Persistence of Member Contribution Under Churn

Robust Decentralized Learning

Luka Roginic
Supervisors: Bart Cox, Jérémie Decouchant
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to the EEMCS Faculty of Delft University of Technology,
In Partial Fulfilment of the Requirements
For the **Bachelor of Computer Science and Engineering**
June 8, 2025

Name of the student: Luka Roginic
Final project course: CSE3000 Research Project
Thesis committee: Bart Cox, Jérémie Decouchant, Anna Lukina

Persistence of Member Contribution Under Churn

Luka Roginić, Bart Cox, Jérémie Decouchant
Delft University of Technology

***Abstract*—Decentralized learning is a paradigm that enables machine learning in a distributed and decentralized manner. A common challenge in this setting is the presence of non-identically and independently distributed (non-IID) data across clients. Under such conditions, it has been shown that node churn, where clients leave and rejoin the system, leads to reduced generalization performance and slower convergence. This degradation occurs because certain data classes may exist only on a few clients. Thus, if those clients drop out, the global model may lose access to important parts of the data distribution. This setting poses an important question: *How can we mitigate the impact of node churn in decentralized learning systems to maintain some persistence of member contributions?* To address this challenge, we empirically study the effectiveness of data augmentation, specifically extending local datasets with small synthetic datasets received from neighbors and generated using their local data. We further enhance this approach with supervised contrastive loss applied to synthetic and local data together, to which we refer to as *synthetic anchors*. Through experiments on the MNIST and CIFAR10 datasets, we demonstrate that data augmentation and synthetic anchors effectively mitigate the effects of churn and help preserve member contribution in decentralized learning.**

I. Introduction

Decentralized learning is a paradigm to perform machine learning from multi-source data in a distributed and decentralized manner. Decentralized learning is strongly connected to federated learning in which a central server orchestrates the learning of each member. Some advantages of decentralized learning over federated learning include increased ownership, privacy, and scalability [1–3]. One of the possible use cases of decentralized learning is utilizing data generated by edge devices, such as mobile phones and Internet of Things devices. Such data is often proprietary, discouraging owners from sharing it [4, 5]. Additionally, the increase in data protection regulations may legally prohibit the transmission of such data from devices, particularly when it contains sensitive personal information [6]. The need for decentralized learning solutions is further supported by research indicating the necessity of such approaches, as shown by Wang et al. [7].

Decentralized learning has been extensively studied, and many algorithms have been proposed to enable collaborative model training in a decentralized setting [8]. Various aspects of decentralized learning have been investigated, including its complexity [9], comparisons with federated learning [1–3], its performance under non-independent and identically distributed (non-IID) data [10–12], and its robustness to node churn [13]. Although churn is a well-known issue in distributed systems, and it has been thoroughly studied how to mitigate or prevent it [14, 15], there is still room for exploring it in decentralized learning. There is a lack of a systematic approach to studying how churn impacts decentralized learning and how it can be mitigated such that some member contribution is preserved, especially under non-IID conditions, where it becomes most evident [13]. A possible situation where churn appears in decentralized learning is when data is coming from mobile phones which often switch between being online and offline [3]. Existing efforts often treat performance of decentralized learning under churn as secondary issue, overlooking its significance. This leads us to our research question: *How can we mitigate the impact of node churn in decentralized learning systems to maintain some persistence of member contributions?*

In this work, we aim to answer this research question by empirically investigating how decentralized learning aided by methods based on dataset condensation performs under churn and non-IID settings. Before training starts each member synthesizes a small synthetic dataset that carries condensed information about their local data. This synthetic dataset is then exchanged with neighbors to be used in further training. In the first method, called data augmentation, members augment the synthetic datasets received from their neighbors to their local train set. Then they continue training as in the baseline algorithm. The second method builds upon data augmentation. After augmenting the neighbors' synthetic datasets, the loss function is extended to include supervised contrastive loss. This loss encourages samples of the same class to be pulled closer together, making the synthetic data act as anchors to the local dataset. We refer to this approach as synthetic anchors. This method is inspired by DeSA [10]. The overview of both methods can be found in Figure 1. To the best of our knowledge, we are the first to conduct an empirical study on how decentralized learning aided by dataset condensation based methods performs under churn and non-IID conditions, and how they help preserve member contributions. This work makes the following contributions:

- We empirically evaluate how data augmentation with dataset condensation performs under churn and contributes to preserving client contribution.
- We empirically evaluate how synthetic anchors perform under churn and contribute to preserving client contribution.

Our results show that data augmentation improves performance under churn across both MNIST and CIFAR10 datasets. On MNIST, test accuracy increases by up to 6.73%, from 90.67% to 97.40%, while accuracy on missing members' data(MMD) improves by as much as 6.67%. On CIFAR10, data

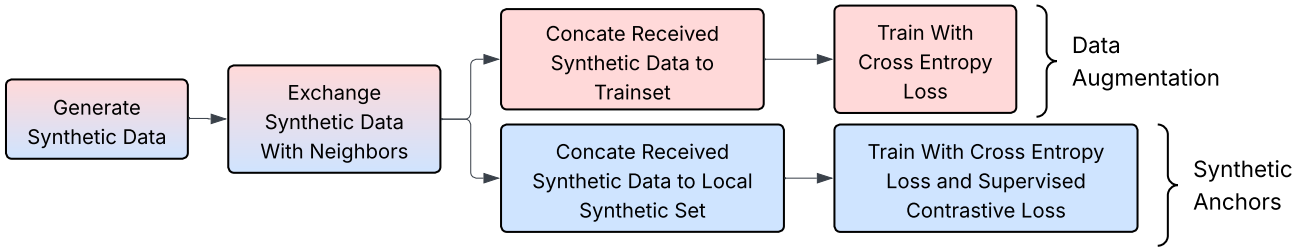


Fig. 1: Overview of Data Augmentation and Synthetic Anchors

augmentation yields test accuracy gains of up to 5.32% and improvements of up to 7.89% on MMD accuracy. Additionally, data augmentation improves the stability of the model, reflected by lower standard deviations. Similarly, synthetic anchors provide improvements of up to 9.29% in test accuracy and 6.75% in MMD accuracy, while also reducing the standard deviation of average accuracy across the network.

II. Background

A. Decentralized Learning

Decentralized learning is a collaborative machine learning approach where members train a model without relying on a central server. Each member trains its own local model and exchanges information, usually model weights, with neighbors during communication rounds. Upon receiving information from their neighbors, each member uses it to infer knowledge about the global model.

A popular example of decentralized learning is gossip learning [8]. Gossip learning is characterized by its sampling strategy, online learning model, and ensembling method. During training, each node updates its local model, most often using stochastic gradient descent(SGD). Afterward, the node selects neighbors to send its model to using a sampling method. Upon receiving models from its neighbors, the node combines them with its own model using an ensembling method. A common ensembling method is plain averaging, where a node updates its model by averaging it with those received from its neighbors. Let $\mathcal{N}_i$ denote the set of neighbors of node i, and let $\theta_j^{(t)}$ represent the model parameters received from neighbor j at time t. The updated model at node i is given by:

$$\theta_i^{(t+1)} = \frac{1}{|\mathcal{N}_i|+1} \left(\theta_i^{(t)} + \sum_{j \in \mathcal{N}_i} \theta_j^{(t)} \right),$$

where $\theta_i^{(t)}$ is the current model of node i. This averaging strategy treats all models equally.

Decentralized learning systems are further characterized by data distribution among the members. Two distinct cases are identically and independently distributed (IID) data and non-IID data. Non-IID data is often modeled using a Dirichlet distribution, which is characterized by an α parameter indicating the degree of non-IID-ness [16] with a lower value of α resulting in more unbalanced distribution across clients. Another important aspect of decentralized systems is network topology. Denser topologies can accelerate convergence, while sparser networks may have the opposite effect. A common setting in decentralized learning involves sparse network topologies.

The advantages of decentralized systems compared to centralized ones include the absence of a single point of failure, which makes the system more resilient; improved scalability, since no new infrastructure is needed when new members join; and enhanced privacy, as there is no central node aggregating all the information [1–3].

B. Churn

In distributed systems, it is inevitable that members will join and leave over time, which is known as churn. Every robust decentralized system must be designed to handle churn. Members may leave permanently or rejoin the system at later stages. Systems must define how to respond when members leave gracefully (e.g., by sending a leave message) or unexpectedly due to failures. While graceful departures can often be treated similarly to failures, they may still warrant separate handling in systems where graceful exits are common, potentially allowing for better performance. Churn is further characterized by dropout rate, how often members leave the system, and the offline duration, how long members stays outside the system after leaving [13]. We distinguish a special case of churn, in which members do not return, from other churn patterns. A popular way to model churn is by using traces from real devices, for example smartphones [3].

C. Dataset Condensation

The computational cost of training state-of-the-art models across various fields has been rapidly increasing due to the growing size of models and datasets. Designing or adapting deep learning models to new tasks requires even more computational resources, as it often involves training multiple models on the same dataset to evaluate design choices such as loss functions, architectures, and hyperparameters. Therefore, there is a need for solutions that can reduce model training time [17]. One approach is to create smaller training sets that preserve the essential information of the original dataset to a predetermined degree. Dataset condensation generates a compact synthetic dataset on which models can be trained to achieve performance comparable to models trained on

the full dataset. Common strategies for dataset condensation include gradient matching, aligning gradients between real and synthetic data, and distribution matching, which aligns their feature embeddings [17, 18]. The goal of dataset condensation can be formalized as:

$$\mathbb{E}_{x\sim P_D}\left[\ell\left(\phi_{\theta_{\mathcal{T}}}(x), y\right)\right] \simeq \mathbb{E}_{x\sim P_D}\left[\ell\left(\phi_{\theta_{\mathcal{S}}}(x), y\right)\right],$$

where P_D is the real data distribution, ℓ is the loss function, ϕ is a model parameterized by θ, and $\phi_{\theta_{\mathcal{T}}}$, $\phi_{\theta_{\mathcal{S}}}$ are networks trained on the real dataset $\mathcal{T}$ and the synthetic dataset $\mathcal{S}$, respectively.

III. Related Work

A. Decentralized learning

One of the most widely studied decentralized learning approaches is gossip learning [2, 8]. Decentralized learning has also been extensively compared to centralized methods such as federated learning [1–3]. Lu et al. provide tight lower bounds on the time complexity of decentralized learning algorithms, as well as an algorithm that achieves it [9]. However, these works do not address churn or non-IID data

B. Churn

Liu et al. study gossip learning under different churn scenarios, such as non-IID data, offline duration, and dropout rates [13]. Their study shows that churn is most impactful in non-IID scenarios. However, they do not address how to mitigate the impact of churn or preserve member contributions. Dinani et al. propose two novel approaches for modeling dynamic networks with churn using time-varying graphs based on the Erdős–Rényi and Barabási–Albert models [19]. Their results suggest churn may help if nodes rejoin intelligently. However, their work considers dynamic topologies, while ours assumes static. Hegedűs et al. compare the performance of federated learning and gossip learning under various conditions, including churn [2]. Their findings indicate that churn delays convergence and introduces noise in gossip learning. However, they do not explore methods to mitigate churn or preserve client contributions. Stutzbach et al. study churn in peer-to-peer networks, while Berta et al. provide an empirical study of churn in mobile systems [24, 25]. Rhea et al. address churn in distributed hash tables [14], and Godfrey et al. study how to reduce churn by selecting a subset of available nodes [15]. These works do not focus on decentralized learning specifically but contribute to a broader understanding of churn in decentralized systems.

