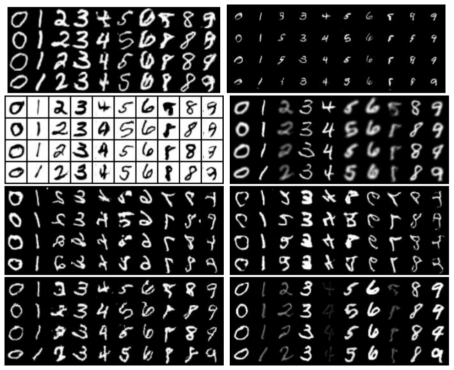


Figure 5. Samples from a conditional GAN. Top to bottom, $\mathbf{z}^c$ is changed while $\mathbf{z}^s$ and $\mathbf{y}$ are fixed. Left to right, $\mathbf{z}^c$ is fixed while $\mathbf{z}^s$ and $\mathbf{y}$ are changed. $\mathbf{z}^c$ and $\mathbf{z}^s$ are the same across clients.

CelebA. The federated dataset is created by a partition based on attributes. Given 40 attributes and 10 clients, each client is given 4 unique attributes, which all of its data have. The content part should then capture the general facial structures, and the style should capture the other non-global variations. Our results again confirm the empirical feasibility of our model.

6 Discussion

Our framework leaves some room for improvements and further work to be done in multiple directions. Here, we discuss some of those directions and highlight interesting aspects we can exploit.

Client shift. Our architecture adapts to different clients i under covariate shift $q_i(\mathbf{x})$ and prior shift $q_i(\mathbf{y})$. Can we extend it to the case of temporal heterogeneity? For example, our model might not work well with clients with diurnal activity.

Model improvement. How can we design a generator that can effectively incorporate clients' variations? Is the choice of private parameters mostly problem-specific? What about different modalities of data, such as language and sensorimotor control? Can we use an attention-based mechanism for controlling the aggregation or sampling mechanisms for our model?

Generator on server. Our model is as private as `FedAvg`. In fact, the federated part of the generator can stay completely in the server. Can we train the federated discriminator completely on the server as well to ensure full privacy? The server would then need discriminative features of the data from the client, which might reveal the information about the client. Still, it would be interesting to explore the benefits of having the generator be completely on the server.

Conclusion. We introduced a framework combining federated learning, generative adversarial learning, and (causal) representation learning together by leveraging the federated setting for learning a client-invariant representation based on a causal model of the data-generating process, which we model as a GAN. We disentangle the content and style latents using a contrastive regularizer on the discriminator. Experiments validate our framework and show that our model is effective at personalized generation and self-supervised classification and benefits further from supervised data.

References

M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary. Federated learning with personalization layers, 2019. URL https://arxiv.org/abs/1912.00818.

S. Augenstein, H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, and B. A. y Arcas. Generative models for effective ML on private, decentralized datasets. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJgaRA4FPH.

H. Barlow. Redundancy reduction revisited. *Network: computation in neural systems*, 12(3):241, 2001.

D. Bui, K. Malik, J. Goetz, H. Liu, S. Moon, A. Kumar, and K. G. Shin. Federated user representation learning. *arXiv preprint arXiv:1909.12535*, 2019.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. *Advances in Neural Information Processing Systems*, 30, 2017.

E. Diao, J. Ding, and V. Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

C. Fan and P. Liu. Federated generative adversarial learning, 2020. URL https://arxiv.org/abs/2005.03793.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

L. Gresele, J. Von Kügelgen, V. Stimper, B. Schölkopf, and M. Besserve. Independent mechanism analysis, a new concept? *Advances in neural information processing systems*, 34:28233–28248, 2021.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

S. Horváth, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12876–12889. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/6aed000af86a084f9cb0264161e29dd3-Paper.pdf.

J. Jeong and J. Shin. Training gans with stronger augmentations via contrastive discriminator. *arXiv preprint arXiv:2103.09742*, 2021.

P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning, 2019. URL https://arxiv.org/abs/1912.04977.

M. Kang and J. Park. Contragan: Contrastive learning for conditional image generation. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21357–21369. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f490c742cd8318b8ee6dca10af2a163f-Paper.pdf.

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence, 2016a. URL https://arxiv.org/abs/1610.02527.

J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency, 2016b. URL https://arxiv.org/abs/1610.05492.

L. Kong, S. Xie, W. Yao, Y. Zheng, G. Chen, P. Stojanov, V. Akinwande, and K. Zhang. Partial disentanglement for domain adaptation. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11455–11472. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/kong22a.html.

P. H. Le-Khac, G. Healy, and A. F. Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020. doi: 10.1109/ACCESS.2020.3031549.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks, 2018. URL https://arxiv.org/abs/1812.06127.

K.-Y. Liang, A. Srinivasan, and J. C. Andresen. Modular federated learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2022. doi: 10.1109/ijcnn55064.2022.9892377. URL https://doi.org/10.1109%2Fijcnn55064.2022.9892377.

X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas. Communication-efficient learning of deep networks from decentralized data. 2016. doi: 10.48550/ARXIV.1602.05629. URL https://arxiv.org/abs/1602.05629.

T. Miyato and M. Koyama. cgans with projection discriminator. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=ByS1VpgRZ.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

J. Pearl. *Causality*. Cambridge university press, 2009.

K. Pillutla, K. Malik, A.-R. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao. Federated learning with partial model personalization. In K. Chaudhuri, S. Jegelka, L. Song,

C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17716–17758. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/pillutla22a.html.

M. Rasouli, T. Sun, and R. Rajagopal. Fedgan: Federated generative adversarial networks for distributed data, 2020. URL https://arxiv.org/abs/2006.07228.

B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5): 612–634, 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, B. A. y. Arcas, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, S. Diggavi, H. Eichner, A. Gadhikar, Z. Garrett, A. M. Girgis, F. Hanzely, A. Hard, C. He, S. Horvath, Z. Huo, A. Ingerman, M. Jaggi, T. Javidi, P. Kairouz, S. Kale, S. P. Karimireddy, J. Konecny, S. Koyejo, T. Li, L. Liu, M. Mohri, H. Qi, S. J. Reddi, P. Richtarik, K. Singhal, V. Smith, M. Soltanolkotabi, W. Song, A. T. Suresh, S. U. Stich, A. Talwalkar, H. Wang, B. Woodworth, S. Wu, F. X. Yu, H. Yuan, M. Zaheer, M. Zhang, T. Zhang, C. Zheng, C. Zhu, and W. Zhu. A field guide to federated optimization, 2021. URL https://arxiv.org/abs/2107.06917.

Y. Wang and M. I. Jordan. Desiderata for representation learning: A causal perspective. *arXiv preprint arXiv:2109.03795*, 2021.

R. Yonetani, T. Takahashi, A. Hashimoto, and Y. Ushiku. Decentralized learning of generative adversarial networks from non-iid data, 2019. URL https://arxiv.org/abs/1905.09684.

N. Yu, G. Liu, A. Dundar, A. Tao, B. Catanzaro, L. S. Davis, and M. Fritz. Dual contrastive loss and attention for gans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6731–6742, 2021.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *International conference on machine learning*, pages 819–827. PMLR, 2013.

