future internet

MDPI

Review

A Review and Experimental Evaluation on Split Learning

Zhanyi Hu [1,†], Tianchen Zhou [1,†], Bingzhe Wu [2], Cen Chen [1,*] and Yanhao Wang [1,*]

[1] School of Data Science and Engineering, East China Normal University, Shanghai 200062, China; 51255903110@stu.ecnu.edu.cn (Z.H.); tianchen@stu.ecnu.edu.cn (T.Z.)
[2] Tencent AI Lab, Shenzhen 518054, China; bingzhewu@tencent.com
* Correspondence: cenchen@dase.ecnu.edu.cn (C.C.); yhwang@dase.ecnu.edu.cn (Y.W.)
† These authors contributed equally to this work.

Abstract: Training deep learning models collaboratively on decentralized edge devices has attracted significant attention recently. The two most prominent schemes for this problem are Federated Learning (FL) and Split Learning (SL). Although there have been several surveys and experimental evaluations for FL in the literature, SL paradigms have not yet been systematically reviewed and evaluated. Due to the diversity of SL paradigms in terms of label sharing, model aggregation, cut layer selection, etc., the lack of a systematic survey makes it difficult to fairly and conveniently compare the performance of different SL paradigms. To address the above issue, in this paper, we first provide a comprehensive review for existing SL paradigms. Then, we implement several typical SL paradigms and perform extensive experiments to compare their performance in different scenarios on four widely used datasets. The experimental results provide extensive engineering advice and research insights for SL paradigms. We hope that our work can facilitate future research on SL by establishing a fair and accessible benchmark for SL performance evaluation.

Keywords: split learning; deep learning; edge computing

check for updates

Academic Editors: Qiang Duan and Zhihui Lu

Received: 16 January 2025
Revised: 9 February 2025
Accepted: 11 February 2025
Published: 13 February 2025

Citation: Hu, Z.; Zhou, T.; Wu, B.; Chen, C.; Wang, Y. A Review and Experimental Evaluation on Split Learning. *Future Internet* **2025**, *17*, 87. https://doi.org/10.3390/fi17020087

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Nowadays, a lot of edge devices, including sensors, smartphones, and wearables, are producing huge volumes of valuable data. Such data can be used to collectively train high-quality Deep Learning (DL) models through edge computing [1], which have broad real-world applications in various domains, including Industry 4.0 [2], healthcare [3], fintech [4], transportation [5], and energy [6]. However, these data are usually owned by organizations whose regulations do not allow for data sharing and exchange. For example, the exchange of sensitive raw patient data between healthcare providers for medical analysis can cause serious privacy concerns [7,8], thus having been prohibited by laws for privacy protection [9]. In such scenarios, data are scattered across different silos or edge devices and cannot be collected on a central server for the training of DL models.

To address the above problems, various schemes have been proposed to enable collaborative DL model training on decentralized silos or devices without raw data sharing and exchange [10,11]. Among these frameworks, Federated Learning (FL) [12] and Split Learning (SL) [13,14] are the most prominent. FL allows different clients to collaboratively train a shared DL model while keeping all training data locally and only exchanging gradients to ensure privacy. However, due to the demands of real-world applications, DL models have become larger and more complex to achieve satisfactory performance. Despite the increasing computational capabilities of modern hardware, training large-scale DL models on resource-constrained edge devices, such as sensors, smartphones, and wearables [15],

remains a great challenge. Consequently, SL is introduced as an alternative to FL for resource-constrained collaborative training, where a set of participants (i.e., clients) with limited resources can train a large-scale DL model with the assistance of a supercomputer center (i.e., server). The core idea of SL is to divide a large model into smaller client-side models and a server-side model. In this way, clients only perform lightweight computations and transmit intermediate-layer activations, rather than raw data, to the server to meet data regulatory requirements. Therefore, the key distinction between SL and FL is that clients hold the complete model in FL, whereas clients hold only a part of the complete model in SL.

In recent years, there have been several surveys and experimental evaluations for FL, as summarized in Table 1. These studies have covered multiple aspects of FL, including but not limited to types and methods of FL, privacy and security issues in FL, FL in edge environments, the design of incentive mechanisms, model aggregation and client selection methods for FL, and the applications of FL in healthcare, industrial engineering, intrusion detection, and other domains. However, although SL has also been extensively studied and there have been a variety of SL paradigms [13,16–18] in the literature, to the best of our knowledge, there are only a few surveys and evaluations for SL [19–21]. Furthermore, the surveys in [20,21] are specific to security and privacy issues in SL and do not provide comprehensive discussions and evaluations of SL paradigms. The evaluations in [19], which were published in 2020, focus more on FL than SL and do not compare the more recent FL and SL paradigms.

Table 1. Summary of surveys and experimental evaluations on FL and SL topics.

Domain	Topic	References	Summary of Contributions
FL	Evaluation	[22]	Compare the performance of different FL methods.
	Concept and Method	[23–25]	Provide an overview of basic FL types and methods.
	Application	[26]	Review the application of FL in the industrial engineering domain.
	Privacy and Security	[27,28]	Discuss the security and privacy issues in FL.
	Framework	[29–33]	Provide a general framework and benchmark for FL.
	Edge Computing	[34]	Discuss the design of FL methods in the Internet of Things.
	Method and Application	[35]	Outline FL types, methods, and perspective applications.
	Application	[36]	Discuss the applications and future prospects of FL in the healthcare domain.
	Mechanism Design	[37]	Review the incentive mechanisms designed for FL.
	Mechanism Design	[38]	Review the model aggregation mechanisms for FL.
	Application	[39,40]	Discuss the application of FL in intrusion detection.
	Edge Computing	[41]	Review lightweight FL methods on resource-constrained devices.
	Privacy and Security	[42]	Discuss data and model poisoning backdoor attacks and defenses for FL.
SL	Edge Computing	[19]	Compare SL and FL methods for the Internet of Things.
	Privacy and Security	[20]	Discuss the issues about data leakage and protection in SL.
	Privacy and Security	[21]	Evaluate the robustness of SL paradigms against model inversion attacks.
	Concept and Method; Evaluation	This Work	Review SL concepts and paradigms; Compare the performance of different SL methods.

Therefore, several critical problems for SL research have not yet been well addressed, as listed below.

- Lack of Systematic Review and Categorization of SL Paradigms: SL paradigms are highly diverse in terms of label sharing, model aggregation, cut layer selection, etc. There has not been a systematic review and categorization of the existing SL paradigms. In particular, paradigms that combine FL with SL or even obscure the distinctions between Vertical FL (VFL) [43] and SL cannot be precisely categorized in the existing literature. This poses challenges for researchers to compare SL paradigms, who often find themselves having to reinvent the wheel.

- Lack of SL Benchmarks and Evaluations: Recent years have witnessed rapid advancements in SL. However, the lack of benchmarks and evaluations has made it difficult to fairly compare the performance of SL paradigms. Existing SL paradigms are often evaluated in datasets from various domains, including vision [13], healthcare [14], and finance [44]. Furthermore, data partitioning methods may vary from one study to another. For example, for SL in the non-IID setting, some studies divide the dataset based on the labels of samples, while others utilize the Dirichlet distribution. A few existing studies do not even evaluate the performance of their paradigms on non-IID data. Therefore, in Table 2, we first summarize the main datasets used in recent SL studies and the partitioning methods on these datasets.

Table 2. Overview of datasets and partitioning methods used in recent SL studies.

Domain	Dataset Name	Data Partitioning	References
Vision	MNIST [45]	IID IID and Non-IID	[13,46–48] [18,49–56]
	Fashion-MNIST [57]	IID IID and Non-IID	[18,46,58] [18,50–52,55,59,60]
	CIFAR [61]	IID IID and Non-IID	[13,18,46–48,62–64] [17,50–54,59,65–70]
	HAM1000 [71]	IID IID and Non-IID	[46] [67]
	CINIC-10 [72]	IID and Non-IID	[67,69]
	ImageNet [73]	IID	[13,47]
Healthcare	CheXpert [74]	IID IID and Non-IID	[14] [16,75]
	ECG [76]	IID IID and Non-IID	[77] [19,52]
	Diabetic Retinopathy [78]	IID	[14]
Finance	Bank [79]	IID	[80]
	Adult [81]	IID	[80]
Graph	CORA [82]	IID and Non-IID	[83]
	PubMed [84]	IID and Non-IID	[83]

In this paper, we provide a systematic survey and comparison of the existing SL paradigms. We first classify existing SL paradigms along different dimensions and provide a detailed discussion on each of them. Our selection criteria focus on whether the SL paradigm demonstrates effective solutions in reducing communication and computation costs, ensuring label protection and addressing data and computation-communication heterogeneity. To comprehensively evaluate the performance of these SL paradigms, we performed extensive experiments on four widely used datasets that cover a wide range of tasks including image classification, medical data analysis, tabular prediction, and drug discovery in both IID and non-IID settings. By analyzing the experimental results, we provide extensive engineering advice for ML developers to choose appropriate SL paradigms according to their specific needs. We also provide research insights on the robustness and efficiency of existing SL paradigms for future improvements. The main contributions of this paper are summarized as follows:

- Systematic Review of Existing SL Paradigms: We review existing SL paradigms and classify them according to their characteristics. Generally, we categorize them into three types: model-split-only, weight aggregation-based, and intermediate data

aggregation-based approaches. We annotate them along four dimensions and provide suggestions for which scenarios are suitable for each of them.

- Detailed Experimental Evaluation of Existing SL Paradigms in Different Scenarios: We conducted extensive experiments to compare existing SL paradigms in different scenarios using the SLPerf framework.

In the remainder of this paper, we first review the SL paradigms in Section 2. In addition, we present an extensive experimental evaluation of SL paradigms in Section 3. Finally, we conclude the entire article in Section 4.

2. Taxonomy and Review of SL Paradigms

In this section, we provide a taxonomy and review of SL paradigms. To better distinguish them from FL, the SL paradigms that we consider generally involve a model split between the server and clients. However, with advances in SL and FL, several FL paradigms have incorporated some SL semantics within the FL framework, such as dynamically adapting the model size through pruning during training [85]. Therefore, we also include some FL paradigms that use submodels during training in our discussion. To comprehensively introduce these paradigms, we categorize them into three types based on whether intermediate data (e.g., smashed data or gradients) or model weights are aggregated during the training process. The first type, i.e., model split-only approach, refers to the SL paradigms where the model architecture is partitioned across different participants, but no data or model weight aggregation is performed during training. The second type, referred to as weight aggregation-based approach, utilizes techniques such as FedAvg [12] to aggregate model weights from different client devices to obtain a global model. The third type, called intermediate data aggregation-based approach, aggregates intermediate data from different client devices, such as gradients or smashed data. By categorizing SL paradigms into these three types, our goal is to provide a clear and concise understanding of different approaches to their design ideas.

According to the above classification, we conduct a comprehensive comparison of the SL paradigms in Table 3, which lists the main SL paradigms and compares them along four dimensions. Here, "Commun. Cost Reduction" represents whether a paradigm reduces communication costs between clients and the server; "Label Protection" indicates whether a paradigm can prevent label leakage; "Data Heterogeneity" refers to whether a paradigm mitigates model performance degradation due to data heterogeneity between different clients; and "Commun. or Comput. Heterogeneity" indicates whether a paradigm considers training time variations among clients caused by differences in communication bandwidth or computation resources. Next, we review the three types of SL paradigm in more detail.