C. Decentralized learning and non-IID data

Huang et al. propose DeSA, a decentralized learning algorithm that uses synthetic anchors to address non-IID data [10]. Although they use synthetic data as anchors, they do not focus on how these techniques help preserve client contributions or mitigate the impact of churn. Chengxi Li et al. introduce Def-KT, an algorithm that leverages knowledge distillation to improve model generalization under non-IID conditions [11]. SS-DGST uses gradient tracking and snapshots to improve convergence in decentralized SGD with random topologies [20]. However, these works neither address churn nor incorporate synthetic data. De Luca et al. apply data augmentation to address non-IID data in federated learning, whereas our focus is on decentralized learning [21]. Furthermore, they do not use dataset condensation. Sha et al. propose a novel algorithm to minimize the effect of non-IID data in federated learning by utilizing Centered Kernel Alignment-based member selection and dataset condensation [22]. ShiMao et al. use dataset condensation with the principle of least privilege to construct FLiP, an algorithm designed to help preserve privacy in federated learning[26]. Song et al. propose FedD3, a one-shot federated method using dataset condensation for non-IID settings [23]. The works of Sha et al., Song et al., and ShiMao et al. are not applicable to decentralized learning and do not model churn. A summary of similarities between our work and the studies in this section can be found in table I.

D. Dataset Condensation

Zhao et al. developed a new method for dataset condensation based on distribution matching, with improved computational complexity compared to previous work [17]. Another method for dataset condensation is gradient matching, proposed by Zhao et al. [18]. Dhasade et al. introduce Quickdrop, an algorithm that uses dataset condensation to improve unlearning in federated learning [27]. Dong et al. show that data condensation preserves privacy [28]. Carlini et al. question the privacy properties of dataset condensation [29]. The works in this section do not touch the topic of decentralized learning, unlike ours.

IV. System Model

The network topology in this experiment is a static, regular graph, where no edges are added or removed during training. The network is closed, meaning all members are present before training begins and no new members join or leave.

Training proceeds with synchronized iterations: each node must receive messages from all of its neighbors before proceeding to the next iteration. Churn is simulated by having nodes skip training and send a "churned" message to their neighbors. Although nodes do not actually leave, this approach effectively captures the impact of churn. Churn is simulating members leaving due to failure, and we assume that all members behave consistently toward all other members.

Each method is evaluated against a baseline and assessed by test accuracy on the global test set and on MMD.

V. Methodology

A. Churn

Permanent churn is modeled using a schedule, which determines which members will permanently leave in which iteration. All members that leave do so before convergence of the model. Furthermore, it is guaranteed that the schedule leaves the network connected. Probabilistic churn is modeled using a Bernoulli distribution: in each round, a node has a certain probability of leaving the training process and, if absent, probability of rejoining.

Title	Decentralized learning	Modeling churn	Mitigating churn	Preserving member contribution	Mitigating non-iid data	Dataset condensation
DeSA [10]	✓	✗	✗	✗	✓	✓
Liu et al. [13]	✓	✓	✗	✗	✗	✗
Dinanai et al. [19]	✓	✓	✗	✗	✗	✗
Def-KT [11]	✓	✗	✗	✗	✓	✗
SS-DSGT [20]	✓	✗	✗	✗	✓	✗
Luca et al. [21]	✗	✗	✗	✗	✓	✗
DCFL [22]	✗	✗	✗	✗	✓	✓
FedD3 [23]	✗	✗	✗	✗	✓	✓
This work	✓	✓	✓	✓	✓	✓

TABLE I: Related work

B. Data Augmentation

Intuition. Churn is most impactful in decentralized learning under non-IID settings because exit of a client can cause the loss of entire data classes from the training process. To address this, each client generates a small synthetic dataset representing the client's local data distribution. This synthetic dataset is then transmitted to neighbors, who use it in their local training. We hypothesize that the aggregated synthetic dataset from the neighbors will provide a more balanced data distribution and preserve member contributions.

Approach. Luca et al. demonstrated that data augmentation can improve federated learning under non-IID conditions [21]. However, their approach applies only minor transformations to local data, which limits information selected samples carry. This limitation does not exist in dataset condensation, where synthetic samples can be more information dense [17]. We combine dataset condensation with augmentation in a decentralized learning setting. At the start, each member generates a small synthetic dataset that reflects its local data distribution. These synthetic datasets are significantly smaller than the original datasets, and their size per class is determined by:

$$\text{synthetic\_size}_{i,c} = \max\left(10,\ 0.01 \times \left|\text{original\_data}_{i,c}\right|\right)$$

where *i* denotes the member and *c* denotes the class. To generate the synthetic dataset, we apply distribution matching by Zhao et al. [17], which leverages the empirical Maximum Mean Discrepancy loss [30], defined as:

$$D_i^{\text{Syn}} = \arg\min_{D} \left\| \frac{1}{|D_i|} \sum_{(x,y)\in D_i} \psi^{\text{rand}}(x|y) - \frac{1}{|D|} \sum_{(x,y)\in D} \psi^{\text{rand}}(x|y) \right\|^2 \tag{1}$$

Here, ψ^{rand} denotes a feature extractor, which is a randomly initialized neural network. We used three different networks: CNN, LeNet, and ResNet which are periodically exchanged to provide diverse views of the data and improve generalization. The dataset D_i represents the local data available to client i, while D_i^{Syn} corresponds to the synthetic dataset being optimized. We use the Adam optimizer [31], while Zhao et al. use stochastic gradient descent. Lastly, each data point is augmented with color, crop, cutout, scale, and rotate transformations during loss computation to enhance performance. This formulation allows the construction of the synthetic anchor dataset in a class-balanced manner, avoiding label distribution bias toward any particular subset of classes. A diagram illustrating the overall procedure is shown in Figure 2.

After generating the synthetic dataset, each member sends their synthetic dataset to their neighbors. Received datasets are collected before training begins and augmented to the local dataset. Afterwards, training proceeds as in the baseline algorithm. Detailed algorithm can be found in appendix A.

C. Synthetic Anchor

Intuition. Synthetic anchors extend the data augmentation approach by leveraging the idea that synthetic samples are more information dense than local data by incorporating supervised contrastive loss. This loss brings embeddings of the same class closer and pushes apart those of different classes. By keeping synthetic data fixed during training, local data is drawn toward them, effectively turning synthetic samples into anchors for global information. We hypothesize that combining cross entropy loss, which focuses on classification accuracy, with supervised contrastive loss, which emphasizes semantic clustering, will yield complementary objectives that improve generalization and preserve member contribution.

Approach. Huang et al. [10] demonstrated that synthetic anchors can improve model's generalizability under non-IID data setting. Our approach adapts their algorithm, DeSA, to mitigate impact of churn and help preserve client contribution. The method begins by generating synthetic data using distribution matching, as in the data augmentation approach. Synthetic data is then exchanged with neighbors, who, upon receiving the data merge it with their own synthetic data, in contrast to data augmentation where synthetic data is merged into the local train set. This design choice enables us to leverage the global information encoded in synthetic samples through supervised contrastive loss [32]. By detaching synthetic samples (i.e., not updated) during training, they become anchors for global information, aligning local features with global ones. Due to the difference in size between the local training set and the synthetic dataset, batches from each set are sampled separately to ensure synthetic data is used in every training round. This allows each client to align its local private data with the global

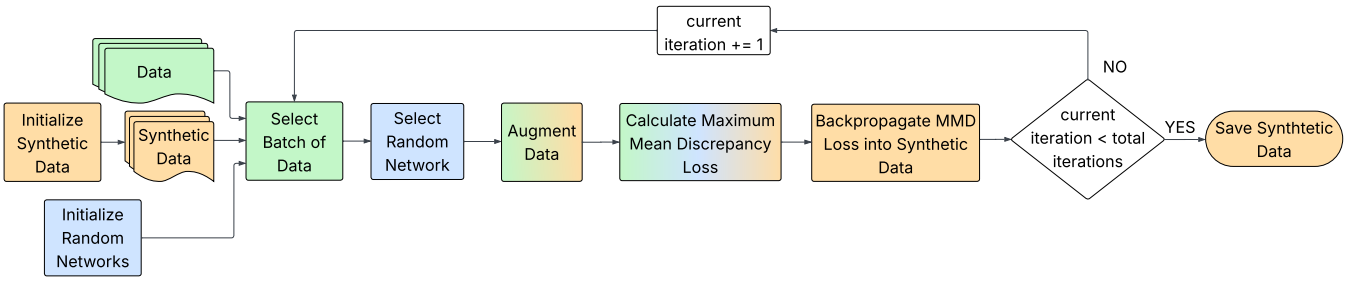


Fig. 2: Synthetic Data Generation

data distribution. The final supervised contrastive loss is defined as follows:

$$\mathcal{L}_{\text{SCL}}(\psi_i) = \mathbb{E}[d(\psi_i(D^{\text{syn}}) \| \psi_i(D_i))],$$

Where d stands for supervised contrasted loss distance computed as follows:

$$d(\psi_i; D^{\text{syn}}, D_i) = \sum_{j \in \mathcal{B}} \frac{1}{|\mathcal{B}^j_{y_j}|} \sum_{\mathbf{x}_p \in \mathcal{B}^j_{y_j}} \log \frac{\exp(\psi_i(\mathbf{x}_j) \cdot \psi_i(\mathbf{x}_p)/\tau_{\text{temp}})}{\sum_{\mathbf{x}_a \in \mathcal{B}_j} \exp(\psi_i(\mathbf{x}_j) \cdot \psi_i(\mathbf{x}_a)/\tau_{\text{temp}})} \quad (2)$$

Here, $\mathcal{B}_j$ represents a batch that includes both the local raw data D_i and the global synthetic anchor data D^{syn}, but without the specific sample j. The subset $\mathcal{B}^j_{y_j} \subset \mathcal{B}_j$ includes only those samples with label y_j. The scalar τ_{temp} is a temperature parameter controlling the sharpness of the distribution. The final loss is then a composition of cross entropy loss and supervised contrastive loss and is defined as follows:

$$\mathcal{L} = \lambda_{\text{CE}} \mathcal{L}_{\text{CE}}(D_i, D^{\text{syn}}, M_i) + \lambda_{\text{SCL}} \mathcal{L}_{\text{SCL}}(D_i, D^{\text{syn}}, M_i)$$

Here, $\mathcal{L}_{\text{CE}}$ is the standard cross-entropy loss computed on the union of the local dataset D_i and the synthetic dataset D^{syn}. The term $\mathcal{L}_{\text{SCL}}$ denotes the supervised contrastive loss. The scalars λ_{CE} and λ_{SCL} control the strength of cross entropy loss and supervised contrastive loss, respectively. Detailed algorithm can be found in appendix A.

VI. Experimental Setup

We built our solutions on top of the decentralizepy framework [33]. We used decentralizepy's implementation of Decentralized Parallel Stochastic Gradient Descent (DPSGD) as the baseline. Each node locally trains a LeNet model using SGD, with learning rate of 0.01. The communication protocol between the members is TCP. The network is a static, regular graph with 16 nodes of either degree 3 or 5. Churn is simulated via a predefined schedule where 3, 5, or 8 members leave before convergence. The schedule can be seen in table II. Churn is further modeled as a Bernoulli process, where each node has a probability of 80%, 90%, or 95% of leaving the system and a 50% probability of returning. Each method is evaluated in both settings.

