

Privacy in Collaborative Deep Learning Systems: A Taxonomy and Archetypes

SASCHA RANK, Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Karlsruhe, Germany, KASTEL Security Research Labs, Karlsruhe, Germany, and School of Computation, Information and Technology, Technical University of Munich, Heilbronn, Germany

DAVID JIN, Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Karlsruhe, Germany and KASTEL Security Research Labs, Karlsruhe, Germany

SCOTT THIEBES, School of Economics and Management, Tongji University, Shanghai, China

ALI SUNYAEV, School of Computation, Information and Technology, Technical University of Munich, Heilbronn, Germany

Collaborative deep learning (CDL) systems can help overcome resource constraints in neural network training and use by enabling multiple parties to share resources. Despite their promising benefits, they also raise privacy and confidentiality concerns. To address these concerns, CDL systems integrate privacy-preserving techniques (PPTs), such as differential privacy, homomorphic encryption, secure multi-party computation, and trusted execution environments. Researchers and practitioners face a plethora of CDL system designs that are scattered across research streams. To support consistent description and informed selection of CDL systems, we present a privacy-focused taxonomy and describe four CDL system archetypes and their use of PPTs.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** → **Machine learning**; • **Security and privacy**; • **Software and its engineering** → **Designing software**;

Additional Key Words and Phrases: Deep learning, privacy-preserving techniques, differential privacy, homomorphic encryption, secure multi-party computation, trusted execution environments, federated learning

ACM Reference Format:

Sascha Rank, David Jin, Scott Thiebess, and Ali Sunyaev. 2026. Privacy in Collaborative Deep Learning Systems: A Taxonomy and Archetypes. *ACM Comput. Surv.* 58, 11, Article 275 (April 2026), 37 pages. <https://doi.org/10.1145/3801094>

This work was supported by KASTEL Security Research Labs.

Authors' Contact Information: Sascha Rank (corresponding author), Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Karlsruhe, Baden-Württemberg, Germany, KASTEL Security Research Labs, Karlsruhe, Baden-Württemberg, Germany, and School of Computation, Information and Technology, Technical University of Munich, Heilbronn, Baden-Württemberg, Germany; e-mail: sascha.rank@tum.de; David Jin, Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Karlsruhe, Baden-Württemberg, Germany and KASTEL Security Research Labs, Karlsruhe, Baden-Württemberg, Germany; e-mail: david.jin@kit.edu; Scott Thiebess, School of Economics and Management, Tongji University, Shanghai, China; e-mail: scott.thiebess@tongji.edu.cn; Ali Sunyaev, School of Computation, Information and Technology, Technical University of Munich, Heilbronn, Baden-Württemberg, Germany; e-mail: sunyaev@tum.de.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 0360-0300/2026/04-ART275

<https://doi.org/10.1145/3801094>

1 Introduction

Training **neural networks (NNs)** with **deep learning (DL)** requires large amounts of high-quality training data [129] and substantial computational resources [50]. Data owners, whether individuals or organizations, often encounter resource constraints in practice: they lack the resources required to train NNs with adequate predictive performance on their own [104].

Collaborative DL (CDL) systems are a promising approach to overcome resource constraints in DL. In CDL systems, parties share resources to train NNs or to use NNs for inference. During training, CDL systems allow data owners to train NNs with their combined training data and to use third parties' computational resources [4, 84]. During inference, CDL systems enable data owners to use third parties' NNs and computational resources to make predictions [29, 106]. Two fundamental types of CDL systems can be distinguished: centralized and distributed CDL systems [105]. In centralized CDL systems, data owners transmit training or inference data to one or more central servers [49]. In distributed CDL systems, such as federated learning systems, data owners keep their training and inference data local and only exchange intermediate results of local computations (e.g., gradients or NN parameters) with other data owners or servers [79].

Despite the promising benefit of overcoming resource constraints, CDL systems also cause confidentiality and privacy concerns [80, 86]. If data owners transmit plain training data or inference data to central servers, confidential or private information is directly exposed to other parties. Even if they only exchange intermediate results, other parties may extract confidential or private information from intermediate results (e.g., with membership inference or reconstruction attacks [112]). To address confidentiality and privacy concerns, researchers and practitioners developed various CDL system designs that integrate **privacy-preserving techniques (PPTs)**. PPTs are technical approaches that are used to protect confidential or private information in training data, inference data, and NNs. Prevalent PPTs used in CDL systems include **differential privacy (DP)**, **homomorphic encryption (HE)**, **secure multi-party computation (MPC)**, and **trusted execution environments (TEEs)**.

