FEDERATED TRAINING OF DUAL ENCODING MODELS ON SMALL NON-IID CLIENT DATASETS

Raviteja Vemulapalli, Warren Richard Morningstar, Philip Andrew Mansfield,
Hubert Eichner, Karan Singhal, Arash Afkanpour, Bradley Green
Google Research
{ravitejavemu,wmorning,memes,huberte,
karansinghal,arashaf,brg}@google.com

ABSTRACT

Dual encoding models that encode a pair of inputs are widely used for representation learning. Many approaches train dual encoding models by maximizing agreement between pairs of encodings on centralized training data. However, in many scenarios, datasets are inherently decentralized across many clients, motivating federated learning. In this work, we focus on federated training of dual encoding models on decentralized data composed of many small, non-IID (independent and identically distributed) client datasets. Existing approaches require large and diverse training batches to work well and perform poorly when naively adapted to the setting of small, non-IID client datasets using federated averaging. We observe that large-batch loss computation can be simulated on small individual clients for loss functions that are based on encoding statistics. Based on this insight, we propose a novel federated training approach, *Distributed Cross Correlation Optimization (DCCO)*, which trains dual encoding models using encoding statistics aggregated across clients, without sharing individual samples or encodings. Our experimental results on two datasets demonstrate that the proposed approach outperforms federated variants of existing approaches by a large margin.

1 INTRODUCTION

Dual encoding models are a class of models that generate a pair of encodings for a pair of inputs using one or two encoding networks. These models are widely used for representation learning of both unimodal and multi-modal data (Chen et al., 2020a; Chen & He, 2021; He et al., 2020; Grill et al., 2020; Zbontar et al., 2021; Radford et al., 2021; Jia et al., 2021). While several approaches exist for training dual encoding models in centralized settings, training these models on decentralized datasets is less explored.

Federated learning (McMahan et al., 2017) is a widely-used approach for learning from decentralized datasets without transferring raw data to a central server. In many real-world scenarios, individual client datasets are small and non-IID (independent and identically distributed), e.g., in *cross-device* federated settings (Kairouz et al., 2021). For example, in the context of mobile medical apps such as Aysa (AskAysa, 2022) and DermAssist (DermAssist, 2022), each user contributes only a few (1-3) images. Motivated by this, we focus on federated training of dual encoding models on decentralized data composed of a large number of small, non-IID client datasets.

Recently, several approaches have been proposed for training dual encoding models in centralized settings based on contrastive losses (He et al., 2020; Chen et al., 2020a;b), statistics-based losses (Zbontar et al., 2021; Bardes et al., 2022), and predictive losses (Grill et al., 2020; Chen & He, 2021). One way to enable federated training of dual encoding models is to adapt these existing approaches using the Federated Averaging (FedAvg) strategy of McMahan et al. (2017). As described in Section 2, all of these approaches require large and diverse training batches to work well, and their performance degrades when trained on small, non-IID client datasets.

To enable large-batch federated contrastive training, Zhang et al. (2020) and Wu et al. (2022b) propose to share individual sample encodings between clients, raising privacy concerns. Zhuang et al. (2021; 2022) extend BYOL (Grill et al., 2020) to federated settings by using a separate target encoder on each client. He et al. (2021) extends SimSiam (Chen & He, 2021) to federated settings by using a separate personalized model on each client in addition to a shared model. Makhija et al. (2022) goes a step further and removes the shared model. All of these approaches focus on cross-silo settings with small number of clients and thousands of samples per client. Makhija et al. (2022) showed that the performance of these methods degrades significantly as the number of clients increases. Different from these works, we focus on datasets composed of a large number of small, non-IID client datasets.

In this work we observe that, in the case of statistics-based loss functions, we can simulate large-batch loss computation on each individual (small) client, by first aggregating encoding statistics from many clients and then sharing these aggregated large-batch statistics with all the clients that contributed to them. Based on this observation, we propose a novel approach, *Distributed Cross Correlation Optimization (DCCO)*, for federated training of dual encoding models on small, non-IID client datasets. The proposed approach simulates large-batch training with the loss function of Zbontar et al. (2021), which we refer to as *Cross Correlation Optimization (CCO)* loss. This is achieved without sharing individual data samples or their encodings between clients.

Federated dual encoding models have also been explored for recommendation systems (Ning et al., 2021; Wu et al., 2022a), where a finite lookup table of item encodings is learned. In this work, we focus on continuous signals such as images where the concept of item lookup table is inapplicable.

Major Contributions

- We observe that large-batch training of dual encoding models can be simulated on decentralized datasets by using loss functions based on encoding statistics aggregated across clients, without sharing individual samples or their encodings.
- Building on this insight, we present Distributed Cross Correlation Optimization (DCCO), a novel approach for training dual encoding models on decentralized datasets composed of a large number of small, non-IID client datasets.
- We prove that when we perform one step of local training in each federated training round, one round of DCCO training is equivalent to one step of centralized training on a large batch composed of all samples across all clients participating in the federated round.
- We evaluate the proposed approach on two datasets and show that it outperforms FedAvg variants of contrastive and CCO training by a significant margin. The proposed method also significantly outperforms supervised training from scratch, demonstrating its effectiveness for decentralized self-supervised learning.

