

A deep cut into Split Federated Self-supervised Learning

Marcin Przewięźlikowski^{1,2,3}[0000–0003–4772–3268], Marcin Osial^{1,2,3}[0000–0002–7414–3608], Bartosz Zieliński^{1,3}[0000–0002–3063–3621], and Marek Śmieja¹[0000–0003–2027–4132]

¹ Jagiellonian University, Faculty of Mathematics and Computer Science

² Jagiellonian University, Doctoral School of Exact and Natural Sciences

³ IDEAS NCBR

Abstract. Collaborative self-supervised learning has recently become feasible in highly distributed environments by dividing the network layers between client devices and a central server. However, state-of-the-art methods, such as MocoSFL, are optimized for network division at the initial layers, which decreases the protection of the client data and increases communication overhead. In this paper, we demonstrate that splitting depth is crucial for maintaining privacy and communication efficiency in distributed training. We also show that MocoSFL suffers from a catastrophic quality deterioration for the minimal communication overhead. As a remedy, we introduce **M**omentum-**A**ligned **c**ontrastive **S**plit **F**ederated **L**earning (MonAcoSFL), which aligns online and momentum client models during training procedure. Consequently, we achieve state-of-the-art accuracy while significantly reducing the communication overhead, making MonAcoSFL more practical in real-world scenarios⁴.

Keywords: Federated learning · Self-supervised learning · Contrastive learning

1 Introduction

Collaborative learning techniques allow multiple participating parties to jointly train models without compromising the confidentiality of their private data, making them increasingly popular [28,31,33,38,24]. Among these techniques, Federated Learning (FL) [28] stands out as the most prevalent framework. In FL, the learning task is solved by a federation of participating devices (which we refer to as clients) coordinated by a central server. Each client has a local training dataset (unseen by the server) and computes an update to the current global model maintained by the server (only this update is communicated). Moreover, a part of the model can be trained on the server side, resulting in a setup called Split Federated Learning (SFL) [31].

⁴ Our codebase is available at <https://github.com/gmum/MonAcoSFL>

Federated learning has exhibited considerable success in supervised learning tasks [28,10,32]. However, the assumption of full labeling may sometimes be impractical due to the challenges and expertise required for accurate labeling [27]. That is why more practical methods focus on unlabeled data [37,38], combining FL with classic self-supervised learning (SSL) [16]. A notable example is MocoSFL, which is based on Split Federated Learning (SFL) and Momentum Contrast (MoCo) [17,24].

MocoSFL can achieve good accuracy with relatively low memory requirements and can support a large number of clients. However, it was optimized for network division at the initial layers (one or three layers on client devices), which has certain disadvantages. First, it decreases the protection of the client data (Figure 2). Secondly, it increases communication overhead (information transferred between clients and server), as shown in Figure 1.

In this paper, we delve into the relationship between communication overhead and the splitting point. We identify the optimal splitting point and highlight the poor performance of MocoSFL when aligned with it. As a remedy, we introduce **M**omentum-**A**ligned **C**ontrastive **S**FL (MonAcoSFL). In contrast to MocoSFL, which synchronizes only the online client models during the training, MonAcoSFL also synchronizes their momentum models. This change is crucial because it prevents the divergence of online and momentum models and reduces confusion during training (see Figure 5).

Through extensive experiments, we empirically confirm that the synchronizing scheme of MocoSFL indeed creates misalignment between its online and momentum models, which is not the case in MonAcoSFL. Crucially, we show that when training mobile-friendly architectures split at deeper, more communication-efficient points, MonAcoSFL improves over the performance of MocoSFL [24] by a significant margin, confirming its practicality.

Our contributions can be summarized as follows:

- We analyze the practical trade-off in communication efficiency of split federated learning and show that MocoSFL performs poorly for the most optimal splits.
- We re-think model synchronization in MocoSFL and introduce MonAcoSFL, which reduces the divergence between online and momentum clients at each synchronization step.
- We present an extensive experimental study and confirm that MonAcoSFL maintains high performance even when applied to deeper splits of the backbone model, increasing its practical applicability to real-world scenarios.

2 Related work

Federated learning (FL) [28,36,26,25] is a method of training models across multiple decentralized entities or devices while keeping the data localized, without the need to send it to a centralized location [28]. Models are individually trained and then aggregated by a global server, ensuring data privacy by sharing only model parameters among clients. The seminal FedAVG algorithm [28], which

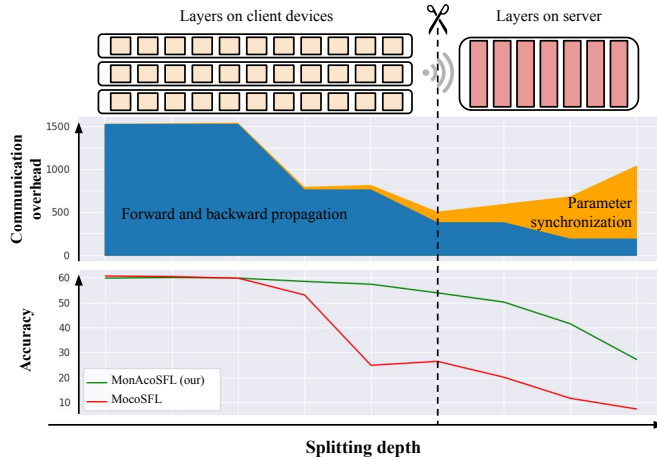


Fig. 1: Communication overhead and accuracy for MocoSFL and MonAcoSFL depending on the splitting depth. In this example, the minimal communication overhead is obtained for dividing the model into 11 and 7 layers on the client and server sides, respectively. For such an optimal case, the accuracy of MocoSFL drops significantly, in contrast to the accuracy of MonAcoSFL. Notice that computational overhead consists of forward and backward propagation (marked as blue) and parameter synchronization (marked as orange).