Data synthesis is run for 15,000 iterations using the Adam optimizer, with a learning rate of 0.01 and β values set to 0.5 and 0.9, which control the first and second moments of the gradients, respectively. We evaluate on MNIST and CIFAR-10 datasts. MNIST consists of 60,000 grayscale images of size 28×28 across 10 classes, while CIFAR-10 contains 50,000 RGB images of size 32×32, also spanning 10 classes. training runs for 500 iterations on MNIST and 3,000 on CIFAR-10. For synthetic anchors, static weights are set as $\lambda_{\text{CE}} = 1$ and $\lambda_{\text{SCL}} = 2$. We also experiment with dynamic weights, defined as:

$$\lambda_{\text{CE}} = \frac{\text{current iteration}}{\text{total iterations}}, \quad \lambda_{\text{SCL}} = 2 \cdot (1 - \lambda_{\text{CE}})$$

Data is distributed using Dirichlet partitioning with a concentration parameter $\alpha = 0.01$. The same random seed was reused to avoid regenerating synthetic data. Data distribution for each node can be found in appendix B.

Scenario	Churn Schedule (node : leaving iteration)
3 churn	`3:6, 7:11, 12:22`
5 churn	`2:5, 4:10, 6:13, 10:25, 14:30`
8 churn	`0:28, 1:3, 3:8, 6:13, 8:18, 12:19, 14:23, 15:20`

TABLE II: Churn schedule for permanent churn experiments.

VII. Results and Evaluation

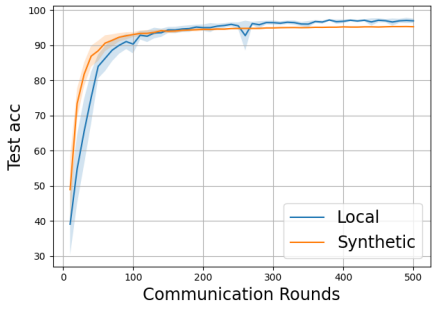
A. Dataset Condensation

The quality of synthetic data plays a significant role in our approach, so it is important to first demonstrate that it is of sufficient quality. In Figure 3, we compare DPSGD trained solely on local data versus DPSGD trained only on synthetic data generated from each node's local data. The results show that both models achieve similar final accuracy, with the synthetic-data-trained models exhibiting significantly less noise. This validates the effectiveness of the synthetic data.

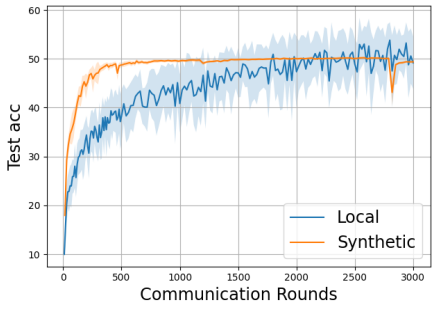
In our experiments, we use DPSGD augmented with synthetic data under no churn as the baseline. A comparison of final accuracies between augmented and unaugmented DPSGD in the no-churn setting is shown in Table III.

B. Experiments with MNIST

In Table IV, we present the final accuracies for training on the MNIST dataset, evaluated against the baseline. One can observe that data augmentation improves accuracy in 11 out of 12 cases. It increases accuracy from 90.93% to 95.52% when 8 members leave in the 3-degree topology, and from



(a) MNIST



(b) CIFAR

Fig. 3: Test accuracy of model trained on synthetic vs local data.

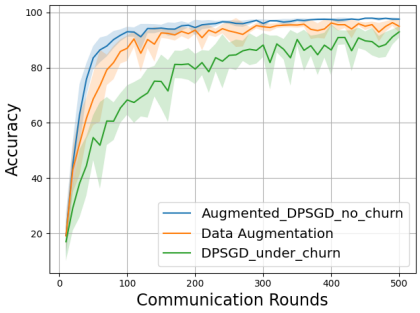


Fig. 4: Data augmentation in 3 degree 3 members leaving scenario.

DPSGD	CIFAR10		MNIST	
	Degree 3	Degree 5	Degree 3	Degree 5
Not Aug.	49.2±5.4%	54.8±4.0%	94.8±2.5%	96.9±0.6%
Aug.	55.9±2.8%	56.8±4.4%	97.6±0.4%	98.2±0.2%

TABLE III: Test accuracy of unaugmented(not Aug.) and augmented(Aug.) DPSGD under no churn on CIFAR-10 and MNIST datasets.

90.67% to 97.40% when the probability of being active is 80% in the 5-degree topology. Data augmentation also improves convergence, as illustrated in Figure 4. It also reduces variance in 8 out of 12 cases. When testing on missing members' data (MMD), data augmentation provides comparable results, most notably when 8 members leave in the 3-degree topology, raising the accuracy from 87.39% to 94.06%, as seen in Table IV.

However, there are several scenarios where the system is not heavily impacted by churn, especially in the 5-degree topology. This motivates us to move to the CIFAR10 dataset and limit MNIST testing to data augmentation. Plots for the full training runs can be found in Appendix C.

C. Experiments with CIFAR10

In Table V, we present a comparison of models trained on the CIFAR10 dataset using data augmentation and synthetic anchors under permanent churn. Our results show that data augmentation consistently improves test set and MMD accuracy compared to unaugmented DPSGD under churn. The improvements range from 2.47% to 5.32% in test accuracy and from 0.77% to 7.91% in MMD accuracy.

While synthetic anchors are designed to build upon data augmentation, they fail to outperform it in 4 out of 6 scenarios in test accuracy and in 5 out of 6 scenarios in MMD accuracy. Synthetic anchors improve accuracy from -0.98% to 5.17% in test accuracy and from -1.08% to 5.88% in MMD accuracy compared to DPSGD under churn. One area where synthetic anchors outperform data augmentation is stability, with lower standard deviation in 9 out of 12 cases across both test and MMD accuracy. However, Table V does not reveal the full picture. As shown in Figure 5, which illustrates the 3-degree, 3-members-leaving scenario, synthetic anchors converge much faster compared to data augmentation. This trend is consistent across all churn settings, as further illustrated in plots provided in Appendix C.

This observation motivated the introduction of dynamic weights for the cross-entropy and supervised contrastive losses composing synthetic anchors. We define the weights as linear functions of the current iteration and the total number of iterations, hence naming this approach Linear Synthetic Anchors. The benefits of this method are evident in Figure 5. These are further supported by Table V, where linear synthetic anchors outperform both data augmentation and synthetic anchors when 3 or 5 members leave, across both test and MMD accuracy, except for data augmentation when 3 members leave in the 5-degree topology. The most significant improvements occur in the 3-degree, 3-members-leaving case, where test accuracy increases from 41.36% to 50.82% and MMD accuracy from 39.26% to 46.01%. However, the benefits diminish when 8 members leave, with linear synthetic anchors outperforming synthetic anchors only in the 3-degree topology. Nevertheless, linear synthetic anchors exhibit the best overall stability, showing lower standard deviation than both data augmentation and synthetic anchors in 21 out of 24 pairwise comparisons across test and MMD accuracy.

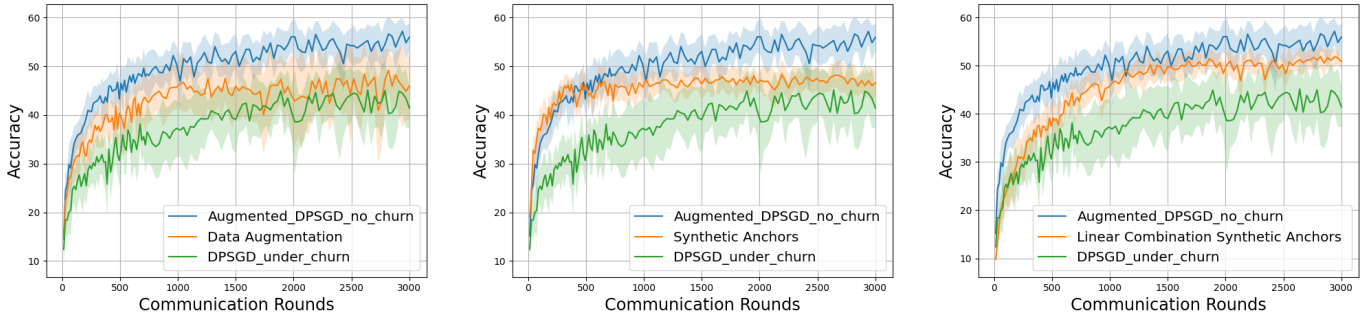
Lastly, Table VI presents the performance of data augmentation, synthetic anchors, and linear synthetic anchors under probabilistic churn. While the impact of churn is generally lower in this setting, all proposed methods outperform DPSGD under churn in every scenario. Moreover, linear synthetic anchors achieve the highest test accuracy in 5 out of 6 settings and the lowest standard deviation in 3 out of 6 cases. The most notable gain is observed when the probability of being active is 95%, with an improvement of 2.02%. Plots of training runs corresponding to accuracies reported in Table V and Table VI can be found in Appendix C.

VIII. Discussion

The results clearly show that data augmentation helps mitigate the impact of churn and preserve member contributions. Although accuracy improves on missing members' data, it remains lower than on the regular test set. This suggests current

Setting	Degree 3 Acc. (%)	Degree 3 MMD Acc. (%)	Degree 3 Δ DPSGD	Degree 5 Acc. (%)	Degree 5 MMD Acc. (%)	Degree 5 Δ DPSGD
DPSGD	97.56 ± 0.42	–	–	98.23 ± 0.21	–	–
DPSGD - 3 members leaving	92.99 ± 2.05	93.84 ± 3.41	-4.57	95.93 ± 0.65	92.95 ± 1.16	-2.30
Data Augmentation	94.99 ± 2.24	95.15 ± 2.70	-2.57	97.19 ± 0.67	94.90 ± 1.06	-1.04
DPSGD - 5 members leaving	96.81 ± 0.55	95.07 ± 1.38	-0.75	96.46 ± 1.37	95.53 ± 1.45	-1.77
Data Augmentation	95.44 ± 5.78	95.12 ± 3.38	-2.12	96.72 ± 1.91	96.34 ± 0.98	-1.51
DPSGD - 8 members leaving	90.93 ± 5.04	87.39 ± 8.47	-6.63	95.20 ± 1.43	93.07 ± 2.42	-3.03
Data Augmentation	95.52 ± 2.45	94.06 ± 3.81	-2.04	96.76 ± 0.70	96.15 ± 0.84	-1.47
DPSGD- p_{active}=0.80	93.10 ± 4.36	–	-4.46	90.67 ± 9.06	–	-7.56
Data Augmentation	94.96 ± 3.60	–	-2.60	97.40 ± 0.50	–	-0.83
DPSGD- p_{active}=0.90	95.41 ± 2.25	–	-2.15	96.14 ± 3.01	–	-2.09
Data Augmentation	95.56 ± 2.91	–	-2.00	97.79 ± 0.51	–	-0.44
DPSGD- p_{active}=0.95	96.90 ± 1.02	–	-0.66	97.61 ± 0.43	–	-0.62
Data Augmentation	97.49 ± 0.54	–	-0.07	98.07 ± 0.32	–	-0.16

TABLE IV: Final test and missing members data (MMD) accuracies for MNIST dataset under permanent and probabilistic churn. Δ DPSGD denotes the difference in test accuracy from the DPSGD baseline.



(a) Data Augmentation (b) Synthetic Anchors (c) Linear Synthetic Anchors

Fig. 5: Accuracy throughout training of Data Augmentation, Synthetic Anchors, and Linear Synthetic Anchors under 3-degree, 3-members leaving scenario.