Practitioners intending to use CDL systems are confronted with various CDL systems designs that integrate PPTs in manifold ways. CDL system designs that integrate PPTs differ regarding key characteristics that influence their suitability for use cases. In some CDL systems, for example, data owners transmit unprotected intermediate results to a central server. The central server applies a DP mechanism to prevent data owners from inferring private information from training data of other data owners [139]. Such CDL systems may be suitable if data owners trust the central server to correctly apply a DP mechanism. However, such CDL systems are unsuitable if data owners do not trust the central server. Hence, not all CDL systems are equally suitable for every use case. To make informed decisions when selecting CDL systems for use cases, practitioners must understand the range of available CDL system designs, their key characteristics, and their use of PPTs.

Research on CDL system designs that integrate different PPTs is scattered across research streams with heterogeneous terminology. For example, CDL systems that use DP mechanisms are often described in DL and statistics research [3, 22, 92], whereas CDL systems that use MPC tend to be described in security research [4, 61, 62]. Hence, understanding the range of available CDL system designs requires drawing parallels between CDL systems that use distinct PPTs from a dispersed body of literature. To support researchers and practitioners in gaining a comprehensive understanding, a synthesis of CDL systems, their key characteristics, and their use of PPTs is needed. Therefore, we set out to answer the following **research questions (RQs)**:

- *RQ1: What key characteristics differentiate CDL system designs?*
- *RQ2: How do common CDL system designs use PPTs?*

To answer our RQs, we adopted a three-phased research approach. First, we searched for relevant publications on CDL systems that use PPTs with a snowballing approach [140]. Second, we developed a privacy-focused CDL system taxonomy based on 122 scientific publications describing 149 CDL systems by following a systematic taxonomy development method [90]. Third, we applied a taxonomy-based clustering approach to derive archetypes of CDL systems [51] and analyzed how CDL systems that implement these archetypes use PPTs.

This article makes three main contributions. First, we present a systematically developed, privacy-focused CDL system taxonomy comprising 4 categories with 16 dimensions and 49 characteristics. The CDL system taxonomy supports researchers and practitioners in consistently classifying and describing CDL systems and their characteristics. Second, we present four CDL system archetypes. The CDL system archetypes illustrate prevalent CDL system designs and their common characteristics. For researchers and practitioners, the CDL system archetypes provide a starting point for selecting CDL systems suitable for use cases. Third, we describe how CDL systems implement the CDL system archetypes using PPTs to protect training data, inference data, and NNs. This provides a basis for understanding which privacy goals can be fulfilled with PPTs in CDL system designs.

This article unfolds as follows. Section 2 introduces relevant background knowledge on DL, CDL systems, privacy-related attacks in DL, and PPTs used in CDL systems. In Section 3, we describe how we developed a privacy-focused CDL system taxonomy and CDL system archetypes based on scientific literature. Section 4 presents our privacy-focused CDL system taxonomy and its categories, dimensions, and characteristics. In Section 5, we delineate the developed CDL system archetypes and describe how prevalent CDL system designs integrate PPTs to implement these archetypes. In Section 6, we discuss the principal findings, contributions, and limitations of our work and describe promising directions for future research. Section 7 concludes this article.

2 Background

2.1 Deep Learning

DL is a machine learning technique that uses NNs to solve tasks [63]. NNs are composed of neurons that receive inputs, process these inputs, and produce outputs. For each input $x_1, \dots, x_n \in \mathbb{R}$, a neuron has an associated weight $w_1, \dots, w_n \in \mathbb{R}$. Neurons compute the weighted sum of their inputs $y = w_1x_1 + \dots + w_nx_n + b$, where $b \in \mathbb{R}$ is an additional bias term. The weighted sum y is passed into a non-linear activation function $f : \mathbb{R} \rightarrow \mathbb{R}$ to produce the output $z = f(y)$ of the neuron. An example of a typical activation function is the rectified linear unit function $ReLU(x) = \max(0, x)$.

Neurons in NNs are commonly arranged in layers. The first layer of a NN, the input layer, receives input data (e.g., an image). The following layers are referred to as hidden layers. Hidden layers extract increasingly abstract features from input data (e.g., edges, contours, and object parts). The last layer, the output layer, produces the results of a NN (e.g., which objects are identified on an image) [40]. NNs are called feed-forward NNs if data flows in only one direction, from the input layer to the output layer. Conversely, NNs that feed outputs of neurons back into neurons in the same or previous layers are referred to as recurrent NNs. Modern NNs use various types of layers, such as attention layers, convolutional layers, and recurrent layers.

A NN architecture defines its structure, including the number and types of layers, the number of neurons per layer, and the activation functions used. Different NN architectures are suitable for different types of tasks. For instance, NNs with convolutional layers are especially suitable for image processing tasks. Meanwhile, NNs with recurrent layers are more suitable for tasks that benefit from remembering information from prior inputs, such as natural language processing tasks and time series analysis tasks. Suitable NN architectures for tasks can be identified through manual tuning or automated neural architecture search techniques [33].