2 Problems with Existing Approaches

Contrastive loss functions Contrastive losses maximize the similarity between two encodings of a data sample while pushing encodings of different samples apart. This is highly effective when each sample is contrasted against a large set of diverse samples (Chen et al., 2020a;b; He et al., 2020). When combined with FedAvg, the effectiveness of contrastive training decreases as each sample is contrasted against a small set of within-client samples, which may be relatively similar.

Dependence on batch normalization Approaches such as BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2021) use predictive losses that encourage two encodings of a data sample to be predictive of each other. Though these approaches do not explicitly push the encodings of different samples apart, they work well when trained with large batches in centralized settings. Importantly, they use batch normalization (Ioffe & Szegedy, 2015), whose efficacy decreases rapidly when batches become smaller (Wu & He, 2018). When training on small, non-IID client datasets in federated settings, group normalization (Wu & He, 2018) is typically used instead of batch normalization (Hsieh et al., 2020; Hsu et al., 2020). When we experimented with BYOL and SimSiam by replacing batch normalization with group normalization, the models did not train well (see Appendix C). This dependence on batch normalization suggests that these approaches are not a good fit for federated training on small, non-IID client datasets.

Statistics-based loss functions While the above contrastive and predictive losses directly use individual sample encodings in their computation, the CCO loss introduced by Zbontar et al. (2021) is a function of encoding statistics computed over a batch of samples. This loss function maximizes the correlation coefficient values of matching dimensions and minimizes the correlation coefficient values of non-matching dimensions of the two encodings of a dual encoding model. Since CCO loss is a function of batch statistics, its efficacy decreases with smaller batch sizes, and it performs poorly in federated settings when used for training on small, non-IID client datasets (see Sec. 4).

3 Proposed Approach

3.1 Cross Correlation Optimization Loss

Let X and Y be the two inputs to a dual encoding model, and $F = [F_i] \in \mathcal{R}^d$ and $G = [G_j] \in \mathcal{R}^d$ be their encodings, respectively. The CCO loss used for training dual encoding models in Zbontar et al. (2021) is given by [1]

$$\mathcal{L}_{CCO} = \sum_{i=1}^{d}(1-C_{ii})^2 + \lambda \sum_{i=1}^{d} \frac{1}{d-1} \sum_{\substack{j=1 \\ j \neq i}}^{d} C_{ij}^2, \;\; C_{ij} = \frac{\langle F_i G_j \rangle - \langle F_i \rangle \langle G_j \rangle}{\sqrt{\langle (F_i)^2 \rangle - \langle F_i \rangle^2} \sqrt{\langle (G_j)^2 \rangle - \langle G_j \rangle^2}}, \quad (1)$$

where C_{ij} represents the correlation coefficient between i^{th} component of F and j^{th} component of G. Here, $\langle \, \rangle$ represents average values computed using a batch of samples. The first term in the CCO loss encourages the two encodings of a data sample to be similar by maximizing the correlation coefficient between the matching dimensions of the encodings, and the second term reduces the redundancy between output units by decorrelating the different dimensions of the encodings.

3.2 Motivation: Aggregating and Redistributing Encoding Statistics

Both contrastive and CCO losses are highly effective when used with large training batches, and their efficacy decreases as batch size decreases. Contrastive losses directly use individual sample encodings, and it is unclear how large-batch contrastive loss can be computed on small clients without sharing individual sample encodings between clients. Unlike contrastive losses, CCO loss only uses encoding statistics $\langle F_i \rangle, \langle (F_i)^2 \rangle, \langle G_j \rangle, \langle (G_j)^2 \rangle, \langle F_i G_j \rangle$. Although CCO loss is a nonlinear function of these statistics, the statistics themselves are average values, i.e. linear combinations of individual sample values. Hence, we can compute large-batch statistics as a weighted average of small-batch statistics computed on individual clients, without sharing individual sample encodings. This makes it possible to simulate large-batch CCO loss on individual clients by first aggregating encoding statistics from many clients and sharing these aggregated large-batch statistics with all the clients that contributed to them. Based on this insight, we propose our DCCO approach that simulates large-batch training using small, non-IID client datasets.

The strategy of aggregating and redistributing statistics can also be used to implement multi-client synchronous batch normalization in federated settings. However, since batch normalization is typically used at every layer of the encoder network, this will add multiple additional rounds of server-client communication (as many as the number of batch normalization layers in the network) which is undesirable. In contrast, the proposed DCCO approach uses only one additional round of communication by aggregating statistics at the end of the network before computing the loss function.

3.3 Distributed Cross Correlation Optimization (DCCO)

Figure 1 presents the proposed DCCO training approach. In each training round, a central server samples K clients from the pool of available clients and broadcasts the current model to these clients. Each client k uses this model to encode its local data and compute local encoding statistics $\langle F_i \rangle_k, \langle (F_i)^2 \rangle_k, \langle G_j \rangle_k, \langle (G_j)^2 \rangle_k, \langle F_i G_j \rangle_k$ based on N_k samples. These local encoding statistics are then aggregated by the central server using weighted averaging, and the aggregated statistics are shared with the participating clients. During this aggregation, each local encoding statistic is given

[1] Zbontar et al. (2021) do not refer to this loss as CCO loss; we use this more general term instead of *BarlowTwins* loss to emphasize the loss function's applicability to cases where the two encoder networks differ.