conducts a straightforward weighted averaging of clients’ model weights, remains a solid foundational framework for numerous approaches [37,38,23]. One of the most important variants of FL is Split Federated Learning (SFL), which involves storing a part of the trained model on a centralized server, reducing the computational overhead of clients at the cost of additional communication overhead due to sending latent vectors from clients to the server [31]. In addition, SFL creates a trade-off between the clients’ energy consumption and privacy [22].

Self-supervised learning (SSL), a pivotal development in unsupervised representation learning, shines particularly in its joint-embedding form within the realm of computer vision [8,9,17,4]. This approach leverages contrastive learning [3,29] to modulate similarity and disparity among augmented views. To prevent trivial solutions, such methods often rely on substantial memory banks, like MoCo [17,12], or necessitate large batch sizes as observed with SimCLR [11], to amass a significant number of meaningful negative examples. On the other hand, non-contrastive methods such as BYOL [16] do not require negative examples in their objectives, yet still require large batch sizes during training [16,13,9]. A common architectural component of several popular SSL approaches is the maintaining of "momentum" models to achieve asymmetry of representations [17,12,14,9,2].

Early attempts at deploying the SSL methods in Federated setting have grappled with the complexity of data diversity [5] and addressed key privacy consid-

erations [34]. Methods like FedU [37] utilize the BYOL framework for federated SSL, while FedEMA [38] refines this by adaptively updating client networks with insights from the global model’s trends. However, the above methods cannot be scaled beyond cross-silo settings, involving up to 100 client devices. Li et.al. demonstrate the key ingredients of scaling SSL to highly distributed cross-client environments (as many as 1000 clients) in the form of MocoSFL – a combination of MoCo and the Split Federated Learning paradigm [24].

3 Preliminaries

In this section, we first describe the problem of federated learning in a self-supervised regime and its challenges. Then, we recall the MocoSFL [24], a recently published method, which we use as a basis for our MonAcoSFL.

3.1 Federated self-supervised learning

Problem statement. In this paper, we tackle the problem of *federated self-supervised learning*. It considers multiple participating devices (called clients) and a centralized server that collaboratively trains a single model based on data gathered by the former. However, the devices are prohibited from sending their raw data to the server e.g. for privacy reasons. Instead, the training is conducted in a *federated learning* manner, where each client trains a copy of the model on its local data and periodically synchronizes it with other clients. Another assumption is that data gathered by the devices do not contain labels. Thus, the model must be trained by optimizing a *self-supervised* objective. After the training, the resulting model can be finetuned to the target tasks using labeled data and distributed to all participating devices.

Split federated learning (SFL) [28] is a practical approach to federated learning, which splits the model into client and server parts. For a formal definition, let us define N as the number of participating clients and f_{ϕ_i} as a copy of model f_{ϕ} stored by the i -th client parametrized by ϕ_i . Furthermore, let f_{ϕ} be composed of two parts $f_{\phi^s} \circ f_{\phi^c}$, where $\phi = \phi^s \cup \phi^c$, f_{ϕ^c} is distributed among client devices, and f_{ϕ^s} is stored on a centralized server. There are multiple copies of f_{ϕ^c} but only one version of parameters⁵ ϕ^s .

Client models start with the same initial parameters ($\phi_1^c = \phi_2^c = \dots = \phi_N^c$) and are trained in two phases:

1. Optimization of $\phi_1^c, \dots, \phi_N^c, \phi^s$ w.r.t. to training objective \mathcal{L} .
2. Synchronization of $\phi_1^c, \dots, \phi_N^c$ by overwriting each ϕ_i^c with $\hat{\phi}^c = \sum_{i=0}^N \frac{\phi_i^c}{N}$.

⁵ Note that FL is a special case of SFL where $f = f^c$, whereas a situation where $f = f^s$ is equivalent to regular training on a single device.

Each client processes only its local data during the optimization step, which leads to increasingly diverging $\phi_1^c, \dots, \phi_N^c$. That is why the synchronization step is required.

In standard Federated learning, the communication is limited to transferring parameters for model synchronization. In SFL, we must also transfer from clients the activations of f^c (for forward propagation by the server) and their gradients w.r.t. to \mathcal{L} from the f^s residing on the server (for backpropagation in local models).