A typical DL lifecycle comprises two principal stages: a training stage and an inference stage. In the training stage, a learning algorithm uses training data to iteratively optimize parameters (e.g., weights and biases) of a NN by minimizing a loss function. Learning algorithms in DL typically use two main processes: forward propagation and backward propagation. During forward propagation, input data flows through the layers of a NN to generate an output (i.e., a prediction). A loss function evaluates how well the NN currently performs by measuring the difference between the NN's predictions and target values (e.g., labels). During backward propagation, the learning algorithm calculates how each parameter contributed to the difference between predictions and target values and adjusts each parameter to minimize the difference in subsequent iterations. In the inference stage, a trained NN is used with inference data (i.e., unseen data instances) to make predictions. Only forward propagation is required in the inference stage because parameters are not adjusted.

2.2 Collaborative Deep Learning Systems

CDL systems can help overcome resource constraints in DL, such as limited availability of training data or computational resources. In CDL systems, multiple parties share resources to train NNs or to use NNs for inference. Parties are individuals or organizations that participate in CDL systems. Two principal types of parties that actively participate in CDL systems can be distinguished based on their goals and the resources they provide: data owners and servers. Data owners aim at training NNs or to use NNs for inference. They provide training data (in the training stage) or inference data (in the inference stage) to CDL systems. Depending on the CDL system design, data owners provide varying amounts of computational resources. Servers aim at supporting the execution of the training or the inference stage. They provide computational resources to CDL systems. Unlike data owners, they do not provide their own training data or inference data. However, servers may receive training data or inference data from data owners during the training or inference stage. CDL systems use varying numbers of servers. Some CDL systems do not use any servers.

Extant literature commonly distinguishes between two types of CDL systems: centralized CDL systems and distributed CDL systems [68]. In centralized CDL systems, training and inference stages are executed in a centralized manner. One or more servers hold (parts of) a NN. In the training stage, data owners upload training data to servers. Servers train a NN with training data of data owners. After training, servers either return the trained NN to data owners or retain it for the inference stage. In the inference stage, servers use a NN to make predictions. Servers either use NNs retained from the training stage or NNs trained outside the CDL system. NNs trained outside the CDL system are provided to CDL systems by an external NN owner [19] or by a party in the CDL system (e.g., a server) [82]. Data owners transmit inference data to servers. Servers use inference data and a NN to make predictions and return these predictions to data owners.

In distributed CDL systems, data owners do not transmit plain training data or inference data to other parties. Instead, they compute intermediate results based on local training data or inference data and transmit these intermediate results to other parties [55]. Distributed CDL systems can be further categorized into distributed CDL systems with and without servers. In distributed CDL systems with servers, servers orchestrate the execution of training and inference stages. For example, in federated learning systems, a central server broadcasts a global NN to data owners. Data owners train the NN with their local training data and return intermediate results (e.g., gradients or NN parameters) to the central server. The central server aggregates the intermediate results (e.g., by averaging gradients) and updates the global NN [79]. In distributed CDL systems without servers, data owners take on the tasks of central servers. For example, in gossip learning systems, each data owner receives intermediate results from other data owners and uses these intermediate results to update its local NN [47]. In swarm learning systems, data owners use blockchain smart contracts to aggregate intermediate results and update a global NN [138].

Table 1. Overview of Privacy-Related Attacks in DL and References to Black-Box and White-Box Examples

Attack Type	Goal	Black-Box Examples	White-Box Examples
Membership inference	Determine whether specific data instances were part of the training dataset of NNs	Shokri et al. [124], Choquette-Choo et al. [23]	Rahman et al. [103], Nasr et al. [86]
Model extraction	Infer information about the architecture and parameters of NNs or fully reconstruct NNs	Tramèr et al. [132], Yu et al. [146]	–
Property inference	Extract information about the training dataset of NNs that may be unrelated to the task of NNs	Song and Raghunathan [127], Mahloujifar et al. [78]	Ganju et al. [37], Song and Raghunathan [127]
Reconstruction	Recreate data samples or generate class representatives of data samples used to train NNs	Fredrikson et al. [35], Salem et al. [117]	He et al. [46], Zhang et al. [149]

2.3 Privacy-Related Attacks in Deep Learning

To understand the need for PPTs in CDL systems, knowledge of privacy-related attacks in DL is required. Data owners in CDL systems must exchange data (either plain training and inference data or intermediate results) with other parties. Even if adversaries do not gain direct access to training data, inference data, or NNs, they can exploit access to intermediate results or outputs of DL stages to extract private information from data or NNs. Privacy-related attacks in DL can be broadly categorized into black-box and white-box attacks [112]. Black-box attacks assume that adversaries have limited access to a NN. Adversaries can query a NN with arbitrary inference data and observe the corresponding outputs, but they do not know the architecture or parameters of a NN. In contrast, white-box attacks assume that adversaries know the architecture or parameters of a NN. The following sections describe the most relevant types of privacy-related attacks in DL. Table 1 summarizes relevant types of privacy-related attacks and provides references to black-box and white-box examples.

