DICE: DATA INFLUENCE CASCADES IN DECENTRALIZED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Decentralized learning offers a promising approach to crowdsource computational workloads across geographically distributed compute interconnected through peer-to-peer networks, accommodating the exponentially increasing compute demands in the era of large models. However, the absence of proper incentives in locally connected decentralized networks poses significant risks of free riding and malicious behaviors. Data influence, which ensures fair attribution of data source contributions, holds great potential for establishing effective incentive mechanisms. Despite the importance, little effort has been made to analyze data influence in decentralized scenarios, due to non-trivial challenges arising from the distributed nature and the localized connections inherent in decentralized networks. To overcome this fundamental challenge, we propose DICE, the first framework to systematically define and estimate **D**ata **I**nfluence **C**ascad**E**s in decentralized environments. DICE establishes a new perspective on influence measurement, seamlessly integrating self-level and community-level contributions to capture how data influence cascades implicitly through networks via communication. Theoretically, the framework derives tractable approximations of influence cascades over arbitrary neighbor hops, uncovering for the first time that data influence in decentralized learning is shaped by a synergistic interplay of data, communication topology, and the curvature information of optimization landscapes. By bridging theoretical insights with practical applications, DICE lays the foundations for incentivized decentralized learning, including selecting suitable collaborators and identifying malicious behaviors. We envision DICE will catalyze the development of scalable, autonomous, and reciprocal decentralized learning ecosystems.

1 INTRODUCTION

Machine learning has made remarkable progress in the past a few years, driven primarily by large language models (LLMs) such as GPT-4 (Achiam et al., 2023), Llama 3 (Dubey et al., 2024), Claude 3 (Anthropic, 2024) and Gemini 1.5 (Reid et al., 2024), which have surpassed human performance on several key benchmarks (Maslej et al., 2024). Compute scaling, highlighted by Ho et al. (2024) as central to these gains, is forecasted by Epoch AI to grow four to fivefold annually in cutting-edge models (Sevilla & Roldán, 2024). This exponential growth in computational demands necessitates substantial financial investments; for example, training OpenAI's GPT-4 requires approximately $78 million in compute costs (Maslej et al., 2024). Such exorbitant expenses are far beyond the reach of most individuals and academic institutions, resulting in a concentration of access to the most advanced frontier models within a small set of well-funded large corporations.

Currently, large-scale training and inference processes are primarily conducted within luxury data centers. Decentralized training, inspired by swarm intelligence (Bonabeau et al., 1999; Surowiecki, 2004; Mavrovouniotis et al., 2017), offers cost-efficient alternative by crowdsourcing computational workload with geographically distributed edge compute (Yuan et al., 2022; Borzunov et al., 2023). One notable example of decentralized computing's substantial computational potential is the Bitcoin system, whose instantaneous 16 GW power consumption (CCAF, 2023) triples the estimated 5 GW of the world's largest planned GPU cluster (Gardizy & Efrati, 2024).

Despite the advantage, the absence of a central authority complicating coordination among participants. Additionally, contributing to decentralized training involves non-negligible costs for edge users, prompting a natural question: why are edge users willing to participate in the decentralized training process? Game theory insights suggest that with appropriate incentives, even self-interested individuals can contribute to a system that achieves socially desirable outcomes (Nisan et al., 2007). To fully leverage the computational potential of decentralized swarms, incentive mechanisms akin to those in the Bitcoin system are essential to ensure fair compensation for contributors[1]. A critical component of these incentive mechanisms is the accurate quantification of individual contributions, or proof of work, which is necessary to foster "benign collaboration" among decentralized swarms without centralized governance. Therefore, the fundamental question arises:

Fundamental question: *How to quantify individual contributions in decentralized learning?*

Addressing this question establishes the foundation for promoting valuable contributions and discouraging free-riding and malicious behaviors, which are critical to a collaborative decentralized learning environment based on reciprocity (Gouldner, 1960; Fehr & Gächter, 2000).

With the ever-growing demand for high-quality data in modern machine learning (Hoffmann et al., 2022; Penedo et al., 2023; Li et al., 2023; Longpre et al., 2024; Villalobos et al., 2024), the influence of data plays an increasingly important role (Sorscher et al., 2022; Grosse et al., 2023; Xia et al., 2024). It ensures fair attribution of data source contributions, which is pivotal in establishing effective incentive mechanisms. Besides serving as an incentive, data influence has been extensively applied in various machine learning domains, including few-shot learning (Park et al., 2021), dataset pruning (Sorscher et al., 2022; Yang et al., 2023), distillation (Loo et al., 2023), fairness improving (Li & Liu, 2022), machine unlearning (Guo et al., 2020; Sekhari et al., 2021), explainability (Koh & Liang, 2017; Han et al., 2020; Grosse et al., 2023), as well as training-set attacks (Demontis et al., 2019; Jagielski et al., 2021) and defenses (Hammoudeh & Lowd, 2022).

Despite its importance, understanding and measuring data influence in fully decentralized environments remains largely untouched. Unlike centralized scenarios, where data influence is confined to a single model and can be statically analyzed after training, decentralized learning involves collaborative training among multiple participants connected via localized communication. In these settings, the influence of a single data instance impacts its local model and propagates to neighbors and even higher-order neighbors through iterative parameter exchange—a phenomenon we term the cascading effect. Unfortunately, existing data influence estimators tailored for centralized settings cannot characterize such dynamic transmission of data influence without significant modifications.

To address these challenges, we propose DICE (**D**ata **I**nfluence **C**ascad**E**), the first framework for systematically defining and estimating data influence in decentralized learning environments. We summarize our major contributions as follows:

- **Conceptual contributions**: DICE introduces the concept of a ground-truth data influence for decentralized learning, seamlessly integrating direct and indirect contributions to capture influence propagation across multiple hops during training.
- **Theoretical contributions**: Building on this foundation, we derive tractable approximations of ground-truth DICE for an arbitrary number of neighbor hops, establishing a foundational framework to systematically characterize the flow of influence across decentralized networks. These theoretical results uncover, for the first time, that data influence in decentralized learning extends beyond the data itself and the local model, as seen in centralized training. Instead, it is a joint product of three critical factors: the original data, the topological importance of the data keeper, and the curvature information of intermediate nodes mediating propagation. This dependency highlights the intricate interplay between data quality, network structure, and the optimization landscape, deepening our understanding of data influence and its pivotal role in shaping decentralized learning outcomes.

In our vision, we anticipate that the DICE framework will pave the way for novel incentive mechanism designs and the establishment of an economic framework for decentralized learning, including data and parameter markets. DICE also hold great potential to address critical challenges such as identifying new suitable collaborators, and detecting free-riders and malicious behaviors, under the constraints of limited local communication. We envision these impactful applications may contribute to the practical realization of scalable, autonomous, and reciprocal decentralized learning ecosystems.

[1]In this paper, the terms contributor, node, agent and participant interchangeably refer to an edge individual.

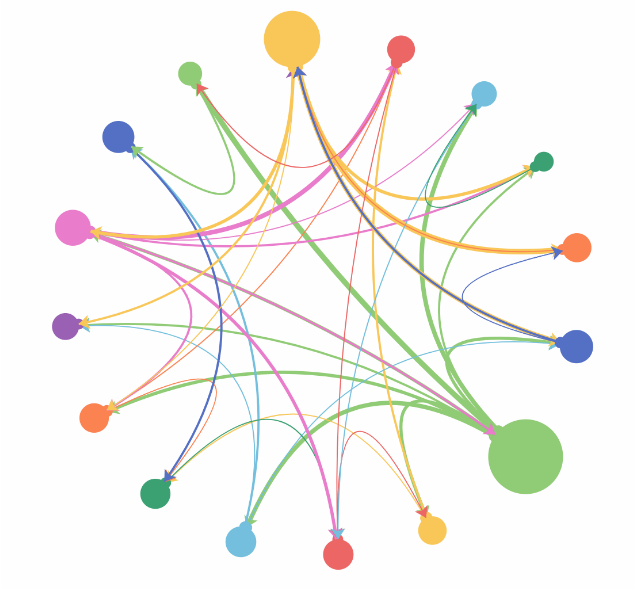

Figure 1: Visualization of influence cascades during decentralized traininggg with ResNet-18 on CIFAR-10 under a designed communication matrix (see Figure D.22). The thickness of edges represents the strength of communication links (i.e., weights in $\boldsymbol{W}$), while node sizes correspond to the one-hop DICE-E influence scores (see Proposition 1) computed for the same data batch across different participants. The differences in node sizes underscore how the identical data exerts varying influences depending on the node where it is stored, highlighting the critical role of topology in shaping data influence within decentralized learning.

2 Related work

Data Influence Estimation. Data influence quantifies the contribution of training data to model predictions (Chen et al., 2024; Ilyas et al., 2024). Data influence estimators are broadly categorized into static and dynamic approaches[2]. Specifically, static approaches include both retraining-based and one-point methods. Retraining-based methods, such as leave-one-out (Cook, 1977), Shapley value (Shapley, 1953), and Datamodels (Ilyas et al., 2022), assess data influence by retraining the model on (subsets of) the training data. These methods offer a conceptually straightforward computation of data influence and are grounded in strong theoretical foundations, but they are often computationally expensive due to the requirement for retraining.

In contrast, one-point influence methods approximate the effect of retraining using a single trained model. A well-established one-point method is the canonical influence function (Koh & Liang, 2017), developed from statistics (Hampel, 1974; Chatterjee et al., 1982), which examines how infinitesimal perturbations of a training example affect the empirical risk minimizers (ERM) (Vapnik & Chervonenkis, 1974). The influence function has been extended to incorporate higher-order information (Basu et al., 2020) and scaled for larger models (Guo et al., 2021; Schioppa et al., 2022), including LLMs (Grosse et al., 2023). While these static influence measures have elegant theoretical foundations, they are limited in characterizing how training data influences the training process.

Alternatively, dynamic methods enhance influence estimation by considering the evolution of model parameters across training iterations (Charpiat et al., 2019). Notable examples in this category include TracIn (Pruthi et al., 2020) and In-Run Data Shapley (Wang et al., 2024), which track the influence of training data points by averaging gradient similarities over time. The practicality of dynamic influence estimators is demonstrated by their applications in improving training processes in modern setups (Xia et al., 2024). Recently, Nickl et al. (2023) adopt a novel memory-perturbation equation

[2]Typically, data influence estimators are classified as retraining-based and gradient-based methods (Hammoudeh & Lowd, 2024). For enhanced logical coherence in this paper, we group retraining-based methods and one-point gradient-based techniques under the static category. This classification does not contradict conventional categorizations.

framework to derive dynamic influence estimation of model trained under different centralized optimization algorithms, including SGD, RMSprop and Adam.

However, the existing static and dynamic influence estimation methods primarily target centralized scenarios, and little progress has been made in analyzing data influence in fully decentralized environments. To the best of our knowledge, the only closely related work is by Terashita & Hara (2022), who proposed a decentralized hyper-gradient method and provided novel insights on applying hyper-gradients to compute a centralized formulation of data influence. Nevertheless, their estimation method is static and cannot capture data cascades arising from gossip communication during decentralized training. In contrast, our framework, DICE, is specifically designed for fully decentralized environments, allowing it to provide a fine-grained characterization of the unique influence cascades inherent in these settings.

Incentivized Decentralized Learning. Most existing incentive mechanisms for collaborative learning are designed for federated learning (Zeng et al., 2021). For instance, Wang et al. (2023) propose an Incentive Collaboration Learning (ICL) framework to promote collaboration. Their focus is on mechanism design rather than the precise quantification of individual contributions. In contrast, our work lays the foundation for such incentive mechanisms by accurately measuring individual-level contributions. In federated learning, the Shapley value has been effectively utilized to quantify participant contributions (Jia et al., 2019; Ghorbani & Zou, 2019; Wang et al., 2019; 2020). Our approach differs fundamentally in two key aspects: first, we focus on fully decentralized settings without central servers, although our framework supports federated learning scenarios; second, our work considers influence cascades between participants, an completely new perspective that has not been explored in existing literature. Regrading decentralized learning, we are only aware of the work by Yu et al. (2023) presenting a blockchain-based incentive mechanism for fully decentralized learning. However, their mechanism relies on smart contracts and differs from our focus.

3 Notations and Preliminaries

This section introduces notations and essential preliminaries for decentralized learning. For more detailed background, please refer to Appendix A.1.

Setup. In the context of crowdsourcing workload, we consider a general distributed optimization problem over a connected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of participants and $\mathcal{E}$ denotes the communication links between them. The participants collaboratively minimize a weighted sum of local objectives (T. Dinh et al., 2020; Hanzely & Richtárik, 2020; Even et al., 2024):

$$\min_{\boldsymbol{\theta}=\{\boldsymbol{\theta}_k\in\mathbb{R}^d\}_{k\in\mathcal{V}}} \left[l(\boldsymbol{\theta}) \triangleq \sum_{k\in\mathcal{V}} q_k l_k(\boldsymbol{\theta}_k) \right], \tag{1}$$

where $q_k \geq 0$ with $\sum_{k\in\mathcal{V}} q_k = 1$, and each local objective $l_k(\boldsymbol{\theta}_k) = \mathbb{E}_{\boldsymbol{z}_k\sim\mathcal{D}_k}\left[L(\boldsymbol{\theta}_k; \boldsymbol{z}_k)\right]$ is defined by the expectation over the local data distribution $\mathcal{D}_k$. Empirical risk minimization involves optimizing the sample average approximation:

$$\hat{l}(\boldsymbol{\theta}) = \sum_{k\in\mathcal{V}} q_k \hat{l}_k(\boldsymbol{\theta}_k) \quad \text{where} \quad \hat{l}_k(\boldsymbol{\theta}_k) = \frac{1}{n_k}\sum_{i=1}^{n_k} L(\boldsymbol{\theta}_k; \boldsymbol{z}_{k_i}). \tag{2}$$

Here, n_k is the number of samples in participant k, and $\{\boldsymbol{z}_{k_i}\}_{i=1}^{n_k}$ are drawn from $\mathcal{D}_k$.

Decentralized learning aims to minimize the global objective $l(\boldsymbol{\theta}) = \sum_{k\in\mathcal{V}} q_k l_k(\boldsymbol{\theta}_k)$ with only local computations and gossip communications among neighboring participants (Xiao & Boyd, 2004; Nedic & Ozdaglar, 2009). The communication protocol is governed by a weighted adjacency matrix $\boldsymbol{W} \in [0,1]^{n\times n}$, where $\boldsymbol{W}_{k,j} \geq 0$ represents the strength of connection from participant j to participant k, with $\boldsymbol{W}_{k,j} > 0$ if $(k,j) \in \mathcal{E}$. This matrix characterizes the communication topology, defining how information propagates through the network. In this paper, $\boldsymbol{W}$ is designed to be row-stochastic, satisfying $\sum_{j=1}^{n} \boldsymbol{W}_{k,j} = 1$ for all $i \in \mathcal{V}$[3].