Self-supervised learning (SSL) is a paradigm of learning representations without data labels [1,4]. Currently, the prevalent approaches to SSL are the joint-embedding architectures [17,12,11,16,9,2], where model f learns by optimizing *contrastive objectives*. Formally, let $\mathbf{x}', \mathbf{x}''$ be two augmentations of a sample $\mathbf{x} \sim X$. Contrastive objectives enforce the similarity of $f(\mathbf{x}')$ and $f(\mathbf{x}'')$ while avoiding trivial solutions, i.e., producing identical embeddings of unrelated data samples. For this purpose, most joint-embedding methods [11,9] use objective functions requiring large batch sizes.

Due to the large-data requirements and the computational overhead associated with the contrastive objectives, deploying SSL methods in highly distributed Federated environments is challenging in practice [37,38,24].

3.2 Momentum contrastive split federated learning

Momentum Contrastive Split Federated Learning (MocoSFL) [24] addresses the practical challenges of deploying SSL methods in distributed environments by combining SFL [31] with Momentum Contrastive Learning (MoCo) [17] method.

Similarly to MoCo, the contrastive objective of MocoSFL is calculated with InfoNCE [29] based on the *memory of embeddings* M :

$$\mathcal{L}_{\text{InfoNCE}}(\mathbf{z}', \mathbf{z}'', M) = -\log \frac{\exp(\mathbf{z}' \cdot \mathbf{z}'' / \tau)}{\exp(\mathbf{z}' \cdot \mathbf{z}'' / \tau) + \sum_{\mathbf{z}_M \in M} \exp(\mathbf{z}' \cdot \mathbf{z}_M / \tau)} \quad (1)$$

where $\mathbf{z}' = f_\phi(\mathbf{x}')$, $\mathbf{z}'' = f_{EMA(\phi)}(\mathbf{x}'')$, \mathbf{x}' and \mathbf{x}'' are two augmentations of a single data sample, and \mathbf{z}_M denotes the embeddings stored in the memory M (after each training step, M is updated with \mathbf{z}'' in a first-in-first-out manner). Moreover, ϕ are parameters of the *online* model, and $EMA(\phi)$, being the exponential moving average of ϕ , are parameters of the *momentum* model. Because MocoSFL operates in the SFL setup, each client contains its version of parameters ϕ_i^c and $EMA(\phi_i^c)$, and there is one version of parameters ϕ^s and $EMA(\phi^s)$ on the server. Similarly, the memory M is also maintained by the server.

To our best knowledge, MocoSFL is the only SSL method operating in a challenging cross-client federated learning with over 100 clients, each contributing as few as 250 data samples from different distributions [24]. It is possible by using a large memory of negative examples from all clients, which minimizes the detrimental effect of potentially small batch sizes of individual clients [7,11] and reducing the chance of overfitting to any individual client distribution [35].

Moreover, relegating most model layers and the contrastive objective to the server reduces the computational burden on the clients [31].

4 MonAcoSFL: Momentum-Aligned Contrastive Split Federated Learning

In this section, we first describe the limitations of MocoSFL and show that its performance deteriorates using higher split layers (see Section 4.1). Then, we analyze the source of this deterioration and introduce MonAcoSFL as a proposed solution (see Section 4.2).

4.1 Limitations of MocoSFL

The main limitation of MocoSFL is a drop in performance when splitting the network at the higher layers. In consequence, it can be effectively used only with a few layers on the client side. It leads to two crucial concerns. First, sending representations (activations) from low layers makes the model more sensitive to data leakage and attacks such as the Model Inversion Attack (MIA) [15]. Second, low-layer representations are much larger than those returned by higher layers, which increases communication overhead. Below, we describe those problems in detail.

Privacy concerns caused by sending representations from low layers is illustrated in Figure 2. One can observe that representations of low ResNet18 [18] layers highly resemble the respective input data, in contrast to the activations from the higher layers. In fact, in principle, models that learn perceptive features (such as MoCo) do not retain reconstructive features in their high representations [4]. Thus, in SFL, increasing the number of layers on the client side reduces the privacy risks associated with broadcasting network representations.

Communication overhead. Network architectures used in mobile devices, such as ResNet [18] or MobileNet [30], progressively downsample the spacial dimensions of representations obtained from successive layers while simultaneously increasing their channel dimensions. As a result, the overall representation size *decreases* with network depth, and the communication in the SFL optimization phase requires less bandwidth. However, more layers on client devices require exchanging more parameters during the SFL synchronization phase, which translates to a larger bandwidth. Therefore, a trade-off exists between those two communication overheads, as illustrated in Figure 3 with an optimal split in the higher layers.

Deteriorated performance for higher split layer. Based on the above considerations, it is evident that choosing a larger splitting depth can be attractive due to privacy and efficiency reasons. Thus, a natural question arises – **how does splitting depth affect the performance of MocoSFL?** To answer this, we

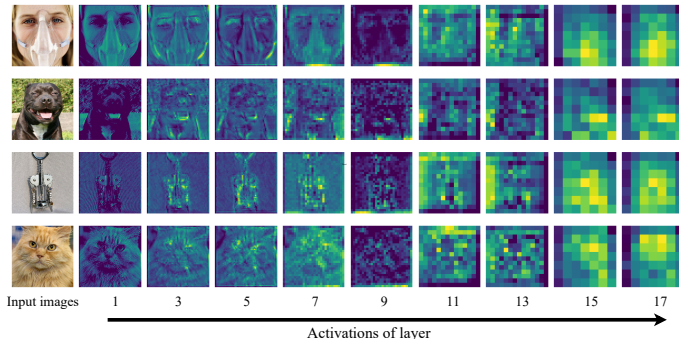


Fig. 2: Activations obtained from successive layers of untrained ResNet-18 for a sample ImageNet image. One can observe that representations of low layers highly resemble the respective input data, increasing the privacy risks associated with broadcasting network activations.