2.3.1 Membership Inference Attacks. Membership inference attacks aim at determining whether specific training data instances were part of the training dataset of a NN [112]. Such attacks exploit that NNs tend to behave differently for data instances they were trained on compared with data instances they encounter for the first time. For example, the confidence of a NN in its predictions is often higher for data instances from the training dataset compared with unseen data instances [124]. Adversaries can execute membership inference attacks with black-box [124] or white-box access [86] to a NN. For instance, black-box membership inference attacks can be executed by training multiple shadow NNs that mimic the behavior of a target NN. Adversaries train these shadow NNs on a dataset similar to that of the target NN. Then, adversaries query the shadow NNs with data instances they were trained on and with unseen data instances. Finally, adversaries train a binary classifier on the shadow NN outputs to distinguish between data instances from the training dataset of the target NN and unseen data instances [124]. White-box membership inference attacks are typically more effective as adversaries can exploit additional information [112].

2.3.2 Model Extraction Attacks. Model extraction attacks aim at inferring information about the architecture and parameters of NNs or to fully reconstruct NNs. As white-box attacks assume that an adversary already has full access to a NN, model extraction attacks typically assume black-box access to a NN. Model extraction attacks can also be used as preparation to execute further attacks, such as membership inference attacks [112]. Adversaries execute model extraction attacks, for example, by repeatedly querying NNs to generate pairs of inference data instances and corresponding NN outputs. These pairs are then used to train a shadow NN that approximates the behavior of a target NN [132].

2.3.3 Property Inference Attacks. Property inference attacks aim at extracting information about the training dataset of a NN that may be unrelated to the actual task of the NN and that is not explicitly encoded as features in the training dataset [112]. For example, adversaries may use property inference attacks to infer whether the training dataset of a NN contains people of a certain age while the actual task of the NN is to detect diseases. Property inference attacks can be executed with black-box [127] or white-box access [37] to a NN. Like membership inference attacks, property inference attacks can be executed using shadow NNs. Adversaries create two auxiliary datasets: one with the target property, and one without the target property. Then, they train two shadow NNs using the auxiliary datasets and repeatedly query these two shadow NNs to obtain outputs. Finally, adversaries train a binary classifier to distinguish between outputs originating from NNs with and without the target property in their training data [80].

2.3.4 Reconstruction Attacks. Reconstruction attacks aim at recreating data instances from the training dataset of a NN. Adversaries can execute reconstruction attacks with black-box [35] or white-box access [46] to a NN. Reconstruction attacks can also be executed using shadow NNs [46]. There are two types of reconstruction attacks: those that recover actual data instances from training data and those that generate class representatives [112]. Class representatives are synthetic data instances that may not exist in the training dataset but approximate the average characteristics of a class. For example, for face recognition, a class representative could be a synthetic image that, while it depicts a realistic human face, does not exist in the training dataset.

2.4 Privacy-Preserving Techniques

To protect training data, inference data, and NNs, CDL systems integrate PPTs (i.e., DP, HE, MPC, and TEEs). The following sections briefly introduce each of these PPTs. Section 5 details how PPTs are used in prevalent CDL system designs to protect training data, inference data, and NNs.

2.4.1 Differential Privacy. DP is a mathematical framework that provides quantifiable definitions of privacy. To satisfy these definitions, DP mechanisms are used. DP mechanisms aim at enabling the extraction of useful information from a dataset while protecting individual data instances in the dataset. Useful properties of DP mechanisms include their sequential and parallel composability and their invariance to post-processing [101]. Intuitively, a mechanism fulfills the definition of DP if no output is significantly more or less likely based on whether a specific data instance was included in a dataset or not. Formally, a randomized mechanism $\mathcal{M}$ satisfies the definition of ϵ -DP if for any two adjacent datasets D_1, D_2 and all subsets S of the output space, the ratio of likelihoods to get a particular output is bounded by e^ϵ , that is, $P[\mathcal{M}(D_1) \in S] \leq e^\epsilon \cdot P[\mathcal{M}(D_2) \in S]$ [30]. Datasets are adjacent if they differ in exactly one data instance. ϵ is referred to as the privacy budget. Smaller values of ϵ provide stronger privacy guarantees but also reduce the accuracy of a DP mechanism. In addition to ϵ -DP, alternative definitions of DP have been proposed that generalize or modify the ϵ -DP definition. Examples include (ϵ, δ) -DP [31], concentrated DP [32], Rényi-DP [81], and zero-concentrated DP [16].

DP can be applied at different stages of a typical DL lifecycle: to training data before training, to training algorithms during training, or to predictions of NNs during inference. A common approach is to apply DP to gradients during training with the **differentially private stochastic gradient descent (DP-SGD)** algorithm [101]. The DP-SGD algorithm satisfies (ϵ, δ) -DP by clipping the L_2 -norm of gradients and adding calibrated Gaussian noise. The privacy budget spent over multiple training iterations with the DP-SGD algorithm can be tracked with the moments accountant method [1].