2.1. Model Split-Only Approaches

A typical paradigm of this type is Vanilla Split Learning (VSL) [13], as shown in Figure 1, which partitions the model into two submodels, namely the server-side model θ^s and the client-side model θ^c. As such, the complete model is represented as $\theta = \theta^s \circ \theta^c$. Assuming that there are K clients, the local dataset for the k-th client ($k \in [1, \ldots, K]$) is denoted as $\mathcal{D}_k$, and , thus, the total dataset is $\mathcal{D} = \cup_{k=1}^{K} \mathcal{D}_k$. The client model θ^c is shared among all clients. When the k-th client training is performed, it uses θ^c and samples (X_k, y_k) from its local dataset $\mathcal{D}_k$ to compute the model output $\mathbf{z}_k = \theta^c(X_k)$, which is called the smashed data [16,46,86], to transmit to the server. The server, in turn, uses $\mathbf{z}_k$ to compute the loss $\mathcal{L}_k$, update the server model, and send the backward propagated gradients ($\mathbf{g}_k = \bigtriangledown \mathcal{L}_k$) back to the client for updating θ^c. Once all samples in $\mathcal{D}_k$ have been used, the weights of θ^c are transmitted to the next client for training.

Table 3. Comparison of existing SL paradigms along four dimensions. Here, "√" and "×" indicate whether an SL paradigm takes into account the corresponding dimension in its design or not.

Type	Paradigm	Commun. Cost Reduction	Label Protection	Data Heterogeneity	Commun. or Comput. Heterogeneity
Model Split-only	Vanilla [13]	×	×	×	×
	U-shape [13]	×	√	×	×
	PSL [49]	×	×	×	×
	AsyncSL [86]	√	×	×	×
	PC-SSL [87]	×	√	×	×
	COMSPLIT [88]	√	×	×	√
	deprune [89]	√	×	×	×
	CARD [90]	√	×	×	√
Weight Aggregation-based	SplitFed [46]	×	×	√	×
	FSL [91]	√	√	×	×
	ParallelSFL [92]	√	×	√	√
	FeSTA [16]	×	√	×	×
	FeSViBS [93]	×	√	×	×
	3C-SL [17]	√	×	×	×
	HSFL [94]	√	×	×	√
	CPSL [18]	√	×	×	√
	DFL [69]	√	×	×	√
	CHEESE [52]	√	×	×	√
	SplitFedZip [95]	√	×	×	×
	SplitLLM [96]	√	×	×	×
	FjORD [85]	√	×	√	√
	SAFL [97]	√	×	√	×
Intermediate Data Aggregation-based	SGLR [58]	×	×	√	×
	LocFedMix-SL [98]	×	×	×	×
	CutMixSL [59]	√	×	×	×
	Mix2SFL [60]	√	×	√	×
	TTSL [99]	√	×	×	×

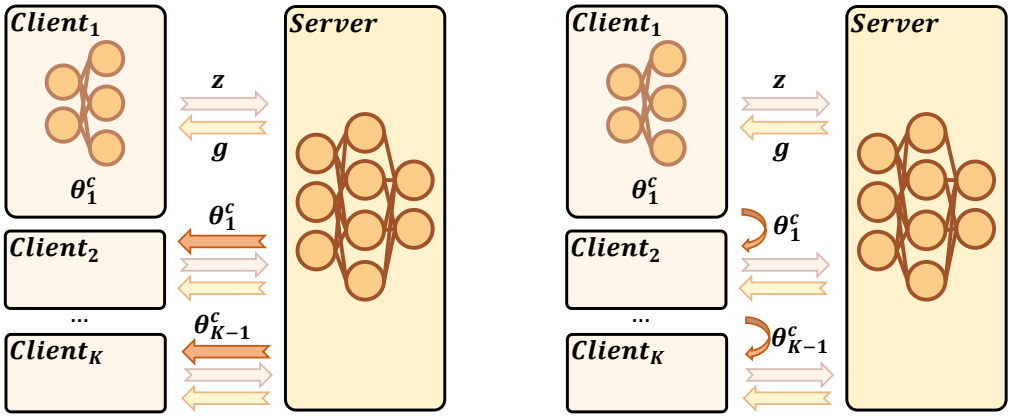


Figure 1. Illustration of the two VSL paradigms: (**a**) VSL (centralized) and (**b**) VSL (peer-to-peer).

Since each client must wait for the previous client to complete a training round before interacting with the server, it is important to note that, in the vanilla paradigm, the clients are trained synchronously. Based on how the next client acquires the global client model weights, the vanilla paradigm can be further divided into two paradigms:

- Centralized: The client uploads the model weights to the server or a trusted third party. The next client then downloads the model weights and proceeds to train the model, as illustrated in Figure 1a.

- Peer-to-Peer: The server sends the address of the previous client to the current client to transfer the model weights, as depicted in Figure 1b.

Synchronous training in VSL can result in significant delays. To address this issue, the Parallel Split Learning (PSL) paradigm [49] has been proposed, as shown in Figure 2a. In PSL, the weights of the client model are not shared. The k-th client stores its own local model θ_k^c. Therefore, all clients can initiate the forward propagation process and transmit the smashed data $\mathbf{z}_k$ to the server in parallel. Once the forward and backward propagation processes are completed, the server returns the gradients $\mathbf{g}_k$ derived from $\mathbf{z}_k$ to the k-th client, thereby reducing the training time. Therefore, the objective function in PSL can be denoted as

$$\min_{\theta^s, \mathbf{\Theta}^c} \sum_{k=1}^{K} \mathbb{E}_{\mathcal{D}_k} \mathcal{L}(\theta^s(\theta_k^c(X_k)), y_k), \tag{1}$$

where $\mathbf{\Theta}^c = \{\theta_k^c | k \in [K]\}$.

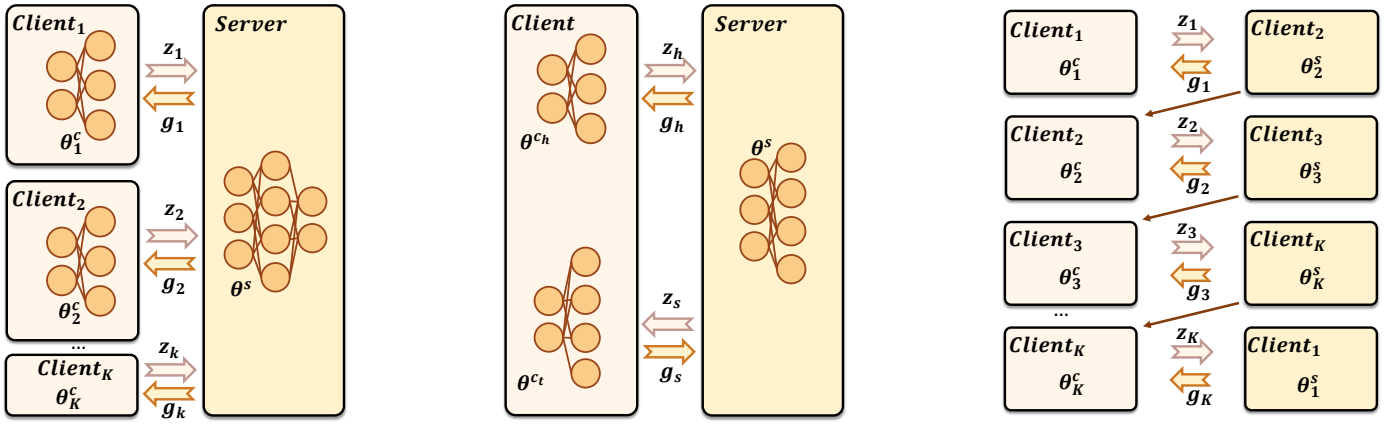


Figure 2. Illustration of three model split-only SL paradigms: (**a**) PSL; (**b**) U-shaped; and (**c**) PC-SSL.

In both VSL and PSL, clients send the labels of the samples to the server, which may cause label leakage. To address this problem, the U-shaped paradigm [13] has been introduced. In this paradigm, as shown in Figure 2b, the original model is divided into three parts: head θ^{c_h}, body θ^s, and tail θ^{c_t}. θ^{c_h} and θ^{c_t} are on the client side, while θ^s is on the server side. The client only sends smashed data $\mathbf{z}_h$ to the server; the server receives and forwards the smashed data for forward propagation and then sends the computed results back to the client. The client uses θ^{c_t} to calculate the loss and gradient, updates the tail model with backpropagation, and sends the gradient to the server. The server updates θ^s and finally sends the backpropagation result to the client to update θ^{c_h}. The above process ensures that only smashed data and gradients, rather than labels, are shared between clients and the server, thus avoiding information leakage caused by label sharing.

Due to the exchange of smashed data and gradients between clients and the server in each batch, the communication cost of the U-shaped paradigm can be quite high. To address this issue, the Split Learning using Asynchronous Training (AsyncSL) paradigm [86] is proposed, which is a loss-based asynchronous training scheme that updates the client-side model less frequently and only sends/receives smashed data/gradients in selected epochs. The process involves a server state, as shown in Figure 3a. In State $\mathcal{A}$, the clients send smashed data to the server and the server sends the gradient back. In State $\mathcal{B}$, only the client sends smashed data to the server and the server does not send the gradient back. In State $\mathcal{C}$, no information exchange occurs between the clients and the server. The state transition depends on the change in the loss function $\Delta\mathcal{L}$, as shown in Figure 3a.

Also, to reduce communication costs, the 3C-SL paradigm that combines circular convolution [100] with SL is proposed in [17]. As shown in Figure 3b, in this paradigm, clients use cyclic convolution to compress multiple features into a single one, which is decoded on the server side through cyclic correlation. Although this process may introduce noise, its impact on performance is mostly negligible. Sometimes, it can even result in regularization effects and improve performance. Similarly, deprune [89] is an adaptive compression-aware SL paradigm designed to improve communication efficiency. This paradigm utilizes pruning and transfer learning to train models with different compression levels for the data being transferred during SL.

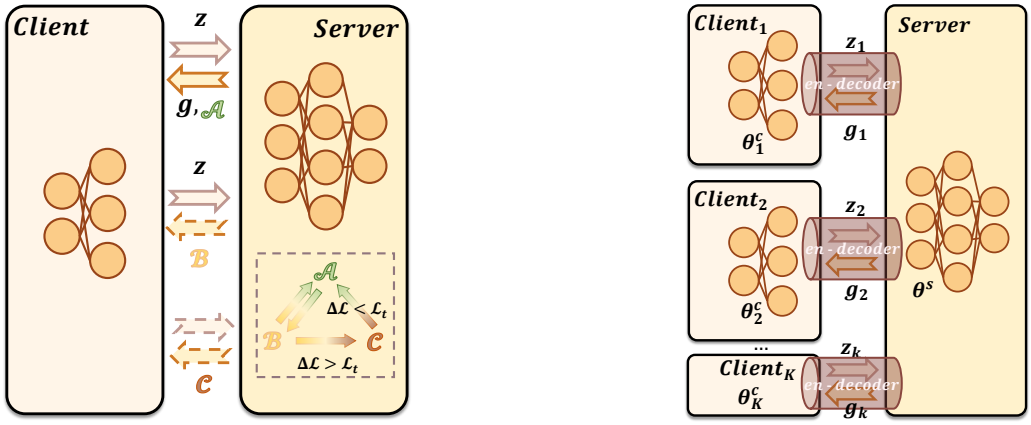


Figure 3. Illustration of the two model split-only SL paradigms with communication cost reduction: (**a**) AsyncSL and (**b**) 3C-SL. In (**a**), the dotted box indicates the process of deciding whether to transmit the data and the arrows with dotted margins denotes the transmissions that are not performed.

The motivation of the Peer-Coordinated Sequential Split Learning (PC-SSL) paradigm [87] is to enable each client to independently make accurate and generalized decisions without relying on communication with the server. Therefore, as shown in Figure 2c, the training process in this paradigm progresses sequentially from the first to the last user. And a cycle ends as the last user becomes the next user. Each step between a user's client and server reflects traditional local training; each step between the server of a user and the client of the next user represents a vanilla peer-to-peer SL.