Setting	Degree 3 Test Acc. (%)	Degree 3 MMD Acc. (%)	Degree 3 Δ DPSGD	Degree 5 Test Acc. (%)	Degree 5 MMD Acc. (%)	Degree 5 Δ DPSGD
DPSGD Augmented	55.94 ± 2.76	–	–	56.84 ± 4.39	–	–
DPSGD - 3 members leaving	41.43 ± 4.20	39.26 ± 7.76	-14.51	46.89 ± 4.73	45.31 ± 4.53	-9.94
Data Augmentation	45.95 ± 6.94	44.08 ± 10.75	-10.00	50.85 ± 3.43	53.22 ± 4.80	-5.99
Synthetic Anchors	46.60 ± 2.61	41.95 ± 5.38	-9.34	48.45 ± 3.16	44.80 ± 2.88	-8.38
Linear Synthetic Anchors	50.82 ± 1.75	46.01 ± 4.27	-5.12	50.68 ± 2.88	49.72 ± 1.74	-6.15
DPSGD - 5 members leaving	40.37 ± 9.18	39.64 ± 7.23	-15.57	46.19 ± 4.18	33.08 ± 9.43	-10.65
Data Augmentation	45.69 ± 10.36	40.41 ± 12.12	-10.25	48.66 ± 5.37	38.11 ± 7.98	-8.17
Synthetic Anchors	42.80 ± 9.64	40.35 ± 12.35	-21.31	48.95 ± 2.26	36.98 ± 6.14	-7.88
Linear Synthetic Anchors	47.29 ± 10.90	41.82 ± 11.32	-8.65	50.79 ± 2.08	40.97 ± 5.51	-6.05
DPSGD - 8 members leaving	44.20 ± 6.67	32.91 ± 8.03	-11.74	44.77 ± 6.15	32.60 ± 6.34	-12.07
Data Augmentation	47.32 ± 5.92	37.76 ± 6.84	-9.52	49.72 ± 5.56	37.34 ± 8.62	-7.11
Synthetic Anchors	43.22 ± 4.17	31.86 ± 4.72	-12.72	47.72 ± 3.85	35.69 ± 4.01	-9.12
Linear Synthetic Anchors	46.46 ± 4.18	35.33 ± 5.07	-9.48	45.89 ± 3.68	35.53 ± 3.99	-10.95

TABLE V: Final test and **missing members data(MMD)** accuracies for CIFAR10 dataset under permanent churn. Δ from DPSGD denotes change in test accuracy from the baseline.

Setting	Degree 3	Degree 5
DPSGD Augmented	55.94 ± 2.76	56.84 ± 4.39
DPSGD - p_{active}=0.80	52.07 ± 3.27	54.67 ± 3.12
Data Augmentation	52.32 ± 4.74	55.83 ± 2.14
Synthetic Anchors	51.86 ± 4.25	55.14 ± 2.65
Linear Synthetic Anchors	53.79 ± 2.71	54.68 ± 2.72
DPSGD - p_{active}=0.90	50.64 ± 4.13	52.33 ± 2.52
Data Augmentation	51.41 ± 5.26	53.26 ± 2.38
Synthetic Anchors	49.70 ± 2.61	54.67 ± 1.56
Linear Synthetic Anchors	52.63 ± 3.56	55.41 ± 1.62
DPSGD - p_{active}=0.95	47.83 ± 4.66	50.02 ± 3.20
Data Augmentation	48.86 ± 3.57	51.82 ± 3.12
Synthetic Anchors	46.26 ± 3.63	51.29 ± 2.82
Linear Synthetic Anchors	49.42 ± 2.95	52.04 ± 2.77

TABLE VI: Final test accuracies for CIFAR10 dataset under probabilistic churn. Δ from DPSGD denotes change in test accuracy from the baseline.

methods cannot fully recover missing members' contributions. It would be interesting to investigate whether alternative data condensation methods can better preserve client contributions and further mitigate the effects of churn. Data augmentation also improves the performance of the baseline model showing promise in enhancing performances of models.

The results also show that while synthetic anchors with static weights offer much faster convergence compared to plain data augmentation, they do not yield improved final accuracy. Faster convergence likely comes from the higher information density of synthetic data. However, the lack of improvement in final performance may be due to the total information contained in the synthetic data being less than the information transmitted and stored in the local data. Therefore, over-reliance on it prevents fully utilizing all available information. In contrast, synthetic anchors with dynamic weights show the most promise. Relying heavily on synthetic data to learn quickly at the beginning , then shifting toward local data for fine-tuning, appears to be a more effective strategy. It would be interesting to explore different dynamic weighting strategies between the cross-entropy loss and supervised contrastive loss composing synthetic anchors. The performance drop observed in the dynamic approach when 8 members leave, compared to the static approach, could be explained by a misalignment between the linear weight transition and the actual churn schedule.

One can also notice that models consistently perform better in denser graph topologies, likely due to increased information flow compared to sparser counterparts, as well as more synthetic data being shared.

The communication overhead is minimal since synthetic datasets are small and sent only once. However, the computational overhead remains substantial in single-model training scenarios, potentially limiting practical use. Exploring computationally lighter data synthesis methods, shorter training runs, or replacing dataset condensation with simpler augmentations (e.g., transformations of local data) would be worthwhile. However, such alternatives are likely to retain less information about the departing members. Another potential extension would be to propagate synthetic data further through the network, although this could pose scalability challenges in larger topologies.

Current results are limited by network size, dataset complexity, and model architecture. Additionally, probabilistic churn is modeled in a simplified manner. It would be valuable to explore how data augmentation performs in more complex settings. Lastly, permanent leaving was modeled using a single permanent churn schedule. This was done to obtain reliable data for one representative scenario, given computational and time constraints. Future work could explore how data augmentation and synthetic anchors behave in larger networks and across a variety of churn scenarios.

IX. Conclusion

We empirically analyzed how data augmentation and synthetic anchors help mitigate churn and preserve member contributions. By extensively testing on the MNIST and CIFAR-10 datasets and using different network topologies, we found that these methods mitigate the impact of churn and help preserve member contributions. Our findings indicate that data augmentation enhances model performance under churn for both datasets. On MNIST, it leads to test accuracy improvements of up to 6.73%, from 90.67% to 97.40%. Furthermore, missing members data accuracy improves as much as 6.67%. For CIFAR-10, data augmentation increases test accuracy by up to 5.32% and improves missing members' data accuracy by up to 7.89%. Moreover, it contributes to greater model stability, as evidenced by reduced standard deviation.

While synthetic anchors with static weights under perform compared to data augmentation, synthetic anchors with dynamic weights show the most promise. Dynamic synthetic anchors outperform both data augmentation and static anchors in test and missing members' data accuracy when 3 or 5 members leave, with the exception of the 3-member case in the 5-degree topology, where data augmentation remains superior. Their improvement go up to 9.29% on test accuracy and 6.75% on MMD accuracy. Although their advantage diminishes when 8 members leave, linear synthetic anchors demonstrate the highest overall stability, achieving lower standard deviations than both data augmentation and static anchors in 21 out of 24 pairwise comparisons across test and missing members' data accuracy.

Nevertheless, we see many directions for future research such as how data augmentation and synthetic anchors perform in more complex settings, how they interact with different data condensation methods, and how different balances between loss functions affect performance.

X. Responsible Research

Reproducibility. To ensure reproducibility, we explicitly define all parameter values used in our experimental setup. The

complete codebase is publicly available in a TU Delft GitLab repository. The repository includes all scripts and configuration files necessary to reproduce the experiments. A README file is provided to ease navigation and usage.

Integrity. This research adheres to the principles outlined in the Netherlands Code of Conduct for Research Integrity [34], specifically Chapter 3 on standards for good research practices.

We address a relevant scholarly question and transparently disclose funding sources and stakeholders, aligning with the principles for research design. The empirical method used is well-suited to our research objectives, and the results are presented accurately and without fabrication, in accordance with the standards for research conduct. All contributors have been clearly acknowledged, and the implications of our findings are discussed transparently, thereby adhering to the standards for reporting results.

Ethics. All experiments were conducted on publicly available datasets; therefore, there are no ethical concerns related to data collection. However, there are potential privacy risks associated with sharing synthetic data, as highlighted by Nicholas Carlini et al. [29]. These risks should be further investigated.

Use of AI. Large language models (ChatGPT) were used to correct grammar and sometimes styling of the text. Furthermore, they were used as help in the implementation of the experiments and algorithms.

References

[1] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu. *Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent*. 2017.

[2] I. Hegedűs, G. Danner, and M. Jelasity. "Gossip Learning as a Decentralized Alternative to Federated Learning". In: *Distributed Applications and Interoperable Systems*. Cham: Springer International Publishing, 2019, pp. 74–90.

[3] I. Hegedűs, G. Danner, and M. Jelasity. "Gossip Learning as a Decentralized Alternative to Federated Learning". In: *Lecture Notes in Computer Science*. Vol. LNCS-11534. Distributed Applications and Interoperable Systems. Kongens Lyngby, Denmark: Springer International Publishing, June 2019, pp. 74–90.

[4] A. Mălan, J. Decouchant, T. Guzella, and L. Chen. "CCBNet: Confidential Collaborative Bayesian Networks Inference". In: *arXiv preprint arXiv:2405.15055* (2024).

[5] A. Shankar, L. Y. Chen, J. Decouchant, D. Gkorou, and R. Hai. "Share Your Secrets for Privacy! Confidential Forecasting with Vertical Federated Learning". In: *arXiv preprint arXiv:2405.20761* (2024).

[6] European Union. *General Data Protection Regulation (GDPR) – Legal Text*. https://gdpr-info.eu/. Accessed: 2025-06-02. European Union, 2018.

[7] J. Wang, B. Cao, P. S. Yu, L. Sun, W. Bao, and X. Zhu. *Deep Learning Towards Mobile Applications*. 2018. arXiv: 1809.03559 [cs.LG].

[8] R. Ormándi, I. Hegedűs, and M. Jelasity. "Gossip learning with linear models on fully distributed data". In: *Concurrency and Computation: Practice and Experience* 25.4 (May 2012), pp. 556–571.

[9] Y. Lu and C. D. Sa. "Decentralized Learning: Theoretical Optimality and Practical Improvements". In: *J. Mach. Learn. Res.* 24 (2023), 93:1–93:62.

[10] C.-Y. Huang, K. Srinivas, X. Zhang, and X. Li. "Overcoming Data and Model Heterogeneities in Decentralized Federated Learning via Synthetic Anchors". In: *arXiv preprint arXiv:2405.11525* (2024).

[11] C. Li, G. Li, and P. K. Varshney. "Decentralized Federated Learning via Mutual Knowledge Transfer". In: *CoRR* abs/2012.13063 (2020).

[12] B. Cox, L. Y. Chen, and J. Decouchant. "Aergia: leveraging heterogeneity in federated learning systems". In: *Proceedings of the 23rd ACM/IFIP International Middleware Conference*. 2022, pp. 107–120.

[13] T. Liu, Y. Cui, X. Hu, Y. Xu, and B. Liu. "On the Convergence of Gossip Learning in the Presence of Node Inaccessibility". In: *ICC 2024 - IEEE International Conference on Communications*. 2024, pp. 4197–4202.

[14] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. *Handling Churn in a DHT*. Tech. rep. UCB/CSD-03-1299. Dec. 2003.