2.4.2 Homomorphic Encryption. HE schemes enable arithmetic operations on encrypted data. After decryption, the result of an operation performed on homomorphically encrypted data matches

the result of the same operation performed on the corresponding unencrypted data. Based on the type and number of operations that can be performed, HE schemes can be categorized into *partially* HE, *somewhat* HE, *leveled* HE, and *fully* HE schemes [2]. *Partially* HE schemes allow either addition (e.g., Paillier [95]) or multiplication (e.g., RSA [113]) for an unlimited number of times. *Somewhat* HE schemes (e.g., BGN [12]) and *leveled* HE schemes (e.g., CKKS [20]) support addition and multiplication, but for a limited number of times. *Fully* HE schemes allow an unlimited number of addition and multiplication operations (e.g., TFHE [21]). There are single-key and multi-key variants of many HE schemes. Single-key HE schemes only allow operations on ciphertexts encrypted with the same key. Multi-key HE schemes allow operations on ciphertexts encrypted under different keys (e.g., by different parties) [25].

During encryption, HE schemes add random noise to encrypted data to ensure ciphertext indistinguishability (i.e., that multiple encryptions of the same plaintext produce different ciphertexts). Each operation performed on ciphertext increases the amount of noise in the result. After a predetermined number of operations, the noise exceeds a threshold, and the ciphertext can no longer be decrypted accurately. To allow an unlimited number of operations, fully HE schemes incorporate bootstrapping [38]. Bootstrapping outputs a fresh ciphertext of the same plaintext with reduced noise.

2.4.3 Secure Multi-Party Computation. MPC protocols enable multiple parties to jointly compute functions over their inputs without revealing these inputs. Two-party computation (2PC) protocols are MPC protocols between exactly two parties. Foundational research proved that any computable function can be computed securely with general-purpose MPC protocols [39, 145]. However, general-purpose MPC protocols are too inefficient for many practical applications because of their high computation and communication overhead. Therefore, researchers developed specialized MPC protocols that focus on improving efficiency for practical applications [150]. MPC builds on a variety of primitives, including HE, oblivious transfer, garbled circuits, and secret sharing. The following briefly introduces the primitives commonly used in MPC protocols.

Oblivious transfer protocols are 2PC protocols between a sender and a receiver [102]. They allow the receiver to choose and obtain a subset of multiple messages held by the sender. The sender does not learn the choice of the receiver, whereas the receiver does not learn the unchosen messages. In the simplest case of a 1-out-of-2 oblivious transfer, the sender has two messages m_0 and m_1 . The receiver has a choice bit b and wants to receive the message m_b without the sender learning b . The idea of oblivious transfer can be generalized to k -out-of- n oblivious transfer. Here, the receiver chooses and obtains k messages from a set of n messages held by the sender [15].

Garbled circuits are 2PC protocols between a garbler and an evaluator [145]. Garbled circuit protocols consist of four main steps. First, the garbler constructs a Boolean circuit representing the function to be computed. The garbler assigns two cryptographic labels to each wire (corresponding to 0 and 1) and encrypts each possible output label of a gate using the corresponding input labels as encryption keys. Second, the garbler sends the encrypted circuit and its own input labels to the evaluator. The evaluator obtains its input labels via oblivious transfer without revealing its choices to the garbler. Third, the evaluator evaluates the circuit by sequentially decrypting gate outputs using the provided input labels without learning intermediate values. Finally, depending on the protocol variant, the garbler, the evaluator, or both parties obtain the final output values. If the garbler should obtain the final output values, the evaluator sends the final output label to the garbler. The garbler maps the final output label to actual output values.

Secret sharing is a cryptographic technique that enables the distribution of a secret as secret shares to a set of n parties. To reconstruct the secret, all or a specific number of parties must combine their secret shares. Secret sharing schemes that enable a specific number of parties to reconstruct a secret are referred to as threshold secret sharing schemes. In an (n, t) -threshold secret

sharing scheme, a dealer splits a secret into n shares and distributes the shares to n parties. At least t parties are required to reconstruct the secret. There are various types of secret sharing, including additive secret sharing, Boolean secret sharing, and Shamir's secret sharing. In additive secret sharing, a dealer shares a secret value x with n parties by selecting random values $x_1, \dots, x_n$ such that $x = x_1 + \dots + x_n$. In Boolean secret sharing, a secret bit b is shared with n parties such that $b = b_1 \oplus \dots \oplus b_n$ [24]. Shamir's secret sharing is a threshold secret sharing scheme. In Shamir's secret sharing, a dealer shares a secret s by constructing a random polynomial $f(x)$ of degree $t - 1$ where $f(0) = s$. Then, the dealer selects n points on the polynomial and gives one point to each party. Since t points are required to define a polynomial of degree $t - 1$, at least t parties must combine their secret shares to reconstruct the secret [122]. Linear operations, such as adding two secret-shared values, adding constants to secret-shared values, or multiplying secret-shared values with constants, can be executed without interaction between parties holding secret shares. To enable efficient multiplication of two secret-shared values, Beaver triples can be used [9]. A Beaver triple consists of three values a , b , and c , such that $c = a \times b$. Beaver triples are pre-computed and distributed to the parties that hold secret shares. Importantly, each Beaver triple can only be used once in a protocol to avoid leakage of private data.