R. S. Zimmermann, Y. Sharma, S. Schneider, M. Bethge, and W. Brendel. Contrastive learning inverts the data generating processx. 139:12979–12990, 2021. URL http://proceedings.mlr.press/v139/zimmermann21a.html.

A Algorithm

In this section, we write the training algorithm in detail for completeness. The training is a straightforward application of `FedAvg` on GANs without its private parameters.

Algorithm 1 Contrastive GANs with Partial FedAvg

```
1:  Input: Clients $i \in [M]$, datasets $X_i = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{|X_i|}$, target distribution $q(\cdot|i)q(i)$ over $X_i$, GAN models $p_i(\cdot;\theta_i)$, GAN objectives $f_i$, server and clients first-order optimizers $\text{OPT}_0$ and $\text{OPT}_i$ with step sizes $\eta_0$ and $\eta_i$, client's loop stopping probability $\pi_i \propto |X_i|^{-1}$, and $T_D$ iterations for training $D$ vs. $G$.
2:  Output: $\theta_0^{\text{fed}}$ and $\theta_i^{\text{pvt}}, \forall i$.
3:  Set $\theta_i := \theta_0, \forall i \in [M]$
4:  for $\tau = 1, \cdots, \tau_{\max}$ do
5:      Sample subset of clients $I \sim q(i)$
6:      for $i \in I$ in parallel do
7:          for $t = 1, \cdots, t_{\max}$ do
8:              train$_G \leftarrow \mathbb{1}[t \bmod T_D + 1 = 0]$
9:              $\theta_i(t+1) \leftarrow \text{GANOPT}(i, \theta_i(t), \text{train}_G; f_i, q_i)$
10:             if $1 \sim \text{Bernoulli}(\pi_i)$ then
11:                 $\Delta_i \leftarrow \theta_i(1) - \theta_i(t)$
12:                 $\Delta_i^{\text{pvt}} \leftarrow 0$
13:                 break loop
14:             end if
15:         end for
16:     end for
17:     $\theta_0(\tau+1) \leftarrow \text{OPT}_0\left(\frac{1}{|I|}\sum_{i\in I}\Delta_i;\, \theta_0(\tau), \eta_0\right)$
18:     $\theta_i^{\text{fed}} \leftarrow \theta_0^{\text{fed}}$ for all clients $i \in [M]$
19: end for
20: procedure GANOPT($i, \theta_i$, train$_G$; $f_i, q_i$)
21:     if train$_G$ then
22:         return $\text{OPT}_i\left(\nabla_{\theta_i^G} f_i(\theta_i, \cdot);\, \theta_i, \eta_i\right)$
23:     else
24:         Sample $\xi \sim q(\cdot|i)$
25:         return $\text{OPT}_i\left(-\nabla_{\theta_i^D} f_i(\theta_i, \xi);\, \theta_i, 2\eta_i\right)$
26:     end if
27: end procedure
```

Here, we let $\textsc{Opt}(\theta, \nabla f(\theta))$ be a first-order optimizer, such as SGD or Adam, that takes the current parameters θ, and the gradient $\nabla f(\theta)$ and returns the updated parameters given step size η. We also define a subroutine $\text{GAN}\textsc{Opt}$ that runs a single descent or ascent step on the GAN's objective. The term train$_G$ is a boolean variable that specifies whether the generator or the discriminator should be trained. In the GAN literature, the discriminator is often trained 3 to 5 times as much as the generator. In our experiments, we let $T_D = 3$, so D is trained 3 times before G is trained, but we also update θ^D with double the learning rate, following the Two Time-scale Update Rule. As for the client's loop, we can simply train it on the dataset for one epoch before breaking the loop so that $t_{\text{max}} = |X_i|$ and $\pi_i = 0$, but we add a loopless option as well by letting $t_{\text{max}} = \infty$ and $\pi_i \propto |X_i|$ for the same training behaviour on average.

Note that we could have removed line 12 of the algorithm and let line 11 be $\Delta_i^{\text{fed}} \leftarrow \theta_i^{\text{fed}}(1) - \theta_i^{\text{fed}}(t)$, but we chose the current procedure to emphasize that the server's update on the private parameters is always set to 0.

B Model Architecture

Our model's architecture is composed of regular modules that are widely used in the GAN literature. The generator is a style-based generator with a ResNet-based architecture (Gulrajani et al., 2017), where we use a conditional batch normalization layer (De Vries et al., 2017) to add the style, which is generated via a style vectorizer (Karras et al., 2019). The discriminator's architecture is ResNet-based as well (He et al., 2016) with the addition of spectral normalization (Miyato et al., 2018). In the case of conditioning on labels, we follow the construction of the projection discriminator as in (Miyato and Koyama, 2018), so we add an embedding layer to the discriminator, and we adjust the conditional batch normalization layers in the generator by separating the batch into two groups channel-wise, and condition each group separately. Finally, we add a projector to the discriminator for contrastive regularization, which is a single hidden layer ReLU network. Recall that we use two discriminators, one is private, and the other is federated.

As for the model architecture for the linear regression dataset, we simplify the discriminator to a 2-layer neural net with 8 hidden dimensions. The generator is a linear layer and the style vectorizer is also a linear layer with a latent vector of dimension 1. The projectors are linear layers as well.

The variables that control the size of our networks are the image size, the discriminator's feature dimension, and the latent variables' dimension. We make the dimension of the content and style latent variables equal for simplicity. For MNIST, we choose the feature dimension to be 64 and the latent dimension to be 128. For CelebA, we choose the feature dimension to be 128 and the latent dimension to be 256. Kindly refer to the code for more specific details.

C Experiments

In this section, we describe the experiments we run in more detail and show results extra results supporting our framework.

For all our experiments, the client models are handled by "workers". The workers only train their models on a specific subset of clients that does not change. We often assign a worker for each client, but due to constraints on computational resources, we might create a smaller number of workers and assign a specific subset of clients for each worker. We run all experiments on a single NVIDIA A100 SXM GPU 40GB.

Our choice for all the optimizers, including the server's optimizer, is Adam (Kingma and Ba, 2014) with $\beta_1 = 0.5$ and $\beta_2 = 0.9$. We choose Adam due to its robustness and speed for training GANs. We found that choosing learning rates 0.01 and 0.001 for the server optimizer and the client optimizer, respectively, is a good starting point. We also use an exponential-decay learning rate schedule for both the server and the clients, with a decay rate of 0.99 for MNIST (0.98 for CelebA) per communication round. This can help stabilize the GAN's output.

C.1 Linear Regression on Simpson's Paradox

We generate a dataset that demonstrates Simpson's Paradox for M clients as follows. We fix a weight, say, $w := 1$. Then for each client $i \in \{1, \cdots, M\}$, we sample $\mathbf{x}$ uniformly from $[M-i, M-i+1]$ and a bias b_i from $[L(i-1), Li]$ for some constant L. Finally, for each client i, we generate n_i points $y \sim w\mathbf{x} + b_i$ and then normalize $\mathbf{x}*$ and $\mathbf{y}$ by subtracting the mean and dividing by the standard deviation. In our experiments, we choose $M := 8$, $L := 4$, and $n_i :=$ 50 for all i. For training a PaDPaF model on this dataset, we noticed that it is much better for D^c to have slightly larger feature dimension, like 16, whereas D_i^s can be smaller, like 2.