COMSPLIT [88] is a communication-aware SL paradigm designed for heterogeneous IoT environments. It enhances SL by adapting to various wireless channel conditions, integrating RNNs for time-series data, and supporting devices with different computational capacities. With an early exit strategy and improved channel modeling, COMSPLIT ensures robust and efficient SL that is demonstrated by water quality monitoring applications.

The rapid development of Large Language Models (LLMs) marks a breakthrough in AI. A few studies have considered how to train LLMs in scenarios with limited computational resources using SL. Li et al. [90] proposed CARD, a method that combines LoRA-based fine-tuning with SL. They performed a detailed analysis of training delay and energy consumption factors for CARD. Consequently, they formulated an optimization problem to minimize both the training delay and the energy consumption of the server. The problem is then decomposed into multiple sub-problems, and the selection of LLM split layers and server resource allocation are jointly determined.

2.2. Weight Aggregation-Based Approach

The integration of SL and FL aims to capitalize on the strengths of both schemes. Since SL is adept at handling resource-limited clients, such as IoT devices, it divides the entire

model into smaller submodels that can be trained independently on the server and clients. Meanwhile, FL excels at addressing non-IID data scenarios. Based on this observation, numerous weight aggregation-based SL paradigms have been proposed. These paradigms train a global client model by combining locally trained models from clients using weight aggregation methods such as FedAvg [12]. In these paradigms, throughout the training process, some or all of the clients send their submodels to the server or a trusted third party for weight aggregation at specific stages.

In SplitFed [46], the clients and the server train their models on their local datasets, similar to what occurs in PSL. Multiple clients train their models in parallel, and after each client finishes a training round, its local model weights are transmitted to the server and aggregated using FedAvg [12] as follows:

$$\theta^c = \sum_{k=1}^{K} \frac{|\mathcal{D}^c|}{|\mathcal{D}|} \theta_k^c.$$

The process is illustrated in Figure 4a. Thapa et al. [46] also propose a variant of SplitFed, SplitFedv2, where clients perform the training round linearly, rather than in parallel. In PPSFL [55], a Group Normalization layer is added to the SplitFed paradigm, which does not participate in aggregation, to prevent privacy leakage.

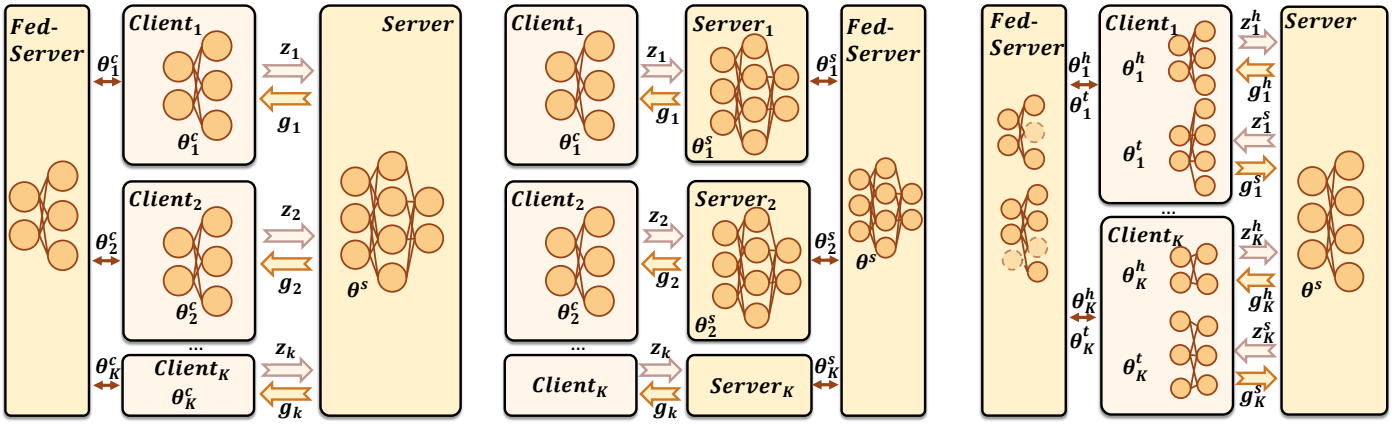


Figure 4. Illustration of weight aggregation-based SL paradigms: (**a**) SplitFed; (**b**) FSL; and (**c**) FeSTA.

In Federated Split Learning (FSL) [91], each client trains its local model using an SL paradigm in conjunction with its corresponding edge server, rather than interacting with a central server. After one epoch has been completed, the model weights of the edge server are aggregated via the central server. In particular, the key distinction between SplitFed and FSL pertains to whether the aggregated model corresponds to the client side or the edge server side of the model. The process is illustrated in Figure 4b. VehicularFSL [101] applies FSL to vehicular networks, where each vehicle (client) collaborates with its nearest roadside infrastructure for training. Considering the statistical and system heterogeneity in edge systems, assigning appropriate working nodes to specific clusters for efficient model training is a challenge. ParallelSFL [92], built upon SplitFed and FSL, develops an effective clustering strategy by optimizing a utility function related to training efficiency and model accuracy. It further addresses the issue of system heterogeneity by assigning different and appropriate local update frequencies to each cluster.

With the concept of U-shaped paradigms and using the Vision Transformer (ViT) [102], the Federated Split Task Agnostic (FeSTA) paradigm [16] was introduced. In this paradigm, as illustrated in Figure 4c, the architectures of the client models may vary with different

tasks. The server holds the Transformer body and a model parameter aggregator that first initializes the weights of the Transformer body and task-specific heads and tails for each task. Clients are divided into groups according to their tasks and trained in parallel. After each training epoch, the clients send the head and tail parameters to the server; the server aggregates the model parameters for the same task and returns them to the clients, using techniques such as FedAvg [12]. This paradigm is suitable for training models for multiple similar tasks as it has a shared Transformer body. Quite similar to FeSTA, but unlike FeSTA where the server sends the class token obtained from the last ViT block to the clients, FeSViBS [93] samples the output of one of all ViT blocks and then utilizes a shared projection network to reduce the output dimension before transmitting it to the clients.

Hybrid Split and Federated Learning (HSFL) [94] and Cluster-based Parallel SL (CPSL) [18] are two paradigms designed to address the practical constraints of power, bandwidth, and channel quality typically present in client devices. As the communication costs of FL and SL depend on multiple factors, HSFL finds an optimal allocation strategy based on the current channel conditions and the diversity index [103], allowing some clients to use the SL method and others to use the FL method. Regardless of the chosen training method, client model aggregation is necessary after each training cycle. CPSL, on the other hand, is based on SplitFed and selects appropriate partition layers and training clusters based on the current channel quality and client data holdings to reduce communication costs. Both methods utilize greedy algorithms to optimize their selection problems. Also, the Weighted Sampled Split Learning (WSSL) paradigm [68] was proposed to address the challenge of non-IID data in SL. It employs a dynamic client selection strategy based on client contributions measured by the normalized weight, ensuring consistent training and improving model output. Owing to the client selection strategy, WSSL goes beyond typical SL paradigms by enhancing model robustness, reducing overfitting risk, and promoting fairness.

FedSplitX [63] addresses system heterogeneity with limited computation capabilities by partitioning a large model into client- and server-side models at different cut layers. This accommodates heterogeneous capabilities, creating flexibility with multiple model pairs based on different cut layers. The framework introduces auxiliary networks for these layers, enabling local loss-based learning, reducing communication costs, and improving performance. For local model optimization, FedSplitX aggregates parameters separately for client- and server-side models using FedAvg.

Since the heterogeneity of the computation and communication capacities becomes a bottleneck of the SL training speed, Dynamic Federated split Learning (DFL) [69] incorporates resource-aware split computation of deep neural networks and dynamic clustering of training participants based on the similarity of their submodel layers. Through resource-aware split learning, DFL adjusts the allocation of training tasks to participants with limited resources, aligning with their computational capabilities and the dynamic nature of communication resources. To handle training data heterogeneity without compromising privacy or transmitting raw data over networks, DFL employs the Centered Kernel Alignment (CKA) [104]. This approach effectively assesses the similarity between layers, allowing for a more personalized and efficient training.

The CHEESE paradigm [52] attempts to address the heterogeneity issue from a different perspective. It initially clusters clients through Device-to-Device (D2D) communication. Each client is then allocated a part of the model based on its computational power, clients within the same cluster collaboratively train a complete model, and finally, models among different clusters are aggregated using FedAvg.

SplitLLM [96] combines LLM training with SplitFed and FSL by splitting the pre-trained model across the cloud, edge, and user devices, updating and aggregating only the LoRA adapters, which reduces memory usage and computational workload.

FjORD [85] is an FL paradigm that adjusts the model size based on available client resources. Although it does not strictly adhere to the definition of SL, it is relevant due to its support for heterogeneity and "splitting" through ordered dropout. It employs Ordered Dropout (OD) for importance-based pruning, enabling devices to train submodels and contribute to the global model. In addition, it incorporates a self-distillation method to enhance the performance of smaller submodels. FjORD also allows for dynamic model scaling during inference, improving accuracy across various datasets without the need for retraining. Like FjORD, SAFL [97] is an FL paradigm in which clients train submodels. The process unfolds in two stages: first, clients are grouped based on data similarities, and pruning criteria are applied to identify optimal submodels; second, clients train the pruned models and participate in server-based aggregation. This approach reduces computational overhead, improves inference accuracy, and ensures efficient and tailored models for each client, especially in heterogeneous data environments.

2.3. Intermediate Data Aggregation-Based Approaches

Unlike model weight aggregation, some SL paradigms aggregate intermediate data, such as gradients and smashed data, from different clients to exchange local information while minimizing communication costs.

In SGLR [58], as shown in Figure 5a, the server aggregates the local gradients of a subset of clients rather than the model parameters. This approach reduces the amount of information shared among clients and eliminates the client decoupling problem. Inspired by existing work on learning rate acceleration [105,106], the learning rate in SGLR is divided into two parts to improve the overall training performance and address the server-side large effective batch problem [58].

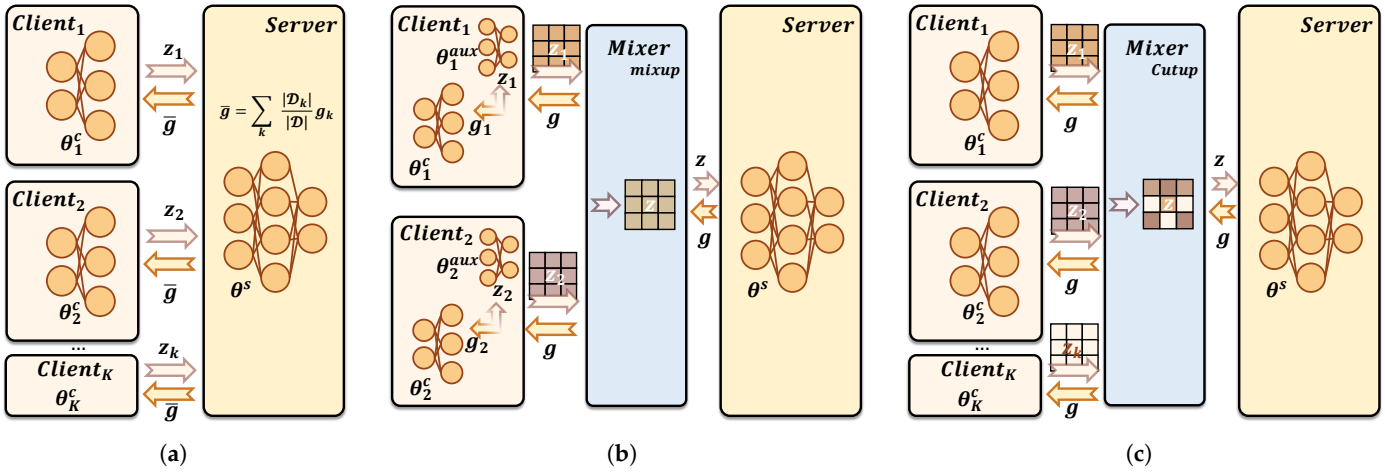


Figure 5. Illustration of intermediate data aggregation-based SL paradigms: (**a**) SLGR; (**b**) LocFedMix-SL; and (**c**) CutMixSL.