2.4.4 Trusted Execution Environments. TEEs are hardware-enforced execution environments that isolate computations from untrusted code, including privileged software such as the operating system. They aim at ensuring confidentiality and integrity of computations inside the TEE [116]. An important feature of TEEs is their remote attestation. Remote attestation allows third parties to verify the integrity of code running inside a TEE. Remote attestation relies on a hardware-based root of trust. To enable remote attestation, processors that support TEEs embed cryptographic keys to certify their authenticity. After successful remote attestation, third parties can establish an encrypted communication channel with the TEE to securely exchange sensitive data. Examples of TEE implementations include ARM TrustZone [100] and Intel Software Guard Extensions [26].

3 Methods

Our research approach comprised three phases: literature search, taxonomy development, and archetype development. The following describes each phase in detail. Figure 1 summarizes our research approach.

3.1 Literature Search

In the first phase, we applied the snowballing approach proposed by Wohlin [140] to identify scientific literature on CDL systems that use PPTs. The snowballing approach uses a start set of relevant papers and identifies additional relevant papers through backward and forward searches. For an overview of the snowballing literature search and the number of papers identified and excluded in each step, we refer to Section A.1 in the appendix.

To build a start set of CDL systems that use PPTs, we searched the databases ACM Digital Library, IEEE Xplore, Scopus, and Web of Science with the following search string: *TI-KE-AB ((priva* OR confidential*) AND (neural network OR deep learning) AND system)*. Given the extensive body of research on CDL systems, we did not aim at analyzing all CDL systems but rather a representative sample to develop a taxonomy. Therefore, we followed the suggestion by Wohlin [140] and limited our initial search to highly cited papers. In particular, we included only papers with at least 100 citations at the time of our search, according to the database where we found the article. This threshold was applied only to build the start set, not during the subsequent snowballing. Our search in January 2024 yielded a list of 256 papers. After screening this list and removing 81 duplicates

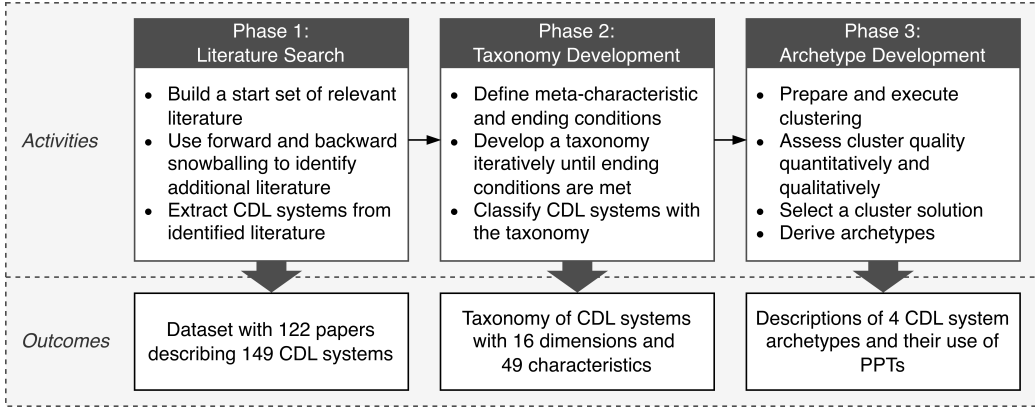


Fig. 1. Overview of our research approach.

and 1 retracted article, we ended up with 174 unique papers. We then assessed the titles, abstracts, and full texts of the 174 unique papers for eligibility. We removed 112 papers that did not describe at least one CDL system in detail. Moreover, we excluded 41 papers that were limited to specialized application areas (e.g., electrocardiogram classification or intrusion detection) as we were interested in general-purpose CDL systems. This process yielded a start set of 21 papers for snowballing.

To identify additional papers that describe CDL systems, we applied one iteration of backward and forward snowballing [140]. For backward snowballing, we screened the reference list of each paper in our start set. We identified candidate papers for potential inclusion based on their title, as well as the place and context of the reference. For forward snowballing, we used Google Scholar to identify papers that reference papers in our start set. We screened the titles and abstracts of the first 100 results on Google Scholar for each paper in our start set to identify candidate papers. Based on their full texts, we decided whether to include candidate papers identified through forward and backward snowballing. We identified 101 additional papers that describe CDL systems through backward and forward snowballing. Combined with the start set of 21 papers, our snowballing literature search yielded a final set of 122 papers.