[15] P. B. Godfrey, S. Shenker, and I. Stoica. *Minimizing Churn in Distributed Systems*. Tech. rep. UCB/EECS-2006-25. Mar. 2006.

[16] B. Cox, J. Galjaard, A. Shankar, J. Decouchant, and L. Y. Chen. "Parameterizing Federated Continual Learning for Reproducible Research". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2023, pp. 478–486.

[17] B. Zhao and H. Bilen. *Dataset Condensation with Distribution Matching*. 2022.

[18] B. Zhao, K. R. Mopuri, and H. Bilen. "Dataset Condensation with Gradient Matching". In: *CoRR* abs/2006.05929 (2020).

[19] M. A. Dinani, A. Di Maio, and G. Rizzo. "Gossip Learning in Edge-Retentive Time-Varying Random Graphs with Node Churn". In: *2024 IEEE Annual Congress on Artificial Intelligence of Things (AIoT)*. 2024, pp. 53–59.

[20] H. Di, H. Ye, X. Chang, G. Dai, and I. Tsang. "Double Stochasticity Gazes Faster: Snap-Shot Decentralized Stochastic Gradient Tracking Methods". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 10765–10791.

[21] A. B. de Luca, G. Zhang, X. Chen, and Y. Yu. *Mitigating Data Heterogeneity in Federated Learning with Data Augmentation*. 2022.

[22] S. Sha and Y. Sun. *DCFL: Non-IID awareness Data Condensation aided Federated Learning*. 2023. arXiv: 2312.14219 [cs.LG].

[23] R. Song, D. Liu, D. Z. Chen, A. Festag, C. Trinitis, M. Schulz, and A. Knoll. *Federated Learning via Decentralized Dataset Distillation in Resource-Constrained Edge Environments*. 2023. arXiv: 2208.11311 [cs.LG].

[24] D. Stutzbach and R. Rejaie. "Understanding churn in peer-to-peer networks". In: *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*. IMC '06. New York, NY, USA: ACM, 2006, pp. 189–202.

[25] Á. Berta, V. Bilicki, and M. Jelasity. "Defining and understanding smartphone churn over the internet: A measurement study". In: *14-th IEEE International Conference on Peer-to-Peer Computing*. 2014, pp. 1–5.

[26] S. Xu, X. Ke, X. Su, S. Li, H. Wu, S. Zhong, and F. Xu. *Privacy-Preserving Federated Learning via Dataset Distillation*. 2024. arXiv: 2410.19548 [cs.LG].

[27] A. Dhasade, Y. Ding, S. Guo, A.-m. Kermarrec, M. D. Vos, and L. Wu. *QuickDrop: Efficient Federated Unlearning by Integrated Dataset Distillation*. 2024.

[28] T. Dong, B. Zhao, and L. Lyu. *Privacy for Free: How does Dataset Condensation Help Privacy?* 2022. arXiv: 2206.00240 [cs.CR].

[29] N. Carlini, V. Feldman, and M. Nasr. *No Free Lunch in "Privacy for Free: How does Dataset Condensation Help Privacy"*. 2022. arXiv: 2209.14987 [cs.LG].

[30] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. "A Kernel Two-Sample Test". In: *The Journal of Machine Learning Research* 13 (Mar. 2012), pp. 723–773.

[31] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[32] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. "Supervised Contrastive Learning". In: *CoRR* abs/2004.11362 (2020).

[33] A. Dhasade, A.-M. Kermarrec, R. Pires, R. Sharma, and M. Vujasinovic. "Decentralized Learning Made Easy with DecentralizePy". In: *Proceedings of the 3rd Workshop on Machine Learning and Systems*. EuroMLSys '23. Rome, Italy: Association for Computing Machinery, 2023, pp. 34–41.

[34] Association of Universities in the Netherlands (VSNU), KNAW, NFU, NWO, and TO2 Federation. *Netherlands Code of Conduct for Research Integrity*. Version: 2018. Published under CC-BY 4.0 license. 2018.

APPENDIX A
ALGORITHM PSEUDOCODE

Algorithm 1 Data Augmentation for client i

for all $j \in \mathcal{N}(C_i)$ **do**
    trainset $\leftarrow$ trainset $\cup$ ReceiveSyntheticSet(C_j)
end for
for $t = 1, \ldots, T$ **do**
    Receive models $\{M_j \mid j \in \mathcal{N}(C_i)\}$
    $M_i \leftarrow$ Average$(M_i, \{M_j\})$
    $\mathcal{L}_{\text{CE}} \leftarrow$ CrossEntropyLoss$(trainset, M_i)$
    $M_i \leftarrow M_i - \eta \nabla_{M_i} \mathcal{L}_{\text{CE}}$
end for

Fig. 6: Pseudocode for the data augmentation executed by client i.

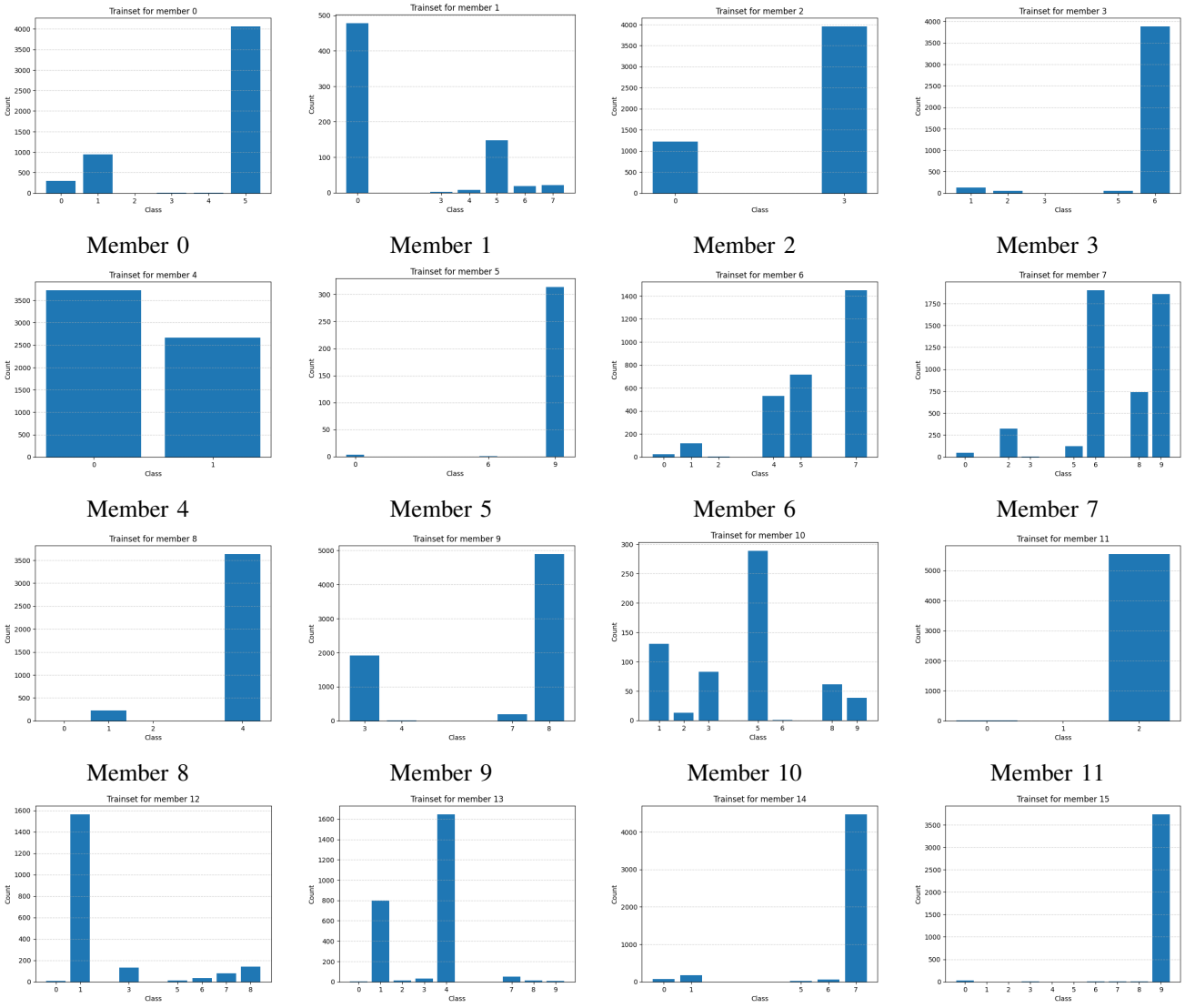
Algorithm 2 Synthetic Anchor for client i

for all $j \in \mathcal{N}(C_i)$ **do**
    $D^{\text{syn}} \leftarrow D^{\text{syn}} \cup$ ReceiveSyn(C_j)
end for
for $t = 1, \ldots, T$ **do**
    Receive models $\{M_j \mid j \in \mathcal{N}(C_i)\}$
    $M_i \leftarrow$ Average$(M_i, \{M_j\})$
    $\mathcal{L}_{\text{CE}} =$ CrossEntropyLoss$(D_i \cup D^{\text{syn}}, M_i)$
    // Detach synthetic data
    $\mathcal{L}_{\text{SCL}} =$ SupervisedContrastiveLoss$(D_i, D^{\text{syn}}, M_i)$
    $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{SCL}} \mathcal{L}_{\text{SCL}}$
    $M_i = M_i - \eta \nabla_{M_i} \mathcal{L}$
end for

Fig. 7: Pseudocode for the synthetic anchors executed by client i.

APPENDIX B
DATASET DISTRIBUTION

MNIST



Member 0 Member 1 Member 2 Member 3

Member 4 Member 5 Member 6 Member 7

Member 8 Member 9 Member 10 Member 11

Member 12 Member 13 Member 14 Member 15

Fig. 8: MNIST local data distribution per member.