In LocFedMix-SL [98], as shown in Figure 5b, the server mixes the smashed data from different clients to create new smashed data for forward propagation. This smashed data mixing increases the effective batch size at the model's top layers to match its higher effective learning rate. Each client has an auxiliary network designed to maximize mutual information between the smashed data and the input data. This makes the relationship between the smashed data and model input tighter, improving model performance and training efficiency.

In CutMixSL [66], SL and ViT are combined and a new type of data called CutSmashed is proposed, as shown in Figure 5c. In this method, each client creates their own CutSmashed data by randomly masking a portion of their original smashed data. The clients then send these partial data to a mixer, which combines them and sends them to the server for training. This method reduces communication costs as clients only need to send a portion of their data, and the server can use multi-casting to send back the gradient. Furthermore, CutSmashed data provide better privacy protection as reconstructing the untransmitted regions is difficult.

To address the scalability issue in PSL paradigms such as SplitFed [46], Mix2SFL [60] introduces SmashMix and GradMix on the basis of SplitFed. The SmashMix involves pairwise mixing of smash data as input for the server-side model, while the GradMix refers to averaging the obtained gradients and returning them to the client. This approach can also accelerate the convergence speed, thereby reducing communication costs.

In the context of energy theft detection [107], the Three-Tier Split Learning (TTSL) [99] was proposed to reduce communication costs and protect privacy. In TTSL, multiple aggregators are employed to average and aggregate the smashed data of the clients before sending them to the server. Through this approach, it is more difficult for honest but curious servers to infer private information from aggregated smashed data, and the communication cost is also reduced.

3. Experimental Evaluation

We implemented several mainstream SL paradigms using the SLPerf framework [108]. In this section, we present comprehensive empirical studies of these paradigms across a wide range of tasks, including image classification, medical data analysis, and tabular prediction. By summarizing and analyzing these results, we provide rich engineering advice to guide ML developers in choosing appropriate SL paradigms according to their specific needs. We also provide research insights to improve current SL paradigms in terms of robustness, efficiency, and privacy protection. We first introduce the experimental setup in Section 3.1. Then, we demonstrate experimental results and provide detailed analyses in Section 3.2.

3.1. Experimental Setup

Among all experiments, we trained each model using SGD with a fixed learning rate of 0.01 on mini-batches of size 64. We evaluated different SL paradigms on five commonly used datasets, namely MNIST, CIFAR-10, CIFAR-100, Adult, and ogbg-molhiv, which cover a wide range of data types, including images, tabular data, and graphs. The detailed information for each dataset is introduced as follows.

- MNIST [45] contains handwritten digits sampled from postal codes (available online: http://yann.lecun.com/exdb/mnist/ (accessed on 10 February 2025)). It comprises a total of 70,000 samples and is divided into 60,000 training samples and 10,000 testing samples. The digits are placed in the center of a 28×28 grid. The goal of the multi-class classification task is to recognize the handwritten digits from 0 to 9. On this dataset, we use LeNet-5 [45] for training and split the network layers in the first block (after the 2D MaxPool layer) so that the client-side model consists of a convolutional layer, a ReLU activation layer, and a 2D MaxPool layer.
- CIFAR [61] has two different variants, CIFAR-10 and CIFAR-100, which contain 60,000 32×32 RGB color images with 10 and 100 class labels, respectively. Images are uniformly distributed over classes, and each image has only one label. The dataset is divided into 50,000 training samples and 10,000 testing samples. The task is to classify objects in the images. Compared to MNIST, CIFAR contains real-world objects, intro-

ducing more noise and fewer similar features between classes, making it more difficult for classification on non-IID data. In particular, different SL paradigms may learn at different rates and be more sensitive to non-IID data. ResNet56 [109] is employed on the CIFAR-10 dataset and ResNet50 is utilized on the CIFAR-100 dataset. The networks are split at the first ReLU activation layer of both ResNet models, which means that the client-side model consists of a convolutional layer, a batch normalization layer, and a ReLU activation layer.

- Ogbg-molhiv is obtained from the Open Graph Benchmark repository (available online: https://ogb.stanford.edu/docs/graphprop/ (accessed on 10 February 2025)). It consists of molecular samples and labels indicating whether the molecules can inhibit HIV replication. The goal of this dataset is to learn a function mapping from the molecular to the label. This task can be modeled as a graph learning task wherein a molecule can be naturally treated as a graph. Specifically, for a given molecule, atoms are represented as nodes, and chemical bonds between them are represented as edges. The node feature is a nine-dimensional vector that contains atomic number, chirality, and other atomic characteristics. The edge feature is three-dimensional. The task in our experiments is a typical case of molecular property prediction, which plays a key role in the AI-drug domain. In real-world scenarios, the drug data are always distributed over different pharmaceutical companies and cannot be shared due to business concerns. Therefore, how to build an ML model without sharing the raw drug data is a hot topic in the research community. To the best of our knowledge, we are the first to apply SL paradigms to this problem and have preliminarily verified its effectiveness. On the dataset, we use a Graph Convolutional Network (GCN) model split at the first embedding layer.
- Adult is a classic dataset with 48,842 instances (available online: https://archive.ics.uci.edu/dataset/20/census+income (accessed on 10 February 2025)) downloaded from the UCI ML repository. The dataset has 14 features such as country, age, workclass, education, etc. The goal for this binary classification task is to predict whether the income exceeds 50 K/year based on the census information. We used a network consisting of a fully connected (FC) layer, a ReLU activation layer, and another FC layer, which is split at the first ReLU activation layer.

3.2. Experimental Results and Analyses

In this subsection, we demonstrate the experimental results and answer the research questions accordingly (as listed in Table 4).

Table 4. Overview of research questions that we answer.

Research Question	Result	General Conclusion
How do different SL paradigms perform when training in the IID setting?	Table 5	All SL paradigms can achieve a similar accuracy and are comparable with the centralized one in the IID setting.
How do different SL paradigms perform when training in the non-IID setting?	Figures 6 and 7 Tables 6 and 7 Figure 8	In the non-IID setting, most SL paradigms can achieve a much lower accuracy to the centralized training, except SplitFed, which is more robust to the non-IID data.
Why does training in the non-IID setting aggravate backward client decoupling?	Figure 9	In the non-IID setting, the local optimum of each client is different, resulting in different behaviors of the model.
What are the merits and disadvantages of SL compared to FedAvg?	Figures 7 and 10	When training on IID data, SL can converge faster than FL. When training on non-IID data, SplitFed can surpass FedAvg.
What are the factors that affect the communication cost of SL?	Tables 8–10	The choice of the cut layer, the SL paradigm, and the size of the dataset will affect the communication cost of training.
Can SL help clients improve their performance?	Figure 11	Except Vanilla in extreme non-IID setting, other SL paradigms can improve the performance of the client in most cases.

In particular, our analyses were carried out along the following four dimensions.

- Comparison between SL paradigms in Table 3. We chose five SL paradigms based on categorization, including Vanilla, U-shaped, PSL, SGLR, and SplitFed. We provide results for the performance of the five SL paradigms when training on IID and non-IID data in Sections 3.2.1 and 3.2.2.
- Comparison with typical FL paradigms such as FedAvg [12]. We provide results and analyses in Section 3.2.3.
- Comparison and analysis of communication costs of the five SL paradigms. The results are presented in Section 3.2.4.
- Comparison with on-device local training. This part aims to figure out whether SL can improve the model performance by leveraging more data from different organizations. Detailed results are presented in Section 3.2.5.

3.2.1. Comparison in the IID Setting

First, we investigated the performance of different SL paradigms in the IID setting. The results are shown in Table 5. We compare the following five paradigms: Vanilla, U-shape, PSL, SGLR, and SplitFed. For most of the datasets we study in this paper, all SL paradigms achieved a similar accuracy and are comparable to the centralized one on IID data. For the ogbg-molhiv dataset, the model used is not suitable for U-shaped and SGLR. Given the significant imbalance in the labels of the dataset, we performed experiments specifically with $\alpha = 0.1$, not with $\alpha = 0$. For example, on the CIFAR-10 dataset with three clients, the vanilla SL paradigm achieved an accuracy of 90.08% while centralized training achieved 91.14%. In contrast, a commonly used U-shaped paradigm reached an accuracy of centralized training. For the graph dataset, the SL paradigms also achieved an AUC of 0.74, compared to an AUC of 0.7472 for centralized training.

Table 5. Results of different SL paradigms on IID and non-IID data. Here, the results for models trained in the centralized setting are provided in bold for comparison, and $\alpha = \infty$ indicates that the dataset is IID among clients [108].

Paradigm	**Vanilla**			**U-Shaped**			**PSL**			**SplitFed**			**SGLR**		
α	**0**	**0.1**	∞	**0**	**0.1**	∞	**0**	**0.1**	∞	**0**	**0.1**	∞	**0**	**0.1**	∞
MNIST (Acc = **98.43%**)	18.26%	72.88%	97.92%	34.86%	67.64%	97.13%	32.97%	70.19%	97.28%	79.32%	97.68%	98.82%	33.12%	86.03%	97.34%
CIFAR-10 (Acc = **91.14%**)	34.24%	35.34%	90.08%	40.23%	64.84%	91.39%	36.91%	35.13%	90.67%	61.55%	77.81%	90.89%	45.88%	67.65%	89.54%
CIFAR-100 (Acc = **69.88%**)	27.49%	43.42%	69.92%	17.69%	39.94%	69.38%	23.84%	33.40%	60.43%	46.20%	50.56%	54.41%	18.61%	35.21%	53.17%
Adult (Acc = **81.73%**)	51.71%	56.12%	79.79%	53.19%	60.42%	81.63%	52.39%	59.57%	80.78%	72.17%	78.32%	80.27%	59.53%	61.01%	79.32%
ogbg-molhiv (AUC = **74.72%**)	-	70.04%	74.28%	-	70.53%	73.41%	-	-	-	-	65.24%	73.74%	-	-	-

3.2.2. Comparison in the Non-IID Setting

In practice, the non-IID setting is more frequently considered for SL. To simulate non-IID settings, we employed label-based splitting for the MNIST and CIFAR-10 datasets. For the Adult dataset, we used attribute-based splitting based on relationships. Following prior work, we first sampled $p_{i,k}$ from Dirichlet distribution Dir(α) and then assigned $p_{i,k}$ proportion of the samples of class/attribute i to client k, where we set α from 0.05 to 0.3 to adjust the level of heterogeneity of the data. The smaller α, the larger the heterogeneity of the data. When $\alpha = 0$, samples from different clients have no overlap in the label/attribute space. For example, the MNIST dataset with 10 labels is divided into five clients by

assigning all samples with two labels to each client. Figure 6 shows the effect of different αs on data partitioning.