If we train a naive linear regression model on this dataset for 50 epochs with a batch size 10, we would get a mean-squared error about 0.182, whereas similarly training a linear regression model for each client, and then freezing the models and train again a linear routing model would yield an error about 0.004, which is two orders of magnitude smaller. See Fig. 6 for an illustration. Code for reproducing plots and similar errors for this experiment are provided in the linked repository in the main text.

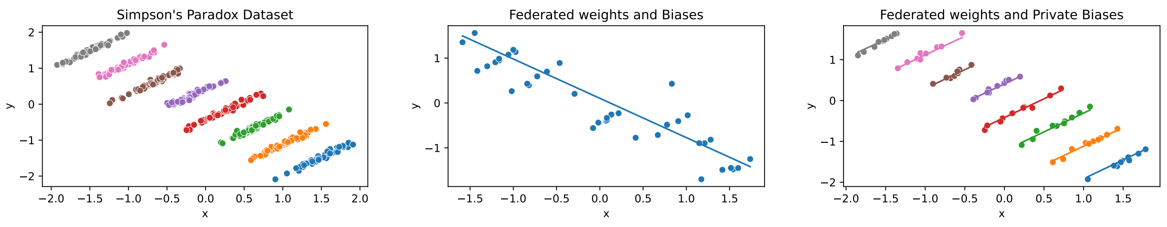
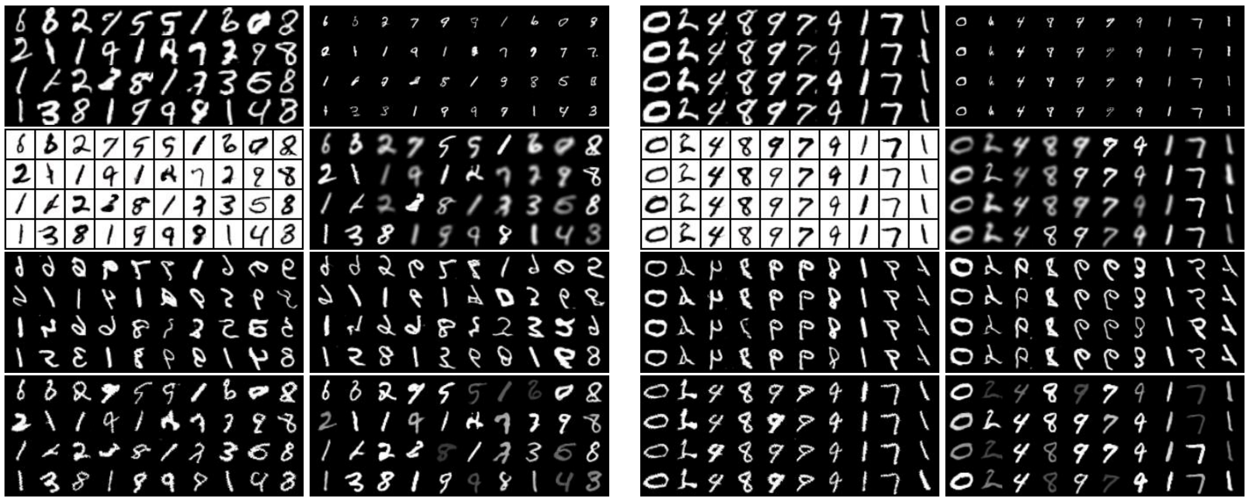


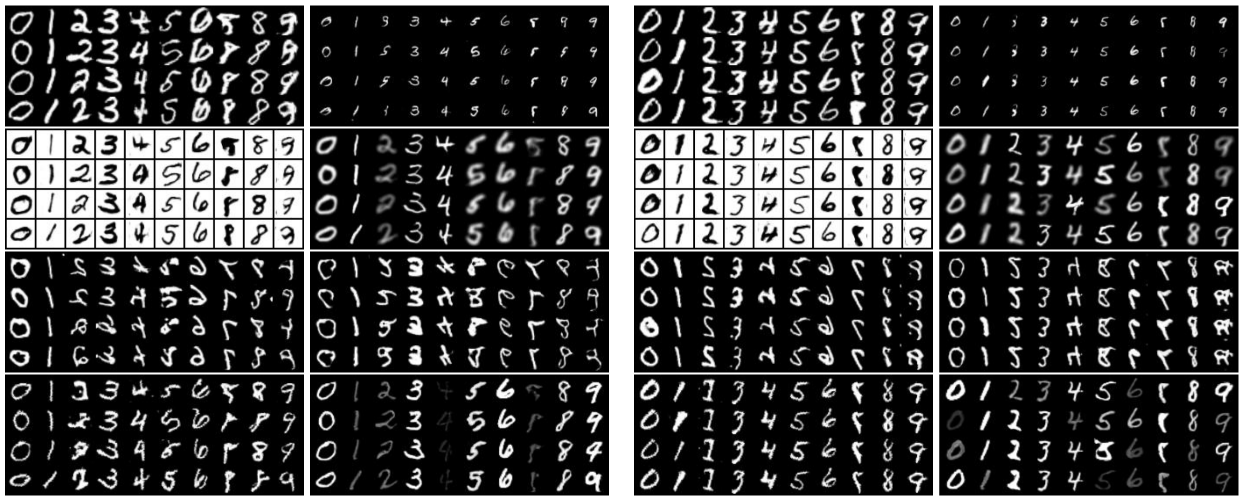
Figure 6. A single linear regression model on simpson's Paradox dataset vs. federated linear regression models with private biases. Note that only a sample is shown in the regression models. The full dataset is shown to the left.



(a) $\mathbf{z}^c$ and $\mathbf{z}^s$ are different across all cells per client. In other words, $\mathbf{z}^c$ and $\mathbf{z}^s$ are the same across clients, but different within client.

(b) Top to bottom, $\mathbf{z}^s$ is changed while $\mathbf{z}^c$ and $\mathbf{y}$ are fixed. Left to right, $\mathbf{z}^s$ is fixed while $\mathbf{z}^c$ and $\mathbf{y}$ are changed.

Figure 7. Samples from a GAN without conditioning. $\mathbf{z}^c$ and $\mathbf{z}^s$ are consistent across clients.



(a) Top to bottom, $\mathbf{z}^c$ is changed while $\mathbf{z}^s$ and $\mathbf{y}$ are fixed. Left to right, $\mathbf{z}^c$ is fixed while $\mathbf{z}^s$ and $\mathbf{y}$ are changed.

(b) Top to bottom, $\mathbf{z}^s$ is changed while $\mathbf{z}^c$ and $\mathbf{y}$ are fixed. Left to right, $\mathbf{z}^s$ is fixed while $\mathbf{z}^c$ and $\mathbf{y}$ are changed.