APPENDIX C
EXPERIMENTS

CIFAR10

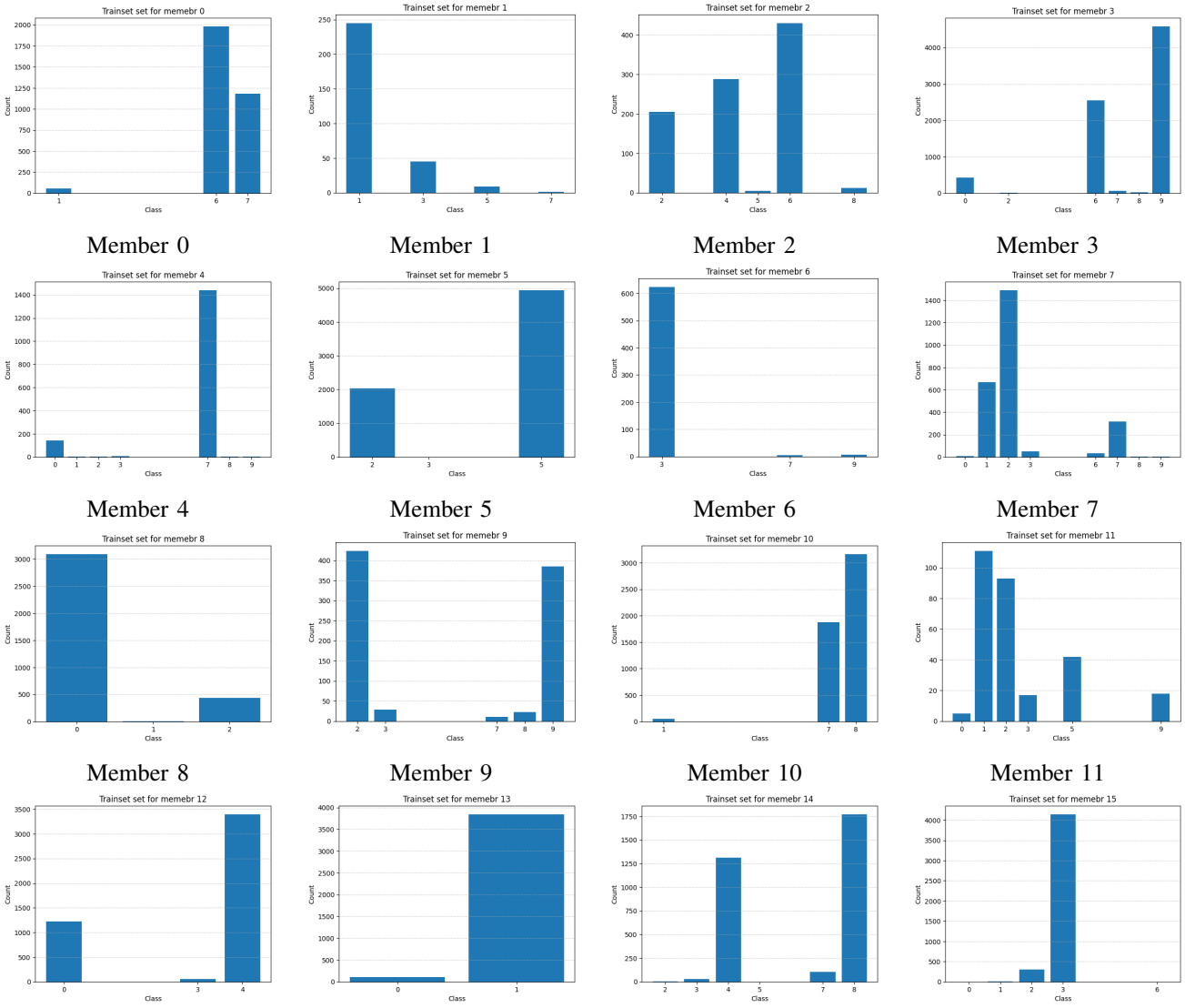


Fig. 9: CIFAR10 local data distribution per member.

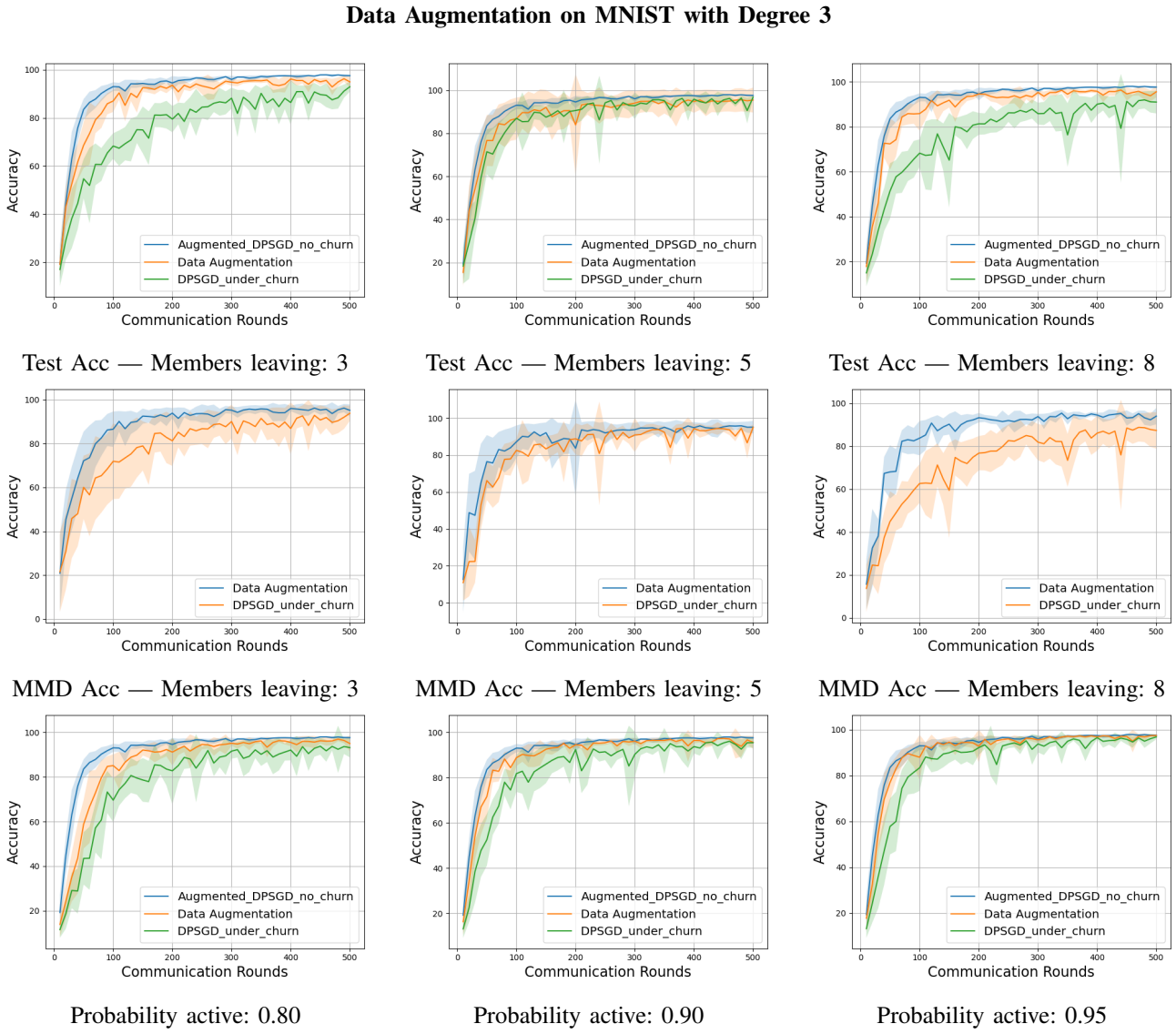


Fig. 10: Data augmentation performance on MNIST dataset in 3 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.

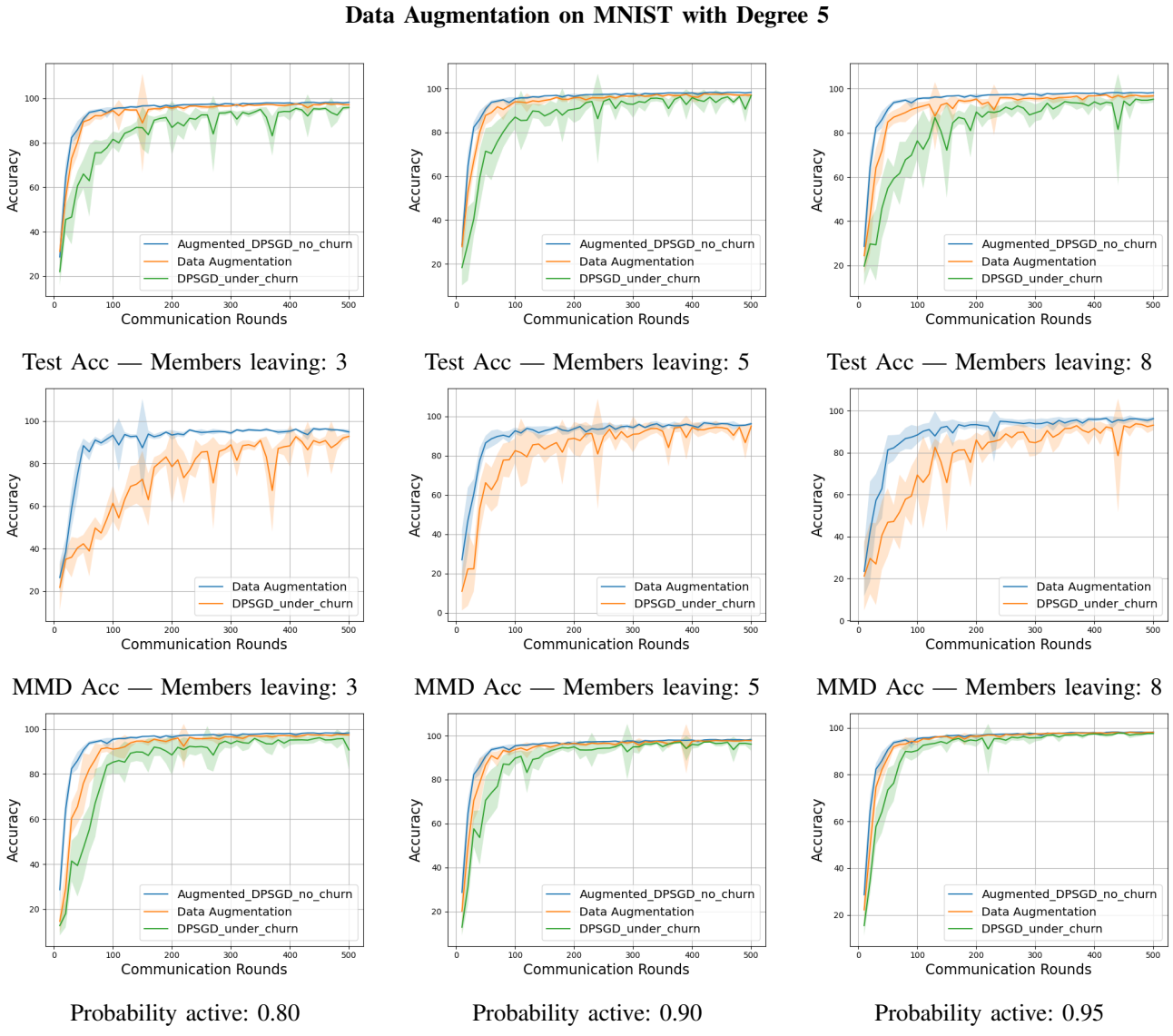


Fig. 11: Data augmentation performance on MNIST dataset in 5 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.