benchmark its performance on the CIFAR-10 dataset [20] for 5, 20, and 200 clients, following the experimental setup of [24] (see Section 5.1) and report the results in Figure 4⁶. Surprisingly, for higher split layers, we observe a catastrophic deterioration in the accuracy of the trained models.

To summarize, pretraining mobile-friendly architectures with MocoSFL with a splitting depth larger than 7 is unreliable, and using a lower splitting depth poses concerns to privacy and communication overhead.

4.2 MonAcoSFL

Before providing our solution to the limitations of MocoSFL, we will first outline the main reason behind them. For this purpose, let us analyze the MocoSFL training procedure in detail. As we described in Section 3.1, client models start with the same initial parameters $\phi_1^c = \phi_2^c = \dots = \phi_N^c$. However, during training, they move away from one another due to different local datasets, and they must be periodically synchronized to stabilize the training. Like in regular SFL schemes, MocoSFL synchronizes *only the online client models*. However, such rapid modification of online parameters breaks the underlying assumption of MoCo that respective online and momentum models encode similar representations at all times, which is essential for minimizing the contrastive objective [17]. We illustrate this phenomenon on the left side of Figure 5. This problem grows with the increased splitting depth because more parameters become misaligned, which explains the decrease in performance.

To overcome this limitation, we propose **Momentum-Aligned Contrastive SFL** (MonAcoSFL) presented on the right side of Figure 5. It ensures the alignment of respective online and momentum client models by synchronizing the

⁶ Note that Li et. al. [24] evaluate MocoSFL only with split layer set to 1 or 3.

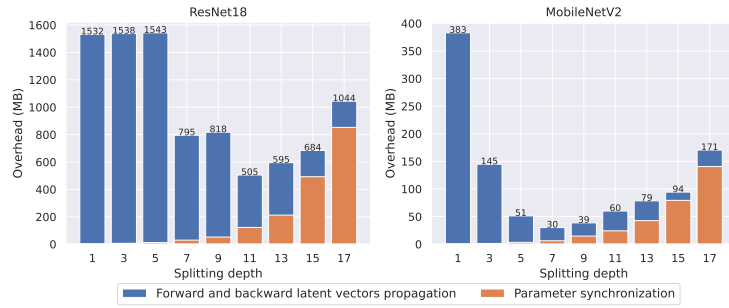


Fig. 3: Communication overhead of a single client device for one training epoch of ResNet-18 [18] and MobileNet-V2 [30] for different splitting depths. The 11-th and 7-th layers are the most communication-efficient for ResNet-18 and MobileNetV2, respectively. Note that the training epoch corresponds to 250 images of resolution 224×224 are processed, and 10 synchronizations of parameters. Moreover, the blue bars correspond to communication in the optimization phase, and the orange bars correspond to parameter synchronization.

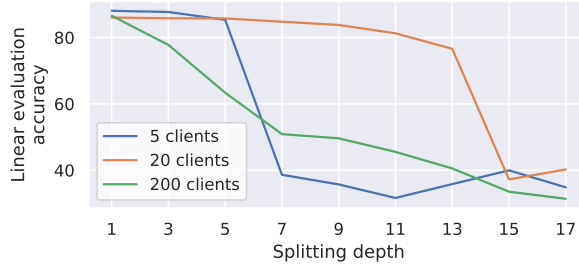


Fig. 4: Accuracy of MocoSFL drops significantly with increased splitting depth regardless of the number of clients. Here, presented for CIFAR-10 [20].

respective client momentum models whenever the online models are synchronized, i.e. by overwriting momentum model of each client with:

$$\widehat{EMA}(\phi^c) = \frac{\sum_{i=0}^N EMA(\phi_i^c)}{N}$$

Note that since $EMA(\phi_1^c), \dots, EMA(\phi_N^c)$ are the EMAs of the individual $\phi_1^c, \dots, \phi_N^c$, their average always corresponds to the EMA of the average value of the online parameters ($\hat{\phi}^c$), i.e: $\widehat{EMA}(\phi^c) = EMA(\hat{\phi}^c)$.