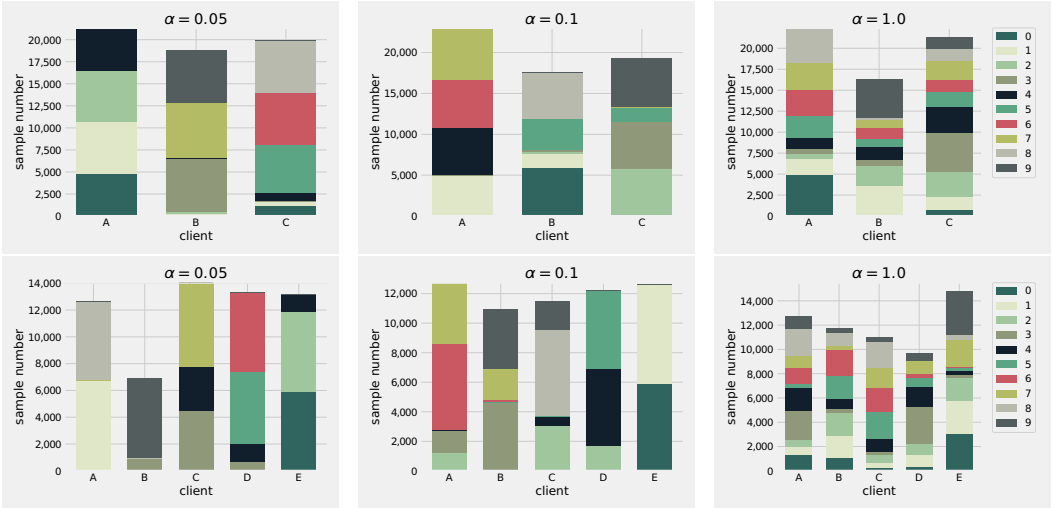


Figure 6. Illustration of data partitioning based on the Dirichlet distribution. In the first row, the number of clients is set to 3; and in the second row, the number of clients is set to 5.

The overall results are also shown in Table 5. We select two representative αs to demonstrate the results of SL paradigms at different data heterogeneity levels. The client number is set to 3 for all results (more results with five clients are shown in Table 7). We summarize the following observations from the results.

- Most SL paradigms, except SplitFed, are observed to struggle in generalizing to the non-IID setting, particularly when data heterogeneity is high (when α is small). The results in Table 5 demonstrate that, when the samples are strictly divided by their labels (i.e., $\alpha = 0$), most paradigms do not perform well on the CIFAR-10 dataset. For example, Vanilla only achieved an accuracy of 34.24%, which is significantly lower than that of centralized learning (91.14%). On the Adult dataset, most SL paradigms perform similarly to random guesses (i.e., ≈50%). When the samples are not strictly divided by their labels/attributes (e.g., $\alpha = 0.1$), the accuracy of all SL paradigms becomes slightly closer to that of centralized learning. However, except for SplitFed, most of the SL paradigms still perform much poorer than centralized learning.
- Compared to other methods, SplitFed demonstrated higher robustness to non-IID data. As shown in Figure 7, the performance of SplitFed remains stable when the data are divided by the Dirichlet distribution with a small α (e.g., $\alpha = 0.1$), achieving an accuracy of 97.68%, which is close to the accuracy achieved on centralized training. However, the accuracy ratios of Vanilla and PSL are low when the value of α is small. We also observe similar trends when training on the CIFAR-10 dataset. This may be because SplitFed allows the clients to send their model parameters to the server, which facilitates the exchange of information about local data. Unlike SplitFed, the clients in SGLR share the information by only averaging the local gradients of the cut layer. As a result, for small values of α, it outperforms other paradigms except for SplitFed. For example, SGLR achieved an accuracy of 67.65% when $\alpha = 0.1$, but Vanilla has an accuracy of only 34.24% in the same setting.

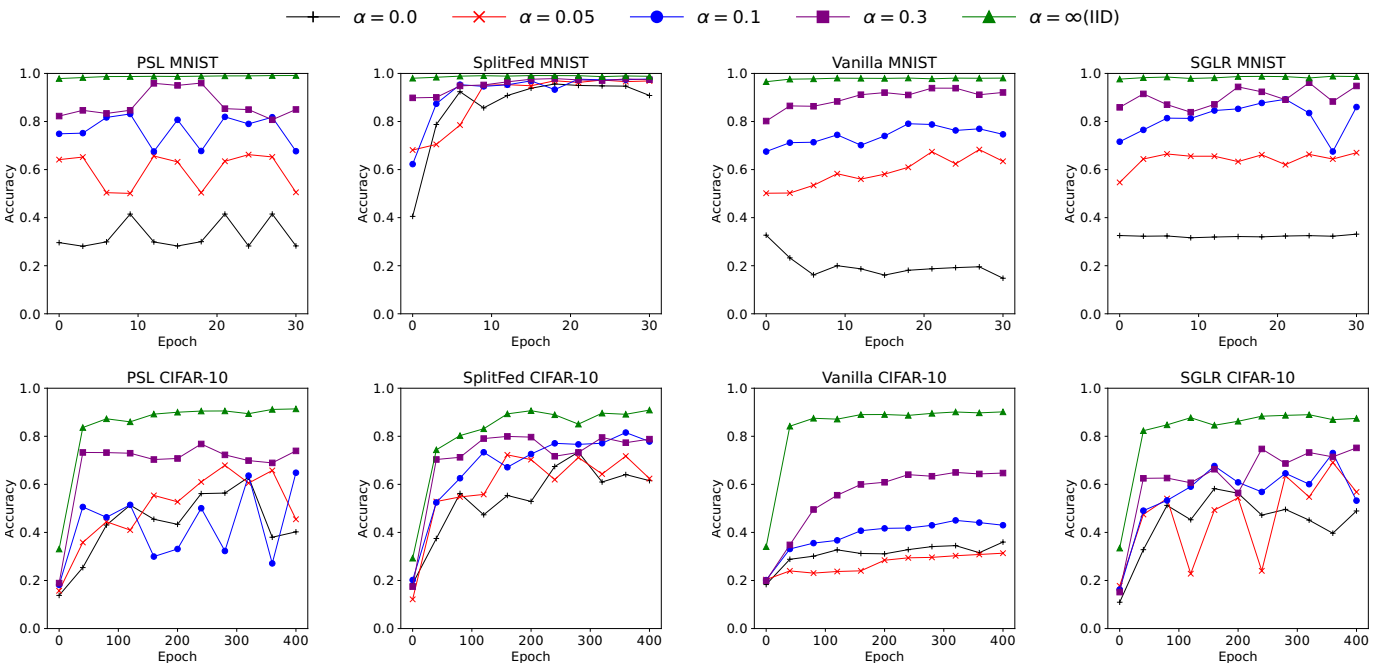


Figure 7. Performance of four SL paradigms on different αs when the number of clients is 3 on the MNIST and CIFAR-10 datasets. Here, the x-axis represents the number of epochs.

- Vanilla performs worse than other paradigms and even worse than local training in some cases. As shown in Table 5, Vanilla only achieves a 18.26% accuracy when $\alpha = 0$. The training mode of Vanilla prevents the model from converging in many cases.
- In addition, we observe that the client number can also affect the model performance. Specifically, in Tables 6 and 7, when the number of clients is 5 and α is small, most SL paradigms achieve a much lower accuracy than when the number of clients is 3. This may be because the data become more heterogeneous as the number of clients increases. As an example, a client may have samples from 3 or 4 classes when the number of clients is 3, but it can only have samples from 2 classes when the number of clients is 5.

Table 6. Accuracy of SL paradigms on the MNIST dataset after 30 epochs with 3 clients [108].

Paradigm	**IID** ($\alpha = \infty$)	**Dirichlet Distribution**			
		$\alpha = 0$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.3$
Vanilla	97.92%	18.26%	69.37%	72.88%	90.24%
U-shaped	97.13%	32.97%	60.24%	70.19%	93.78%
SplitFed	98.82%	79.32%	96.87%	97.68%	97.39%
PSL	97.28%	34.86%	50.54%	67.64%	85.00%
SGLR	97.34%	33.12%	60.71%	86.03%	94.79%

Table 7. Accuracy of SL paradigms on the MNIST dataset after 30 epochs with 5 clients [108].

Paradigm	**IID** ($\alpha = \infty$)	**Dirichlet Distribution**			
		$\alpha = 0$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.3$
Vanilla	98.17%	10.46%	10.31%	12.53%	93.87%
U-shaped	97.46%	19.12%	10.41%	10.84%	92.36%
SplitFed	98.76%	90.13%	95.95%	96.67%	98.39%
PSL	98.37%	19.86%	70.63%	78.63%	95.22%
SGLR	98.26%	10.20%	56.91%	86.03%	90.87%

- Also, non-IID data may exacerbate the backward client decoupling problem [58], which is that the models trained on different clients' data may have significant discrepancies. For example, in PSL, the accuracy of different clients becomes increasingly different as the heterogeneity of the data increases, as shown in Figure 8. This is because in PSL, the clients only use their local data's gradients for training, and the clients become decoupled during the training. This leads to excessive differences between clients, and the smashed data from different clients show a larger difference than that from different labels. To support this observation, we performed t-SNE visualization on the smashed data sent from the client during the testing phase and colored them with different labels and different clients, as shown in Figure 9. Clearly, in the non-IID setting, data points from the same client are clustered together, whereas points from the same label are not. This causes the server-side model to learn the features of different clients first and the label second, which implies that different clients cannot learn from each other. Therefore, the accuracy of different clients is largely dependent on the quality of their local data, making the difference between their accuracy significant. In the IID setting, the backward client decoupling problem is minor.

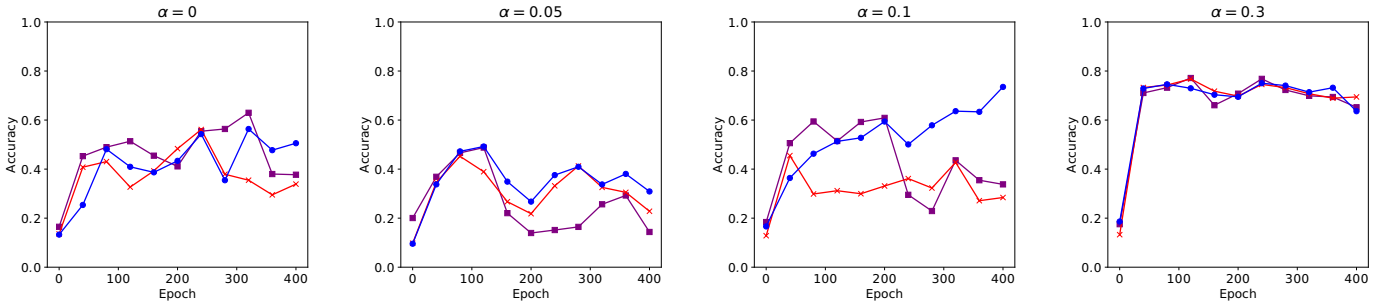
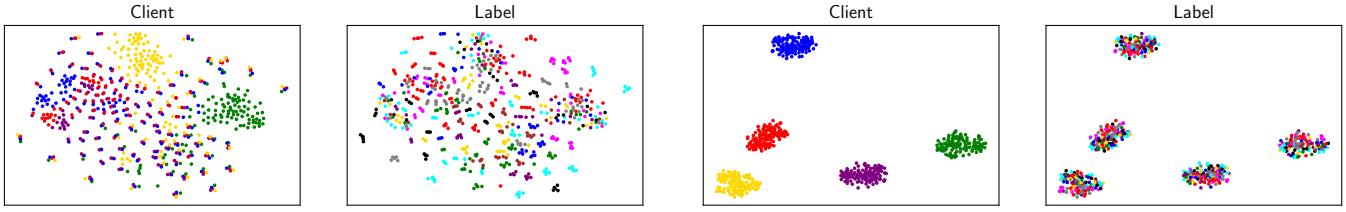


Figure 8. Performance of each client in PSL for different αs when the number of clients is 3. We use a colored line to indicate the performance of each client.



(**a**) (**b**)

Figure 9. Visualization results for "smashed data" of PSL using t-SNE, where points are colored with client and label, respectively. (**a**) IID; (**b**) Non-IID ($\alpha = 0$).

3.2.3. Comparison with Typical FL Paradigms

In this section, we compare the performance of FedAvg with that of SL paradigms on the MNIST and CIFAR-10 datasets. The experimental results of the SL paradigms are shown in Figure 7, and those of FedAvg are shown in Figure 10.