[3]The weighted adjacency matrix $\boldsymbol{W}$ is typically assumed to be doubly-stochastic. The row-stochastic assumption is weaker, yet the convergence of decentralized SGD is still guaranteed (Yuan et al., 2019; Xin et al., 2019).

Decentralized stochastic gradient descent (Yuan et al., 2016; Lian et al., 2017; Koloskova et al., 2020) is an standard implementation of decentralized training, which alternates between performing local stochastic gradient steps and aggregating parameters through gossips, as shown below:

Algorithm 1 Decentralized Stochastic Gradient Descent (Decentralized SGD)

```
Require: $G = (\mathcal{V}, \mathcal{E})$, $\boldsymbol{W} \in [0,1]^{n \times n}$, $\{\boldsymbol{\theta}_k^0\}_{k \in \mathcal{V}}$, learning rates $\{\eta^t\}_{t=1}^T$, mini-batch $\{\mathcal{B}_k^t\}_{k \in \mathcal{V}, t=1}^T$
1: for $t = 1$ to $T$ do in parallel for all participants $k \in \mathcal{V}$
2:     Mini-batch gradient update: $\boldsymbol{\theta}_k^{t+\frac{1}{2}} \leftarrow \boldsymbol{\theta}_k^t - \frac{1}{|\mathcal{B}_k^t|} \sum_{\boldsymbol{z} \in \mathcal{B}_k^t} \nabla L(\boldsymbol{\theta}_k^t; \boldsymbol{z})$
3:     Information sharing: Send $\boldsymbol{\theta}_k^{t+\frac{1}{2}}$ and $\frac{\eta^t}{|\mathcal{B}_k^t|} \sum_{\boldsymbol{z} \in \mathcal{B}_k^t} \nabla L(\boldsymbol{\theta}_k^t; \boldsymbol{z})$ to $\{l \in \mathcal{V} | \boldsymbol{W}_{l,k} > 0\}$[4]
4:     Gossip averaging from in-neighbors: $\boldsymbol{\theta}_k^{t+1} \leftarrow \sum_{j \in \mathcal{N}_{\text{in}}(k)} \boldsymbol{W}_{k,j}^t \boldsymbol{\theta}_j^{t+\frac{1}{2}}$
   End for
```

Remark 1. The weighted adjacency matrix in Algorithm 1 can vary across iterations, resulting in time-varying collaborations among participants. Additionally, FedAVG (McMahan et al., 2017) is a special case of Algorithm 1 where the averaging step is performed globally. This demonstrates that our framework accommodates decentralized learning with dynamic communication topologies and is applicable to both federated and decentralized learning paradigms, even though the primary focus is on fully decentralized learning without central servers.

4 Data Influence Cascades

In this section, we introduce DICE, a comprehensive framework for measuring data influence in decentralized environments. Subsection 4.1 introduces the ground-truth influence measures designed for decentralized learning and Subsection 4.2 provides their dynamic gradient-based estimations.

4.1 Ground-truth Influence in Decentralized Learning

To ensure a logical and coherent flow, we first introduce the fundamental concepts of data influence in centralized settings and then discuss the significant challenges involved in extending these ideas to decentralized environments. In conventional centralized setups, the influence of an individual data instance can be assessed by evaluating the counterfactual change in learning performance through leave-one-out retraining (LOO) (Cook, 1977), defined as follows:

Definition 1 (Leave-one-out Influence)**.**

$$\mathcal{I}_{\text{LOO}}(\boldsymbol{z}, \boldsymbol{z}') = L(\boldsymbol{\theta}^*; \boldsymbol{z}') - L(\boldsymbol{\theta}^*_{\setminus \boldsymbol{z}}; \boldsymbol{z}'), \tag{3}$$

where $\boldsymbol{z}$ denotes the training data instance under influence assessment, $\boldsymbol{z}'$ is the loss-evaluating instance, $\boldsymbol{\theta}^*$ and $\boldsymbol{\theta}^*_{\setminus \boldsymbol{z}}$ are the models trained on the entire dataset $\mathcal{S}$ and $\mathcal{S} \setminus \{z\}$, respectively.

Intuitively, Equation (3) quantifies the influence of $\boldsymbol{z}$ by its individual impact on test loss reduction. A smaller LOO value indicates a significant contribution to learning, which aligns with the concept that the data influence is reflected in its ability to enhance model performance. LOO influence is often considered as the "gold standard" for evaluating how well influence estimators approximate the ground-truth influence in the data influence literature (Koh & Liang, 2017; Basu et al., 2021).

However, extending LOO to decentralized scenarios introduces non-trivial challenges due to the distributed nature and the localized connections in decentralized learning, reflected in Equation (2) and Algorithm 1. In centralized setups, the core idea of LOO is to link data influence to variations in loss or parameter outcomes. In contrast, decentralized learning systems involve multiple participants sharing model parameters through inter-participant communications. As a result, alterations in model parameters caused by a data-level modification propagate throughout the whole network.

[4] In decentralized learning literature, it is common for each participant to share only local parameters with its neighboring participants (Lian et al., 2017; Koloskova et al., 2020). We note that sharing local gradients with neighbors maintains the decentralized learning paradigm and does not significantly compromise privacy, as a participant can reconstruct the gradients of its neighbors using the shared parameters (Mrini et al., 2024).

A natural way to measure the influence of one participant in such collaborative environments is through evaluating its contribution to the whole community (Wang et al., 2020; Yu et al., 2020), which aligns with the customer-centric principle (Drucker, 1985) in determining value[5]. In decentralized learning, when a participant transmits its training assets (e.g., model parameters or gradients) to neighboring participants—akin to offering a product—the recipients derive possible utility from these training assets and may provide reciprocal feedback, such as sharing their own assets in return. This dynamic positions the neighbors as "customers", thereby entrusting them with the rights to determine the value of the assets provided by the supplier. With these insights in mind, we recognize that assessing data influence in decentralized scenarios is far more complex, as summarized below:

Key observations: *In decentralized learning,*
1) neighbors who serves as customers hold the rights to determine data influence;
2) data influence is not static but spreads across participants through gossips during training.

Unfortunately, existing static estimators only calculate the loss change after training and thus cannot characterize the dynamic transmission of data influence within the whole decentralized learning community. Based on the above discussion, we posit that a "gold-standard" influence measure in decentralized scenarios should satisfy the following requirements:

- Quantify community-level influence: Measure the impact of training data instances on the collective utility of the community.
- Depend on training dynamics: Measure the influence based on the training process to characterize the propagation of influence on decentralized networks.

In the following, we introduce the ground-truth influence measures tailored to the requirements of decentralized environments, termed as the *ground-truth influence cascade (DICE-GT)*.

Definition 2 (One-hop Ground-truth Influence Cascade)**.** The one-hop DICE-GT value quantifies the influence of a data instance $\boldsymbol{z}_j^t$ from participant j on a loss-evaluating instance $\boldsymbol{z}'$ within itself and its immediate neighbors. Formally, for a given participant $j \in \mathcal{V}$:

$$\mathcal{I}^{(1)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') = \underbrace{q_j \left(L(\boldsymbol{\theta}_j^{t+\frac{1}{2}}; \boldsymbol{z}') - L(\boldsymbol{\theta}_j^t; \boldsymbol{z}') \right)}_{\text{direct marginal contribution of } \boldsymbol{z}_j^t \text{ to } j} + \underbrace{\sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} q_k \left(L(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}') - L(\boldsymbol{\theta}_{k \setminus \boldsymbol{z}_j^t}^{t+1}; \boldsymbol{z}') \right)}_{\text{indirect marginal contribution of } \boldsymbol{z}_j^t \text{ to one-hop neighbors}},$$

where $\boldsymbol{\theta}_j^{t+\frac{1}{2}}$ denotes the updated model parameters of j after training on $\boldsymbol{z}_j^t$ at iteration t (see Algorithm 1). For each one-hop out-neighbor $k \in \mathcal{N}^{(1)}_{\text{out}}(j)$, $\boldsymbol{\theta}_k^{t+1}$ denotes the averaged model parameters after receiving updated parameters $\{\boldsymbol{\theta}_l^{t+\frac{1}{2}} | \boldsymbol{W}_{k,l} > 0\}$ influenced by $\boldsymbol{z}_j^t$, while $\boldsymbol{\theta}_{k \setminus \boldsymbol{z}_j^t}^{t+1}$ represents the model parameters of k without the influence from $\boldsymbol{z}_j^t$, i.e., $\boldsymbol{\theta}_{k \setminus \boldsymbol{z}_j^t}^{t+1} = \sum_{l \in \mathcal{N}_{\text{out}}(k) \setminus j} \boldsymbol{W}_{k,l}^t \boldsymbol{\theta}_l^{t+\frac{1}{2}} + \boldsymbol{W}_{k,j}^t \boldsymbol{\theta}_l^t$.

The economic intuition behind the DICE-GT value is that it captures both the direct marginal contribution of a data instance to itself and its subsequent impact on immediate neighbors. Specially, the first term $L(\boldsymbol{z}'; \boldsymbol{\theta}_j^{t+\frac{1}{2}}) - L(\boldsymbol{z}'; \boldsymbol{\theta}_j^t)$ captures the inter-node direct influence of training data instance $\boldsymbol{z}_j^t$ on the test loss change at node j, which corresponds to the TracInIdeal influence in (Pruthi et al., 2020) designed for centralized scenarios. The second term $\sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} (L(\boldsymbol{z}'; \boldsymbol{\theta}_k^{t+1}) - L(\boldsymbol{z}'; \boldsymbol{\theta}_{k \setminus \boldsymbol{z}_j^t}^t))$ measures the intra-node influence unique in decentralized learning, which aggregates the indirect influences on all one-hop neighbors, i.e., direct neighbors, of node j.

In decentralized learning environments, data influence propagates not only to immediate neighbors but also to multi-hop neighbors through the communication topology. To characterize this multi-hop influence, we extend the ground-truth influence cascade measure to arbitrary r-hop neighbors.

Definition 3 (Multi-hop Ground-truth Influence Cascade)**.** The multi-hop DICE-GT value quantifies the cumulative influence of a data instance $\boldsymbol{z}$ on a loss-evaluating instance $\boldsymbol{z}'$ across all nodes within

[5]Peter Drucker's customer-centric value principle from *Innovation and Entrepreneurship* states, "Quality in a product or service is not what the supplier puts in. It is what the customer gets out of it.".

r-hop neighborhoods of participant j. Formally, for a given participant $j \in \mathcal{V}$:

$$\mathcal{I}^{(r)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') = q_j \left(L(\boldsymbol{\theta}_j^{t+\frac{1}{2}}; \boldsymbol{z}') - L(\boldsymbol{\theta}_j^t; \boldsymbol{z}') \right) + \sum_{s=1}^{r} \sum_{k \in \mathcal{N}^{(s)}_{\text{out}}(j)} q_k \left(L(\boldsymbol{\theta}_k^{t+s}; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+s}_{k \setminus \boldsymbol{z}_j^t}; \boldsymbol{z}') \right),$$

where $\mathcal{N}^{(s)}_{\text{out}}(j)$ denotes the set of s-hop out-neighbors of j (please refer to Appendix A.2 for details of high-order neighbors). $\boldsymbol{\theta}_k^{t+s}$ and $\boldsymbol{\theta}^{t+s}_{k \setminus \boldsymbol{z}_j^t}$ represents the parameters of node k at iteration $t+s$ when the influence from $\boldsymbol{z}_j^t$ are included and excluded, respectively.

Analogous to Definition 2, the first term captures the direct influence of data $\boldsymbol{z}_j^t$ on the loss at node j. The subsequent summation aggregates the indirect influences on all multi-hop neighbors up to r steps away from node j. The reason to measure test loss change at the $t+s$ step is that the impact of $\boldsymbol{z}_j^t$ propagating to $k \in \mathcal{N}^{(s)}_{\text{out}}(j)$ requires s steps. This layered formulation accounts for the multi-hop cascading effects through the network up to the specified order r. $\mathcal{I}^{(r)}_{\text{DICE-GT}}$ captures both immediate and distinct impact within the decentralized learning community, aligning with the dynamic local information propagation observed in decentralized environments.

4.2 Dynamic Gradient-based Estimations

To meet the second aforementioned requirement of decentralized learning, we design dynamic gradient-based estimators for DICE-GT, called the *influence cascade estimations (DICE-E)*.

Proposition 1 (Approximation of One-hop DICE-GT)**.** The one-hop DICE-GT value (see Definition 2) can be linearly approximated as follow:

$$\mathcal{I}^{(1)}_{\text{DICE-E}}(\boldsymbol{z}_j^t, \boldsymbol{z}') = -\eta^t q_j \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t) - \eta^t \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} q_k \boldsymbol{W}^t_{k,j} \nabla L(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t).$$

The proof is included in Appendix C.1.

Additivity. The one-hop DICE-E influence measure is additive over training instances. Specifically, for a mini-batch $\mathcal{B}_j^t$ from participant j, the total influence is the sum of individual influences:

$$\mathcal{I}^{(1)}_{\text{DICE-E}}(\mathcal{B}_j^t, \boldsymbol{z}') = \sum_{\boldsymbol{z}_j^t \in \mathcal{B}_j^t} \mathcal{I}^{(1)}_{\text{DICE-E}}(\boldsymbol{z}_j^t, \boldsymbol{z}'). \tag{4}$$

This property arises from linear Taylor approximation and the mini-batch update rule in decentralized SGD. Specifically, substituting the mini-batch gradient update procedure of Algorithm 1 into the one-hop DICE-E expression and leveraging the linearity of the inner products, we observe that the total influence is additive over the mini-batch. This additivity is crucial for practical implementations, allowing efficient computation of DICE-E influence for large mini-batches.

We then extend the influence approximation to multi-hop neighbors in decentralized learning and show how the influence of a data instance cascades over the decentralized network.

Proposition 2 (Approximation of Two-hop DICE-GT)**.** The two-hop DICE-GT influence $\mathcal{I}^{(2)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}')$ (see Definition 3) can be approximated as follows:

$$\begin{aligned}\mathcal{I}^{(2)}_{\text{DICE-E}}(\boldsymbol{z}_j^t, \boldsymbol{z}') &= \mathcal{I}^{(1)}_{\text{DICE-E}}(\boldsymbol{z}_j^t, \boldsymbol{z}') \\ &- \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} \sum_{l \in \mathcal{N}^{(1)}_{\text{out}}(k)} \eta^t q_l \boldsymbol{W}^{t+1}_{l,k} \boldsymbol{W}^t_{k,j} \nabla L(\boldsymbol{\theta}_l^{t+2}; \boldsymbol{z}')^\top (\boldsymbol{I} - \eta^{t+1} \boldsymbol{H}(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1})) \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t),\end{aligned} \tag{5}$$

where $\boldsymbol{H}(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1})$ denotes the Hessian matrix of L with respect to $\boldsymbol{\theta}$ evaluated at $\boldsymbol{\theta}_k^{t+1}$ and $\boldsymbol{z}_k^{t+1}$. Full proof is included in Appendix C.2.