Figure 8. Samples from a conditional GAN. $\mathbf{z}^c$ and $\mathbf{z}^s$ are consistent across clients.

C.2 MNIST

The MNIST train dataset is partitioned into 8 subsets assigned to 8 clients, and each client is handled by a unique worker. The data augmentations used for each partition are clear from the images. We write the PyTorch code for each data augmentation:

1. Zoom in: `CenterCrop(22)`.
2. Zoom out: `Pad(14)`.
3. Color inversion: `RandomInvert(p=1.0)`.
4. Blur: `GaussianBlur(5, sigma=(0.1, 2.0))`.
5. Horizontal flip: `RandomHorizontalFlip(p=1.0)`.
6. Vertical flip: `RandomVerticalFlip(p=1.0)`.
7. Rotation: `RandomRotation(40)`.
8. Brightness: `ColorJitter(brightness=0.8)`.

In the case of conditioning on $\mathbf{y}$, we further restrict the datasets and drop 50% of the labels **from each partitioned dataset**. This implies that some data will be lost, and each client will see only 50% of the labels. This is intentional as we want to restrict the dataset's size further and test our model's robustness on unseen labels.

The results for training our model on this federated dataset are shown in Fig. 7 for the unconditional case (i.e. $\mathbf{y}$ is unavailable), and Fig. 8 for the conditional case with 50% labels seen per client. The results were generated after training the model for 300 communication rounds. For MNIST, we train the models for a half epoch in each round to further restrict the local convergence for each client.

C.2.1 Adaptation to New Clients (i.e. Data Augmentation)

After training our conditional model on the previous data augmentations, we re-train it for 25 communication rounds on the following data augmentations and show the results:

1. Affine: `RandomAffine(degrees=(30, 70), translate=(0.1, 0.3), scale=(0.5, 0.75))`.
2. Solarize: `RandomSolarize(threshold=192.0)`.
3. Erase: `transforms.RandomErasing(p=1.0, scale=(0.02, 0.1))` (after `ToTensor()`).

This is to show that our conditional model can quickly adapt to new styles without severely affecting its content generation capabilities and its generalization to unseen labels.

C.2.2 Can We Predict the Client from the Image?

We ask a question analogous to the one we are concerned with in representation learning. Instead of predicting the label from the image, we want to predict the augmentation, or the client, that generated the image. We show that we can predict the client with good accuracy (93.3%). See Fig. 11.

The style prediction is done by linear transformations on the content representation and the style representations. For the style representations, we linearly map each representation from each client to a scalar u_i, so that $\boldsymbol{u}$ is a vector of all the scalars. We then pass $\boldsymbol{u}$ through an extra linear transformation to get a vector of logits for $p(i|\mathbf{x})$. Note that this is still a linear transformation of the style representations, but we do it this way to reduce dimensionality.

Note that the content representation can still predict the style with good accuracy. Indeed, we have seen that our generator architecture finds difficulties in assigning rotations to style variations, which is due to the style generator construction as a conditional batch normalization layer, which shows limited capabilities in capturing rotation-like variations. Still, the prediction accuracy from the style representations is better overall, which shows that our model does disentangle, to some extent, the style from the content.

C.2.3 Re-Styling via Content Transfer

We describe a simple method for transferring the content of an image to another style, given the style vectorizers G_i^s. In other words, we can transfer the content of an image taken from one client to another without sharing the image but by finding the approximate latent variables that generate the image and transferring the content latent variable. We shall see that this approach can indeed transfer the content well enough, where the content is up to the generator's expressiveness and its separation of style. For example, we show that our generator finds difficulties in distinguishing some rotations as non-content variations. This is because all other clients have slightly rotated digits as well.

To find the approximate latent variables from the image without any significant overhead, we use the discriminator's representations to predict the latent variables. We show that this approach can predict latents that reconstruct the image very well, and thus we can use the same content latent on the new client.

First, we show the algorithm for reconstructing and re-styling an image from some client. The main idea is to use $\mathcal{L}_{\text{BT}}$ from (8) to maintain an identity correlation between the source content representation and the target (i.e. reconstructed) content representation. We noticed that we do not need to do the same for the style representation. In fact, not enforcing an identity correlation between styles can even improve the results in terms of style consistency.

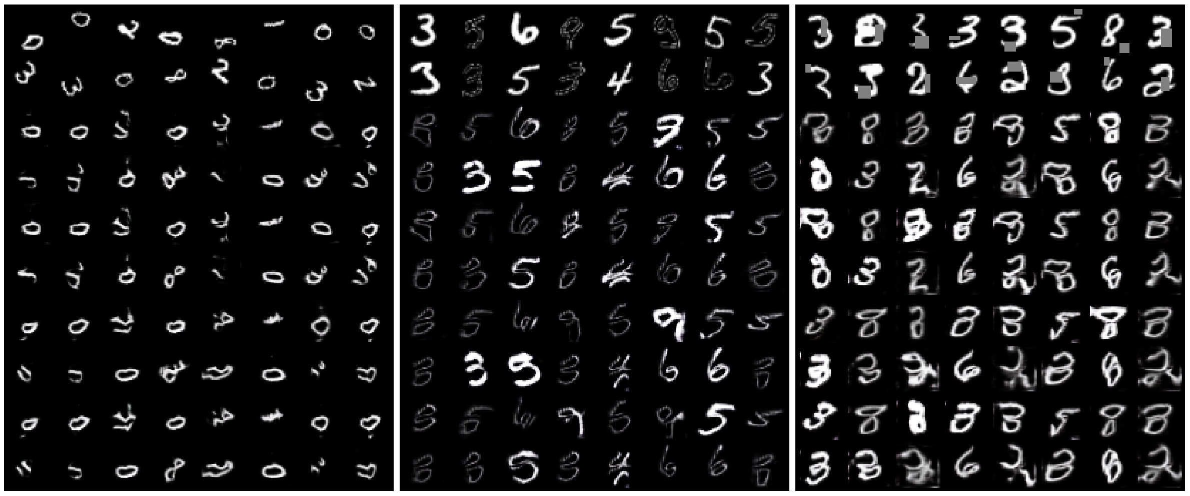
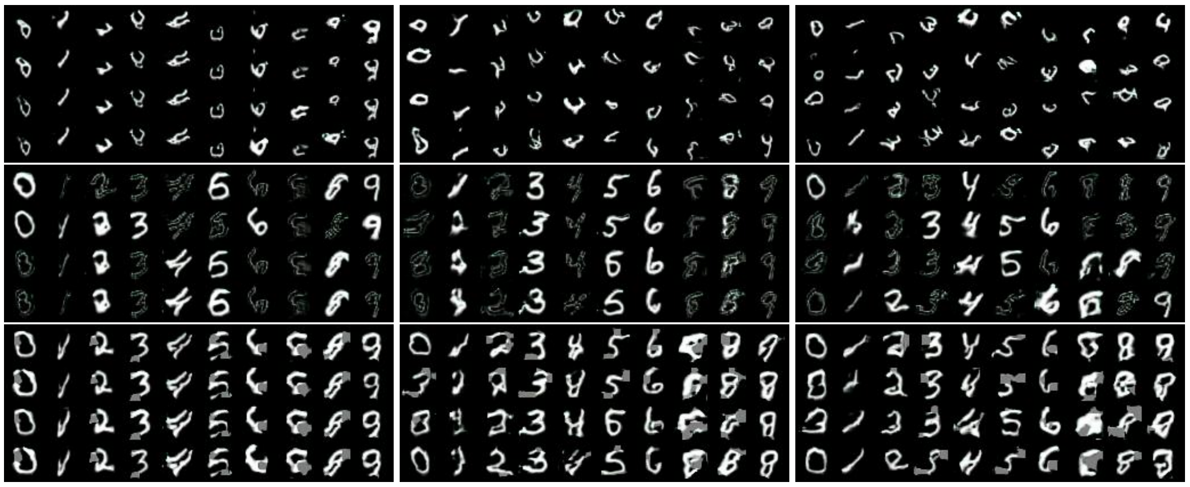