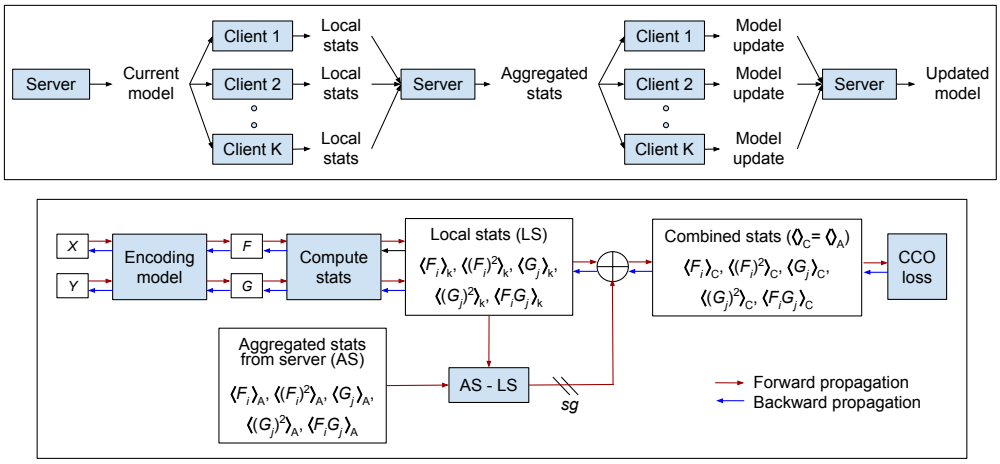


Figure 1: Top: one round of DCCO training. Bottom: local training on client k. Here sg refers to the stop gradient operation, and $\bigoplus$ denotes addition. The combined statistics $\langle\,\rangle_C$ used for loss computation are equal to the aggregated statistics $\langle\,\rangle_A$ from the server. Note that gradient backpropagation happens only through the local statistics $\langle\,\rangle_k$.

a weight that is proportional to the number of samples that contributed to its computation. Then, each client computes a local model update by minimizing the CCO loss function computed using aggregated statistics $\langle\,\rangle_A$. During local training, while aggregated statistics are used for computing the loss function, gradients are backpropagated only through local statistics $\langle\,\rangle_k$ since each client has access only to its local data. Finally, the local model updates are aggregated by the central server and used to update the current model. To aggregate client model updates, we use a weighted average, weighted by examples per client. During the entire training process, individual data samples never leave the clients either in raw or encoded formats.

Our statistics aggregation procedure is same as the FedAvg weight aggregation procedure and thus may contain similar vulnerabilities (from a privacy standpoint) which can be addressed with approaches such as Secure Aggregation (Bonawitz et al., 2017). We plan to explore this in the future.

Our work focuses on training with small client datasets (for example, no more than six images per client in the case of DERM dataset). Hence, in each federated training round, we only perform one step of local training on each participating client. In this setting, one round of DCCO training is equivalent to one centralized training step on a batch composed of all data samples across all clients participating in the federated round. See Appendix A for the proof. This equivalence between federated and centralized training holds only because of the additional step of aggregating and sharing encoding statistics. Naively applying FedAvg with one within-client CCO loss-based training step per round is not equivalent to centralized training, as we show in Section 4.

While this work focuses on the CCO loss of Zbontar et al. (2021), the proposed distributed learning strategy can also be used with other statistics-based loss functions such as Bardes et al. (2022).

4 Experiments

In this section, we evaluate the proposed DCCO approach in the context of self-supervised learning where the two inputs of a dual encoding model are generated by applying random augmentations to a single unlabeled input, and both inputs are processed using the same network.

Experimental setup First, the encoder network is pretrained on an unlabeled, decentralized dataset. Then, a linear classifier is added on top of the encoder network, and either only the newly added classifier (linear evaluation protocol) or the entire network (full finetuning protocol) is trained using a small labeled classification dataset on a central sever. The accuracy of final model is used as the evaluation metric.

Comparisons We compare the proposed approach with FedAvg variants of within-client CCO and contrastive loss-based training. We use CCO loss-based centralized training as upper bound for the proposed approach. We report results for fully-supervised training from scratch (using the limited labeled data) to demonstrate the effectiveness of DCCO as a self-supervised pretraining strategy.

CIFAR-100 dataset (Krizhevsky & Hinton, 2009) We use all 50K training images in the dataset as unlabeled data for pretraining, and a small fraction of them (10% or 1%) as labeled data for supervised finetuning. To study the effect of non-identical client data distribution, we generated IID and non-IID client datasets using the Dirichlet distribution-based sampling process of Hsu et al. (2019) with $\alpha = 1000$ and $\alpha = 0$, respectively. We generated multiple decentralized datasets by varying the total number of clients and the number of samples per client (see Table 1).

DERM dataset (Liu et al., 2020) This dataset consists of de-identified images of skin conditions captured using consumer-grade digital cameras. Each case in the dataset includes one to six images of a skin condition from a single patient. A portion of this dataset is labelled with 419 skin conditions. Following Azizi et al. (2022), we focus on the most common 26 conditions and group the rest into 'Other' class leading to a label space of 27 categories. Following Azizi et al. (2022), we use 207,032 unlabeled images (59,770 cases) for pretraining the encoder network. For federated pretraining, we consider each case as a separate client. For supervised finetuning, we use up to 4,592 labeled cases. The final accuracy is reported on a test set of 4,146 labeled cases. This dataset also provides a validation split with 1,190 labeled cases which we use for tuning some hyperparameters.