Proposition 3 (Approximation of r-hop DICE-GT)**.** The r-hop DICE-GT influence $\mathcal{I}^{(r)}_{\text{DICE-GT}}(\boldsymbol{z}^t_j, \boldsymbol{z}')$ (see Definition 3) can be approximated as follows:

$$\mathcal{I}^{(r)}_{\text{DICE-E}}(\boldsymbol{z}^t_j, \boldsymbol{z}') = -\sum_{\rho=0}^{r} \sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} \eta^t q_{k_\rho} \left(\prod_{s=1}^{\rho} \boldsymbol{W}^{t+s-1}_{k_s,k_{s-1}}\right)$$
$$\nabla L(\boldsymbol{\theta}^{t+\rho}_{k_\rho}; \boldsymbol{z}')^\top \left(\prod_{s=2}^{\rho} \left(\boldsymbol{I} - \eta^{t+s-1} \boldsymbol{H}(\boldsymbol{\theta}^{t+s-1}_{k_s}; \boldsymbol{z}^{t+s-1}_{k_s})\right)\right) \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j). \quad (6)$$

where $k_0 = j$, $P_j^{(\rho)}$ denotes the set of all sequences $(k_1, \ldots, k_\rho)$ such that $k_s \in \mathcal{N}^{(1)}_{\text{out}}(k_{s-1})$ for $s = 1, \ldots, \rho$ (see Definition A.7) and $\boldsymbol{H}(\boldsymbol{\theta}^{t+s}_{k_s}; \boldsymbol{z}^{t+s}_{k_s})$ is the Hessian matrix of L with respect to $\boldsymbol{\theta}$ evaluated at $\boldsymbol{\theta}^{t+s}_{k_s}$ and data $\boldsymbol{z}^{t+s}_{k_s}$. For the cases when $\rho = 0$ and $\rho = 1$, the relevant product expressions are defined as identity matrices, thereby ensuring that the r-hop DICE-E remains well-defined. Full proof is deferred to Appendix C.3.

Multi-hop DICE-E characterizes the cascading effects of data influence through multiple "layers" of communication. In this context, the influence of a data instance from participant j can propagate through a sequence of intermediate nodes, reaching participants that are ρ hops away. This multi-hop propagation mechanism allows DICE-E to account for indirect influences, providing a more comprehensive and nuanced measurement of data influence across the entire decentralized network.

Influence Dynamics: Exponential Decay and Topological Dependency. Proposition 3 demonstrates that the multi-hop influence of a data instance $\boldsymbol{z}^t_j$ is governed by the product of communication weights $\prod_{s=1}^{\rho} \boldsymbol{W}^{t+s-1}_{k_s,k_{s-1}}$ and Hessian-related terms $\prod_{s=2}^{\rho}(\boldsymbol{I} - \eta^{t+s-1}\boldsymbol{H}^{t+s-1}_{k_s})$. This indicates that the multi-hop influence of data in decentralized learning depends on the curvature information of the intermediate nodes and diminishes exponentially with each additional network hop. Moreover, the influence dynamics are inherently tied to the communication graph topology, as reflected by weights such as $\boldsymbol{W}^t_{k,j}$, representing connection strength. Nodes with higher topological importance (e.g., node j with large $\sum_{j=1}^{n} \boldsymbol{W}_{j,k}$) have their data influences propagated more widely and with greater impact on the global utility. This property highlights the interplay between the original data and the network structure in sharping data influence in decentralized learning.

4.3 Practical Applications of DICE-E

In idealized scenarios, participants may seek to estimate the influence of their high-order neighbors on their local utility improvement. However, multi-hop influence estimations can be computationally intensive. Therefore, one-hop DICE-E emerges as a more suitable choice due to its computational efficiency and inherent additivity. Based on Proposition 1, we derive the peer-level contribution, which we refer to as the *proximal influence*.

Definition 4 (Proximal Influence)**.** The proximal influence of a data instance $\boldsymbol{z}^t_j$ from participant j on participant k at iteration t is defined as follows:

$$\mathcal{I}^{k,j}_{\text{DICE-E}}(\boldsymbol{z}^t_j, \boldsymbol{z}') = -\eta^t \boldsymbol{W}^t_{k,j} q_k \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j)^\top \nabla L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}'). \quad (7)$$

This term quantifies the influence of the data instance $\boldsymbol{z}^t_j$ from participant j on the loss reduction experienced by its immediate neighbor k. Importantly, under the information sharing protocol defined in Algorithm 1, participant k has access to q_k, $\boldsymbol{W}^t_{k,j}$, $\nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j)$, and $\nabla L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}')$. Therefore, each participant can compute the proximal contributions of its neighbors. The proximal influence can be utilized in the following scenarios:

Collaborator Selection. In decentralized learning, local data remains private and only local parameter communication is permitted. The absence of a central authority complicates the problem of selecting the most suitable neighbors with high-quality data. Fortunately, DICE offers a mechanism for participants to efficiently estimate the contributions of their neighbors with proximal influence. By assessing the proximal influence of their neighbors, participants can identify the potential collaborators that have the most significant positive impact on their learning process.

To ensure reciprocal collaboration (Gouldner, 1960; Sundararajan & Krichene, 2023), participants can compute *reciprocity factors*, which evaluate the mutual balance of influence.

Definition 5 (Reciprocity Factors)**.** The *reciprocity factor* is defined in two forms:

1. **Proximal Reciprocity Factor:** The reciprocity factor between participants j and k at iteration t is

$$R_{k,j}^{t} = \frac{q_k \boldsymbol{W}_{k,j}^{t} \nabla L(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}_j^{t}; \boldsymbol{z}_k^{t})}{q_j \boldsymbol{W}_{j,k}^{t} \nabla L(\boldsymbol{\theta}_j^{t+1}; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}_k^{t}; \boldsymbol{z}_j^{t})}. \tag{8}$$

2. **Neighborhood Reciprocity Factor:** To evaluate reciprocity at the community level, the neighborhood reciprocity factor for participant j at iteration t is defined as:

$$R_{j}^{t} = \frac{\sum_{k \in \mathcal{N}_{\text{out}}^{(1)}(j)} q_k \boldsymbol{W}_{k,j}^{t} \nabla L(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}_j^{t}; \boldsymbol{z}_j^{t})}{\sum_{l \in \mathcal{N}_{\text{in}}^{(1)}(j)} q_l \boldsymbol{W}_{j,l}^{t} \nabla L(\boldsymbol{\theta}_j^{t+1}; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}_l^{t}; \boldsymbol{z}_j^{t})}. \tag{9}$$

The proximal reciprocity factor measures the balance of influence between two participants, with values near unity indicating equitable mutual contributions. Significant deviations suggest an imbalance, helping participants refine their collaboration strategies. The neighborhood reciprocity factor extends this concept to a participant's local community, evaluating the balance between influence inflow and outflow. This metric supports participants in adjusting their engagement and aids the community in managing membership, such as admitting new members or excluding underperforming participants.

5 Experiments

Computational Resources. The experiments are conducted on a computing facility equipped with 80 GB NVIDIA® A100™ GPUs.

Implementation Details. Vanilla mini-batch Adapt-Then-Communicate version of Decentralized SGD ((Lopes & Sayed, 2008), see Algorithm 1) with commonly used topologies (Ying et al., 2021) is employed to train three layer MLPs (Rumelhart et al., 1986), three layer CNNs (LeCun et al., 1998) and ResNet-18 (He et al., 2016) on subsets of MNIST (LeCun et al., 1998), CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and Tiny ImageNet (Le & Yang, 2015) datasets. The number of participants (one GPU as a participant) is set to 16 and 32, with each participant holding 512 samples. For sensitivity analysis, we evaluate the stability of results under hyperparameter adjustments. The local batch size is varied as 16, 64, and 128 per participant, while the learning rate is set as 0.1 and 0.01 without decay. The code will be made publicly available.

5.1 Influence Alignment

We evaluate the alignment between one-hop DICE-GT (see Definition 2) and its first-order approximation, one-hop DICE-E (see Proposition 1). One-hop DICE-E $\mathcal{I}_{\text{DICE-E}}^{(1)}(\mathcal{B}_j^t, \boldsymbol{z}')$ is computed as the sum of one-sample DICE-E within the mini-batch $\mathcal{B}_j^t$ thanks to the additivity (see Equation (4)). DICE-GT $\mathcal{I}\text{DICE-GT}^{(1)}(\mathcal{B}_j^t, \boldsymbol{z}')$ is calculated by measuring the loss reduction after removing $\mathcal{B}_j^t$ from node j at the t-th iteration. As shown in Figure 2, each plot contains 30 points, with each point representing the result of a single comparison of the ground-truth and estimated influence. We can observe from Figure 2 that DICE-E closely tracks DICE-GT under different settings. The alignment becomes even stronger on simpler data set including CIFAR-10 and CIFAR-100, as detailed in Appendix D.2. These results demonstrate that DICE-E provides a strong approximation of DICE-GT, achieving consistent alignment across datasets (CIFAR-10, CIFAR-100 and Tiny ImageNet) and model architectures (CNN and MLP). Further validation of this alignment is provided in Appendix D.2 to corroborate the robustness of one-hop DICE-E under changing batch sizes, learning rates, and training epochs.

5.2 Anomaly Detection

DICE identifies malicious neighbors, referred to as anomalies, by evaluating their proximal influence, which estimates the reduction in test loss caused by a single neighbor. A high proximal influence

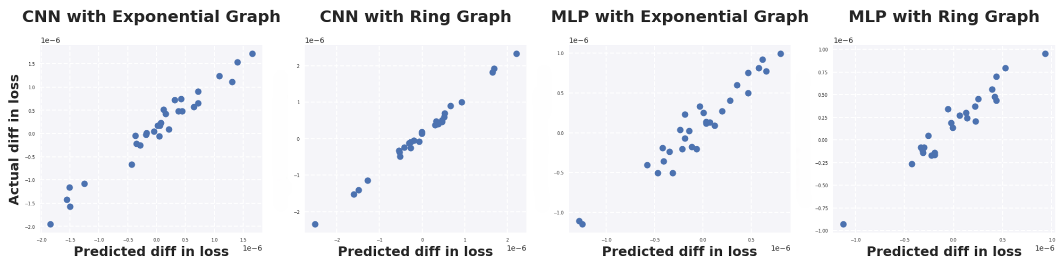

Figure 2: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 32-node ring graph. Each node uses a 512-sample subset of Tiny ImageNet. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.1.

score indicates that a neighbor increases the test loss, negatively impacting the learning process. In our setup, anomalies are generated through random label flipping or by adding random Gaussian noise to features, please kindly refer to (Zhang et al., 2024). Figure 2 illustrates that the most anomalies (in red) are readily detectable with proximal influence values. Additional results in Appendix D.3 further validate the reliability of this approach.

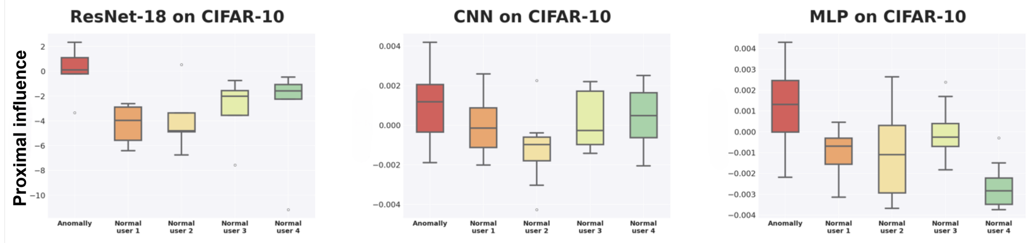

Figure 3: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.1. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by random label flipping, while the other four are normal participants.

5.3 Influence cascades

The topological dependency of DICE-E in our theory reveals the "power asymmetries" (Blau, 1964; Magee & Galinsky, 2008) in decentralized learning. To support the theoretical finding, we examine the one-hop DICE-E values of the same batch on participants with vastly different topological importance. Figure 1 illustrates the one-hop DICE-E influence scores of an identical data batch across participants during decentralized training of a ResNet-18 model on the CIFAR-10 dataset. Node sizes represent the one-hop DICE-E influence scores, quantifying how a single batch impacts other participants in the network. The dominant nodes (e.g., those with larger outgoing communication weights in $\boldsymbol{W}$) exhibit significantly higher influence, as shown in Figure 1 and further detailed in Appendix D.4. These visualizations underscore the critical role of topological properties in shaping data influence in decentralized learning, demonstrating how the structure of the communication matrix $\boldsymbol{W}$ determines the asymmetries in influence.

6 Conclusion

In this paper, we introduce DICE, the first comprehensive framework to quantify data influence in fully decentralized learning environments. DICE characterizes how data influence cascades through the communication network and uncover for the first time the intricate interaction between original data, communication topology and the curvature information of optimization landscapes in shaping influence. Future work can also leverage DICE to design effective incentive schemes (Yu et al., 2020) and build decentralized data and parameter markets (Huang et al., 2023; Fallah et al., 2024).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Samyadeep Basu, Xuchen You, and Soheil Feizi. On second-order group influence functions for black-box predictions. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021.

Peter M. Blau. Exchange and power in social life. 1964.

Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin A Raffel. Distributed inference and fine-tuning of large language models over the internet. In *Advances in Neural Information Processing Systems*, 2023.

CCAF. Cambridge bitcoin electricity consumption index (CBECI). https://ccaf.io/cbnsi/cbeci, 2023.

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input similarity from the neural network perspective. In *Advances in Neural Information Processing Systems*, 2019.

Samprit Chatterjee, R. Dennis Cook, and Sanford Weisberg. Residuals and influence in regression. 1982.

Daiwei Chen, Jane Zhang, and Ramya Korlakai Vinayak. Unraveling the impact of training samples. In *ICLR Blogposts 2024*, 2024. URL https://iclr-blogposts.github.io/2024/blog/unraveling-the-impact-of-training-samples/. https://iclr-blogposts.github.io/2024/blog/unraveling-the-impact-of-training-samples/.

R. Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1): 15–18, 1977.

Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

Peter F. Drucker. *Innovation and Entrepreneurship*. Perennial Library, 1985.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Mathieu Even, Anastasia Koloskova, and Laurent Massoulie. Asynchronous SGD on graphs: a unified framework for asynchronous decentralized and federated optimization. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, 2024.