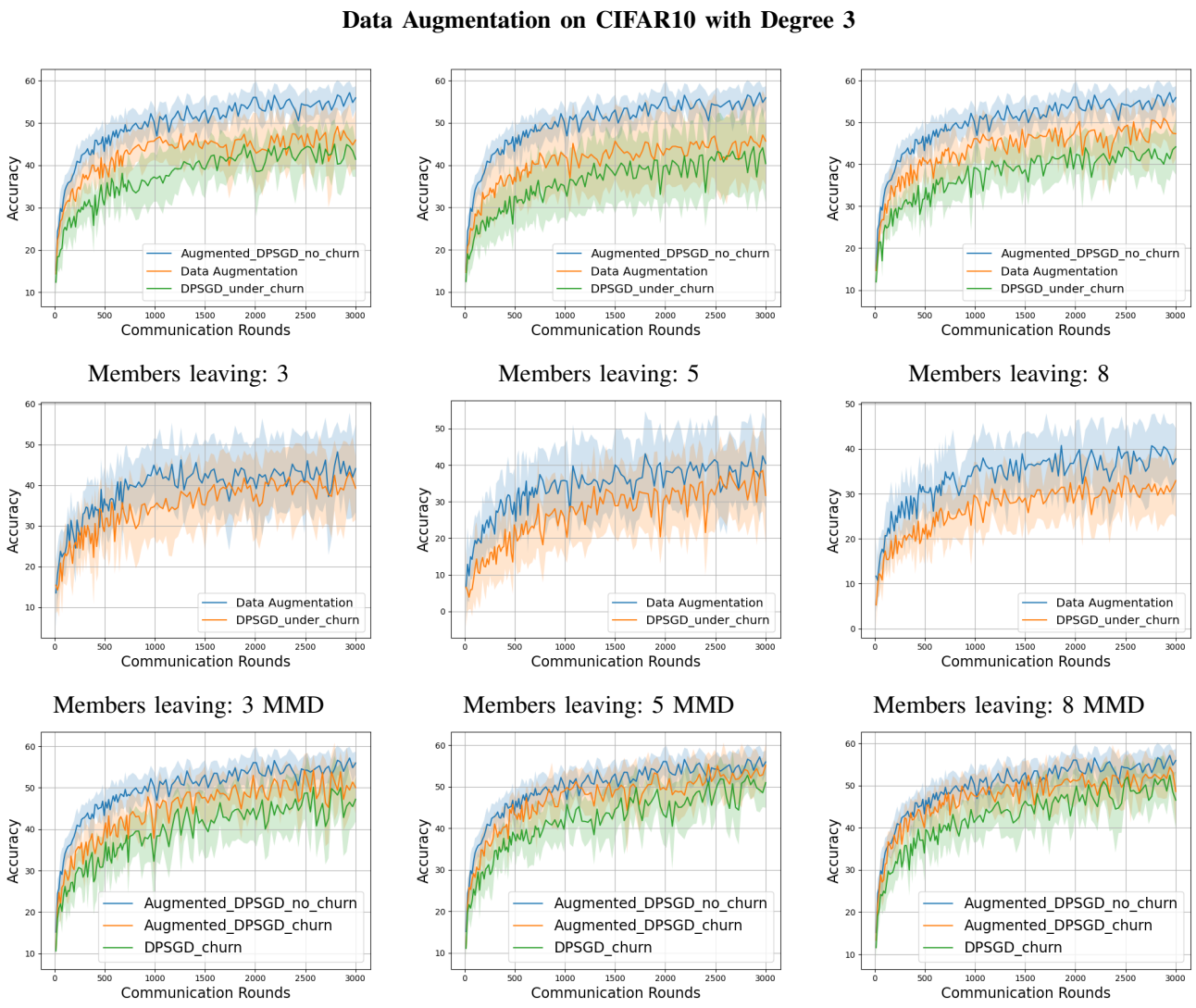
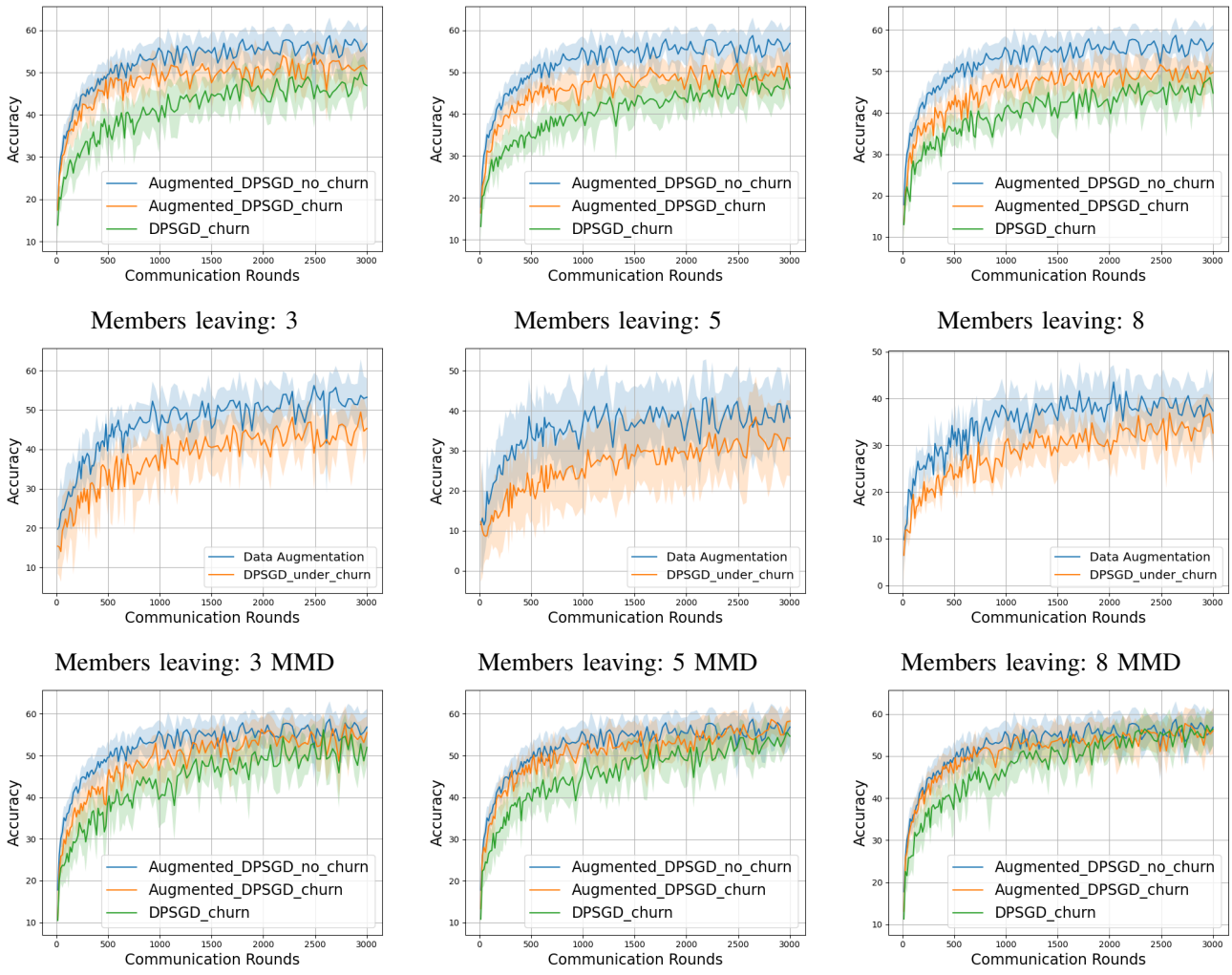


Fig. 12: Data augmentation performance on CIFAR10 dataset in 3 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.

Data Augmentation on CIFAR10 with Degree 5



Members leaving: 3 | Members leaving: 5 | Members leaving: 8

Members leaving: 3 MMD | Members leaving: 5 MMD | Members leaving: 8 MMD

Probability active: 0.80 | Probability active: 0.90 | Probability active: 0.95

Fig. 13: Data augmentation performance on CIFAR10 dataset in 5 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.

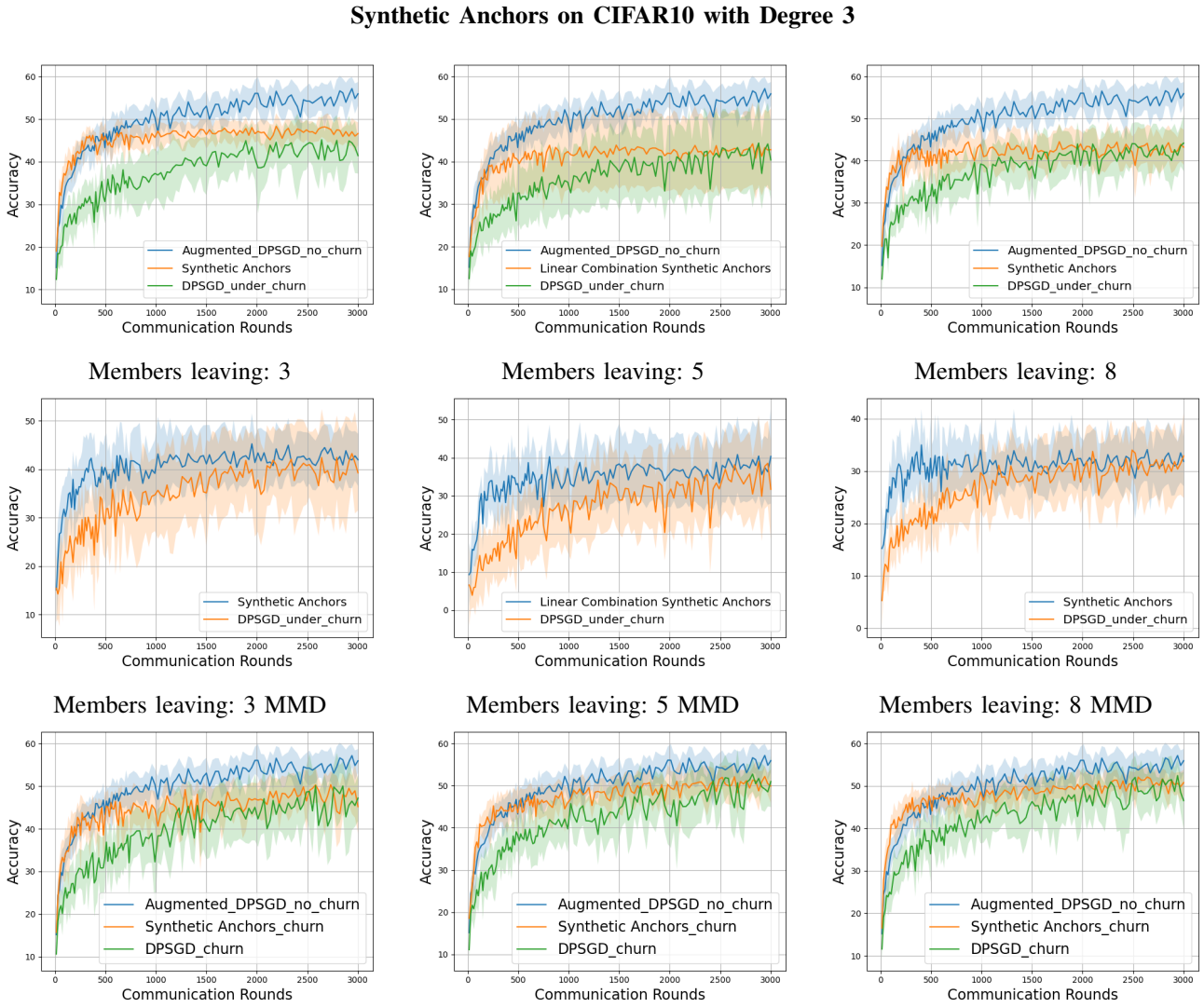


Fig. 14: Synthetic anchors performance on CIFAR10 dataset in 3 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.