Networks Residual networks (He et al., 2016) with 14 and 50 layers are used as encoder networks for experimenting with CIFAR-100 and DERM datasets, respectively. Following Zbontar et al. (2021), a three layer fully-connected projection network is used to increase the dimensionality of the encodings before computing the CCO loss during pretraining, and the projection network is discarded while training the final classifier. The projection network configurations are [1024, 1024, 1024] and [2048, 2048, 4096] for CIFAR-100 and DERM datasets, respectively. For the contrastive loss, following Chen et al. (2020a;b), a fully-connected projection network is used to reduce the dimensionality of the encodings before computing the loss. The projection network configurations are [256, 256, 128] and [2048, 2048, 128] for CIFAR-100 and DERM datasets, respectively. Weight standardization (Qiao et al., 2019) and group normalization (Wu & He, 2018) with 32 groups are used at every layer except the last projection layer. For the DERM dataset, since each case consists of multiple images (up to six), the final classification network performs average pooling of individual image encodings and uses a linear classifier on top of the average-pooled feature to predict the label for a case. Following Azizi et al. (2021), we use 224×224 images as input while pretraining the encoder, and 448×448 images as input while training the classifier.

4.1 Training details

CIFAR-100 We experimented with several decentralized versions by varying the total number of clients and the number of samples per client (see Table 1). During federated pretraining, all models were trained for 100K rounds. For FedAvg variants of CCO and contrastive loss-based training, we observed overfitting. For these approaches, we evaluated multiple pretrained checkpoints under linear evaluation protocol and report the results for the best checkpoints. We did not observe such overfitting for the proposed DCCO approach and report results based on 100K rounds of training. Each client is visited around 1000 times during federated pretraining of 100K rounds. So, for centralized CCO loss-based pretraining, we used 1000 epochs of training with a batch size of 512.

DERM For federated experiments, each case in the dataset is considered as a client dataset. We conducted experiments by varying the number of clients sampled in each round (see Table 2). All models were trained for 75K rounds during federated pretraining. For this dataset (59,770 clients), each client is visited around 320 times if we perform 75K rounds of federated training with 256 clients per round. So, for centralized CCO training, we used 320 epochs with a batch size of 512.

We used a value of 20 for the tradeoff parameter in CCO loss (Eq. 1) and a value of 0.1 for the temperature parameter in contrastive loss (Chen et al., 2020a). In all federated experiments, we used gradient descent with learning rate 1.0 as the local optimizer on clients. Please refer to Appendix B for further details about the training hyperparameters and data augmentations used in all settings.

	NON-IID CLIENT DATASETS				IID CLIENT DATASETS		
TOTAL CLIENTS	50,000	12,500	6,250	3,125	12,500	6,250	3,125
SAMPLES / CLIENT	1	4	8	16	4	8	16
CLIENTS / ROUND	512	128	64	32	128	64	32
LINEAR EVALUATION PROTOCOL (5K LABELED TRAINING SAMPLES)							
CCO + FEDAVG	–	FAILED	28.7	30.7	FAILED	41.2	44.1
CONTRASTIVE + FEDAVG	–	31.1	32.4	31.8	41.1	44.1	46.4
PROPOSED DCCO	51.7	49.6	48.1	45.9	51.4	52.0	51.9
CENTRALIZED CCO	52.6						
SUPERVISED FROM SCRATCH	42.4						
FULL FINETUNING PROTOCOL (5K LABELED TRAINING SAMPLES)							
CCO + FEDAVG	–	FAILED	36.3	38.9	FAILED	43.0	46.0
CONTRASTIVE + FEDAVG	–	40.8	41.8	41.3	44.9	46.8	47.4
PROPOSED DCCO	51.5	50.5	49.3	47.7	51.5	52.1	52.0
CENTRALIZED CCO	52.5						
SUPERVISED FROM SCRATCH	42.4						
LINEAR EVALUATION PROTOCOL (500 LABELED TRAINING SAMPLES)							
CCO + FEDAVG	–	FAILED	13.5	14.5	FAILED	26.2	27.6
CONTRASTIVE + FEDAVG	–	14.8	15.6	15.0	24.7	27.8	29.3
PROPOSED DCCO	33.3	31.2	29.5	26.7	33.3	33.5	33.5
CENTRALIZED CCO	34.0						
SUPERVISED FROM SCRATCH	13.1						

Table 1: Results on CIFAR-100 dataset under linear evaluation and full finetuning protocols.

4.2 RESULTS

CIFAR100 Table 1 shows the performance of various approaches under linear evaluation and full finetuning protocols for multiple decentralized versions of CIFAR-100 dataset. When only 500 images are labeled, training only the linear classifier performed better than finetuning the entire network. So, in this case, we only report results under linear evaluation protocol. The proposed approach outperforms the FedAvg variants of contrastive and CCO training by a significant margin for both IID and non-IID client datasets. Specifically, in the case of non-IID client datasets, the performance gains are in the range of 11-18% under linear evaluation protocol and 6-10% under full finetuning protocol, and in the case of IID client datasets, the gains are in the range of 4-10% under linear evaluation protocol and 4-6% under full finetuning protocol. The proposed approach also outperforms fully-supervised training from scratch by 10-20%, demonstrating the effectiveness of DCCO as a federated self-supervised pretraining strategy.