Alireza Fallah, Michael I Jordan, Ali Makhdoumi, and Azarakhsh Malekian. On three-layer data markets. *arXiv preprint arXiv:2402.09697*, 2024.

Ernst Fehr and Simon Gächter. Fairness and retaliation: The economics of reciprocity. *Journal of Economic Perspectives*, 14(3):159–181, 2000.

Anissa Gardizy and Amir Efrati. Microsoft and OpenAI plot $100 billion stargate AI supercomputer. *The Information*, 2024. URL https://www.theinformation.com/articles/microsoft-and-openai-plot-100-billion-stargate-ai-supercomputer.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *Advances in Neural Information Processing Systems*, 2020.

Alvin W. Gouldner. The norm of reciprocity: A preliminary statement. *American Sociological Review*, 25(2):161–178, 1960.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Zayd Hammoudeh and Daniel Lowd. Identifying a training-set attack's target using renormalized influence estimation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: a survey. *Machine Learning*, 113(5):2351–2403, 2024.

Frank R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, 2016.

Anson Ho, Tamay Besiroglu, Ege Erdil, David Owen, Robi Rahman, Zifan Carl Guo, David Atkinson, Neil Thompson, and Jaime Sevilla. Algorithmic progress in language models. *arXiv preprint arXiv:2403.05812*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, 2022.

Tzu-Heng Huang, Harit Vishwakarma, and Frederic Sala. Train 'n trade: Foundations of parameter markets. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Andrew Ilyas, Kristian Georgiev, Logan Engstrom, and Sung Min (Sam) Park. Data attribution at scale, 2024. URL https://ml-data-tutorial.org/. ICML 2024 Tutorial.

Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. Subpopulation data poisoning attacks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019.

Heasung Kim, Hyeji Kim, and Gustavo De Veciana. Clustered federated learning via gradient-based partitioning. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized SGD with changing topology and local updates. In *International Conference on Machine Learning*, 2020.

Alex Krizhevsky, G Hinton, et al. Learning multiple layers of features from tiny images (tech. rep.). *University of Toronto*, 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

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

Peizhao Li and Hongfu Liu. Achieving fairness at no utility cost via data reweighing with influence. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, 2017.

Shayne Longpre, Robert Mahari, Ariel Lee, Campbell Lund, Hamidah Oderinwale, William Brannon, Nayan Saxena, Naana Obeng-Marnu, Tobin South, Cole Hunter, et al. Consent in crisis: The rapid decline of the ai data commons. *arXiv preprint arXiv:2407.14933*, 2024.

Noel Loo, Ramin Hasani, Mathias Lechner, and Daniela Rus. Dataset distillation with convexified implicit gradients. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 22649–22674, 2023.

Cassio G. Lopes and Ali H. Sayed. Diffusion least-mean squares over adaptive networks: Formulation and performance analysis. *IEEE Transactions on Signal Processing*, 56(7), 2008.

Joe C Magee and Adam D Galinsky. Social hierarchy: The self-reinforcing nature of power and status. *Academy of Management Annals*, 2(1):351–398, 2008.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 25(4):2983–3013, 2023.

Nestor Maslej, Loredana Fattorini, Raymond Perrault, Vanessa Parli, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, and Jack Clark. The AI index 2024 annual report. Technical report, AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, 2024.

Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33: 1–17, 2017.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Abdellah El Mrini, Edwige Cyffers, and Aurélien Bellet. Privacy attacks in decentralized learning. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Peter Nickl, Lu Xu, Dharmesh Tailor, Thomas Möllenhoff, and Mohammad Emtiyaz E Khan. The memory-perturbation equation: Understanding model's sensitivity to data. In *Advances in Neural Information Processing Systems*, 2023.

Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

Seulki Park, Jongin Lim, Younghan Jeon, and Jin Young Choi. Influence-balanced loss for imbalanced visual classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon LLM: Outperforming curated corpora with web data only. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, 2020.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. MIT Press, 1986.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3710–3722, 2021.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *Advances in Neural Information Processing Systems*, 2021.

Jaime Sevilla and Edu Roldán. Training compute of frontier ai models grows by 4-5x per year, 2024. URL https://epochai.org/blog/training-compute-of-frontier-ai-models-grows-by-4-5x-per-year.

Lloyd S. Shapley. A value for n-person games. In *Contributions to the Theory of Games, Volume II*, chapter 17. Princeton University Press, 1953.

Abhishek Singha, Charles Lua, Gauri Guptaa, Ayush Chopraa, Jonas Blanca, Tzofi Klinghoffera, Kushagra Tiwarya, and Ramesh Raskara. A perspective on decentralizing ai. 2024.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems*, 2022.

Mukund Sundararajan and Walid Krichene. Inflow, outflow, and reciprocity in machine learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

James Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies, and Nations*. Doubleday, 2004.

Canh T. Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems*, 2020.

Naoyuki Terashita and Satoshi Hara. Decentralized hyper-gradient computation over time-varying directed networks. *arXiv preprint arXiv:2210.02129*, 2022.

Vladimir Vapnik and Alexey Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moscow, 1974.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? Limits of LLM scaling based on human-generated data. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Guan Wang, Charlie Xiaoqian Dang, and Ziye Zhou. Measure contribution of participants in federated learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 2597–2604, 2019.

Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *arXiv preprint arXiv:2406.11011*, 2024.

Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. *A Principled Approach to Data Valuation for Federated Learning*, pp. 153–167. Springer International Publishing, 2020.

Xinran Wang, Qi Le, Ahmad Faraz Khan, Jie Ding, and Ali Anwar. A framework for incentivized collaborative learning. *arXiv preprint arXiv:2305.17052*, 2023.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 54104–54132, 2024.

Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

Ran Xin, Chenguang Xi, and Usman A. Khan. Frost—fast row-stochastic optimization with uncoordinated step-sizes. *EURASIP Journal on Advances in Signal Processing*, 2019(1):1, 2019.

Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence. In *The Eleventh International Conference on Learning Representations*, 2023.

Bicheng Ying, Kun Yuan, Yiming Chen, Hanbin Hu, Pan Pan, and Wotao Yin. Exponential graph is provably efficient for decentralized deep training. In *Advances in Neural Information Processing Systems*, 2021.

Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A fairness-aware incentive scheme for federated learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020.

Haoxiang Yu, Hsiao-Yuan Chen, Sangsu Lee, Sriram Vishwanath, Xi Zheng, and Christine Julien. idml: Incentivized decentralized machine learning. *arXiv preprint arXiv:2304.05354*, 2023.

Binhang Yuan, Yongjun He, Jared Quincy Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 2022.

Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Kun Yuan, Bicheng Ying, Xiaochuan Zhao, and Ali H. Sayed. Exact diffusion for distributed optimization and learning—part i: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3):708–723, 2019.

Liangqi Yuan, Ziran Wang, Lichao Sun, Philip S. Yu, and Christopher G. Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, pp. 1–1, 2024.

Rongfei Zeng, Chao Zeng, Xingwei Wang, Bo Li, and Xiaowen Chu. A comprehensive survey of incentive mechanism for federated learning. *arXiv preprint arXiv:2106.15406*, 2021.

Chang Zhang, Shunkun Yang, Lingfeng Mao, and Huansheng Ning. Anomaly detection and defense techniques in federated learning: a comprehensive review. *Artificial Intelligence Review*, 57(6): 150, 2024.

A BACKGROUND

A.1 BACKGROUND OF DECENTRALIZED LEARNING

Decentralized training allows collaborative training without the control of central servers. For a more comprehensive overview of decentralized learning, we refer readers to Martínez Beltrán et al. (2023); Singha et al. (2024); Yuan et al. (2024).

We summarize some commonly used notions regarding decentralized training as follows:

Definition A.1 (Doubly Stochastic Matrix)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent a decentralized communication topology, where $\mathcal{V}$ is the set of n nodes and $\mathcal{E}$ is the set of edges. For any $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the doubly stochastic gossip matrix $\boldsymbol{W} = [\boldsymbol{W}_{j,k}] \in \mathbb{R}^{n \times n}$ is defined on the edge set $\mathcal{E}$ and satisfies:

- If $j \neq k$ and $(j,k) \notin \mathcal{E}$, then $\boldsymbol{W}_{j,k} = 0$; otherwise, $\boldsymbol{W}_{j,k} > 0$.
- $\boldsymbol{W}_{j,k} \in [0,1]$ for all j, k, and $\sum_k \boldsymbol{W}_{k,j} = \sum_j \boldsymbol{W}_{j,k} = 1$.

Intuitively, the doubly stochastic property ensures a balanced flow of information during gossip communication, a common assumption in decentralized learning literature. However, in the scenarios we consider, participants may occupy different roles within the network. Influential nodes might have higher outgoing weights, i.e., $\sum_{j=1}^{n} \boldsymbol{W}_{j,k} > 1$.

To accommodate such cases while still ensuring the convergence of decentralized SGD (Yuan et al., 2019; Xin et al., 2019), we introduce a relaxed condition:

Definition A.2 (Row Stochastic Matrix)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a decentralized communication topology, where $\mathcal{V}$ is the set of n nodes and $\mathcal{E}$ is the set of edges. For any $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the row stochastic gossip matrix $\boldsymbol{W} = [\boldsymbol{W}_{j,k}] \in \mathbb{R}^{n \times n}$ is defined on the edge set $\mathcal{E}$ and satisfies:

- If $j \neq k$ and $(j,k) \notin \mathcal{E}$, then $\boldsymbol{W}_{j,k} = 0$; otherwise, $\boldsymbol{W}_{j,k} > 0$.
- $\boldsymbol{W}_{j,k} \in [0,1]$ for all j, k, and $\sum_j \boldsymbol{W}_{k,j} = 1$.

The weighted adjacency matrix $\boldsymbol{W}$ in Algorithm 1 can vary across iterations, resulting in time-varying collaborations among participants. Additionally, FedAVG (McMahan et al., 2017) is a special case of Algorithm 1 where the averaging step is performed globally. This demonstrates that our framework accommodates decentralized learning with dynamic communication topologies and is applicable to both federated and decentralized learning paradigms, even though the primary focus is on fully decentralized learning without central servers.

A.2 BACKGROUND OF MULTI-HOP NEIGHBORS

In graph theory, the concept of neighborhoods is fundamental for understanding the structure and dynamics of graphs. To ensure a coherent and comprehensive flow in Section 4, we provide formal definitions of multi-hop neighborhoods.

The adjacency matrix serves as a powerful tool for representing and analyzing the structure of a graph. Multi-hop neighbors can be precisely defined using the adjacency matrix.

Definition A.3 (Adjacency Matrix)**.** The adjacency matrix A of a graph $G = (\mathcal{V}, \mathcal{E})$ is an $n \times n$ square matrix (where $n = |\mathcal{V}|$) defined by:

$$A_{jk} = \begin{cases} 1 & \text{if } (j,k) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{A.1}$$

The adjacency matrix enables the determination of r-hop neighbors through matrix exponentiation. Specifically, the (j,k)-entry of A^r, denoted as $(A^r)_{jk}$, corresponds to the number of distinct paths of length r from node j to node k.

Definition A.4 (r-hop Neighbor via Adjacency Matrix)**.** The set of r-hop neighbors is formally defined using the adjacency matrix A as:

$$\mathcal{N}^{(r)}(j) = \left\{ k \in \mathcal{V} \,\middle|\, (A^r)_{jk} > 0 \text{ and } \forall s < r,\, (A^s)_{jk} = 0 \right\}. \tag{A.2}$$

This definition indicates that there exists at least one path of length r connecting nodes j and k, and no shorter path exists between them.

Multi-hop neighbors can also be defined via the *shortest path length* between two nodes.

Definition A.5 (Shortest Path Length)**.** In a connected graph $G = (\mathcal{V}, \mathcal{E})$, the shortest path length $d(j, k)$ between nodes $j \in \mathcal{V}$ and $k \in \mathcal{V}$ is the minimum number of edges that must be traversed to travel from j to k.

Building upon this, the set of r-hop neighbors is defined as follows:

Definition A.6 (r-hop Neighbor via Shortest Path Length)**.** For any node $j \in \mathcal{V}$ and a positive integer $r \geq 1$, the set of r-hop neighbors, denoted by $\mathcal{N}^{(r)}(j)$, consists of all nodes that are at a distance of exactly r from node j. Formally,

$$\mathcal{N}^{(r)}(j) = \{k \in \mathcal{V} \mid d(j, k) = r\}, \tag{A.3}$$

where $d(j, k)$ represents the shortest path length between nodes j and k in the graph G.

Furthermore, an alternative perspective on r-hop neighborhoods involves characterizing them through sequences of nodes, which provides a formal framework aligned with influence propagation in decentralized learning.

Definition A.7 (r-hop Neighbor via Node Sequences)**.** For any node $j \in \mathcal{V}$ and a positive integer $r \geq 1$, let $P_j^{(r)}$ denote the set of all sequences $(k_1, \ldots, k_r)$ such that for each $s = 1, \ldots, r$, the node k_s is an out-neighbor of k_{s-1}, with $k_0 = j$. Formally,

$$P_j^{(r)} = \left\{ (k_1, \ldots, k_r) \mid k_s \in \mathcal{N}_{\text{out}}^{(1)}(k_{s-1}) \text{ for } s = 1, \ldots, r \right\}.$$

This definition ensures that each node in the r-hop neighborhood is reachable from node j through a sequence of consecutive immediate out-neighbors within $\rho \leq r$ steps.

This sequence-based characterization of r-hop neighborhoods provides a granular understanding of the pathways through which influence or information can propagate within the network, complementing the previous definitions based on adjacency matrices and shortest path lengths.

B Discussions

B.1 Practical applications of DICE

Decentralized Machine Unlearning. As concerns about data privacy and the right to be forgotten increase, the ability to remove specific data contributions from a trained model becomes important (Guo et al., 2020; Sekhari et al., 2021). In decentralized settings, retraining the model from scratch is often impractical for edge users with limited compute. The proximal influence measure enables participants to estimate the impact of removing a particular data instance from its neighbor. For example, by assessing the influence of $\boldsymbol{z}_j^t$ on neighbors, participants can adjust their local models to mitigate the effects of $\boldsymbol{z}_j^t$ without requesting full retraining of the whole decentralized learning system. This approach facilitates efficient and targeted unlearning procedures, avoiding costly system-wide retraining while respecting individual data privacy requests.