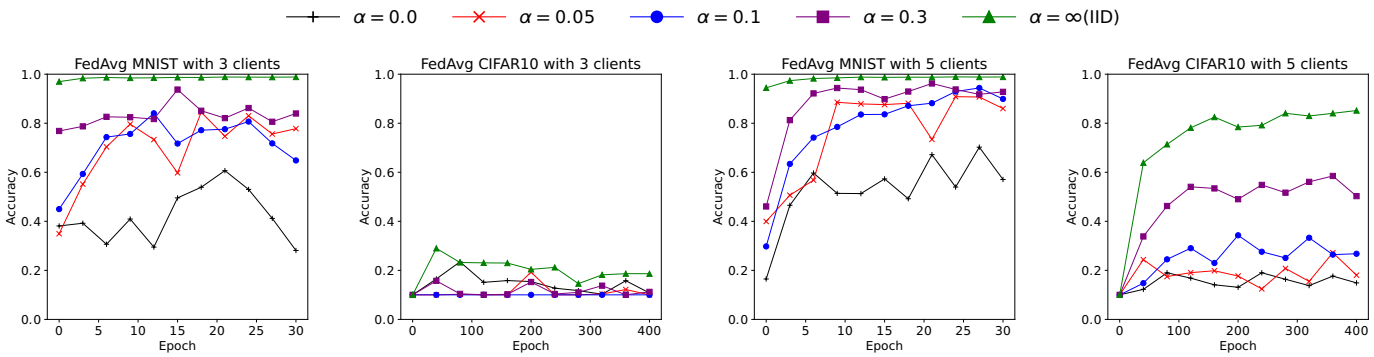


Figure 10. Performance of FedAvg with different αs on the MNIST and CIFAR-10 datasets. In the first two, we set the number of clients to 3; in the latter two, we set the number of clients to 5. The x-axis represents the number of epochs [108].

First, we observe that on both datasets, SplitFed can always converge relatively faster than FedAvg with the same data partition method. For example, on the MNIST dataset, SplitFed is always able to achieve a similar accuracy to that of centralized learning at around 10 epochs, whereas FedAvg does not achieve it except in the IID case. Similar conclusions can be drawn for the CIFAR-10 dataset. The results show that when the value of α is the same, SplitFed consistently achieves a higher accuracy than FedAvg. This is probably because the server in SplitFed holds most of the network layers, which can be trained on the entire dataset.

Second, FedAvg outperforms other SL paradigms in the non-IID setting, particularly when both the dataset and model are simple, such as training LeNet-5 with the MNIST dataset. When α is small, FedAvg shows superior accuracy comparable to most SL paradigms, with SplitFed as the only exception.

However, when the data and models become more complex, FedAvg learns much more slowly than the SL paradigms. For example, on the CIFAR-10 dataset, most SL paradigms can achieve better accuracy than FedAvg, indicating the difficulties of FedAvg in training more complex models.

3.2.4. Communication Cost of SL Paradigms

We first analyze the communication complexities of various paradigms theoretically, as shown in Table 8. Then, we compare three paradigms for their communication costs: PSL, SplitFed, and AsyncSL, which are representative paradigms with intermediate data aggregation, model aggregation, and reduced communication, respectively. The communication cost is computed by the size of the data (in MB) that are transferred between the clients and the server. The $loss_{threshold}$ [86] we set in AsyncSL is 0.1.

Table 8. Communication complexities of different paradigms, where S_{cut} is the size of the cut layer, and S_θ is the size of the model θ.

Paradigm	Communication Cost per Epoch
Vanilla	$2 \cdot S_{cut} \cdot \lvert\mathcal{D}_k\rvert + S_{\theta^c}$
PSL	$2 \cdot S_{cut} \cdot \lvert\mathcal{D}_k\rvert$
SGLR	$2 \cdot S_{cut} \cdot \lvert\mathcal{D}_k\rvert$
SplitFed	$2 \cdot S_{cut} \cdot \lvert\mathcal{D}_k\rvert + 2 \cdot S_{\theta^c}$
U-shaped	$2 \cdot (S_{cut} + S'_{cut}) \cdot \lvert\mathcal{D}_k\rvert$
FedAvg	$2 \cdot S_\theta$

We split the networks at different layers to compare the impact of the choice of the cut layer on communication cost. In Table 9, "Small" indicates that the network in clients consists of a convolution layer; and "Large" indicates that the network consists of two convolution layers. In Table 10, "Small" indicates that the model in clients contains a BN

layer, a ReLU layer, and a pooling layer; "Large" indicates that the model in clients contains all layers before the last pool layer.

Table 9. Communication cost per epoch on the MINST dataset when the number of clients is 3 [108].

Paradigm	Small (MB)		Large (MB)	
	Send	Receive	Send	Receive
SplitFed	18.33	197.59	4.04	58.70
(half data)	9.36	98.86	3.97	29.40
PSL	17.86	197.59	3.93	58.65
(half data)	8.79	98.89	3.88	29.42
AsyncSL	4.58	28.24	1.84	8.39
(half data)	2.43	15.43	1.03	4.32

Table 10. Communication cost per epoch on the CIFAR-10 dataset when the number of clients is 3.

Paradigm	Small (MB)		Large (MB)	
	Send	Receive	Send	Receive
SplitFed	15.17	3120.24	17.09	3120.24
PSL	15.17	3120.24	18.14	3120.24
AsyncSL	8.38	532.12	9.23	563.12

The observations from these results are as follows.

- The dataset size has an impact on the communication cost. For example, as shown in Table 9, the communication cost of all three paradigms in one epoch decreases significantly when only half of the dataset is used and is approximately equal to half.
- Training in different SL paradigms results in different communication costs. For example, since the clients in SplitFed need to transmit the local model parameters to the server, SplitFed sends and receives more data than PSL as shown in Table 9. Because the size of the models we use is much smaller than the size of the smashed data and gradients, its effect can be neglected. However, when the model is more complex or the number of clients is much larger, the effect of the model aggregation on the communication cost cannot be ignored [11].
- The choice of the cut layer may have an impact on the communication cost. As shown in Table 9, when training on the MNIST dataset, the communication cost decreases significantly when clients hold two convolution layers instead of one. This is because the choice of the cut layer affects the size of the smashed data and gradients. However, the cost does not decrease but even increases when training on the CIFAR-10 dataset, as shown in Table 10. Thus, choosing the appropriate layer to split the network is effective in reducing communication costs.

3.2.5. Comparison with On-Device Local Training

One of the primary objectives of SL is to enhance the model's performance for each client without requiring the exchange of raw data. Therefore, we compare the performance of SL with on-device local training. In these experiments, each client follows the same data partitioning process as in the other SL experiments and uses an incomplete dataset to train their local model without any communication.

Our main findings are summarized as follows:

- The results of local training and two SL paradigms on IID data with different numbers of clients are presented in the first row of Figure 11. On the MNIST dataset, there is a slight difference among the three algorithms, with only a small drop in local training accuracy at 10 clients. However, on the CIFAR-10 dataset, local training shows a significant drop, with an accuracy of only 71.47% at 10 clients, which is probably due to the complexity of the learning task. Unlike handwritten characters, CIFAR-10

contains real-world objects, which require a sufficient amount of training data. Both SL paradigms show promising results, indicating that they can improve each client's model performance without exchanging the original data.

- The performance of local training and the two SL paradigms on non-IID data with varying α is shown in Figure 11. On the MNIST dataset, the performance of Vanilla and local training decreases as α decreases, but the overall performance of Vanilla is still higher than that of local training. When $\alpha = 0$, the accuracy of Vanilla also drops to a level similar to local training. On the CIFAR-10 dataset, Vanilla's performance is lower than expected; when α is less than 0.5, it has a lower accuracy than local training. SplitFed performs well on both datasets, but on the CIFAR-10 dataset, there is a large drop in accuracy when α is below 0.1. In the most extreme non-IID setting ($\alpha = 0$), SplitFed achieves an accuracy of 64.73%. SplitFed also has difficulty in this case, but its performance is still better than the other two paradigms. SplitFed can mitigate this problem by increasing the frequency of aggregation.

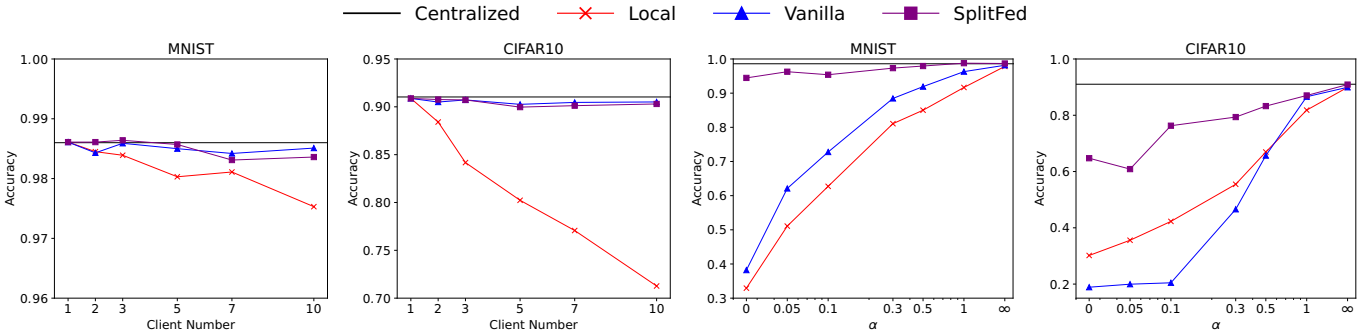


Figure 11. Effect of client number and α on model accuracy [108].

Generally, Vanilla SL helps clients improve performance when data are evenly distributed. However, for complex tasks and highly skewed non-IID distributions, SplitFed should be used to enhance the performance of local devices.

4. Conclusions

In this paper, we first present a systematic review of existing SL paradigms, classifying them according to their core characteristics. We group these paradigms into three main types: model split-only, weight aggregation-based, and intermediate data aggregation-based approaches. For each type, we evaluate its performance across four key dimensions and provide insights into the scenarios where each approach is most suitable. Through comprehensive experiments on five widely used datasets in both IID and non-IID settings, we compare and analyze the performance of various SL paradigms using the SLPerf framework. Our findings highlight several important conclusions: for improved performance in non-IID scenarios, combining SL with FL is beneficial; to reduce communication costs, selecting the appropriate cut layer and SL paradigm is crucial, and so on. These results provide valuable engineering guidance and provide deeper insight for future SL research. Most importantly, our work contributes to the field by establishing a fair and accessible benchmark to evaluate SL performance.

Author Contributions: Software, Z.H. and T.Z.; conceptualization, C.C.; survey, Z.H. and T.Z.; writing—original draft preparation, Z.H. and T.Z.; writing—visualization, Z.H. and T.Z.; writing—review and editing, C.C., Y.W. and B.W.; funding acquisition, C.C.; validation, Z.H. and T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant numbers 62202170 and 62202169.

Data Availability Statement: Real-world datasets that we used in our experiments are publicly available. The code for our experimental evaluation is available at https://github.com/Rainysponge/SLPerf (accessed on 10 February 2025).