In the case of non-IID client datasets, for a fixed global batch size (i.e., the total number of samples participating in a round), the performance of DCCO approach increases as the number of samples per client decreases and the number of clients per round increases, approaching the performance of centralized CCO training when each client has only one sample [2]. This is because the proposed DCCO approach optimizes a loss function based on aggregated global batch statistics and using

[2]Ideally, when each client has only one sample, the performance of DCCO with 512 clients per round should be close to the performance of centralized CCO with a batch size of 512. There is a small ($\sim$1%) performance gap between the two due to the differences in the random data augmentation pipelines implemented in Tensorflow (used for centralized training) and Tensorflow Federated (used for federated training).

LABELED DATA (FINETUNING)	1,524 CASES			3,057 CASES			4,592 CASES		
NUM. CLIENTS (PRETRAINING)	59,770			59,770			59,770		
CLIENTS / ROUND	64	128	256	64	128	256	64	128	256
CONTRASTIVE + FEDAVG	38.9	41.8	43.9	45.3	46.7	47.8	47.7	49.1	51.5
PROPOSED DCCO	42.4	46.3	47.8	46.7	50.3	51.1	48.3	52.8	52.5
CENTRALIZED CCO	48.0			52.0			53.6		
SUPERVISED FROM SCRATCH	30.6			35.9			39.2		

Table 2: Results on DERM dataset under full finetuning protocol.

more clients increases the diversity of samples in the global batch. However, we do not see such trend in the case of IID client datasets because each sample in every client dataset is already sampled in an IID fashion from the full CIFAR-100 dataset.

We do not report results for FedAvg variants of contrastive and CCO losses when each client has only one sample because we need at least two samples to compute these loss functions. The proposed DCCO approach can still be used in this case since it uses statistics aggregated from multiple clients to compute the loss. We do not report results for FedAvg variant of CCO loss when each client has only four samples because training was unstable in this case.

DERM Table 2 shows the performance of various approaches under full finetuning protocol for different amounts of labeled data. The proposed approach clearly outperforms the FedAvg variant of contrastive loss-based training and achieves a performance close to that of centralized CCO training. The proposed approach also results in significant performance gains (13-17%) when compared to fully-supervised training from scratch demonstrating its effectiveness in leveraging unlabeled decentralized datasets. We do not report results for FedAvg variant of CCO loss because training was unstable in this case as each client in this dataset has a maximum of six images.

5 CONCLUSION AND FUTURE WORK

In this work, we proposed an approach for training dual encoding models on decentralized datasets composed of a large number of small, non-IID client datasets. The proposed approach optimizes a loss function based on encoding statistics and simulates large-batch loss computation on individual clients by using encoding statistics aggregated across multiple clients. When each client participating in a training round performs only one local training step, each federated round of DCCO training is equivalent to a centralized training step on a batch consisting of all data samples participating in the corresponding round. Our experimental results on two image datasets show that the proposed approach outperforms FedAvg variants of within-client contrastive and CCO loss-based training. Our proposed approach also outperforms supervised training from scratch by a significant margin; demonstrating its effectiveness as a federated self-supervised learning approach.

This paper focused on unimodal datasets and the CCO loss function of Zbontar et al. (2021). In the future, we plan to experiment with multi-modal datasets, and other statistics-based loss functions such as Bardes et al. (2022). There are also several other interesting research directions we plan to pursue in the future. How can we extend the proposed approach to large client datasets? When training on large client datasets, we may need to perform multiple steps of local training on each client within one federated round. When we perform multiple training steps on a client, in each step, only a small subset of samples that are contributing to the loss function participate in gradient computation. Also, while the model weights change after each local step, the aggregated statistics used in the loss function remain constant. How can we address the effects of these *partial gradients* and *stale statistics*? The proposed DCCO approach uses two communication rounds between server and clients within one federated training round. How can we reduce this to one round?

REFERENCES

AskAysa. Skin condition questions? AI-enabled answers, 2022. URL https://askaysa.com/.

Shekoofeh Azizi, Basil Mustafa, Fiona Ryan, Zachary Beaver, Jan Freyberg, Jonathan Deaton, Aaron Loh, Alan Karthikesalingam, Simon Kornblith, Ting Chen, Vivek Natarajan, and Mohammad Norouzi. Big self-supervised models advance medical image classification. In *ICCV*, 2021.

Shekoofeh Azizi, Laura Culp, Jan Freyberg, Basil Mustafa, Sebastien Baur, Simon Kornblith, Ting Chen, Patricia MacWilliams, S. Sara Mahdavi, Ellery Wulczyn, Boris Babenko, Megan Wilson, Aaron Loh, Po-Hsuan Cameron Chen, Yuan Liu, Pinal Bavishi, Scott Mayer McKinney, Jim Winkens, Abhijit Guha Roy, Zachary Beaver, Fiona Ryan, Justin Krogue, Mozziyar Etemadi, Umesh Telang, Yun Liu, Lily Peng, Gregory S. Corrado, Dale R. Webster, David J. Fleet, Geoffrey E. Hinton, Neil Houlsby, Alan Karthikesalingam, Mohammad Norouzi, and Vivek Natarajan. Robust and efficient medical imaging with self-supervision. *CoRR*, abs/2205.09723, 2022.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

DermAssist. Identify skin conditions with DermAssist, 2022. URL https://health.google/consumers/dermassist/.