To complete the literature search, we extracted CDL systems from the identified literature. In the final set of 122 papers, we identified 149 distinct CDL systems. These form the foundation for taxonomy development and archetype development in the subsequent phases.

3.2 Taxonomy Development

In the second phase, we developed a privacy-focused CDL system taxonomy. Taxonomies are important tools in many disciplines as they support researchers and practitioners in understanding and analyzing complex domains [90]. To develop a privacy-focused CDL system taxonomy, we applied the systematic taxonomy development method proposed by Nickerson et al. [90]. The method comprises 7 steps, with steps 3 to 7 being carried out iteratively.

As a first step, we defined the meta-characteristic for our taxonomy. The meta-characteristic is the most comprehensive characteristic and serves as a basis for the choice of subsequent characteristics in the taxonomy. All characteristics of the taxonomy must logically follow from the meta-characteristic [90]. We defined the meta-characteristic as *principal factors that influence the suitability of CDL systems for use cases*. In the second step, we specified ending conditions. Ending conditions determine when the iterative execution of steps 3 to 7 of the taxonomy development method stops. We adopted the 8 objective and 5 subjective ending conditions proposed by Nickerson

et al. [90]. The objective ending conditions require the dimensions of a taxonomy to have mutually exclusive and collectively exhaustive characteristics. The subjective ending conditions require a taxonomy to be comprehensive, concise, explanatory, extendable, and robust.

We then iteratively executed steps 3 to 7. The iterations followed either an empirical-to-conceptual (inductive) approach or a conceptual-to-empirical (deductive) approach. In empirical-to-conceptual iterations, we carefully analyzed and compared CDL systems described in literature to identify relevant CDL system characteristics. In conceptual-to-empirical iterations, we used conceptual literature and our knowledge of CDL systems to revise the taxonomy. At the end of each iteration, we re-examined all CDL systems from prior empirical-to-conceptual iterations and assigned the characteristics of the revised taxonomy to each CDL system. This provided us with a complete dataset for the archetype development in the third phase. In total, we conducted 6 iterations of steps 3 to 7. For an overview of the taxonomy development iterations, we refer to Section A.2 in the appendix.

3.3 Archetype Development

In the third phase, we used cluster analysis to derive CDL system archetypes that support the selection of CDL systems for use cases. Cluster analysis aims to group objects such that objects within the same group are as similar as possible, whereas objects in different groups are as dissimilar as possible. We applied a taxonomy-based cluster analysis approach comprising three steps to identify prevalent CDL system designs [51]. In a fourth step, we analyzed the CDL systems in each cluster and their use of PPTs.

First, we prepared the data for clustering. We used our data on 149 CDL systems classified using the CDL system taxonomy from the second phase. We carefully assessed the dimensions of the CDL system taxonomy using our domain knowledge and deemed all dimensions relevant for clustering. Moreover, we prepared label-encoded and one-hot-encoded versions of our dataset to enable the use of different clustering algorithms. Our dataset did not have any missing values.

Second, we clustered the 149 CDL systems based on the characteristics of our taxonomy. We followed recommendations from extant literature and compared seven clustering algorithms suitable for categorical data [51]. In particular, we used the following clustering algorithms and similarity measures: CD-clustering with Hamming distance, k-medoids with Hamming distance, k-modes with Ng's dissimilarity [88], k-PbC with information-theoretic-based dissimilarity [89], k-SCC with Lin similarity [13], LSHFk-Centers with DILCA [54], and StepMix with latent class probabilities. We conducted 1,000 runs of each clustering algorithm for numbers of clusters k ranging from 3 to 10 and calculated the average silhouette scores (see Section A.3 in the appendix). Higher silhouette scores indicate better cluster cohesion and separation [51].

Third, we assessed the clustering solutions with a quantitative and a qualitative evaluation. For the quantitative evaluation, we analyzed the average silhouette scores of the selected clustering algorithms and numbers of clusters. The LSHFk-Centers algorithm with 4 clusters reached the highest average silhouette score. Since the highest average silhouette scores for all numbers of clusters were in close proximity, we decided to also evaluate these clustering solutions in a qualitative evaluation. For the qualitative evaluation, we compared and rated the meaningfulness of the clustering solution with the highest silhouette score for each number of clusters. We discussed and compared the explanatory power of these clustering solutions and sought meaningful interpretations for all clusters. Ultimately, we decided that the clustering solution from the LSHFk-Centers algorithm with 4 clusters provided the highest meaningfulness, confirming the quantitative evaluation.