Figure 9. Results after re-training for 1 communication round on the three new data augmentations. Rows 1-2 are real samples and rows 3-8 are generated. Rows 3-6 share the same $\mathbf{z}^c$, and rows 7-10 as well. Rows 3-4 and 7-8 share the same $\mathbf{z}^s$, similarly for rows 5-6 and 9-10.



(a) Changing $\mathbf{z}^s$ per row while fixing $\mathbf{z}^c$. (b) Changing $\mathbf{z}^c$ per row while fixing $\mathbf{z}^s$. (c) Changing both $\mathbf{z}^c$ and $\mathbf{z}^s$.

Figure 10. Results after 25 communication rounds. We can see that solarization is learned as a style, while erosion is learned as a content. The generalization to unseen digits is maintained but less so in the challenging affine augmentation.

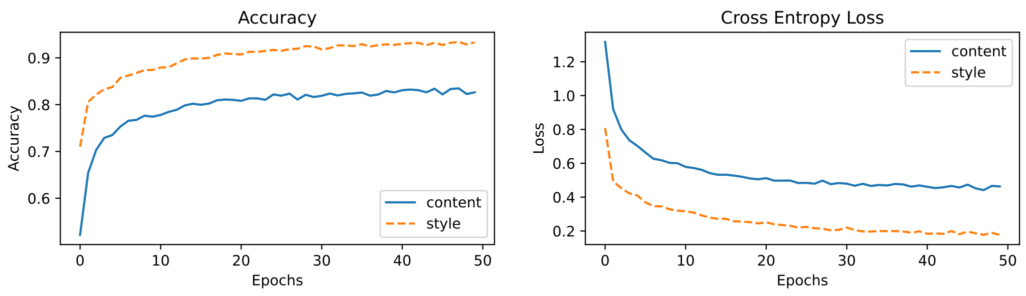


Figure 11. Prediction of style. We see that the content features have good accuracy (82.6%), but the style features can classify the source client with much better accuracy (93.3%).

Algorithm 2 Re-style via content transfer

```
1: Input: Trained models, $t_{\text{max}}$ number of epochs, $\mathbf{x}_a$ (and $\mathbf{y}_a$, if any) a sample from source client $a$, and a target client $b$.
2: Output: Representation-to-latent transforms $\mathfrak{Z}^c$ and $\mathfrak{Z}_i^s$ for each $D_0^c$ and $D_i^s$, resp., $\forall i \in [M]$, and re-styled $\mathbf{x}_b$.
3: for $t = 1, \cdots, t_{\text{max}}$ do
4:     Sample client $i \sim q(i)$, and sample label $\mathbf{y} \sim q_i(\mathbf{y})$, if any.
5:     Sample latents $\mathbf{z}^c \sim q_i(\mathbf{z}^c)$ and $\mathbf{z}^s \sim q_i(\mathbf{z}^s)$.
6:     Sample image $\mathbf{x} \sim p_i(\mathbf{x}|\mathbf{z}^c, \mathbf{z}^s, \mathbf{y})$ (i.e. set $\mathbf{x} = G_0^c(\mathbf{z}^c, G_i^s(\mathbf{z}^s), \mathbf{y})$).
7:     Transform latents $\tilde{\mathbf{z}}^c = \mathfrak{Z}^c(\phi_0^c(\mathbf{x}))$ and $\tilde{\mathbf{z}}_i^s = \mathfrak{Z}_i^s(\phi_i^s(\mathbf{x}))$.
8:     Sample reconstructed image $\tilde{\mathbf{x}} \sim p_i(\mathbf{x}|\tilde{\mathbf{z}}^c, \tilde{\mathbf{z}}^s, \mathbf{y})$.
9:     Minimize $\mathcal{L}_{\text{BT}}(\phi_0^c(\mathbf{x}), \phi_0^c(\tilde{\mathbf{x}}); g_0^c)$ w.r.t. $\mathfrak{Z}^c$ and $\mathfrak{Z}_i^s$ (see (8)).
10: end for
11: Transfer $a \to b$: Sample $\mathbf{x}_b \sim p_b(\mathbf{x}|\mathfrak{Z}_0^c(\phi_0^c(\mathbf{x}_a)), \mathbf{z}_b^s, \mathbf{y}_a)$, where $\mathbf{z}_b^s \sim p_b(\mathbf{z}^s)$.
```

We do not find a good explanation for this. We believe that applying Barlow Twins loss in our setting warrants further empirical and theoretical investigation.

The effectiveness of this procedure can be seen in Figs. 12 and 13, where we transfer the content of some image to a specific client. In Fig. 12, we choose client 8 as the source and client 3 as the target to demonstrate that the reconstruction preserves both the content and style and that the transfer preserves the content. In Fig. 13, notice that, since the MNIST's test set is unseen during training, the unconditional GAN might map some seemingly-easy digits to other similar-looking ones. Simply training on the test set should allow the unconditional GAN to mitigate this effect. Note that this effect is not seen in the conditional GAN, given that the clients see only 50% of the labels during training. It is worth mentioning that, as long as the content representation encoder ϕ_0^c is good enough for images from some unseen domains, like the test set of MNIST, then we can still transfer its content to a known client i with a trained style vectorizer G_i^s.

C.3 CelebA

We test our model on CelebA, which is a more challenging dataset. The dataset contains pictures of celebrities, where each image is assigned a subset of attributes among 40 possible attributes. Thus, we create 40 clients, give a unique attribute to each, and then show them the subset with this attribute. Note that this is not a partition, as some data will be repeated many times, but we want to see if the client will learn a biased style towards the assigned attributes.

To make the simulation feasible, we create 10 workers with their models, each handling a unique subset of 4 clients, and let each worker sample their clients in a round-robin fashion. Thus, each worker will always see at least one of 4 attributes. We train our model for approximately 200 communication rounds, with 2 epochs per round and train the discriminators 5 times as frequently as the generator (i.e. $T_D = 5$).