Abe Fetterman and Josh Albrecht. Understanding self-supervised and contrastive learning with bootstrap your own latent (byol), 2020. URL https://untitled-ai.github.io/understanding-self-supervisedcontrastive-learning.html.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*, 2020.

Chaoyang He, Zhengyu Yang, Erum Mushtaq, Sunwoo Lee, Mahdi Soltanolkotabi, and Salman Avestimehr. SSFL: tackling label deficiency in federated learning via personalized self-supervision. *CoRR*, abs/2110.02470, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The non-iid data quagmire of decentralized machine learning. In *ICML*, 2020.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *CoRR*, abs/1909.06335, 2019.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *ECCV*, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, 2021.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Yuan Liu, Ayush Jain, Clara Eng, David H. Way, Kang Lee, Peggy Bui, Kimberly Kanada, Guilherme de Oliveira Marinho, Jessica Gallegos, Sara Gabriele, Vishakha Gupta, Nalini Singh, Vivek Natarajan, Rainer Hofmann-Wellenhof, Gregory S. Corrado, Lily H. Peng, Dale R. Webster, Dennis Ai, Susan Huang, Yun Liu, R. Carter Dunn, and David Coz. A deep learning system for differential diagnosis of skin diseases. *Nature Medicine*, 26:900–908, 2020.

Disha Makhija, Nhat Ho, and Joydeep Ghosh. Federated self-supervised learning for heterogeneous clients. *CoRR*, abs/2205.12493, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Lin Ning, Karan Singhal, Ellie X. Zhou, and Sushant Prakash. Learning federated representations and recommendations with limited negatives. *CoRR*, abs/2108.07931, 2021.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan L. Yuille. Weight standardization. *CoRR*, abs/1903.10520, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Pierre H. Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel L. Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. BYOL works even without batch statistics. *CoRR*, abs/2010.10241, 2020.

Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fedcl: Federated contrastive learning for privacy-preserving recommendation. *CoRR*, abs/2204.09850, 2022a.

Yawen Wu, Zhepeng Wang, Dewen Zeng, Meng Li, Yiyu Shi, and Jingtong Hu. Decentralized unsupervised learning of visual representations. In *IJCAI*, 2022b.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *CoRR*, abs/1708.03888, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *CoRR*, abs/2010.08982, 2020.

Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. Collaborative unsupervised visual representation learning from decentralized data. In *ICCV*, 2021.

Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Divergence-aware federated self-supervised learning. In *ICLR*, 2022.

A Proof of Equivalence to Centralized Training

Claim: When we perform one step of local training on each participating client in a federated training round, one round of DCCO training is equivalent to one centralized training step on a batch composed of all data samples participating in the round.

Proof: Let F_i^n and G_j^n respectively denote the ith and j^{th} components of encodings F and G of nth sample on a client. Based on the definition of $\langle F_i \rangle_C$, we get

$$\langle F_i \rangle_C = \langle F_i \rangle_k + StopGradient\left[\langle F_i \rangle_A - \langle F_i \rangle_k\right] \implies \frac{\partial \langle F_i \rangle_C}{\partial F_i^n} = \frac{\partial \langle F_i \rangle_k}{\partial F_i^n}. \tag{2}$$

Similarly, by definitions of $\left\langle (F_i)^2 \right\rangle_C, \langle G_j \rangle_C, \left\langle (G_j)^2 \right\rangle_C, \langle F_i G_j \rangle_C$, we get

$$\begin{aligned} \frac{\partial \left\langle (F_i)^2 \right\rangle_C}{\partial F_i^n} = \frac{\partial \left\langle (F_i)^2 \right\rangle_k}{\partial F_i^n}, \frac{\partial \langle G_j \rangle_C}{\partial G_j^n} = \frac{\partial \langle G_j \rangle_k}{\partial G_j^n}, \frac{\partial \left\langle (G_j)^2 \right\rangle_C}{\partial G_j^n} = \frac{\partial \left\langle (G_j)^2 \right\rangle_k}{\partial G_j^n}, \\ \frac{\partial \langle F_i G_j \rangle_C}{\partial F_i^n} = \frac{\partial \langle F_i G_j \rangle_k}{\partial F_i^n}, \frac{\partial \langle F_i G_j \rangle_C}{\partial G_j^n} = \frac{\partial \langle F_i G_j \rangle_k}{\partial G_j^n} \end{aligned} \tag{3}$$

By chain rule, we have

$$\begin{aligned} \frac{\partial \mathcal{L}_{CCO}}{\partial F_i^n} &= \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i \rangle_C} \frac{\partial \langle F_i \rangle_C}{\partial F_i^n} + \frac{\partial \mathcal{L}_{CCO}}{\partial \langle (F_i)^2 \rangle_C} \frac{\partial \left\langle (F_i)^2 \right\rangle_C}{\partial F_i^n} + \sum_{j=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i G_j \rangle_C} \frac{\partial \langle F_i G_j \rangle_C}{\partial F_i^n} \\ \frac{\partial \mathcal{L}_{CCO}}{\partial G_j^n} &= \frac{\partial \mathcal{L}_{CCO}}{\partial \langle G_j \rangle_C} \frac{\partial \langle G_j \rangle_C}{\partial G_j^n} + \frac{\partial \mathcal{L}_{CCO}}{\partial \langle (G_j)^2 \rangle_C} \frac{\partial \left\langle (G_j)^2 \right\rangle_C}{\partial G_j^n} + \sum_{i=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i G_j \rangle_C} \frac{\partial \langle F_i G_j \rangle_C}{\partial G_j^n} \end{aligned} \tag{4}$$