5 Experimental investigation

In this section, we experimentally evaluate MonAcoSFL and compare it with MocoSFL, a recent state-of-the-art method in self-supervised federated learn-

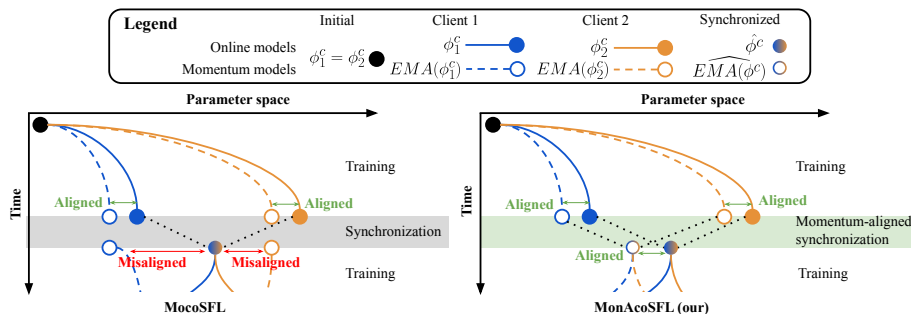


Fig. 5: Visualization of parameters changing in MocoSFL (left) and MonAcoSFL (right) for two clients. The solid and dashed lines represent the progression of online and momentum parameters, respectively. The dotted lines symbolize the synchronization of parameters. The difference between MocoSFL and MonAcoSFL lies in the synchronization procedure that, in the case of MonAcoSFL, ensures that both online and the momentum models remain aligned, preserving their ability to optimize the contrastive objective.

ing. We evaluate both models using classification accuracy on downstream tasks and verifying the privacy of clients data. Finally, we experimentally analyze the alignment of online and momentum models in MoCo framework. For a fair comparison, we closely follow the experimental setup of Li et.al. [24].

5.1 Experimental setup

Hardware We simulate the distributed environment of MocoSFL and MonAcoSFL on a single machine that hosts the client and server models and runs all of them on a single NVidia A100 GPU. We conduct experiments on mobile-friendly ResNet18 [18] and MobileNetV2 [30] backbones.

Data We conduct our experiments on CIFAR-10 and CIFAR-100 [20] datasets. We divide the samples equally between the clients to simulate the situation where each client has access to only a small part of the data. We assume a challenging setting where the data is not Independent and Identically Distributed (non-IID) between the client devices. Namely, we assign for each client images from randomly chosen 2 (for CIFAR-10) or 20 (for CIFAR-100) classes.

Number of clients We compare MocoSFL and MonAcoSFL in both cross-silo (5 or 20 clients) and cross-client (200 clients) settings. To our knowledge, MocoSFL is the only previous self-supervised federated method able to scale to the cross-client setting. We synchronize the models on client devices 10 times per training epoch, which amounts to synchronizing the model every 1000, 250, and 25 encountered images for the 5, 20, and 200-client settings, respectively. We adjust the client-side batch size and client sampling ratio accordingly in order to keep the server-side batch size at around 100. For example, in the 5-client setting,

the batch size is 20 and all clients are used at each training epoch, whereas in the 200-client setting, we use a random choice of 100 clients in each epoch and therefore each client has a batch size of 1.

SSL model We use the MoCo-v2 [12], where the backbone is succeeded by a 2-layer MLP projector network with a hidden size of 1024 [11,6], discarded after SSL pretraining. The server maintains a memory M of 6000 negative embeddings, implemented as a First-In-First-Out (FIFO) queue. After each training step, the queue is updated with the momentum model embeddings of the most recent mini-batch of samples. We train MocoSFL for 200 epochs, with the SGD optimizer, with an initial learning rate of 0.06, 0.9 momentum, and 0.0005 weight decay. The learning rate is scheduled throughout the training according to the cosine scheduler.

Evaluation After each epoch, we perform the k-NN validation of the model using 20% of the validation dataset. K-NN validation is the standard method of evaluating self-supervised representations during their training [21,24]. After training, we use the model which achieved the best k-NN performance. We measure the final performance of the model using the widely used linear evaluation protocol [11,16,38,21,24]. Namely, we attach random linear layer to the frozen pre-trained backbone and train only this layer on the labeled dataset for 100 epochs, with a batch size of 128 and the Adam optimizer [19] with initial learning rate of 0.001 and cosine learning rate scheduler.

5.2 Accuracy performance

Figure 6 presents the accuracy of MonAcoSFL and MocoSFL in the cross-silo (5 or 20 clients) and cross-client (200 clients) settings on the ResNet architecture [20]. Although both models decrease the accuracy with increasing splitting depth, this decrease is much smaller in MonAcoSFL than in MocoSFL. In the most communication-efficient splitting point (layer 11-13), the difference between MonAcoSFL and MoCoSFL exceeds 30 percentage points of accuracy. An even stronger advantage of MonAcoSFL is confirmed on the MobileNet backbone, see Figure 7. In the case of 20 clients, MonAcoSFL improves the accuracy of MoCoSFL by more than 40 percentage points starting from 3th to 15th cut layer on the CIFAR-10 dataset.