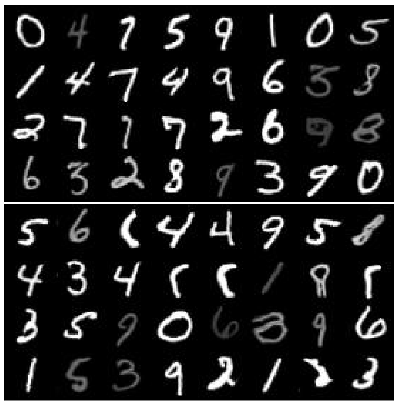
Given some content latent, we can see that the client-specific attributes become more obvious in some clients than others. For example, lipsticks, moustaches, blonde hair, and other attributes are clearer in some clients than in others. See Figs. 15 to 17 for samples from each worker.

For completeness and a better understanding of the generation quality, we list the attributes assigned to each worker:

1. Worker 1: `['Mustache', 'Mouth_Slightly_Open', 'Sideburns', 'Big_Lips']`.
2. Worker 2: `['Attractive', 'Narrow_Eyes', 'Gray_Hair', 'Bald']`.
3. Worker 3: `['Arched_Eyebrows', 'Bangs', 'Chubby', 'Eyeglasses']`.
4. Worker 4: `['Male', 'Rosy_Cheeks', 'Wearing_Necktie', 'High_Cheekbones']`.
5. Worker 5: `['Straight_Hair', 'Wearing_Earrings', 'Black_Hair', 'No_Beard']`.
6. Worker 6: `['5_o_Clock_Shadow', 'Young', 'Wearing_Necklace', 'Wavy_Hair']`.
7. Worker 7: `['Receding_Hairline', 'Bushy_Eyebrows', 'Goatee', 'Heavy_Makeup']`.
8. Worker 8: `['Pointy_Nose', 'Blond_Hair', 'Double_Chin', 'Oval_Face']`.
9. Worker 9: `['Big_Nose', 'Smiling', 'Blurry', 'Brown_Hair']`.
10. Worker 10: `['Wearing_Lipstick', 'Pale_Skin', 'Bags_Under_Eyes', 'Wearing_Hat']`.

Celebrity identification from representation. We also test the content and style features on a celebrity-identification task. The construction of the dataset is done so that we only test whether the style introduces a bias towards some attributes given some content, so we do not expect to see disentanglement between content and style. If we see one

(a) Source (generated) images.


(b) Reconstructed images.

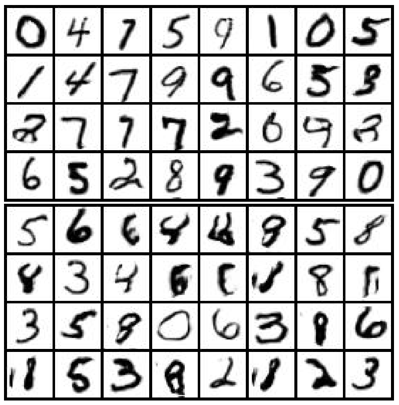
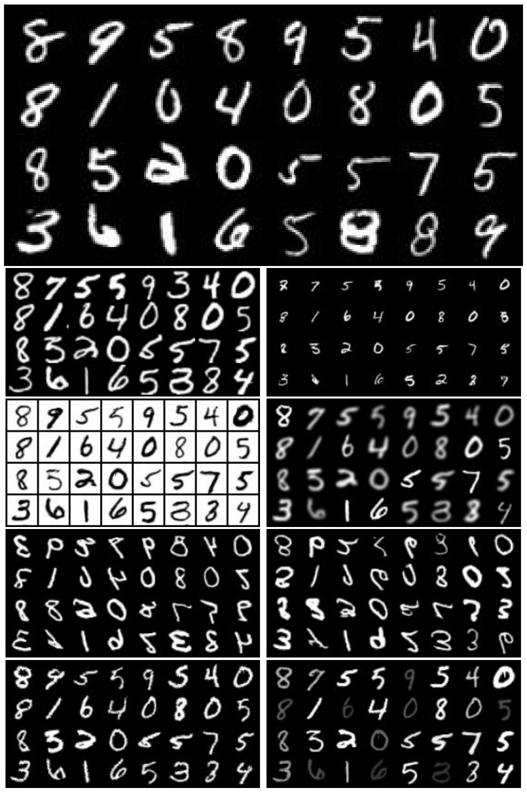

(c) Re-styled with transferred content.

Figure 12. The images show results from the regular, unconditional GANs, whereas the bottom show results for conditional GANs with 50% labels seen per client. Observe how the content stays preserved after transfer. Here, we show an example of transferring the content from client 8 (brightness) to client 3 (color inversion).


(a) Unconditional.

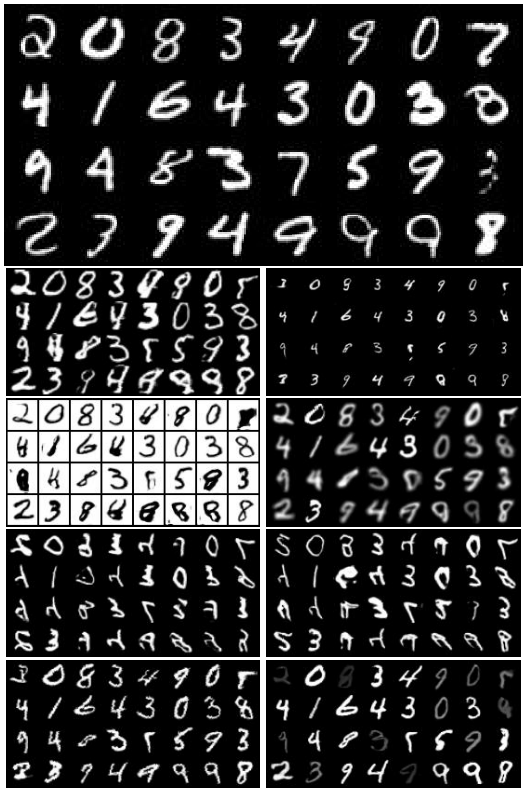

(b) Conditional, 50% labels per client.

Figure 13. Transferring an image from MNIST's test set to all clients. Test image on top transferred image to each client at the bottom.