Substituting Eq. 2 and 3 in Eq. 4, we get

$$\begin{aligned} \frac{\partial \mathcal{L}_{CCO}}{\partial F_i^n} &= \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i \rangle_C} \frac{\partial \langle F_i \rangle_k}{\partial F_i^n} + \frac{\partial \mathcal{L}_{CCO}}{\partial \langle (F_i)^2 \rangle_C} \frac{\partial \left\langle (F_i)^2 \right\rangle_k}{\partial F_i^n} + \sum_{j=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i G_j \rangle_C} \frac{\partial \langle F_i G_j \rangle_k}{\partial F_i^n} \\ &= \frac{1}{N_k} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i \rangle_C} + \frac{2}{N_k} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle (F_i)^2 \rangle_C} F_i^n + \frac{1}{N_k} \sum_{j=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i G_j \rangle_C} G_j^n \\ \frac{\partial \mathcal{L}_{CCO}}{\partial G_j^n} &= \frac{\partial \mathcal{L}_{CCO}}{\partial \langle G_j \rangle_C} \frac{\partial \langle G_j \rangle_k}{\partial G_j^n} + \frac{\partial \mathcal{L}_{CCO}}{\partial \langle (G_j)^2 \rangle_C} \frac{\partial \left\langle (G_j)^2 \right\rangle_k}{\partial G_j^n} + \sum_{i=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i G_j \rangle_C} \frac{\partial \langle F_i G_j \rangle_k}{\partial G_j^n} \\ &= \frac{1}{N_k} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle G_j \rangle_C} + \frac{2}{N_k} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle (G_j)^2 \rangle_C} G_j^n + \frac{1}{N_k} \sum_{i=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i G_j \rangle_C} F_i^n \end{aligned} \tag{5}$$

By chain rule, the gradients for encoding model parameters θ on k^{th} client are given by

$$\left. \frac{\partial \mathcal{L}_{CCO}}{\partial \theta} \right|_k = \sum_{n=1}^{N_k} \sum_{i=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial F_i^n} \frac{\partial F_i^n}{\partial \theta} + \sum_{n=1}^{N_k} \sum_{j=1}^{d} \frac{\partial \mathcal{L}_{CCO}}{\partial G_j^n} \frac{\partial G_j^n}{\partial \theta}, \tag{6}$$

where N_k is the number of samples on the client that contributed to $\mathcal{L}_{CCO}$.

Substituting Eq. 5 in Eq. 6, we get

$$\left.\frac{\partial \mathcal{L}_{CCO}}{\partial \theta}\right|_k = \frac{1}{N_k}\sum_{n=1}^{N_k}\sum_{i=1}^{d}\left(\frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i\rangle_C} + 2\frac{\partial \mathcal{L}_{CCO}}{\partial \langle (F_i)^2\rangle_C}F_i^n + \sum_{j=1}^{d}\frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_iG_j\rangle_C}G_j^n\right)\frac{\partial F_i^n}{\partial \theta} + \frac{1}{N_k}\sum_{n=1}^{N_k}\sum_{j=1}^{d}\left(\frac{\partial \mathcal{L}_{CCO}}{\partial \langle G_j\rangle_C} + 2\frac{\partial \mathcal{L}_{CCO}}{\partial \langle (G_j)^2\rangle_C}G_j^n + \sum_{i=1}^{d}\frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_iG_j\rangle_C}F_i^n\right)\frac{\partial G_j^n}{\partial \theta} \tag{7}$$

The values of stats $\{\langle F_i\rangle_C\}_{i=1}^d$, $\{\langle (F_i)^2\rangle_C\}_{i=1}^d$, $\{\langle G_j\rangle_C\}_{j=1}^d$, $\{\langle (G_j)^2\rangle_C\}_{j=1}^d$, $\{\langle F_iG_j\rangle_C\}_{i,j=1}^d$ and loss $\mathcal{L}_{CCO}$ are same on all the clients participating in a DCCO training round. So, when each client performs one step of local training and the server averages the model updates from the clients by weighing them according to the number of contributing samples on each client, the equivalent global model gradient update is given by

$$\begin{aligned}\frac{\partial \mathcal{L}_{CCO}}{\partial \theta} &= \frac{1}{N}\sum_{k=1}^{K}N_k\left.\frac{\partial \mathcal{L}_{CCO}}{\partial \theta}\right|_k \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{d}\left(\frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_i\rangle_C} + 2\frac{\partial \mathcal{L}_{CCO}}{\partial \langle (F_i)^2\rangle_C}F_i^n + \sum_{j=1}^{d}\frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_iG_j\rangle_C}G_j^n\right)\frac{\partial F_i^n}{\partial \theta} + \\ &\quad \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{d}\left(\frac{\partial \mathcal{L}_{CCO}}{\partial \langle G_j\rangle_C} + 2\frac{\partial \mathcal{L}_{CCO}}{\partial \langle (G_j)^2\rangle_C}G_j^n + \sum_{i=1}^{d}\frac{\partial \mathcal{L}_{CCO}}{\partial \langle F_iG_j\rangle_C}F_i^n\right)\frac{\partial G_j^n}{\partial \theta}\end{aligned} \tag{8}$$

where N is the total number of samples participating in the round.