B.2 Additional related work

Clustered Federated Learning. Clustered Federated Learning (CFL) addresses the challenge of data heterogeneity by grouping clients with similar data distributions and training separate models for each cluster (Mansour et al., 2020; Ghosh et al., 2020; Sattler et al., 2021; Kim et al., 2024). Gradient-based CFL methods (Sattler et al., 2021; Kim et al., 2024) use client gradient similarities to form clusters, with Sattler et al. (2021) employing cosine similarity to recursively partition clients after convergence and Kim et al. (2024) dynamically applying spectral clustering to organize clients based on gradient features during training. These methods effectively capture direct, peer-to-peer gradient relationships to cluster clients with similar data-generating distributions. Both gradient-based CFL and the one-hop DICE estimator (see Proposition 1) utilize gradient similarity information. However, CFL is inherently limited to local interactions, as its gradient similarity metrics are confined to pairwise relationships. In contrast, DICE goes far beyond this scope by systematically quantifying the propagation of influence across multiple hops in a decentralized network. DICE introduces the first comprehensive framework to evaluate multi-hop influence cascades. Mathematically, Proposition 3 highlights how DICE generalizes peer-level gradient similarity into a non-trivial extension for decentralized networks. This includes incorporating key factors including network topology and curvature information, enabling a deeper understanding of how influence flows through the whole decentralized learning systems. A promising future direction is to explore the potential of DICE-E as a more advanced high-order gradient similarity metric for effectively clustering participants in decentralized federated learning.

C PROOF

C.1 PROOF OF PROPOSITION 1

Proposition 1 (Approximation of One-hop DICE-GT). The one-hop DICE-GT value (see Definition 2) can be linearly approximated as follow:

$$\mathcal{I}^{(1)}_{\text{DICE-E}}(\boldsymbol{z}^t_j, \boldsymbol{z}') = -\eta^t q_j \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j) - \eta^t \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} q_k \boldsymbol{W}^t_{k,j} \nabla L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j).$$

Proof. From Definition 2, the one-hop DICE-GT is given by:

$$\mathcal{I}^{(1)}_{\text{DICE-GT}}(\boldsymbol{z}^t_j, \boldsymbol{z}') = q_j \left(L(\boldsymbol{\theta}^{t+\frac{1}{2}}_j; \boldsymbol{z}') - L(\boldsymbol{\theta}^t_j; \boldsymbol{z}') \right) + \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} q_k \left(L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j}; \boldsymbol{z}') \right).$$

We approximate each term using first-order Taylor expansion.

For the first term:

$$\begin{aligned} L(\boldsymbol{\theta}^{t+\frac{1}{2}}_j; \boldsymbol{z}') - L(\boldsymbol{\theta}^t_j; \boldsymbol{z}') &\approx \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}')^\top (\boldsymbol{\theta}^{t+\frac{1}{2}}_j - \boldsymbol{\theta}^t_j) && \text{(C.1)} \\ &= -\eta^t \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j), && \text{(C.2)} \end{aligned}$$

where the last equality follows from the update rule $\boldsymbol{\theta}^{t+\frac{1}{2}}_j = \boldsymbol{\theta}^t_j - \eta^t \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j)$.

For the second term, for each $k \in \mathcal{N}^{(1)}_{\text{out}}(j)$:

$$L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j}; \boldsymbol{z}') \approx \nabla L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}')^\top (\boldsymbol{\theta}^{t+1}_k - \boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j}). \tag{C.3}$$

From the update rule in Algorithm 1, we have:

$$\boldsymbol{\theta}^{t+1}_k = \sum_{l \in \mathcal{N}_{\text{in}}(k)} \boldsymbol{W}^t_{k,l} \boldsymbol{\theta}^{t+\frac{1}{2}}_l \tag{C.4}$$

$$\boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j} = \boldsymbol{W}^t_{k,j} \boldsymbol{\theta}^t_j + \sum_{l \in \mathcal{N}_{\text{in}}(k) \setminus \{j\}} \boldsymbol{W}^t_{k,l} \boldsymbol{\theta}^{t+\frac{1}{2}}_l. \tag{C.5}$$

Thus, the difference becomes:

$$\begin{aligned} \boldsymbol{\theta}^{t+1}_k - \boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j} &= \boldsymbol{W}^t_{k,j} (\boldsymbol{\theta}^{t+\frac{1}{2}}_j - \boldsymbol{\theta}^t_j) \\ &= -\eta^t \boldsymbol{W}^t_{k,j} \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j). && \text{(C.6)} \end{aligned}$$

Therefore,

$$L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j}; \boldsymbol{z}') \approx -\eta^t \boldsymbol{W}^t_{k,j} \nabla L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j). \tag{C.7}$$

Combining the approximations, we obtain:

$$\mathcal{I}^{(1)}_{\text{DICE-E}}(\boldsymbol{z}^t_j, \boldsymbol{z}') = -\eta^t q_j \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j) - \eta^t \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} q_k \boldsymbol{W}^t_{k,j} \nabla L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}')^\top \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j). \tag{C.8}$$

This completes the proof. □

C.2 Proof of Proposition 2

Proposition 2 (Approximation of Two-hop DICE-GT). The two-hop DICE-GT influence $\mathcal{I}^{(2)}_{\text{DICE-E}}(\boldsymbol{z}^t_j, \boldsymbol{z}')$ (see Definition 3) can be approximated as follows:

$$\begin{aligned}&\mathcal{I}^{(2)}_{\text{DICE-E}}(\boldsymbol{z}^t_j, \boldsymbol{z}') = \mathcal{I}^{(1)}_{\text{DICE-E}}(\boldsymbol{z}^t_j, \boldsymbol{z}') \\ &- \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} \sum_{l \in \mathcal{N}^{(1)}_{\text{out}}(k)} \eta^t q_l \boldsymbol{W}^{t+1}_{l,k} \boldsymbol{W}^t_{k,j} \nabla L(\boldsymbol{\theta}^{t+2}_l; \boldsymbol{z}')^\top (\boldsymbol{I} - \eta^{t+1} \boldsymbol{H}(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}^{t+1}_k)) \nabla L(\boldsymbol{\theta}^t_j; \boldsymbol{z}^t_j),\end{aligned} \tag{C.9}$$

where $\boldsymbol{H}(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}^{t+1}_k)$ denotes the Hessian matrix of L with respect to $\boldsymbol{\theta}^{t+1}_k$ evaluated at $\boldsymbol{z}^{t+1}_k$.

Proof. According to Definition 2, the difference between the two-hop and one-hop DICE-GT influences can be expressed as follows:

$$\mathcal{I}^{(2)}_{\text{DICE-GT}}(\boldsymbol{z}^t_j, \boldsymbol{z}') - \mathcal{I}^{(1)}_{\text{DICE-GT}}(\boldsymbol{z}^t_j, \boldsymbol{z}') = \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} \sum_{l \in \mathcal{N}^{(1)}_{\text{out}}(k)} q_l \left(L(\boldsymbol{\theta}^{t+2}_l; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+2}_{l \setminus \boldsymbol{z}^t_j}; \boldsymbol{z}') \right), \tag{C.10}$$

where $\boldsymbol{\theta}^{t+2}_l$ is the parameter of participant l at iteration $t+2$, and $\boldsymbol{\theta}^{t+2}_{l \setminus \boldsymbol{z}^t_j}$ denotes the parameter without the influence of $\boldsymbol{z}^t_j$.

We approximate the loss difference with a first-order Taylor expansion:

$$L(\boldsymbol{\theta}^{t+2}_l; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+2}_{l \setminus \boldsymbol{z}^t_j}; \boldsymbol{z}') \approx \nabla L(\boldsymbol{\theta}^{t+2}_l; \boldsymbol{z}')^\top (\boldsymbol{\theta}^{t+2}_l - \boldsymbol{\theta}^{t+2}_{l \setminus \boldsymbol{z}^t_j}). \tag{C.11}$$

According to the update rule in Algorithm 1, we have

$$\boldsymbol{\theta}^{t+2}_l = \sum_{m \in \mathcal{N}_{\text{in}}(l)} \boldsymbol{W}^{t+1}_{l,m} \boldsymbol{\theta}^{t+\frac{3}{2}}_m, \tag{C.12}$$

$$\boldsymbol{\theta}^{t+2}_{l \setminus \boldsymbol{z}^t_j} = \sum_{m \in \mathcal{N}_{\text{in}}(l)} \boldsymbol{W}^{t+1}_{l,m} \boldsymbol{\theta}^{t+\frac{3}{2}}_{m \setminus \boldsymbol{z}^t_j} = \sum_{m \in \mathcal{N}^{(1)}_{\text{out}}(l) \setminus \{k | k \in \mathcal{N}^{(1)}_{\text{out}}(j)\}} \boldsymbol{W}^{t+1}_{l,m} \boldsymbol{\theta}^{t+\frac{3}{2}}_m + \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} \boldsymbol{W}^{t+1}_{l,k} \boldsymbol{\theta}^{t+\frac{3}{2}}_{k \setminus \boldsymbol{z}^t_j}, \tag{C.13}$$

where k includes all intermediate neighbors connecting participants j and l. The influence of $\boldsymbol{z}^t_j$ only propagate through intermediate neighbors k to l.

The difference in Equation (C.11) then becomes

$$\begin{aligned}&\boldsymbol{\theta}^{t+2}_l - \boldsymbol{\theta}^{t+2}_{l \setminus \boldsymbol{z}^t_j} \\ &= \sum_{m \in \mathcal{N}^{(1)}_{\text{in}}(l)} \boldsymbol{W}^{t+1}_{l,m} \boldsymbol{\theta}^{t+\frac{3}{2}}_m - \left(\sum_{m \in \mathcal{N}^{(1)}_{\text{out}}(l) \setminus \{k | k \in \mathcal{N}^{(1)}_{\text{out}}(j)\}} \boldsymbol{W}^{t+1}_{l,m} \boldsymbol{\theta}^{t+\frac{3}{2}}_m + \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} \boldsymbol{W}^{t+1}_{l,k} \boldsymbol{\theta}^{t+\frac{3}{2}}_{k \setminus \boldsymbol{z}^t_j} \right) \\ &= \sum_{k \in \mathcal{N}^{(1)}_{\text{out}}(j)} \boldsymbol{W}^{t+1}_{l,k} (\boldsymbol{\theta}^{t+\frac{3}{2}}_k - \boldsymbol{\theta}^{t+\frac{3}{2}}_{k \setminus \boldsymbol{z}^t_j}).\end{aligned} \tag{C.14}$$

Next, we approximate $\boldsymbol{\theta}^{t+\frac{3}{2}}_k - \boldsymbol{\theta}^{t+\frac{3}{2}}_{k \setminus \boldsymbol{z}^t_j}$. Considering the update at participant k:

$$\boldsymbol{\theta}^{t+\frac{3}{2}}_k = \boldsymbol{\theta}^{t+1}_k - \eta^{t+1} \nabla L(\boldsymbol{\theta}^{t+1}_k; \boldsymbol{z}^{t+1}_k), \tag{C.15}$$

$$\boldsymbol{\theta}^{t+\frac{3}{2}}_{k \setminus \boldsymbol{z}^t_j} = \boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j} - \eta^{t+1} \nabla L(\boldsymbol{\theta}^{t+1}_{k \setminus \boldsymbol{z}^t_j}; \boldsymbol{z}^{t+1}_k). \tag{C.16}$$

Therefore, we have

$$\boldsymbol{\theta}_k^{t+\frac{3}{2}} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+\frac{3}{2}} = (\boldsymbol{\theta}_k^{t+1} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+1}) - \eta^{t+1}\left(\nabla L(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1}) - \nabla L(\boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+1}; \boldsymbol{z}_k^{t+1})\right). \tag{C.17}$$

Applying the first-order Taylor expansion for the gradient difference yields

$$\nabla L(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1}) - \nabla L(\boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+1}; \boldsymbol{z}_k^{t+1}) \approx \boldsymbol{H}(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1})(\boldsymbol{\theta}_k^{t+1} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+1}). \tag{C.18}$$

Substituting back, we have

$$\boldsymbol{\theta}_k^{t+\frac{3}{2}} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+\frac{3}{2}} \approx (\boldsymbol{I} - \eta^{t+1}\boldsymbol{H}(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1}))(\boldsymbol{\theta}_k^{t+1} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+1}). \tag{C.19}$$

The difference $\boldsymbol{\theta}_k^{t+1} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+1}$ can be approximated as

$$\begin{aligned}
\boldsymbol{\theta}_k^{t+1} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+1} &= \sum_{m\in\mathcal{N}_{\text{in}}(k)} \boldsymbol{W}_{k,m}^t \boldsymbol{\theta}_m^{t+\frac{1}{2}} - \left(\sum_{m\in\mathcal{N}_{\text{in}}(k)\setminus\{j\}} \boldsymbol{W}_{k,m}^t \boldsymbol{\theta}_m^{t+\frac{1}{2}} + \boldsymbol{W}_{k,j}^t \boldsymbol{\theta}_j^t\right) && \text{(C.20)}\\
&= \boldsymbol{W}_{k,j}^t(\boldsymbol{\theta}_j^{t+\frac{1}{2}} - \boldsymbol{\theta}_j^t) && \text{(C.21)}\\
&\approx -\boldsymbol{W}_{k,j}^t \eta^t \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}). && \text{(C.22)}
\end{aligned}$$

Therefore,

$$\boldsymbol{\theta}_k^{t+\frac{3}{2}} - \boldsymbol{\theta}_{k\setminus \boldsymbol{z}_j^t}^{t+\frac{3}{2}} \approx -(\boldsymbol{I} - \eta^{t+1}\boldsymbol{H}(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1}))\boldsymbol{W}_{k,j}^t \eta^t \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t). \tag{C.23}$$

Combining Equation (C.11), Equation (C.14) and Equation (C.23) yields

$$\begin{aligned}
&\mathcal{I}_{\text{DICE-E}}^{(2)}(\boldsymbol{z}_j^t, \boldsymbol{z}') - \mathcal{I}_{\text{DICE-E}}^{(1)}(\boldsymbol{z}_j^t, \boldsymbol{z}')\\
&= -\sum_{k\in\mathcal{N}_{\text{out}}^{(1)}(j)} \sum_{l\in\mathcal{N}_{\text{out}}^{(1)}(k)} \eta^t q_l \boldsymbol{W}_{l,k}^{t+1} \boldsymbol{W}_{k,j}^t \nabla L(\boldsymbol{\theta}_l^{t+2}; \boldsymbol{z}')^\top (\boldsymbol{I} - \eta^{t+1}\boldsymbol{H}(\boldsymbol{\theta}_k^{t+1}; \boldsymbol{z}_k^{t+1}))\nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t).
\end{aligned} \tag{C.24}$$

This completes the proof. □

C.3 Proof of Proposition 3

Proposition 3 (Approximation of r-hop DICE-GT). The r-hop DICE-GT influence $\mathcal{I}_{\text{DICE-E}}^{(r)}(\boldsymbol{z}, \boldsymbol{z}')$ (see Definition 3) can be approximated as follows:

$$\begin{aligned}
\mathcal{I}_{\text{DICE-E}}^{(r)}(\boldsymbol{z}, \boldsymbol{z}') = -\sum_{\rho=0}^{r} \sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} \eta^t q_{k_\rho} \left(\prod_{s=1}^{\rho} \boldsymbol{W}_{k_s,k_{s-1}}^{t+s-1}\right)\\
\nabla L(\boldsymbol{\theta}_{k_\rho}^{t+\rho}; \boldsymbol{z}')^\top \left(\prod_{s=2}^{\rho}\left(\boldsymbol{I} - \eta^{t+s-1}\boldsymbol{H}(\boldsymbol{\theta}_{k_s}^{t+s-1}; \boldsymbol{z}_{k_s}^{t+s-1})\right)\right)\nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t).
\end{aligned} \tag{C.25}$$

where $k_0 = j$, $P_j^{(\rho)}$ denotes the set of all sequences $(k_1, \ldots, k_\rho)$ such that $k_s \in \mathcal{N}_{\text{out}}^{(1)}(k_{s-1})$ for $s = 1, \ldots, \rho$ and $\boldsymbol{H}(\boldsymbol{\theta}_{k_s}^{t+s}; \boldsymbol{z}_{k_s}^{t+s})$ is the Hessian matrix of L with respect to $\boldsymbol{\theta}$ evaluated at $\boldsymbol{\theta}_{k_s}^{t+s}$ and data $\boldsymbol{z}_{k_s}^{t+s}$. For the cases when $\rho = 0$ and $\rho = 1$, the relevant product expressions are defined as identity matrices, thereby ensuring that the r-hop DICE-E remains well-defined.