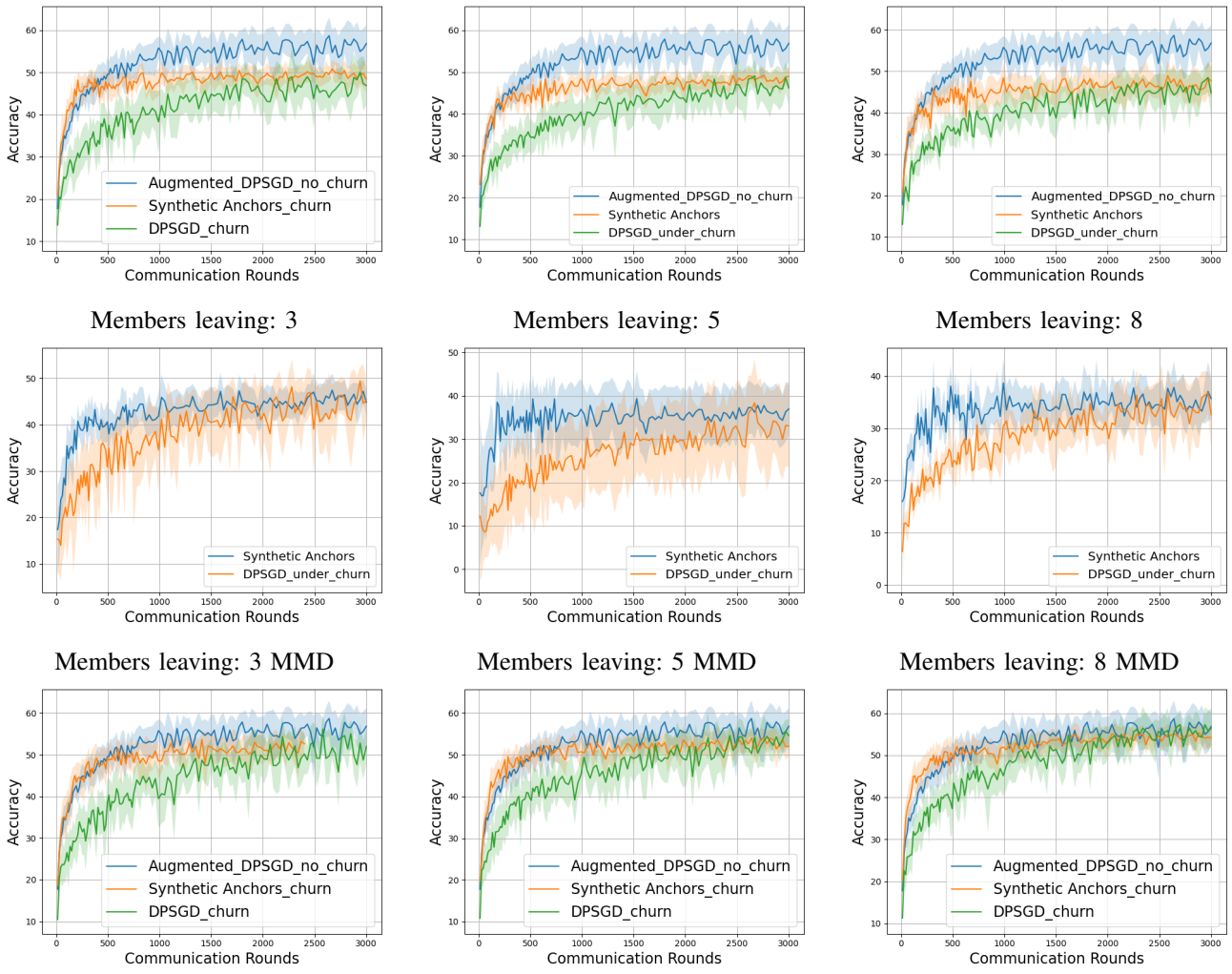
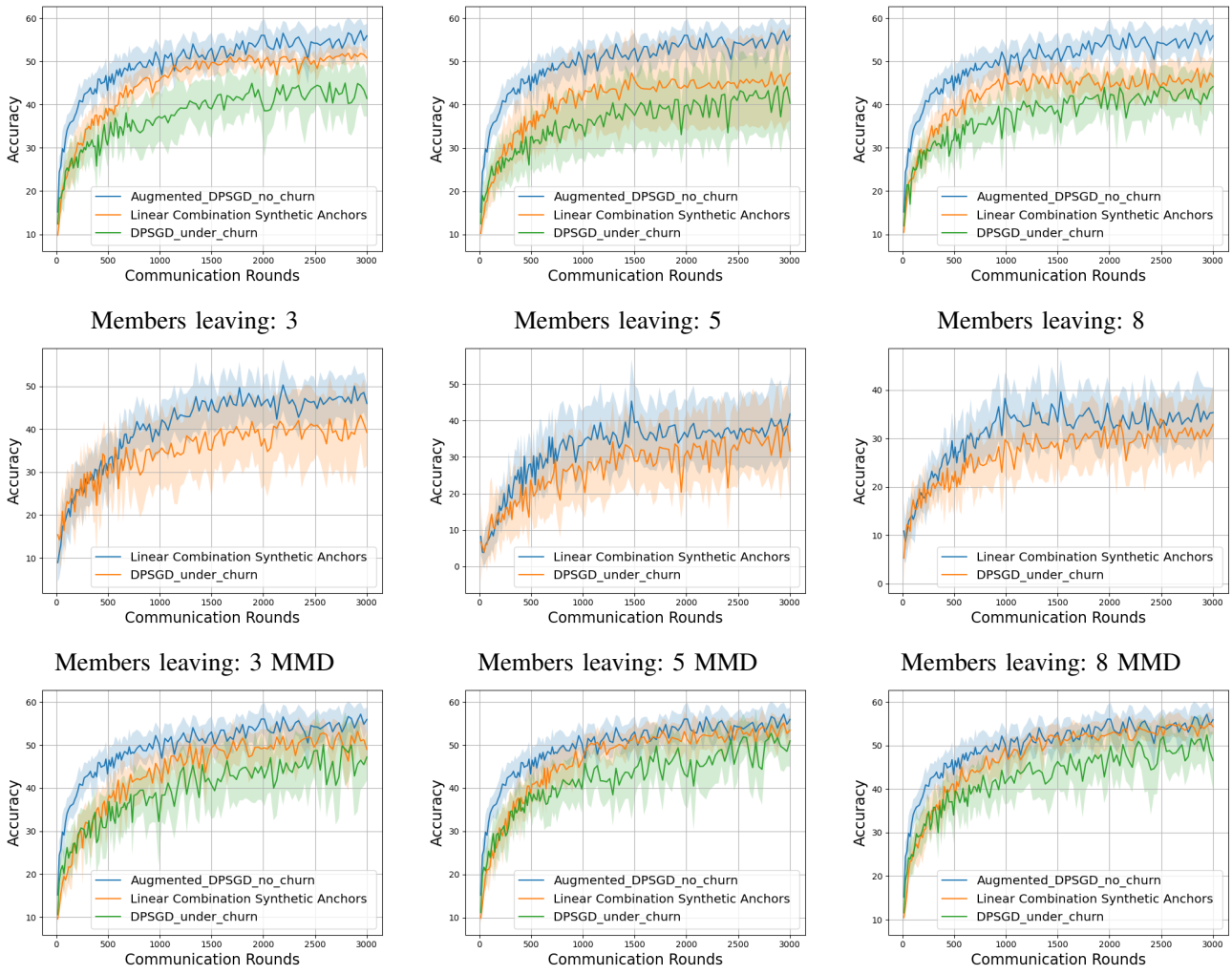


Fig. 15: Synthetic anchors performance on CIFAR10 dataset in 5 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.

Linear Anchors on CIFAR10 with Degree 3



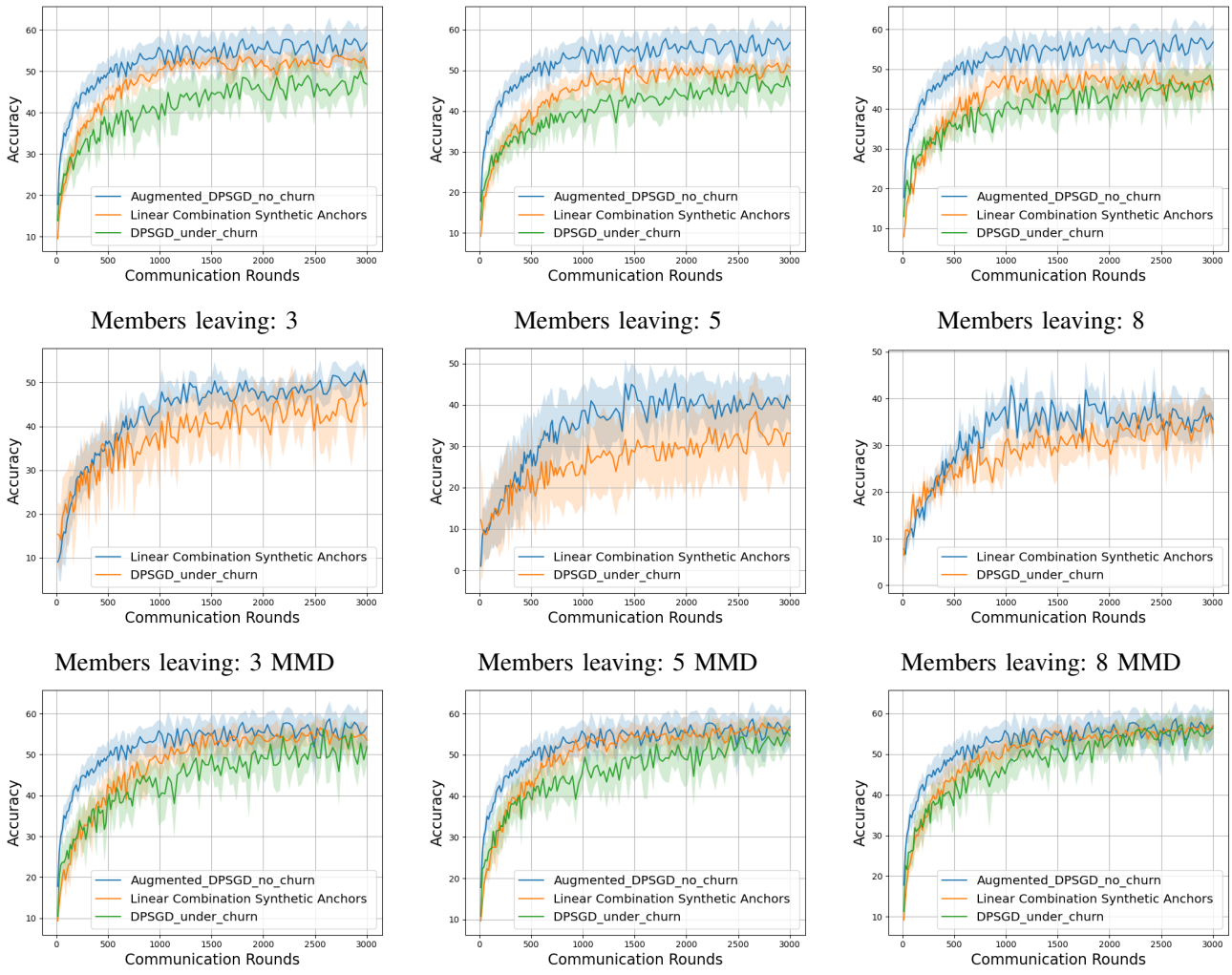
Members leaving: 3 | Members leaving: 5 | Members leaving: 8

Members leaving: 3 MMD | Members leaving: 5 MMD | Members leaving: 8 MMD

Probability active: 0.80 | Probability active: 0.90 | Probability active: 0.95

Fig. 16: Linear synthetic anchors performance on CIFAR10 dataset in 3 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.

Linear Anchors on CIFAR10 with Degree 5



Members leaving: 3 | Members leaving: 5 | Members leaving: 8

Members leaving: 3 MMD | Members leaving: 5 MMD | Members leaving: 8 MMD

Probability active: 0.80 | Probability active: 0.90 | Probability active: 0.95

Fig. 17: Linear synthetic anchors performance on CIFAR10 dataset in 3 degree topology. In the first row we can see permanent churn test accuracy, in the second row MMD accuracy, and in the third test accuracy on probabilistic churn.