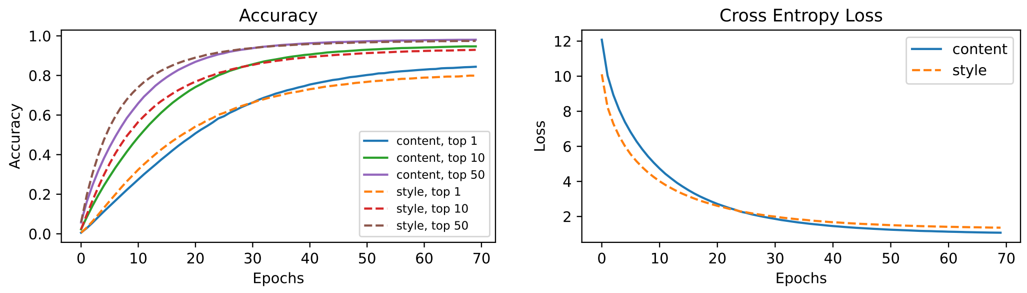
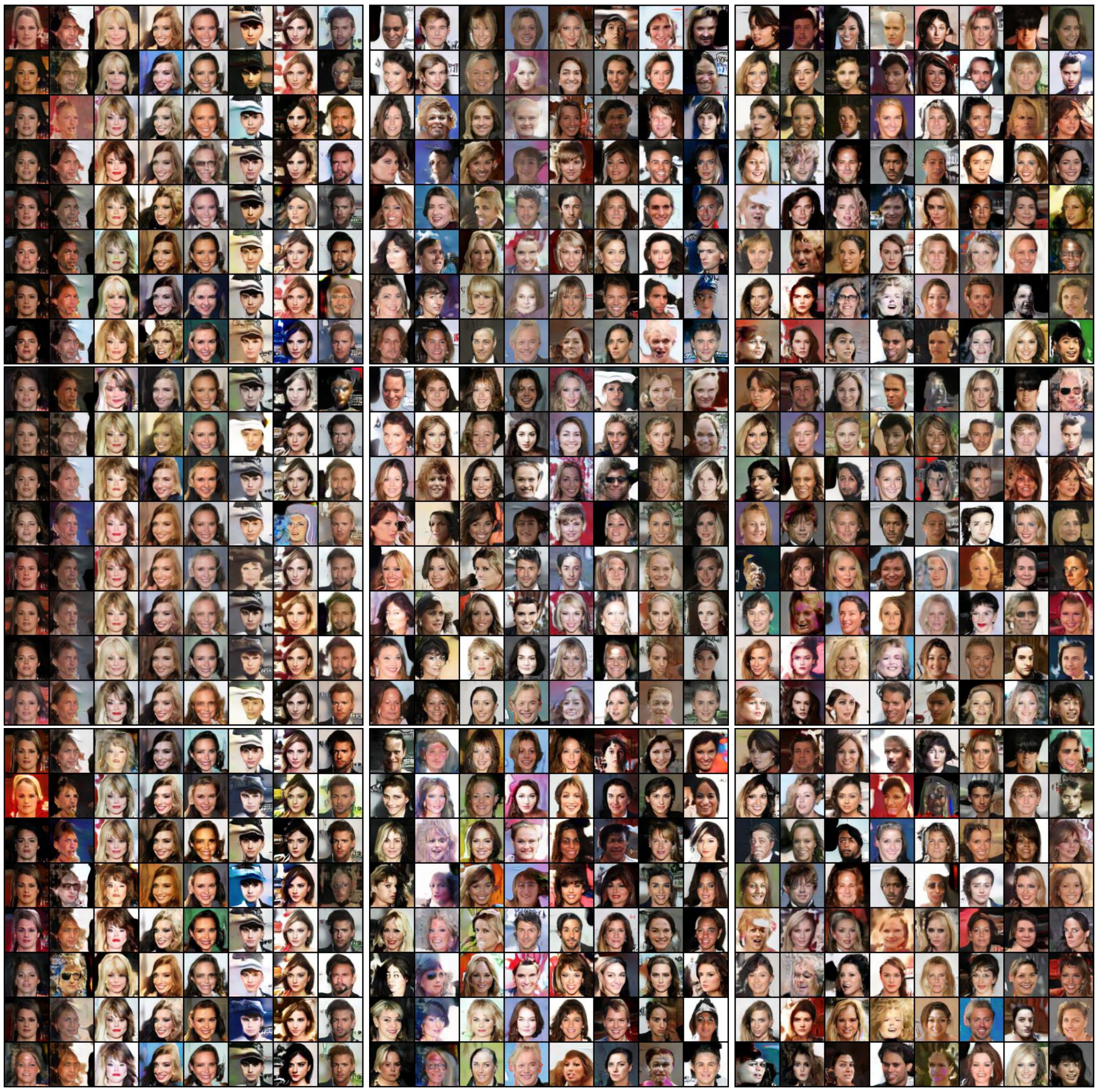


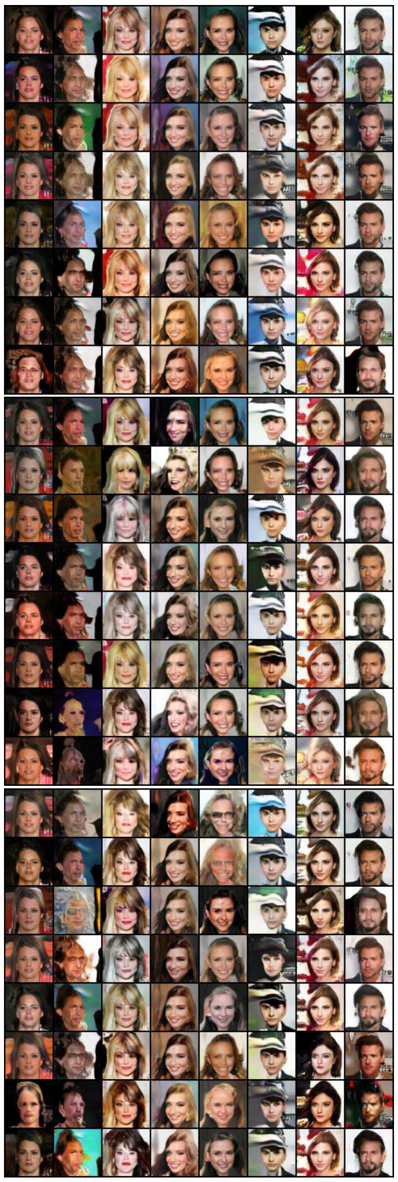
Figure 14. Accuracy and loss after training a linear classifier to predict the celebrity's identity. Note that there are 10,177 identities in total, which makes this task hard, so we also show top-10 and top-50 accuracies.

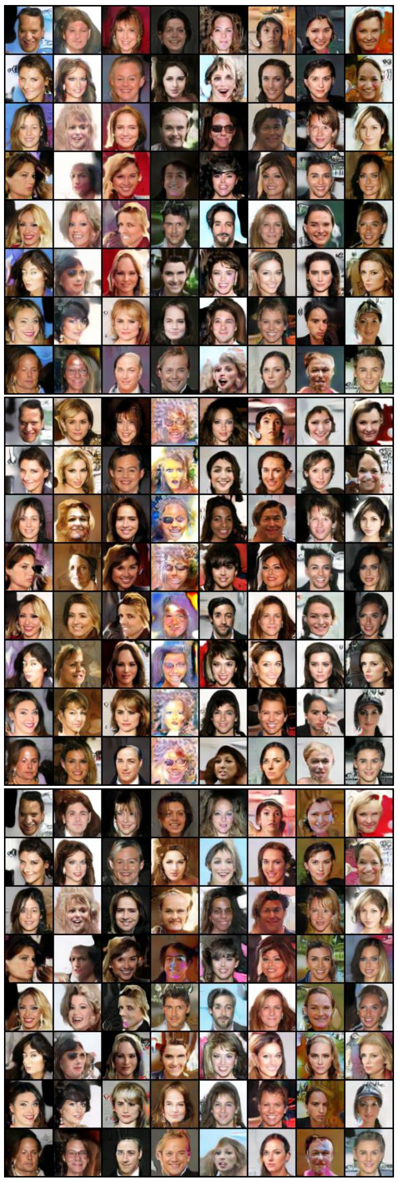
and only one attribute per client, then we might expect an attribute to be a style in the sense that it constitutes a domain, as in our MNIST example. In Fig. 14, we show the accuracy of a linear classifier on the content and style representations. Both achieve decent top-1 accuracies (84% and 79%, resp.), good 90% in top-10 accuracies (95% and 93%, resp.). Note that using top-10 and top-50 is approximately within the same proportion as using top-1 and top-5 for ImageNet, which has 1000 classes.