Proof. From the definition of DICE-GT in Definition 3, the r-hop influence is given by:

$$\mathcal{I}^{(r)}_{\text{DICE-GT}}(\boldsymbol{z}, \boldsymbol{z}') = \sum_{\rho=0}^{r} \sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} q_{k_\rho} \left(L(\boldsymbol{\theta}^{t+\rho}_{k_\rho}; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t}; \boldsymbol{z}') \right), \tag{C.26}$$

where for $\rho = 0$, we have $k_0 = j$, and the term corresponds to the direct influence on participant j.

In the following, we will show that for arbitrary $\rho \in \mathbb{N}^+$,

$$\begin{aligned}
&\Delta\mathcal{I}^{(\rho)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') \triangleq \mathcal{I}^{(\rho)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') - \mathcal{I}^{(\rho-1)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') \\
&\approx - \sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} \eta^t q_{k_\rho} \left(\prod_{s=1}^{\rho} \boldsymbol{W}^{t+s-1}_{k_s,k_{s-1}} \right) \nabla L(\boldsymbol{\theta}^{t+\rho}_{k_\rho}; \boldsymbol{z}')^\top \left(\prod_{s=2}^{\rho} \left(\boldsymbol{I} - \eta^{t+s-1} \boldsymbol{H}^{t+s-1}_{k_s} \right) \right) \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t).
\end{aligned} \tag{C.27}$$

From the definition of DICE-GT for ρ hops, the incremental influence beyond $\rho - 1$ hops is given by:

$$\begin{aligned}
\Delta\mathcal{I}^{(\rho)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') &\triangleq \mathcal{I}^{(\rho)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') - \mathcal{I}^{(\rho-1)}_{\text{DICE-GT}}(\boldsymbol{z}_j^t, \boldsymbol{z}') \\
&= \sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} q_{k_\rho} \left(L(\boldsymbol{\theta}^{t+\rho}_{k_\rho}; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t}; \boldsymbol{z}') \right).
\end{aligned} \tag{C.28}$$

We approximate the loss difference using first-order Taylor expansion:

$$L(\boldsymbol{\theta}^{t+\rho}_{k_\rho}; \boldsymbol{z}') - L(\boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t}; \boldsymbol{z}') \approx \nabla L(\boldsymbol{\theta}^{t+\rho}_{k_\rho}; \boldsymbol{z}')^\top \left(\boldsymbol{\theta}^{t+\rho}_{k_\rho} - \boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t} \right). \tag{C.29}$$

Our goal is to express $\boldsymbol{\theta}^{t+\rho}_{k_\rho} - \boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t}$ in terms of the propagated influence from $\boldsymbol{z}_j^t$ at participant j through the path $(k_1, \ldots, k_\rho)$.

From the update rule in Algorithm 1, we have:

$$\boldsymbol{\theta}^{t+\rho}_{k_\rho} = \sum_{m\in\mathcal{N}_{\text{in}}(k_\rho)} \boldsymbol{W}^{t+\rho-1}_{k_\rho,m} \boldsymbol{\theta}_m^{t+\rho-\frac{1}{2}}, \tag{C.30}$$

$$\boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t} = \sum_{m\in\mathcal{N}_{\text{in}}(k_\rho)} \boldsymbol{W}^{t+\rho-1}_{k_\rho,m} \boldsymbol{\theta}_{m \backslash \boldsymbol{z}_j^t}^{t+\rho-\frac{1}{2}}. \tag{C.31}$$

Thus, the difference becomes

$$\boldsymbol{\theta}^{t+\rho}_{k_\rho} - \boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t} = \sum_{m\in\mathcal{N}_{\text{in}}(k_\rho)} \boldsymbol{W}^{t+\rho-1}_{k_\rho,m} \left(\boldsymbol{\theta}_m^{t+\rho-\frac{1}{2}} - \boldsymbol{\theta}_{m \backslash \boldsymbol{z}_j^t}^{t+\rho-\frac{1}{2}} \right). \tag{C.32}$$

However, only the predecessor participants $k_{\rho-1}$ are influenced by $\boldsymbol{z}_j^t$, so we have:

$$\begin{aligned}
&\boldsymbol{\theta}^{t+\rho}_{k_\rho} - \boldsymbol{\theta}^{t+\rho}_{k_\rho \backslash \boldsymbol{z}_j^t} \\
&= \boldsymbol{W}^{t+\rho-1}_{k_\rho,k_{\rho-1}} \left(\boldsymbol{\theta}_{k_{\rho-1}}^{t+\rho-\frac{1}{2}} - \boldsymbol{\theta}_{k_{\rho-1} \backslash \boldsymbol{z}_j^t}^{t+\rho-\frac{1}{2}} \right) \\
&= \boldsymbol{W}^{t+\rho-1}_{k_\rho,k_{\rho-1}} \left(\boldsymbol{\theta}_{k_{\rho-1}}^{t+\rho-1} - \boldsymbol{\theta}_{k_{\rho-1} \backslash \boldsymbol{z}_j^t}^{t+\rho-1} - \eta^{t+\rho-1} \left(\nabla L(\boldsymbol{\theta}_{k_{\rho-1}}^{t+\rho-1}; \boldsymbol{z}_{k_{\rho-1}}^{t+\rho-1}) - \nabla L(\boldsymbol{\theta}_{k_{\rho-1} \backslash \boldsymbol{z}_j^t}^{t+\rho-1}; \boldsymbol{z}_{k_{\rho-1}}^{t+\rho-1}) \right) \right).
\end{aligned} \tag{C.33}$$

Using a first-order Taylor expansion

$$\nabla L(\boldsymbol{\theta}_{k_{\rho-1}}^{t+\rho-1}; \boldsymbol{z}_{k_{\rho-1}}^{t+\rho-1}) - \nabla L(\boldsymbol{\theta}_{k_{\rho-1}\backslash \boldsymbol{z}_j^t}^{t+\rho-1}; \boldsymbol{z}_{k_{\rho-1}}^{t+\rho-1}) \approx \boldsymbol{H}_{k_{\rho-1}}^{t+\rho-1}\left(\boldsymbol{\theta}_{k_{\rho-1}}^{t+\rho-1} - \boldsymbol{\theta}_{k_{\rho-1}\backslash \boldsymbol{z}_j^t}^{t+\rho-1}\right), \tag{C.34}$$

we obtain

$$\boldsymbol{\theta}_{k_\rho}^{t+\rho} - \boldsymbol{\theta}_{k_\rho \backslash \boldsymbol{z}_j^t}^{t+\rho} \approx \boldsymbol{W}_{k_\rho, k_{\rho-1}}^{t+\rho-1}\left(\boldsymbol{I} - \eta^{t+\rho-1}\boldsymbol{H}_{k_{\rho-1}}^{t+\rho-1}\right)\left(\boldsymbol{\theta}_{k_{\rho-1}}^{t+\rho-1} - \boldsymbol{\theta}_{k_{\rho-1}\backslash \boldsymbol{z}_j^t}^{t+\rho-1}\right). \tag{C.35}$$

By unrolling the recursion, we obtain:

$$\boldsymbol{\theta}_{k_\rho}^{t+\rho} - \boldsymbol{\theta}_{k_\rho \backslash \boldsymbol{z}_j^t}^{t+\rho} \approx \left(\prod_{s=2}^{\rho} \boldsymbol{W}_{k_s, k_{s-1}}^{t+s-1}\left(\boldsymbol{I} - \eta^{t+s-1}\boldsymbol{H}_{k_{s-1}}^{t+s-1}\right)\right)\left(\boldsymbol{\theta}_{k_1}^{t+1} - \boldsymbol{\theta}_{k_1 \backslash \boldsymbol{z}_j^t}^{t+1}\right). \tag{C.36}$$

According to Equation (C.6), $\boldsymbol{\theta}_{k_1}^{t+1} - \boldsymbol{\theta}_{k_1\backslash \boldsymbol{z}_j^t}^{t+1} = -\eta^t \boldsymbol{W}_{k_1,j}^t \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t)$. Then we have:

$$\boldsymbol{\theta}_{k_\rho}^{t+\rho} - \boldsymbol{\theta}_{k_\rho \backslash \boldsymbol{z}_j^t}^{t+\rho} \approx \left(\prod_{s=1}^{\rho} \boldsymbol{W}_{k_s, k_{s-1}}^{t+s-1} \prod_{s=2}^{\rho}\left(\boldsymbol{I} - \eta^{t+s-1}\boldsymbol{H}_{k_{s-1}}^{t+s-1}\right)\right)(-\eta^t \nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t)). \tag{C.37}$$

Substituting back into the difference in DICD-GT:

$$\begin{aligned}
&\Delta \mathcal{I}_{\text{DICE-GT}}^{(\rho)}(\boldsymbol{z}_j^t, \boldsymbol{z}') \\
&\approx \sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} q_{k_\rho} \nabla L(\boldsymbol{\theta}_{k_\rho}^{t+\rho}; \boldsymbol{z}')^\top \left(\boldsymbol{\theta}_{k_\rho}^{t+\rho} - \boldsymbol{\theta}_{k_\rho\backslash \boldsymbol{z}_j^t}^{t+\rho}\right) \\
&\approx -\sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} \eta^t q_{k_\rho}\left(\prod_{s=1}^{\rho} \boldsymbol{W}_{k_s,k_{s-1}}^{t+s-1}\right) \nabla L(\boldsymbol{\theta}_{k_\rho}^{t+\rho}; \boldsymbol{z}')^\top \prod_{s=2}^{\rho}\left(\boldsymbol{I} - \eta^{t+s-1}\boldsymbol{H}_{k_{s-1}}^{t+s-1}\right)\nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t).
\end{aligned} \tag{C.38}$$

Summing over ρ from 0 to r in Equation (C.27), we obtain the final approximation:

$$\begin{aligned}
\mathcal{I}_{\text{DICE-E}}^{(r)}(\boldsymbol{z}, \boldsymbol{z}') = -\sum_{\rho=0}^{r} \sum_{(k_1,\ldots,k_\rho)\in P_j^{(\rho)}} \eta^t q_{k_\rho}\left(\prod_{s=1}^{\rho} \boldsymbol{W}_{k_s,k_{s-1}}^{t+s-1}\right) \\
\nabla L(\boldsymbol{\theta}_{k_\rho}^{t+\rho}; \boldsymbol{z}')^\top\left(\prod_{s=2}^{\rho}\left(\boldsymbol{I} - \eta^{t+s-1}\boldsymbol{H}(\boldsymbol{\theta}_{k_s}^{t+s-1}; \boldsymbol{z}_{k_s}^{t+s-1})\right)\right)\nabla L(\boldsymbol{\theta}_j^t; \boldsymbol{z}_j^t),
\end{aligned} \tag{C.39}$$

which completes the proof. □

D Additional Experiments

D.1 Details of Experimental Setup

We employ the vanilla mini-batch Adapt-Then-Communicate version of Decentralized SGD ((Lopes & Sayed, 2008), see Algorithm 1) with commonly used network topologies (Ying et al., 2021) to train three-layer MLPs (Rumelhart et al., 1986), three-layer CNNs (LeCun et al., 1998), and ResNet-18 (He et al., 2016) on subsets of MNIST (LeCun et al., 1998), CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and Tiny ImageNet (Le & Yang, 2015). The number of participants (one GPU as a participant) is set to 16 and 32, with each participant holding 512 samples. For sensitivity analysis, we evaluate the stability of results under hyperparameter adjustments. The local batch size is varied as 16, 64, and 128 per participant, while the learning rate is set as 0.1 and 0.01 without decay. The code will be made publicly available.

D.2 Influence Alignment

In this experiments, we evaluate the alignment between one-hop DICE-GT (see Definition 2) and its first-order approximation, one-hop DICE-E (see Proposition 1). One-hop DICE-E $\mathcal{I}^{(1)}_{\text{DICE-E}}(\mathcal{B}^t_j, \boldsymbol{z}')$ is computed as the sum of one-sample DICE-E within the mini-batch $\mathcal{B}^t_j$ thanks to the additivity (see Equation (4)). DICE-GT $\mathcal{I}_{\text{DICE-GT}^{(1)}}(\mathcal{B}^t_j, \boldsymbol{z}')$ is calculated by measuring the loss reduction after removing $\mathcal{B}^t_j$ from node j at the t-th iteration. In the following Figures, each plot contains 30 points, with each point representing the result of a single comparison of one-hop DICE-GT and the estimated influence DICE-E. Strong alignments of DICE-GT and DICE-E are observed across datasets (CIFAR-10, CIFAR-100 and Tiny ImageNet) and model architectures (CNN and MLP).

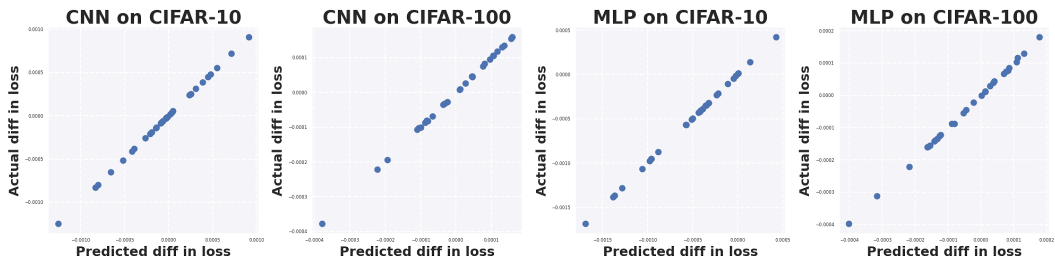

Figure D.1: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 16-node exponential graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.1.