5.3 Privacy Evaluation

We next compare the privacy-preserving capabilities of MonAcoSFL and MoCoSFL with a Model Inversion Attack (MIA) [15]. We assume that the attacker has access to 1% of the training data and trains a decoder for reconstructing images from client model embeddings. We train the decoder with the MSE loss and Adam optimization with a learning rate of 0.001 over 50 epochs and a batch size of 32. We perform an attack on ResNet-18 networks pretrained on CIFAR-100 dataset with 20 clients and varying cut-layers. The attack is conducted

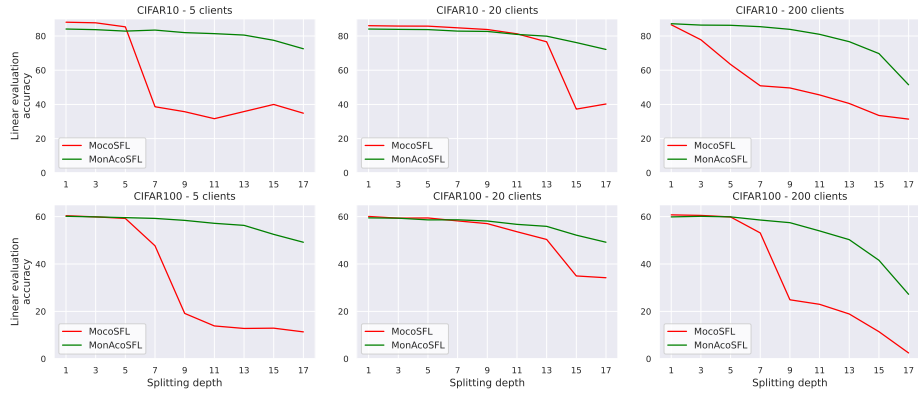


Fig. 6: Linear evaluation of MocoSFL and MonAcoSFL on ResNet18 architecture. MonAcoSFL maintains the accuracy with increasing cut-layers, whereas the performance of MocoSFL rapidly deteriorates.

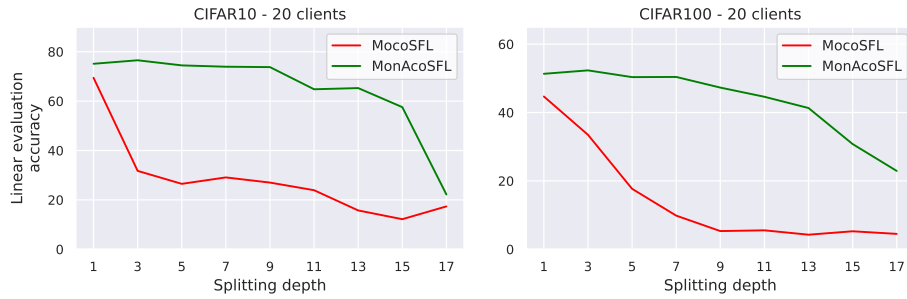
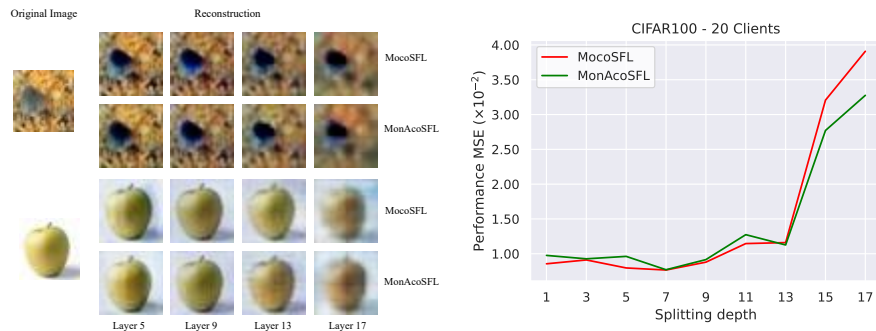


Fig. 7: Linear evaluation accuracy on MobileNetV2 backbone trained with MocoSFL and MonAcoSFL. There is significant discrepancy between MonAcoSFL and MocoSFL at every splitting depth.

without pre-training or employing additional privacy-enhancing techniques like TAResSFL [24].

We compare the original and reconstructed images in terms of MSE in Figure 8b, where a higher MSE indicates better privacy protection. We find that for cut-layers 1-13 the attacks on MocoSFL and MonAcoSFL yield similar reconstruction errors, with the errors of attacks on MonAcoSFL being larger by a small margin. The MSE remains relatively stable for layers 1-7 and increases for layers 9-17, indicating greater privacy of embeddings of these cut-layers. In layers 15-17, MocoSFL achieves larger attack MSE than MonAcoSFL. We attribute this to the fact that the representation of MocoSFL trained with these cut-layers is in general of much worse quality (which results in lower accuracy, see Figure 6) and, as a side-effect, encodes less information about the data. Concluding, using deeper cut-layers (9-13) is not only more efficient from the computational point of view but also increases the protection of client data.



(a) Images reconstructed by the attacker at different layers following MIA on both MocoSFL and MonAcoSFL. (b) MSE of reconstructing the original images by the attacker. Higher MSE values for deeper cut-layers indicates a better resistance to attack (better privacy).

Fig. 8: MIA attack on ResNet-18 models trained with MocoSFL and MonAcoSFL. Deeper splits provide better protection of client data. Both methods achieve similar levels of privacy protection.

5.4 Analysis of models alignment

To empirically verify that MonAcoSFL indeed preserves the alignment of online and momentum model parameters, we measure it throughout the training. We define the average misalignment of online and momentum parameters as their average absolute difference i.e.:

$$\sum_{i=0}^N \frac{|\phi_i^c - EMA(\phi_i^c)|}{N \cdot dim(\phi^c)}, \quad (2)$$

where N denotes the number of clients and $dim(\phi^c)$ is the dimension of client model parameters.