Conflicts of Interest: Dr. Bingzhe Wu was employed by Tencent AI Lab. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2322–2358. [CrossRef]
2. Malik, P.K.; Sharma, R.; Singh, R.; Gehlot, A.; Satapathy, S.C.; Alnumay, W.S.; Pelusi, D.; Ghosh, U.; Nayak, J. Industrial Internet of Things and its Applications in Industry 4.0: State of The Art. *Comput. Commun.* **2021**, *166*, 125–139. [CrossRef]
3. Yin, Y.; Zeng, Y.; Chen, X.; Fan, Y. The internet of things in healthcare: An overview. *J. Ind. Inf. Integr.* **2016**, *1*, 3–13. [CrossRef]
4. Maiti, M.; Ghosh, U. Next-Generation Internet of Things in Fintech Ecosystem. *IEEE Internet Things J.* **2023**, *10*, 2104–2111. [CrossRef]
5. Lin, J.; Yu, W.; Yang, X.; Zhao, P.; Zhang, H.; Zhao, W. An Edge Computing Based Public Vehicle System for Smart Transportation. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12635–12651. [CrossRef]
6. Wu, Y.; Dai, H.; Wang, H.; Xiong, Z.; Guo, S. A Survey of Intelligent Network Slicing Management for Industrial IoT: Integrated Approaches for Smart Transportation, Smart Energy, and Smart Factory. *IEEE Commun. Surv. Tutorials* **2022**, *24*, 1175–1211. [CrossRef]
7. Huang, L.; Shea, A.L.; Qian, H.; Masurkar, A.; Deng, H.; Liu, D. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *J. Biomed. Inform.* **2019**, *99*, 103291. [CrossRef]
8. Liu, Y.; Gadepalli, K.; Norouzi, M.; Dahl, G.E.; Kohlberger, T.; Boyko, A.; Venugopalan, S.; Timofeev, A.; Nelson, P.Q.; Corrado, G.S.; et al. Detecting Cancer Metastases on Gigapixel Pathology Images. *arXiv* **2017**, arXiv:1703.02442.
9. Mantelero, A. The EU Proposal for a General Data Protection Regulation and the roots of the 'right to be forgotten'. *Comput. Law Secur. Rev.* **2013**, *29*, 229–235. [CrossRef]
10. Konečný, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2016**, arXiv:1610.05492.
11. Vepakomma, P.; Swedish, T.; Raskar, R.; Gupta, O.; Dubey, A. No Peek: A Survey of private distributed deep learning. *arXiv* **2018**, arXiv:1812.03288.
12. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
13. Gupta, O.; Raskar, R. Distributed learning of deep neural network over multiple agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [CrossRef]
14. Poirot, M.G.; Vepakomma, P.; Chang, K.; Kalpathy-Cramer, J.; Gupta, R.; Raskar, R. Split Learning for collaborative deep learning in healthcare. *arXiv* **2019**, arXiv:1912.12115.
15. Rajbhandari, S.; Rasley, J.; Ruwase, O.; He, Y. ZeRO: Memory optimizations toward training trillion parameter models. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, GA, USA, 9–19 November 2020; pp. 20:1–20:16.
16. Park, S.; Kim, G.; Kim, J.; Kim, B.; Ye, J.C. Federated Split Task-Agnostic Vision Transformer for COVID-19 CXR Diagnosis. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 24617–24630.
17. Hsieh, C.Y.; Chuang, Y.C.; Wu, A.Y. C3-SL: Circular Convolution-Based Batch-Wise Compression for Communication-Efficient Split Learning. In Proceedings of the IEEE 32nd International Workshop on Machine Learning for Signal Processing, Xi'an, China, 22–25 August 2022; pp. 1–6.
18. Wu, W.; Li, M.; Qu, K.; Zhou, C.; Shen, X.; Zhuang, W.; Li, X.; Shi, W. Split learning over wireless networks: Parallel design and resource management. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 1051–1066. [CrossRef]
19. Gao, Y.; Kim, M.; Abuadbba, S.; Kim, Y.; Thapa, C.; Kim, K.; Camtep, S.A.; Kim, H.; Nepal, S. End-to-End Evaluation of Federated Learning and Split Learning for Internet of Things. In Proceedings of the International Symposium on Reliable Distributed Systems, Shanghai, China, 21–24 September 2020; pp. 91–100.
20. Pham, N.D.; Chilamkurti, N. Data Leakage Threats and Protection in Split Learning: A Survey. *Proc. Int. Conf. Intell. Comput. Emerg. Appl.* **2023**, *3*, 141–147.

21. Na, H.; Oh, Y.; Lee, W.; Choi, D. Systematic Evaluation of Robustness Against Model Inversion Attacks on Split Learning. In Proceedings of the 24th International Conference on Information Security Applications, Jeju Island, Republic of Korea, 23–25 August 2023; pp. 107–118.
22. Nilsson, A.; Smith, S.; Ulm, G.; Gustavsson, E.; Jirstrand, M. A Performance Evaluation of Federated Learning Algorithms. In Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning, Rennes, France, 10 December 2018; pp. 1–8.
23. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
24. Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; Gao, Y. A survey on federated learning. *Knowl. Based Syst.* **2021**, *216*, 106775. [CrossRef]
25. Wen, J.; Zhang, Z.; Lan, Y.; Cui, Z.; Cai, J.; Zhang, W. A survey on federated learning: Challenges and applications. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 513–535. [CrossRef] [PubMed]
26. Li, L.; Fan, Y.; Tse, M.; Lin, K.Y. A review of applications in federated learning. *Comput. Ind. Eng.* **2020**, *149*, 106854. [CrossRef]
27. Lyu, L.; Yu, H.; Yang, Q. Threats to Federated Learning: A Survey. *arXiv* **2020**, arXiv:2003.02133.
28. Yin, X.; Zhu, Y.; Hu, J. A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions. *ACM Comput. Surv.* **2021**, *54*, 131:1–131:36. [CrossRef]
29. He, C.; Li, S.; So, J.; Zhang, M.; Wang, H.; Wang, X.; Vepakomma, P.; Singh, A.; Qiu, H.; Shen, L.; et al. FedML: A Research Library and Benchmark for Federated Machine Learning. *arXiv* **2020**, arXiv:2007.13518.
30. Ryffel, T.; Trask, A.; Dahl, M.; Wagner, B.; Mancuso, J.; Rueckert, D.; Passerat-Palmbach, J. A generic framework for privacy preserving deep learning. *arXiv* **2018**, arXiv:1811.04017.
31. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Parcollet, T.; Lane, N.D. Flower: A Friendly Federated Learning Research Framework. *arXiv* **2020**, arXiv:2007.14390.
32. Caldas, S.; Wu, P.; Li, T.; Konečný, J.; McMahan, H.B.; Smith, V.; Talwalkar, A. LEAF: A Benchmark for Federated Settings. *arXiv* **2018**, arXiv:1812.01097.
33. Zou, T.; GU, Z.; He, Y.; Takahashi, H.; Liu, Y.; Zhang, Y.Q. VFLAIR: A Research Library and Benchmark for Vertical Federated Learning. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
34. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Vincent Poor, H. Federated Learning for Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 1622–1658. [CrossRef]
35. Banabilah, S.; Aloqaily, M.; Alsayed, E.; Malik, N.; Jararweh, Y. Federated learning review: Fundamentals, enabling technologies, and future applications. *Inf. Process. Manag.* **2022**, *59*, 103061. [CrossRef]
36. Antunes, R.S.; André da Costa, C.; Küderle, A.; Yari, I.A.; Eskofier, B. Federated Learning for Healthcare: Systematic Review and Architecture Proposal. *ACM Trans. Intell. Syst. Technol.* **2022**, *13*, 54:1–54:23. [CrossRef]
37. Zhan, Y.; Zhang, J.; Hong, Z.; Wu, L.; Li, P.; Guo, S. A Survey of Incentive Mechanism Design for Federated Learning. *IEEE Trans. Emerg. Top. Comput.* **2022**, *10*, 1035–1044. [CrossRef]
38. Qi, P.; Chiaro, D.; Guzzo, A.; Ianni, M.; Fortino, G.; Piccialli, F. Model aggregation techniques in federated learning: A comprehensive survey. *Future Gener. Comput. Syst.* **2024**, *150*, 272–293. [CrossRef]
39. Khraisat, A.; Alazab, A.; Singh, S.; Jan, T., Jr.; Gomez, A. Survey on Federated Learning for Intrusion Detection System: Concept, Architectures, Aggregation Strategies, Challenges, and Future Directions. *ACM Comput. Surv.* **2024**, *57*, 7:1–7:38. [CrossRef]
40. Zhang, H.; Ye, J.; Huang, W.; Liu, X.; Gu, J. Survey of federated learning in intrusion detection. *J. Parallel Distrib. Comput.* **2025**, *195*, 104976. [CrossRef]
41. Qi, P.; Chiaro, D.; Piccialli, F. Small models, big impact: A review on the power of lightweight Federated Learning. *Future Gener. Comput. Syst.* **2025**, *162*, 107484. [CrossRef]
42. Wan, Y.; Qu, Y.; Ni, W.; Xiang, Y.; Gao, L.; Hossain, E. Data and Model Poisoning Backdoor Attacks on Wireless Federated Learning, and the Defense Mechanisms: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* **2024**, *26*, 1861–1897. [CrossRef]
43. Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 1759–1799. [CrossRef]
44. Yang, S.; Chen, H.; Huang, J.; Yan, Y.; Chen, J.; Xiong, A. Split Learning Based on Self-supervised Learning. In Proceedings of the 12th International Conference on Computer Engineering and Networks, Haikou, China, 4–7 November 2022; pp. 95–104.
45. LeCun, Y.; Boser, B.E.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.E.; Jackel, L.D. Handwritten digit recognition with a back-propagation network. *Adv. Neural Inf. Process. Syst.* **1989**, *2*, 396–404.
46. Thapa, C.; Chamikara, M.A.P.; Camtepe, S. SplitFed: When Federated Learning Meets Split Learning. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 8485–8493. [CrossRef]
47. Kim, J.; Shin, S.; Yu, Y.; Lee, J.; Lee, K. Multiple Classification with Split Learning. In Proceedings of the 9th International Conference on Smart Media and Applications, Jeju, Republic of Korea, 17–19 September 2020; pp. 358–363.
48. Pham, N.D.; Abuadbba, A.; Gao, Y.; Phan, K.T.; Chilamkurti, N. Binarizing Split Learning for Data Privacy Enhancement and Computation Reduction. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3088–3100. [CrossRef]