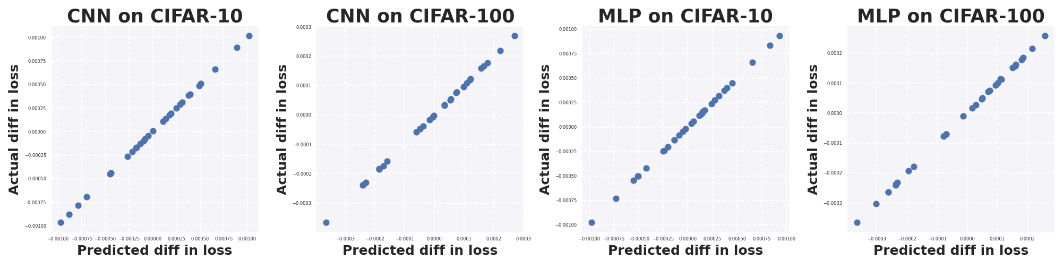

Figure D.2: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 32-node exponential graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.1.

We conduct additional sensitivity analysis experiments to evaluate the robustness of DICE-E under varying hyperparameters, including learning rate, batch size, and training epoch. These results demonstrate that DICE-E provides a strong approximation of DICE-GT, achieving consistent alignment across datasets (CIFAR-10 and CIFAR-100) and model architectures (CNN and MLP) under different batch sizes, learning rates, and training epochs.

D.2.1 Sensitivity Analysis on Batch Size

We conduct experiments to evaluate the robustness of DICE-E under varying batch sizes.

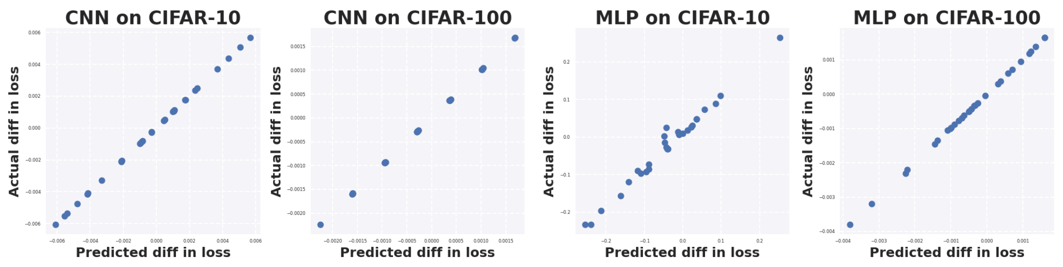

Figure D.3: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 32-node ring graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 16 and a learning rate of 0.1.

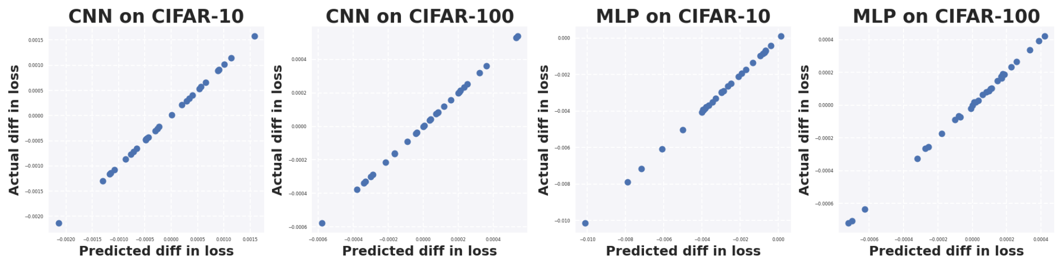

Figure D.4: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 32-node ring graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.1.

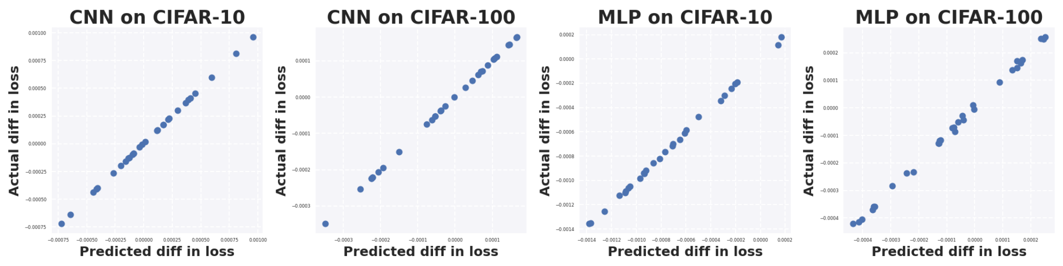

Figure D.5: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 32-node ring graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.1.

D.2.2 Sensitivity Analysis on Learning Rate and the Number of Nodes

We also condcut experiments to evaluate the robustness of DICE-E under varying learning rates and the number of nodes.

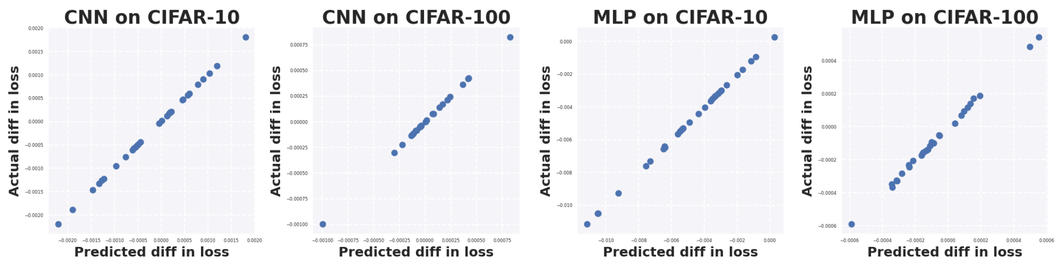

Figure D.6: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 16-node ring graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.1.

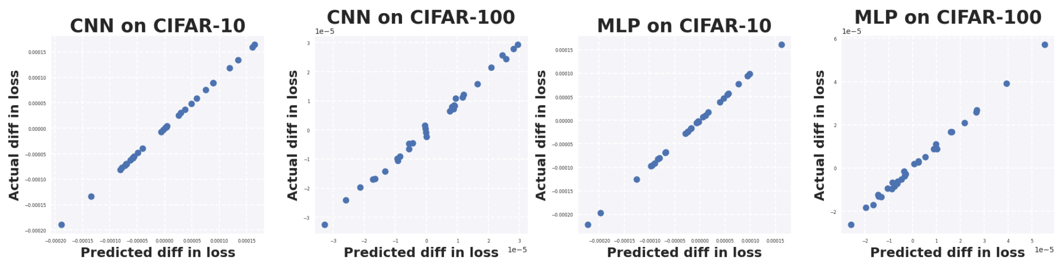

Figure D.7: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 16-node ring graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.01.

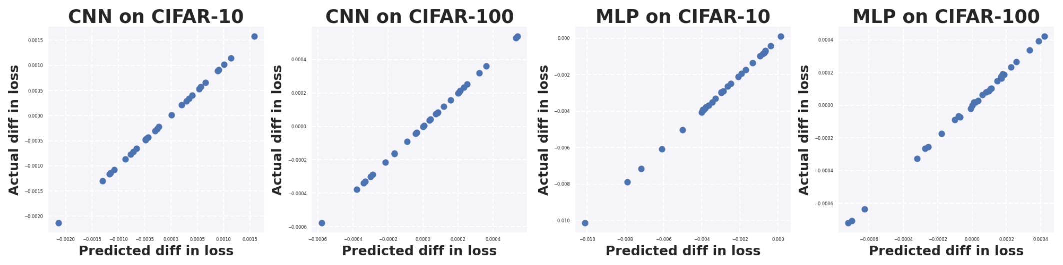

Figure D.8: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 32-node ring graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.1.

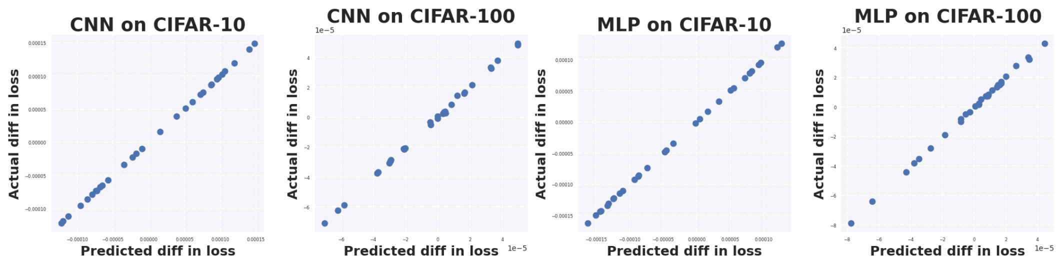

Figure D.9: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 32-node ring graph. Each node uses a 512-sample subset of CIFAR-10 or CIFAR-100. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.01.

D.2.3 Sensitivity Analysis on Training Epochs

We conduct experiments to evaluate the robustness of DICE-E under varying training epochs.

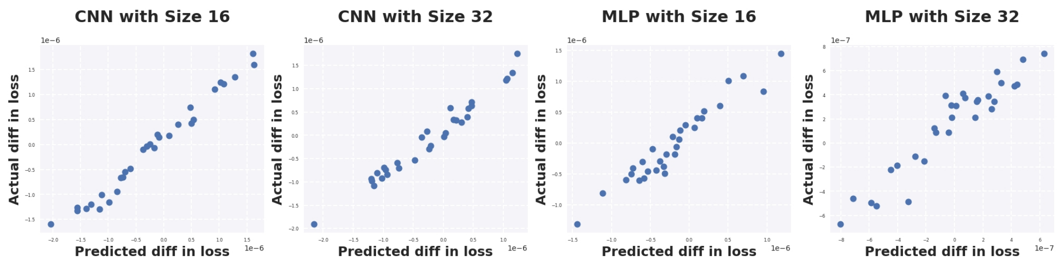

Figure D.10: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on 16 and 32-node exponential graphs. Each node uses a 8192-sample subset of Tiny ImageNet. Models are trained for 10 epochs with a batch size of 128 and a learning rate of 0.1.

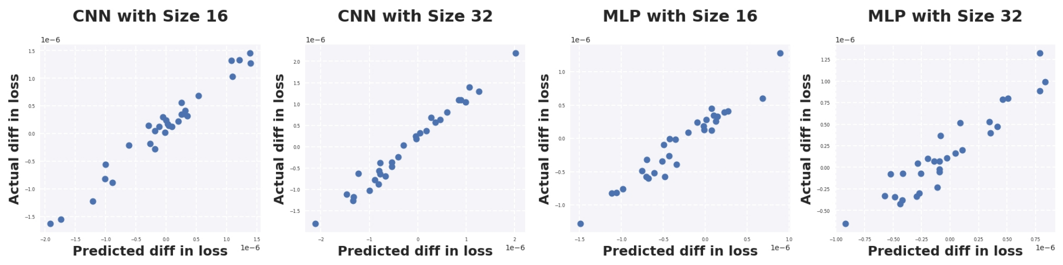

Figure D.11: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on 16 and 32-node exponential graphs. Each node uses a 8192-sample subset of Tiny ImageNet. Models are trained for 10 epochs with a batch size of 128 and a learning rate of 0.1.

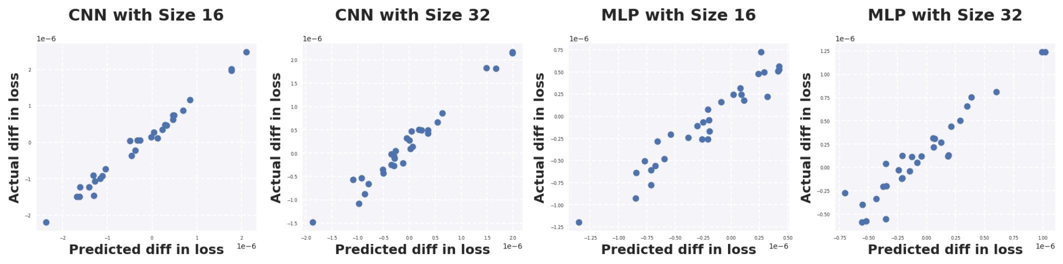

Figure D.12: Alignment between one-hop DICE-GT (vertical axis) and DICE-E (horizontal axis) on a 16 and 32-node exponential graph. Each node uses a 8192-sample subset of Tiny ImageNet. Models are trained for 10 epochs with a batch size of 128 and a learning rate of 0.1.

D.3 Anomaly Detection

We can also use the proximal influence metric to effectively detect anomalies. Specifically, anomalies are identified by observing significantly higher or lower proximal influence values compared to normal data instances. In our setup, anomalies are generated through random label flipping or by adding random Gaussian noise to features. The following Figures illustrates that the most anomalies (in red) is detectable with proximal influence values.

D.3.1 Random label flipping

We can conclude from these experiments that anomalies introduced through random label flipping are readily detectable by analyzing their proximal influence.

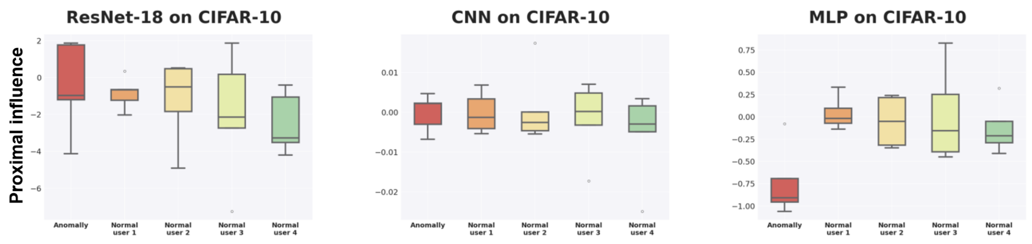

Figure D.13: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 16 and a learning rate of 0.1. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by random label flipping, while the other four are normal participants.

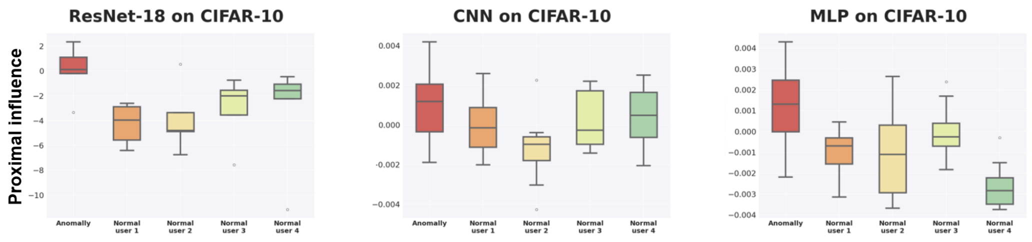

Figure D.14: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.1. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by random label flipping, while the other four are normal participants.