If all the N samples participating in a DCCO round are present on a single client, then according to Eq. 7, the gradients computed on that client using these N samples will be same as the gradients in Eq. 8. Hence one round of federated DCCO training is equivalent to one step of centralized training on a batch composed of all N samples participating in the federated round.

B TRAINING DETAILS

We used a value of 20 for the tradeoff parameter λ in CCO loss (Eq. 1) and a value of 0.1 for the temperature parameter in contrastive loss (Chen et al., 2020a). In all federated experiments, we used gradient descent with learning rate 1.0 as the local optimizer on clients.

B.1 CIFAR-100

Federated pretraining: For updating the model on server, we used Adam optimizer (Kingma & Ba, 2015) with cosine learning rate decay. We experimented with initial learning rates of $5e^{-3}$ and $1e^{-3}$ and report the best results.

Centralized pretraining: We trained for 1000 epochs with a batch size of 512. We used Adam optimizer with initial learning rate of $5e^{-3}$, and cosine learning rate decay.

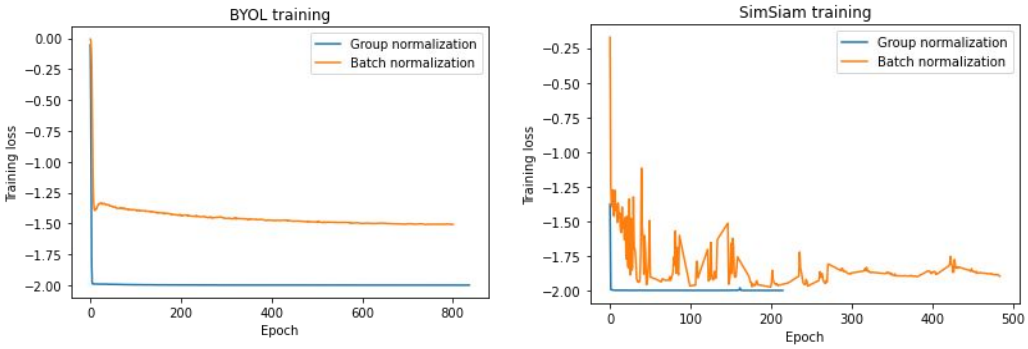
Full network finetuning with 5K samples: We trained for 100 steps using Adam optimizer with batch size 256, initial learning rate $5e^{-3}$, and cosine learning rate decay.

Supervised training from scratch: We trained for 100 epochs using Adam optimizer with batch size 256, initial learning rate 0.01, and cosine learning rate decay.

Linear classifier training with 5K samples: We trained for 1000 steps using LARS optimizer (You et al., 2017) with batch size 512, initial learning rate of 2.0, cosine learning rate decay, and a momentum of 0.9.

Linear classifier training with 500 samples: We trained for 400 steps using LARS optimizer with batch size 128, initial learning rate of 2.5, cosine learning rate decay, and a momentum of 0.9.

Augmentations: During self-supervised pretraining, we used all the data augmentations from Grill et al. (2020) except blur, and during supervised finetuning we only used flip augmentation.



(a) BYOL training

(b) SimSiam training

Figure 2: ResNet14 training on CIFAR-100 dataset.

B.2 DERM

Federated pretraining: For updating the model on server, we used LARS optimizer with cosine learning rate decay and a momentum of 0.9. For DCCO pretraining, the initial learning rate was set to 0.15, 0.9 and 1.8 while using 64, 128 and 256 clients per round, respectively. For FedAvg training with contrastive loss, the initial learning rate was set to 0.15, 0.3 and 0.6 while using 64, 128 and 256 clients per round, respectively. All models were trained for 75K rounds.

Centralized pretraining: We trained for 320 epochs with a batch size of 512. We used LARS optimizer with initial learning rate of 0.6, cosine learning rate decay, and a momentum of 0.9.

Supervised training from scratch and full network finetuning: We used Adam optimizer with a batch size of 128. For each experiment, we used the validation split of this dataset to search for the best learning rate (among $3e^{-4}$ and $1e^{-4}$) and number of training steps (up to 100k).

Augmentations: During self-supervised pretraining, we used random rotation and all the data augmentations from Grill et al. (2020) except solarization. During supervised finetuning, we used all the data augmentations from Azizi et al. (2021).

C BYOL AND SIMSIAM WITH GROUP NORMALIZATION

When we experimented with BYOL and SimSiam by replacing Batch Normalization (BN) with Group Normalization (GN), the models did not train well[3]. Figure 2a shows the training loss when ResNet14 encoder is trained with BYOL approach on CIFAR-100 dataset. When we use GN instead of BN, the loss quickly drops close to its lowest possible value and the model does not learn after that. When we evaluated the BN and GN-based BYOL encoders after training them for 800 epochs with a batch size of 512, we achieved 41% and 3% accuracy, respectively, under linear evaluation protocol. This clearly shows that BN is crucial for BYOL to work well. Similar behavior was observed in the case of SimSiam (see Fig. 2b).

[3] Fetterman & Albrecht (2020) also observes that batch normalization is crucial for BYOL to work. Though Richemond et al. (2020) refutes this and claims that BYOL works well with group normalization, we were not successful in our attempts.