49. Jeon, J.; Kim, J. Privacy-Sensitive Parallel Split Learning. In Proceedings of the International Conference on Information Networking, Barcelona, Spain, 7–10 January 2020; pp. 7–9.
50. Chopra, A.; Sahu, S.K.; Singh, A.; Java, A.; Vepakomma, P.; Sharma, V.; Raskar, R. AdaSplit: Adaptive Trade-offs for Resource-constrained Distributed Deep Learning. *arXiv* **2021**, arXiv:2112.01637.
51. Han, D.J.; Bhatti, H.I.; Lee, J.; Moon, J. Accelerating Federated Learning with Split Learning on Locally Generated Losses. In Proceedings of the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021, Virtual Event, 24 July 2021.
52. Cheng, Z.; Xia, X.; Liwang, M.; Fan, X.; Sun, Y.; Wang, X.; Huang, L. CHEESE: Distributed clustering-based hybrid federated split learning over edge networks. *IEEE Trans. Parallel Distrib. Syst.* **2023**, *34*, 3174–3191. [CrossRef]
53. Kim, S.; Oh, Y.; Jeon, Y. SplitMAC: Wireless Split Learning over Multiple Access Channels. *arXiv* **2023**, arXiv:2311.02405. [CrossRef]
54. Park, J.; Oh, S.; Kim, S.L. SplitAMC: Split Learning for Robust Automatic Modulation Classification. In Proceedings of the IEEE 97th Vehicular Technology Conference, Florence, Italy, 20–23 June 2023; pp. 1–6.
55. Zheng, J.; Chen, Y.; Lai, Q. PPSFL: Privacy-Preserving Split Federated Learning for heterogeneous data in edge-based Internet of Things. *Future Gener. Comput. Syst.* **2024**, *156*, 231–241. [CrossRef]
56. Zhang, Z.; Pinto, A.; Turina, V.; Esposito, F.; Matta, I. Privacy and Efficiency of Communications in Federated Split Learning. *IEEE Trans. Big Data* **2023**, *9*, 1380–1391. [CrossRef]
57. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
58. Pal, S.; Uniyal, M.; Park, J.; Vepakomma, P.; Raskar, R.; Bennis, M.; Jeon, M.; Choi, J. Server-Side Local Gradient Averaging and Learning Rate Acceleration for Scalable Split Learning. *arXiv* **2021**, arXiv:2112.05929.
59. Oh, S.; Baek, S.; Park, J.; Nam, H.; Vepakomma, P.; Raskar, R.; Bennis, M.; Kim, S.L. Privacy-Preserving Split Learning with Vision Transformers using Patch-Wise Random and Noisy CutMix. *arXiv* **2024**, arXiv:2408.01040.
60. Oh, S.; Nam, H.; Park, J.; Vepakomma, P.; Raskar, R.; Bennis, M.; Kim, S.L. Mix2SFL: Two-Way Mixup for Scalable, Accurate, and Communication-Efficient Split Federated Learning. *IEEE Trans. Big Data* **2024**, *10*, 238–248.
61. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009.
62. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv* **2018**, arXiv:1812.00564.
63. Shin, J.; Ahn, J.; Kang, H.; Kang, J. FedSplitX: Federated Split Learning for Computationally-Constrained Heterogeneous Clients. *arXiv* **2023**, arXiv:2310.14579.
64. Ayad, A.; Renner, M.; Schmeink, A. Improving the Communication and Computation Efficiency of Split Learning for IoT Applications. In Proceedings of the IEEE Global Communications Conference, Madrid, Spain, 7–11 December, 2021; pp. 1–6.
65. Joshi, P.; Thapa, C.; Camtepe, S.; Hasanuzzaman, M.; Scully, T.; Afli, H. Splitfed learning without client-side synchronization: Analyzing client-side split network portion size to overall performance. *arXiv* **2021**, arXiv:2109.09246.
66. Baek, S.; Park, J.; Vepakomma, P.; Raskar, R.; Bennis, M.; Kim, S. Visual Transformer Meets CutMix for Improved Accuracy, Communication Efficiency, and Data Privacy in Split Learning. *arXiv* **2022**, arXiv:2207.00234.
67. Mohammadabadi, S.M.S.; Zawad, S.; Yan, F.; Yang, L. Speed Up Federated Learning in Heterogeneous Environment: A Dynamic Tiering Approach. *arXiv* **2023**, arXiv:2312.05642. [CrossRef]
68. Osti, M.; Thakuri, A.; Qolomany, B.; Mulahuwaish, A. Weighted Sampled Split Learning (WSSL): Balancing Privacy, Robustness, and Fairness in Distributed Learning Environments. In Proceedings of the IEEE 29th International Conference on Parallel and Distributed Systems, Danzhou, China, 17–21 December 2023; pp. 443–450.
69. Samikwa, E.; Di Maio, A.; Braun, T. DFL: Dynamic Federated Split Learning in Heterogeneous IoT. *IEEE Trans. Mach. Learn. Commun. Netw.* **2024**, *2*, 733–752. [CrossRef]
70. Shen, J.; Wang, X.; Cheng, N.; Ma, L.; Zhou, C.; Zhang, Y. Effectively Heterogeneous Federated Learning: A Pairing and Split Learning Based Approach. In Proceedings of the IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 4–8 December 2023; pp. 5847–5852.
71. Tschandl, P.; Rosendahl, C.; Kittler, H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Sci. Data* **2018**, *5*, 180161. [CrossRef] [PubMed]
72. Darlow, L.N.; Crowley, E.J.; Antoniou, A.; Storkey, A.J. CINIC-10 is not ImageNet or CIFAR-10. *arXiv* **2018**, arXiv:1810.03505.
73. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
74. Irvin, J.; Rajpurkar, P.; Ko, M.; Yu, Y.; Ciurea-Ilcus, S.; Chute, C.; Marklund, H.; Haghgoo, B.; Ball, R.L.; Shpanskaya, K.S.; et al. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 590–597. [CrossRef]

75. Poirot, M.G. Split Learning in Health Care: Multi-Center Deep Learning Without Sharing Patient Data. Master's Thesis, University of Twente, Enschede, The Netherlands, 2020.
76. Moody, G.B.; Mark, R.G. The impact of the MIT-BIH arrhythmia database. *IEEE Eng. Med. Biol. Mag.* **2001**, *20*, 45–50. [CrossRef] [PubMed]
77. Abuadbba, S.; Kim, K.; Kim, M.; Thapa, C.; Camtepe, S.A.; Gao, Y.; Kim, H.; Nepal, S. Can We Use Split Learning on 1D CNN Models for Privacy Preserving Training? *Proc. Asia Conf. Comput. Commun. Secur.* **2020**, *15*, 305–318.
78. Chang, K.; Balachandar, N.; Lam, C.; Yi, D.; Brown, J.; Beers, A.; Rosen, B.; Rubin, D.L.; Kalpathy-Cramer, J. Distributed deep learning networks among institutions for medical imaging. *J. Am. Med. Inform. Assoc.* **2018**, *25*, 945–954. [CrossRef] [PubMed]
79. Verma, A. Evaluation of classification algorithms with solutions to class imbalance problem on bank marketing dataset using WEKA. *Int. Res. J. Eng. Techn.* **2019**, *5*, 54–60.
80. Ceballos, I.; Sharma, V.; Mugica, E.; Singh, A.; Roman, A.; Vepakomma, P.; Raskar, R. SplitNN-driven Vertical Partitioning. *arXiv* **2020**, arXiv:2008.04137.
81. Becker, B.; Kohavi, R. Adult. UCI Machine Learning Repository. 1996. Available online: https://archive.ics.uci.edu/dataset/2/adult (accessed on 10 February 2025).
82. Cabanes, C.; Grouazel, A.; von Schuckmann, K.; Hamon, M.; Turpin, V.; Coatanoan, C.; Guinehut, S.; Boone, C.; Ferry, N.; Reverdin, G.; et al. The CORA dataset: Validation and diagnostics of ocean temperature and salinity in situ measurements. *Ocean Sci.* **2013**, *9*, 1–18. [CrossRef]
83. Shan, C.; Jiao, H.; Fu, J. Towards Representation Identical Privacy-Preserving Graph Neural Network via Split Learning. *arXiv* **2021**, arXiv:2107.05917.
84. Dernoncourt, F.; Lee, J.Y. PubMed 200k RCT: A Dataset for Sequential Sentence Classification in Medical Abstracts. *Proc. Int. Jt. Conf. Nat. Lang. Process.* **2017**, *2*, 308–313.
85. Horvath, S.; Laskaridis, S.; Almeida, M.; Leontiadis, I.; Venieris, S.; Lane, N. FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12876–12889.
86. Chen, X.; Li, J.; Chakrabarti, C. Communication and Computation Reduction for Split Learning using Asynchronous Training. In Proceedings of the IEEE Workshop on Signal Processing Systems, Coimbra, Portugal, 19–21 October 2021; pp. 76–81.
87. Bedda, K.; Fouda, M.M.; Fadlullah, Z.M. PC-SSL: Peer-Coordinated Sequential Split Learning for Intelligent Traffic Analysis in mmWave 5G Networks. In Proceedings of the IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications, Toronto, ON, Canada, 5–8 September 2023; pp. 1–6.
88. Ninkovic, V.; Vukobratovic, D.; Miskovic, D.; Zennaro, M. COMSPLIT: A Communication-Aware Split Learning Design for Heterogeneous IoT Platforms. *arXiv* **2024**, arXiv:2410.19375. [CrossRef]
89. Mudvari, A.; Vainio, A.; Ofeidis, I.; Tarkoma, S.; Tassiulas, L. Adaptive compression-aware split learning and inference for enhanced network efficiency. *ACM Trans. Internet Techn.* **2024**, *24*, 27:1–27:26. [CrossRef]
90. Li, Z.; Wu, S.; Li, L.; Zhang, S. Energy-Efficient Split Learning for Fine-Tuning Large Language Models in Edge Networks. *IEEE Netw. Lett.* **2025**, *early access*. [CrossRef]
91. Turina, V.; Zhang, Z.; Esposito, F.; Matta, I. Federated or Split? A Performance and Privacy Analysis of Hybrid Split and Federated Learning Architectures. In Proceedings of the IEEE 14th International Conference on Cloud Computing, Chicago, IL, USA, 5–10 September 2021; pp. 250–260.
92. Liao, Y.; Xu, Y.; Xu, H.; Yao, Z.; Huang, L.; Qiao, C. ParallelSFL: A Novel Split Federated Learning Framework Tackling Heterogeneity Issues. In Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, Washington, DC, USA, 18–22 November 2024; pp. 845–860.
93. Almalik, F.; Alkhunaizi, N.; Almakky, I.; Nandakumar, K. FeSViBS: Federated Split Learning of Vision Transformer with Block Sampling. In Proceedings of the 26th International Conference on Medical Image Computing and Computer Assisted Intervention, Part II, Vancouver, BC, Canada, 8–12 October 2023; pp. 350–360.
94. Liu, X.; Deng, Y.; Mahmoodi, T. Energy Efficient User Scheduling for Hybrid Split and Federated Learning in Wireless UAV Networks. In Proceedings of the IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; pp. 1–6.
95. Shiranthika, C.; Hadizadeh, H.; Saeedi, P.; Bajić, I.V. SplitFedZip: Learned Compression for Data Transfer Reduction in Split-Federated Learning. *arXiv* **2024**, arXiv:2412.17150.
96. Mudvari, A.; Jiang, Y.; Tassiulas, L. SplitLLM: Collaborative Inference of LLMs for Model Placement and Throughput Optimization. *arXiv* **2024**, arXiv:2410.10759.
97. Li, N.; Wang, X.; Du, X.; Cai, P.; Wang, T. SAFL: Structure-Aware Personalized Federated Learning via Client-Specific Clustering and SCSI-Guided Model Pruning. *arXiv* **2025**, arXiv:2501.18659.
98. Oh, S.; Park, J.; Vepakomma, P.; Baek, S.; Raskar, R.; Bennis, M.; Kim, S.L. LocFedMix-SL: Localize, Federate, and Mix for Improved Scalability, Convergence, and Latency in Split Learning. In Proceedings of the ACM Web Conference, Lyon, France, 25–29 April 2022; pp. 3347–3357.

99. Alromih, A.; Clark, J.A.; Gope, P. Privacy-Aware Split Learning Based Energy Theft Detection for Smart Grids. In Proceedings of the 24th International Conference on Information and Communications Security, Canterbury, UK, 5–8 September 2022; pp. 281–300.
100. Plate, T.A. Holographic reduced representations. *IEEE Trans. Neural Netw.* **1995**, *6*, 623–641. [CrossRef] [PubMed]
101. Wu, M.; Cheng, G.; Ye, D.; Kang, J.; Yu, R.; Wu, Y.; Pan, M. Federated Split Learning With Data and Label Privacy Preservation in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2024**, *73*, 1223–1238. [CrossRef]
102. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proceedings of the 9th International Conference on Learning Representations, Virtual Event, 3–7 May 2021.
103. Taïk, A.; Mlika, Z.; Cherkaoui, S. Data-Aware Device Scheduling for Federated Edge Learning. *IEEE Trans. Cogn. Commun. Netw.* **2022**, *8*, 408–421. [CrossRef]
104. Kornblith, S.; Norouzi, M.; Lee, H.; Hinton, G. Similarity of Neural Network Representations Revisited. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 3519–3529.
105. Goyal, P.; Dollár, P.; Girshick, R.B.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv* **2017**, arXiv:1706.02677.
106. Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. *arXiv* **2014**, arXiv:1404.5997.
107. Althobaiti, A.; Jindal, A.; Marnerides, A.K.; Roedig, U. Energy theft in smart grids: A survey on data-driven attack strategies and detection methods. *IEEE Access* **2021**, *9*, 159291–159312. [CrossRef]
108. Zhou, T.; Hu, Z.; Wu, B.; Chen, C. SLPerf: A Unified Framework for Benchmarking Split Learning. *arXiv* **2023**, arXiv:2304.01502.
109. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.