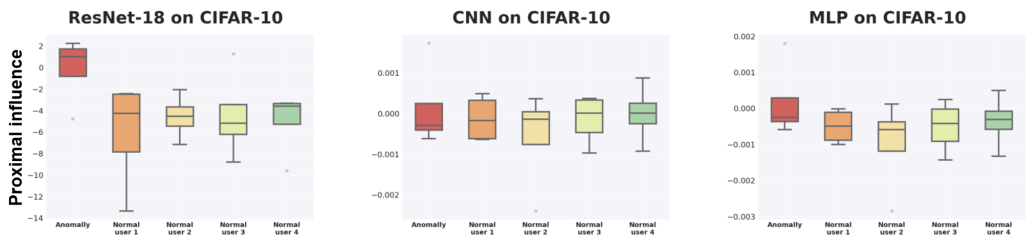

Figure D.15: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.1. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by random label flipping, while the other four are normal participants.

D.3.2 Feature Perturbations

We can conclude from Figure D.19, Figure D.20 and Figure D.21 that most anomalies introduced through adding zero-mean Gaussian noise with high variance are readily detectable by analyzing their proximal influence, which significantly deviates from that of normal data participants.

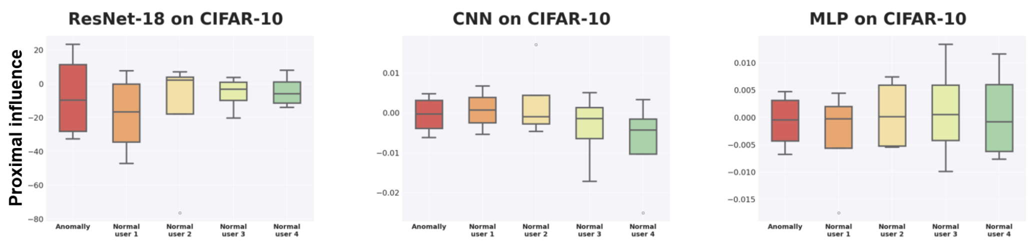

Figure D.16: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 16 and a learning rate of 0.01. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by random label flipping, while the other four are normal participants.

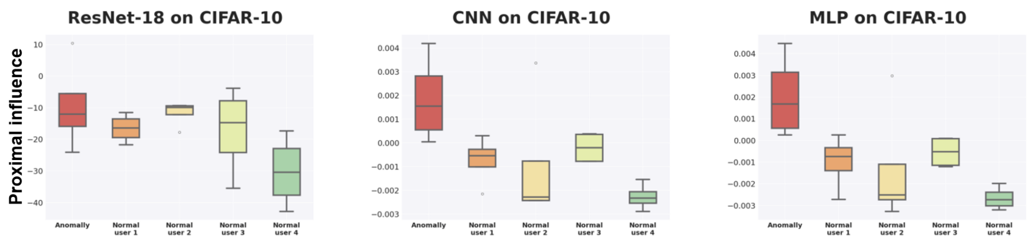

Figure D.17: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.01. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by random label flipping, while the other four are normal participants.

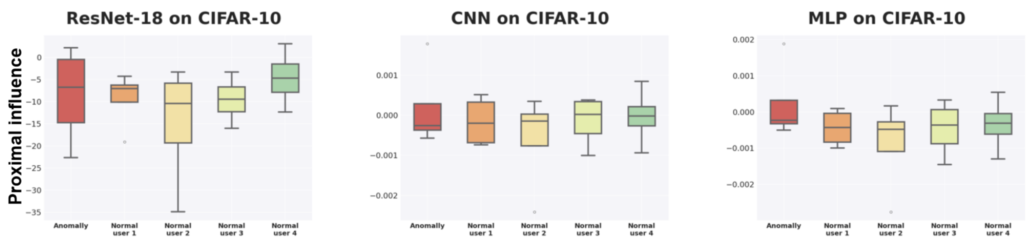

Figure D.18: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.01. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by random label flipping, while the other four are normal participants.

D.4 INFLUENCE CASCADES

The topological dependency of DICE-E in our theory reveals the "power asymmetries" (Blau, 1964; Magee & Galinsky, 2008) in decentralized learning. To support the theoretical finding, we examine the one-hop DICE-E values of the same batch on participants with vastly different topological importance. Figure 1 illustrates the one-hop DICE-E influence scores of an identical data batch across participants during decentralized training of a ResNet-18 model on the CIFAR-10 dataset. Node sizes represent the one-hop DICE-E influence scores, quantifying how a single batch impacts other participants in the network. The dominant nodes (e.g., those with larger outgoing communication weights in $\boldsymbol{W}$) exhibit significantly higher influence, as shown in Figure 1 and further detailed in Figure D.23 and Figure D.24. These visualizations underscore the critical role of topological

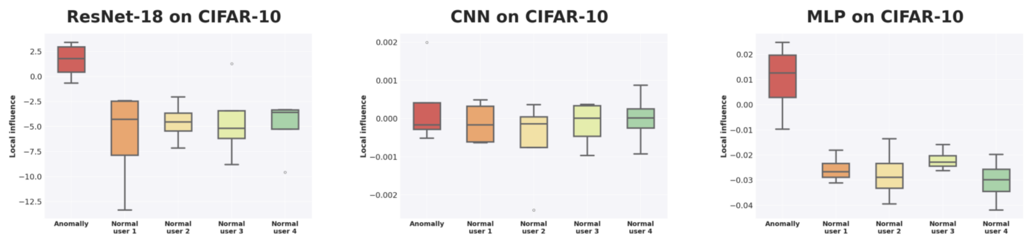

Figure D.19: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 128 and a learning rate of 0.1. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by adding zero-mean Gaussian noise with variance equals 100 on each feature, while the other four are normal participants.

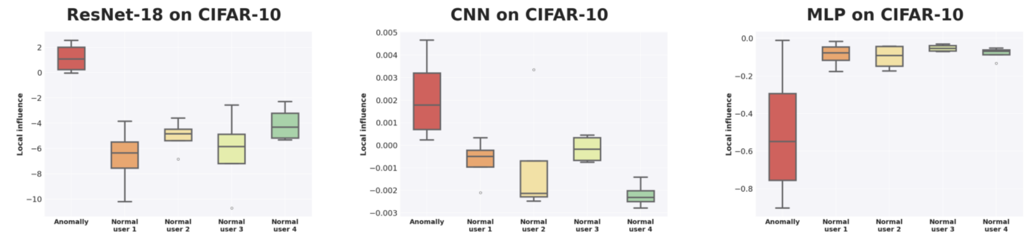

Figure D.20: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 64 and a learning rate of 0.01. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by adding zero-mean Gaussian noise with variance equals 100 on each feature, while the other four are normal participants.

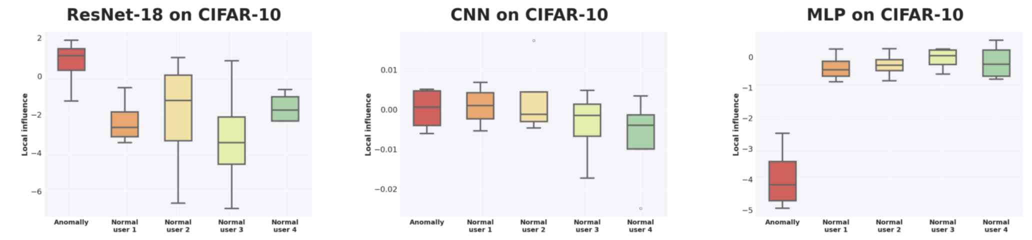

Figure D.21: Anomaly detection on exponential graph with 32 nodes. Each node uses a 512-sample subset of CIFAR-10. Models are trained for 5 epochs with a batch size of 16 and a learning rate of 0.1. In a 32-node exponential graph, each participant connects with 5 neighbors, where the neighbor in red is set as an anomaly by adding zero-mean Gaussian noise with variance equals 100 on each feature, while the other four are normal participants.

properties in shaping data influence in decentralized learning, demonstrating how the structure of the communication matrix $\boldsymbol{W}$ determines the asymmetries in influence.

To better observe and showcase the "influence cascade" phenomenon, we design a communication matrix with one "dominant" participant (node 0), two "subdominant" participants (nodes 7 and 10), and several other common participants. Figure D.22 (Left) visualizes the communication topology, where node sizes indicate out-degree, reflecting their influence, and edge thickness represents the strength of communication links. Node 0 stands out as the dominant participant with the largest size, while nodes 7 and 10 serve as subdominant intermediaries. Figure D.22 (Right) complements this by showing the adjacency matrix $\boldsymbol{W}$ as a heatmap, where the color intensity highlights the magnitude of connection strengths, with the dominant participant exhibiting strong outgoing links across the network. Together, these visualizations highlight the hierarchical structure and asymmetries in the communication matrix, crucial for understanding topological influences in decentralized learning.

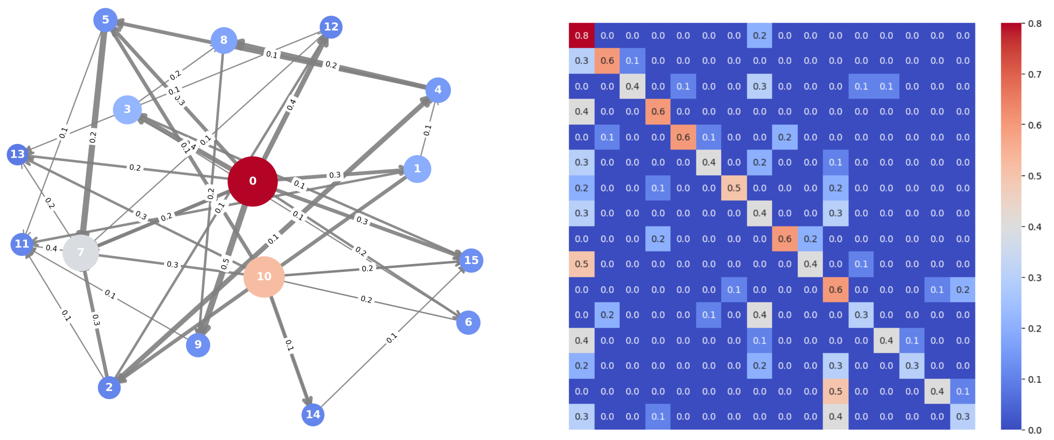

Figure D.22: **Left**: Visualization of the communication topology used in Subsection 5.3, where each node represents a participant, and edges indicate communication links. Node sizes are proportional to their out-degree (sum of outgoing edge weights), reflecting their communication influence within the community. Edge thickness corresponds to the strength of connection (i.e., weight), with directional arrows capturing the flow of information between participants. Self-loops are omitted for simplicity. **Right**: Heatmap representation of the weighted adjacency matrix $\boldsymbol{W}$ used in Subsection 5.3, where each entry $\boldsymbol{W}_{k,j}$ quantifies the communication strength from participant j to k. The color intensity represents the magnitude of the weights.

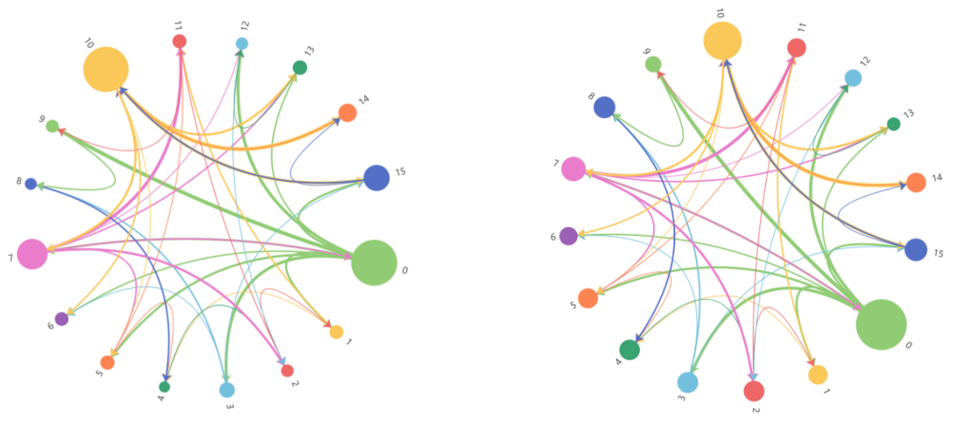

Figure D.23: Visualization of influence cascades during decentralized trainingg with MLP on MNIST (left) and CIFAR-10 (right) under a designed communication matrix (see Figure D.22). The thickness of edges represents the strength of communication links (i.e., weights in $\boldsymbol{W}$), while node sizes correspond to the relative one-hop DICE-E influence scores (see Proposition 1) computed for the same data batch across different participants. The numerical labels on the nodes indicate the corresponding participants, aligning with the participant indices in Figure D.22.

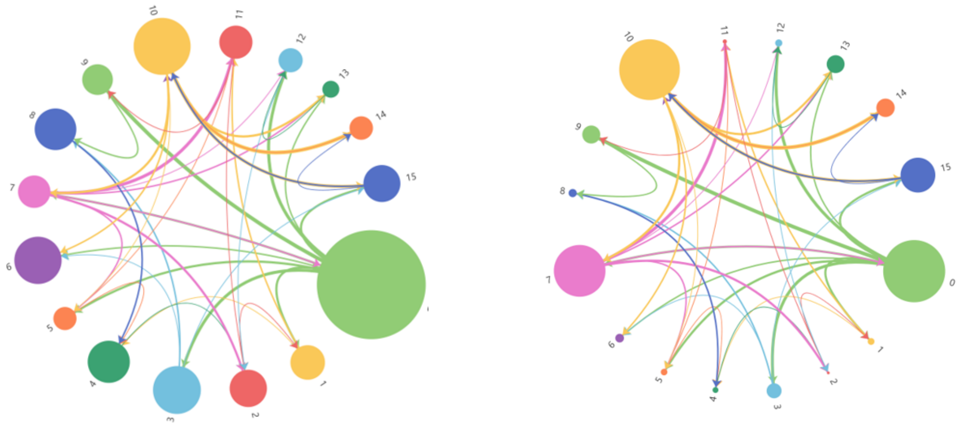

Figure D.24: Visualization of influence cascades during decentralized trainingg with ResNet-18 on CIFAR-10 (left) and CIFAR-100 (right) under a designed communication matrix (see Figure D.22). The thickness of edges represents the strength of communication links (i.e., weights in $\boldsymbol{W}$), while <u>node sizes</u> correspond to the relative one-hop DICE-E influence scores (see Proposition 1) computed for the same data batch across different participants. The numerical labels on the nodes indicate the corresponding participants, aligning with the participant indices in Figure D.22.