(a) Changing $\mathbf{z}^s$ per row while fixing $\mathbf{z}^c$. (b) Changing $\mathbf{z}^c$ per row while fixing $\mathbf{z}^s$. (c) Changing both $\mathbf{z}^c$ and $\mathbf{z}^s$.

Figure 15. Workers 1 at the top, 2 in the middle, and 3 at the bottom.


(a) Changing $\mathbf{z}^s$ per row while fixing $\mathbf{z}^c$.


(b) Changing $\mathbf{z}^c$ per row while fixing $\mathbf{z}^s$.

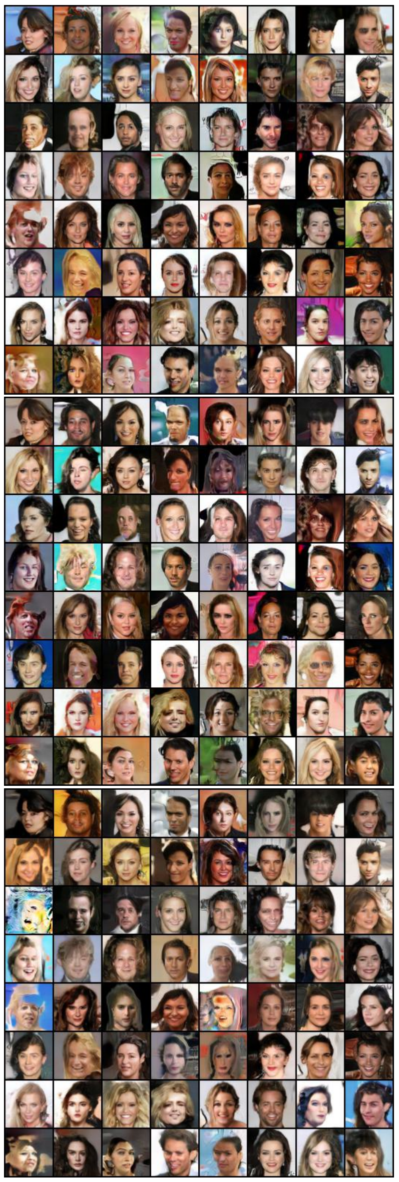
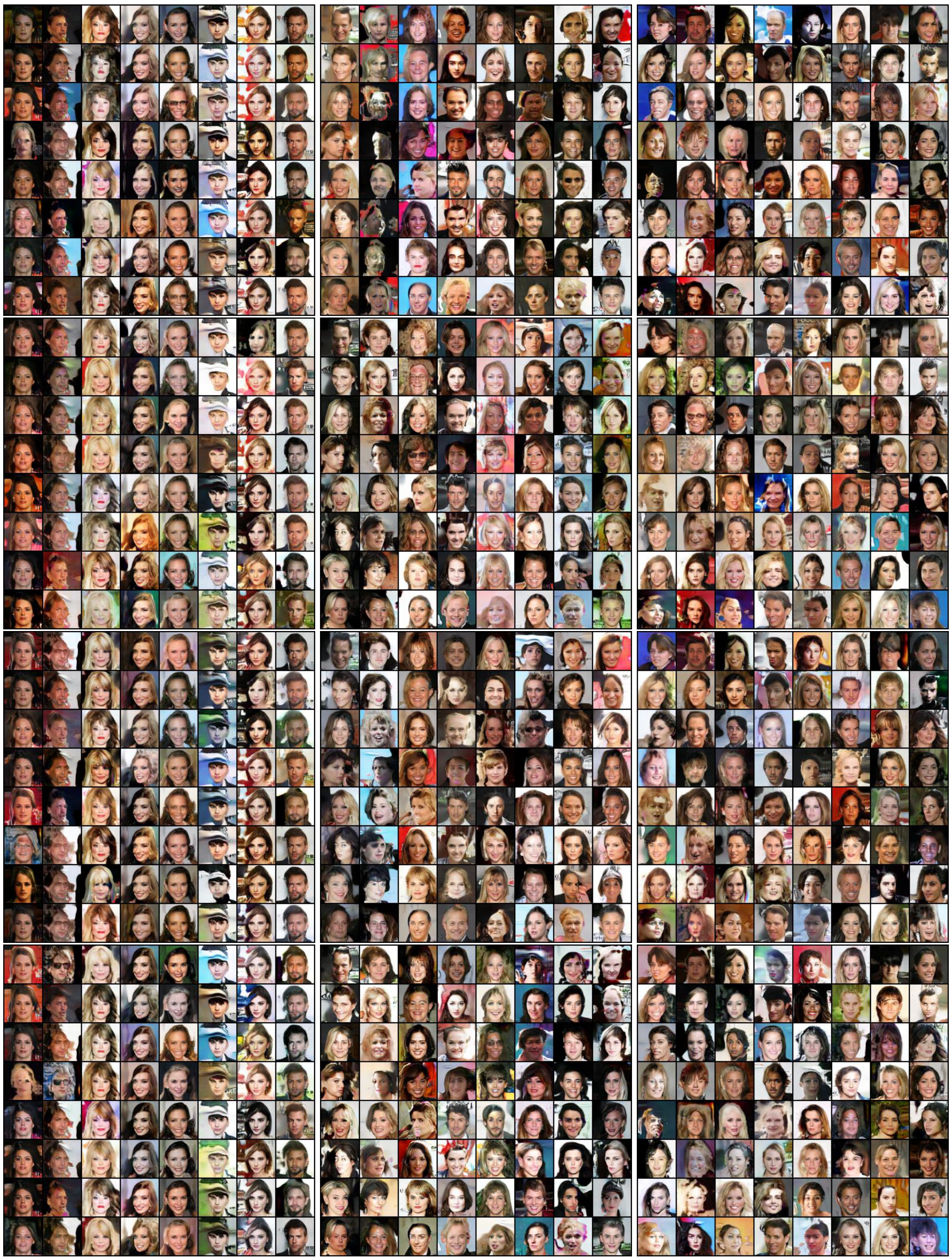

(c) Changing both $\mathbf{z}^c$ and $\mathbf{z}^s$.

Figure 16. Workers 4 at the top, 5 in the middle, and 6 at the bottom.



(a) Changing $\mathbf{z}^s$ per row while fixing $\mathbf{z}^c$. (b) Changing $\mathbf{z}^c$ per row while fixing $\mathbf{z}^s$. (c) Changing both $\mathbf{z}^c$ and $\mathbf{z}^s$.

Figure 17. Workers 7 at the top, 8 in the upper-middle, 9 in the lower-middle, and 10 at the bottom.