We plot in Figure 9 the misalignment values for 1500 initial training steps (out of approximately 50000) of ResNet18 trained on CIFAR-100 by 20 clients, MocoSFL and MonAcoSFL, with split at the 11th layer. Throughout the first 125 steps, both methods display similar misalignments. However, during parameter synchronizations, the difference between the online and momentum models in MocoSFL rapidly increases by an order of magnitude. On the other hand, in MonAcoSFL, the misalignment of momentum and online models does not change significantly throughout the training.

6 Conclusion

In this paper, we conducted an in-depth analysis of MocoSFL – the state-of-the-art method for Federated Self-Supervised Learning [24]. We found that the quality of this method deteriorates in privacy-preserving and communication-efficient

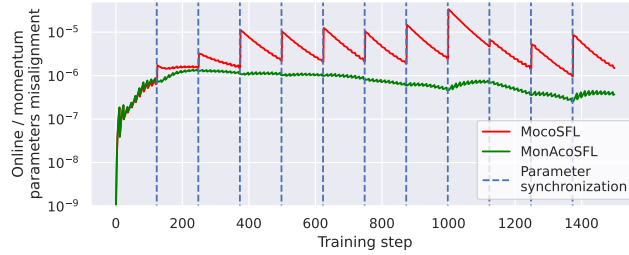


Fig. 9: A comparison of average misalignment of online and momentum client models in the initial steps of training for MonAcoSFL and MocoSFL (lower is better). We mark with blue lines the steps at which the parameter synchronization is performed (every 125 steps). In MocoSFL the misalignment grows rapidly during parameter synchronizations, whereas in MonAcoSFL it remains an order of magnitude smaller, resulting in a more stable training.

settings and identified that the reason for this is its synchronization scheme, which leaves the participating online and momentum models misaligned. We addressed this problem by introducing **M**omentum-**A**ligned **c**ontrastive **S**plit **F**ederated **L**earning (MonAcoSFL) and showed that it significantly improves the performance of MoCoSFL, especially under communication-efficient conditions.

From a theoretical perspective, our solution is centered on the correct synchronization of online and momentum models of self-supervised learners deployed in distributed environments. Given that the online/momentum model pairs are a staple of several modern SSL approaches [16,9,2], our findings can be applicable in federated learning of various SSL methods other than MoCo [17]. From the practical perspective, MonAcoSFL is the first Federated self-supervised method to achieve good results using deeper cut layers, providing efficient communication between clients and server and better protection of client data.

Ethical implications

In this work, we focus on improving the existing Federated self-supervised approaches in regimes where data privacy and communication efficiency are of concern. Our findings can lead to reducing the risks of data leakages when deploying Federated self-supervised learning algorithms in real-world scenarios.

Acknowledgements

This research has been supported by the flagship project entitled "Artificial Intelligence Computing Center Core Facility" from the Priority Research Area Digi-World under the Strategic Programme Excellence Initiative at Jagiellonian University, and by the Horizon Europe Programme (HORIZON-CL4-2022-HUMAN-02) under the project "ELIAS: European Lighthouse of AI for Sustainability", GA no. 101120237. The research of Marcin Przewiężlikowski was supported by

the National Science Centre (Poland), grant no. 2023/49/N/ST6/03268. The research of Marcin Osial was supported by National Science Centre (Poland) grant number 2023/50/E/ST6/00469. The research of Marek Śmieja was supported by the National Science Centre (Poland), grant no. 2022/45/B/ST6/01117. The research of Bartosz Zieliński was supported by National Science Centre (Poland) grant number 2022/47/B/ST6/03397. Some experiments were performed on servers purchased with funds from a grant from the Priority Research Area (Artificial Intelligence Computing Center Core Facility) under the Strategic Programme Excellence Initiative at Jagiellonian University. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016303.

References

1. Albelwi, S.: Survey on self-supervised learning: Auxiliary pretext tasks and contrastive learning methods in imaging. *Entropy* **24**(4) (2022). <https://doi.org/10.3390/e24040551>, <https://www.mdpi.com/1099-4300/24/4/551>
2. Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., Ballas, N.: Self-supervised learning from images with a joint-embedding predictive architecture. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 15619–15629 (June 2023)
3. Bachman, P., Hjelm, R.D., Buchwalter, W.: Learning representations by maximizing mutual information across views. *Advances in neural information processing systems* **32** (2019)
4. Balestrieri, R., Ibrahim, M., Sobal, V., Morcos, A.S., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A.G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsiavash, H., LeCun, Y., Goldblum, M.: A cookbook of self-supervised learning. *ArXiv* **abs/2304.12210** (2023), <https://api.semanticscholar.org/CorpusID:258298825>
5. van Berlo, B., Saeed, A., Ozcebe, T.: Towards federated unsupervised representation learning. In: *Proceedings of the third ACM international workshop on edge systems, analytics and networking*. pp. 31–36 (2020)
6. Bordes, F., Balestrieri, R., Garrido, Q., Bardes, A., Vincent, P.: Guillotine regularization: Why removing layers is needed to improve generalization in self-supervised learning. *Transactions on Machine Learning Research* (2023), <https://openreview.net/forum?id=ZgXfXSz51n>
7. Bulat, A., Sánchez-Lozano, E., Tzimiropoulos, G.: Improving memory banks for unsupervised learning with large mini-batch, consistency and hard negative mining. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 1695–1699 (2021). <https://doi.org/10.1109/ICASSP39728.2021.9414389>
8. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) *Advances in Neural Information Processing Systems*. vol. 33, pp. 9912–9924. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf

9. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)
10. Chen, C., Zhou, J., Zheng, L., Wu, H., Lyu, L., Wu, J., Wu, B., Liu, Z., Wang, L., Zheng, X.: Vertically federated graph neural network for privacy-preserving node classification. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 1959–1965 (2022)
11. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 1597–1607. PMLR (13–18 Jul 2020), <https://proceedings.mlr.press/v119/chen20j.html>
12. Chen, X., Fan, H., Girshick, R.B., He, K.: Improved baselines with momentum contrastive learning. CoRR **abs/2003.04297** (2020), <https://arxiv.org/abs/2003.04297>
13. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15750–15758 (June 2021)
14. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9640–9649 (October 2021)
15. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. p. 1322–1333. CCS '15, Association for Computing Machinery, New York, NY, USA (2015). <https://doi.org/10.1145/2810103.2813677>, <https://doi.org/10.1145/2810103.2813677>
16. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Dohersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., Valko, M.: Bootstrap your own latent - a new approach to self-supervised learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 21271–21284. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf
17. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)
20. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
21. Lee, H., Lee, K., Lee, K., Lee, H., Shin, J.: Improving transferability of representations via augmentation-aware self-supervision. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 17710–17722. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper_files/paper/2021/file/94130ea17023c4837f0dcdda95034b65-Paper.pdf
22. Lee, J., Seif, M., Cho, J., Poor, H.V.: Exploring the privacy-energy consumption tradeoff for split federated learning (2024)

23. Lee, R., Kim, M., Li, D., Qiu, X., Hospedales, T., Huszár, F., Lane, N.D.: Fedl2p: Federated learning to personalize. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), <https://openreview.net/forum?id=FM81CI68Iz>
24. Li, J., Lyu, L., Iso, D., Chakrabarti, C., Spranger, M.: MocoSFL: enabling cross-client collaborative self-supervised learning. In: The Eleventh International Conference on Learning Representations (2023), <https://openreview.net/forum?id=2QGJXyMN0Pz>
25. Li, X., Jiang, M., Zhang, X., Kamp, M., Dou, Q.: Fedbn: Federated learning on non-iid features via local batch normalization. arXiv preprint arXiv:2102.07623 (2021)
26. Liu, Q., Chen, C., Qin, J., Dou, Q., Heng, P.A.: Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1013–1023 (2021)
27. Makhija, D., Ho, N., Ghosh, J.: Federated self-supervised learning for heterogeneous clients (2023), <https://openreview.net/forum?id=bNPth9YMqZ>
28. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.y.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Singh, A., Zhu, J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (20–22 Apr 2017), <https://proceedings.mlr.press/v54/mcmahan17a.html>
29. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
30. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4510–4520. IEEE Computer Society, Los Alamitos, CA, USA (jun 2018). <https://doi.org/10.1109/CVPR.2018.00474>, <https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00474>
31. Thapa, C., Mahawaga Arachchige, P.C., Camtepe, S., Sun, L.: Splitfed: When federated learning meets split learning. Proceedings of the AAAI Conference on Artificial Intelligence **36**(8), 8485–8493 (Jun 2022). <https://doi.org/10.1609/aaai.v36i8.20825>, <https://ojs.aaai.org/index.php/AAAI/article/view/20825>
32. Wu, C., Wu, F., Lyu, L., Qi, T., Huang, Y., Xie, X.: A federated graph neural network framework for privacy-preserving personalization. Nature Communications **13**(1), 3091 (2022)
33. Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y.: A survey on federated learning. Knowledge-Based Systems **216**, 106775 (2021). <https://doi.org/https://doi.org/10.1016/j.knosys.2021.106775>, <https://www.sciencedirect.com/science/article/pii/S0950705121000381>
34. Zhang, F., Kuang, K., Chen, L., You, Z., Shen, T., Xiao, J., Zhang, Y., Wu, C., Wu, F., Zhuang, Y., et al.: Federated unsupervised representation learning. Frontiers of Information Technology & Electronic Engineering **24**(8), 1181–1193 (2023)
35. Zhang, F., Kuang, K., You, Z., Shen, T., Xiao, J., Zhang, Y., Wu, C., Zhuang, Y., Li, X.: Federated unsupervised representation learning (2020)
36. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
37. Zhuang, W., Gan, X., Wen, Y., Zhang, S., Yi, S.: Collaborative unsupervised visual representation learning from decentralized data. In: IEEE/CVF International Conference on Computer Vision (CVPR). pp. 4912–4921 (2021)

38. Zhuang, W., Wen, Y., Zhang, S.: Divergence-aware federated self-supervised learning. In: International Conference on Learning Representations (2022), <https://openreview.net/forum?id=oVE1z8N1Ne>