Fourth, we analyzed and compared the CDL systems in each cluster of the selected clustering solution to develop CDL system archetypes. We extracted characteristics shared by a substantial part of CDL systems in each cluster and analyzed how CDL systems in each cluster use PPTs to

Table 2. A Privacy-Focused CDL System Taxonomy

	Dimension	Characteristics
Privacy	Inference Data Privacy Goal	None Input Privacy
	Training Data Privacy Goal	None Input Privacy Output Privacy Input Privacy and Output Privacy
	NN Architecture Privacy Goal	None From DOs From Servers From DOs and Servers
	NN Parameter Privacy Goal	None From DOs From Servers From DOs and Servers
Processes	Data Owner Interactivity	Non-Interactive Interactive
	DL Stages	Training Inference Training and Inference
	Main Computing Parties	DOs Servers DOs and Servers
Threats	Colluding Parties	None DOs DOs and Servers
	Data Owner Corruption	None Honest-but-Curious Malicious
	Server Corruption	None Honest-but-Curious Malicious
Topology	Communication Paths	DO $\leftrightarrow$ DO DO $\leftrightarrow$ Server DO $\leftrightarrow$ DO and DO $\leftrightarrow$ Server DO $\leftrightarrow$ Server and Server $\leftrightarrow$ Server
	Data Partitioning	Arbitrary Horizontal Vertical
	Number of Data Owners	One Few (≤ 100) Many (> 100)
	Number of Servers	None One Multiple
	Scope of NNs	Global Individual Split
	Trusted Third Party	Not Required Required

Data owners are abbreviated as DOs in characteristics.

meet privacy goals. We describe the CDL system archetypes, their common characteristics, and their use of PPTs in Section 5.

4 A Privacy-Focused Collaborative Deep Learning System Taxonomy

To support researchers and practitioners in consistently classifying CDL systems that use PPTs and in describing their characteristics, we developed a privacy-focused CDL system taxonomy. Our privacy-focused CDL system taxonomy comprises four categories of dimensions: *privacy*, *processes*, *threats*, and *topology*. In total, these categories include 16 dimensions with 49 characteristics. In the following, we describe each category, dimension, and characteristic. Table 2 provides an overview of our privacy-focused CDL system taxonomy.

4.1 Privacy

The privacy category comprises four dimensions with 14 characteristics related to the protection of inference data, training data, and NNs from unauthorized access by third parties.

Inference Data Privacy Goal. Protecting inference data is a key challenge in CDL systems. The two characteristics of the inference data privacy goal dimension describe whether a CDL system is designed to protect inference data from unauthorized access by third parties. CDL systems that are not designed to protect inference data have an inference data privacy goal of *none*. Such CDL systems may be suitable if parties only interact during the training stage or if inference data does not contain private information. CDL systems with the characteristic *input privacy* are designed to protect inference data from parties that execute the computations required for the inference stage. Input privacy covers the inference data itself and intermediate and final results of computations (i.e., predictions). If data owners transmit inference data to untrusted parties who execute computations, these untrusted parties should not be able to learn private information from inference data.

Training Data Privacy Goal. Another key challenge in CDL systems is the protection of training data. The training data privacy goal dimension comprises four characteristics describing the type of protection a CDL system is designed to ensure for training data. The training data privacy goal is *none* for CDL systems that are not designed to protect training data. CDL systems with the

characteristic *input privacy* are designed to protect training data from parties that execute the computations required for the training stage. Input privacy covers the training data itself and intermediate and final results of computations (i.e., NNs). Input privacy is required when data owners transmit private training data or intermediate results based on private training data to untrusted parties who execute computations. CDL systems that are designed to ensure that outputs of DL stages do not reveal private information from training data have the characteristic *output privacy*. Output privacy is needed when outputs of computations (i.e., NNs or predictions made with NNs) are made available to untrusted parties. CDL systems designed to fulfill both previous goals have the characteristic *input privacy and output privacy*.

Neural Network Architecture Privacy Goal. The NN architecture privacy goal dimension describes which parties a CDL system is designed to protect the structure of NNs from. Development of NN architectures requires substantial resources. Moreover, knowledge of NN architectures enables more effective privacy-related attacks. Therefore, NN owners often want to protect NN architectures from unauthorized access by third parties. The NN architecture privacy goal dimension comprises four characteristics. CDL systems that are not designed to protect NN architectures have a NN architecture privacy goal of *none*. Some CDL systems are designed to protect NN architectures only *from data owners* or only *from servers*. CDL systems that are designed to fulfill both previous goals have a NN architecture privacy goal of *from data owners and servers*.

Neural Network Parameter Privacy Goal. The NN parameter privacy goal dimension describes which parties a CDL system is designed to protect NN parameters (e.g., weights and biases) from. NN owners often want to protect NN parameters from unauthorized access by third parties because training NN parameters requires substantial computational resources. The NN parameter privacy goal dimension comprises four characteristics. CDL systems with a NN parameter privacy goal of *none* are not designed to protect NN parameters. Some CDL systems are designed to protect NN parameters either only *from data owners* or only *from servers*. CDL systems that are designed to combine these goals have the characteristic *from data owners and servers*.

4.2 Processes

The processes category comprises three dimensions with eight characteristics related to the execution of DL stages in CDL systems.

Data Owner Interactivity. The data owner interactivity dimension describes the frequency of data owners' interactions with other parties in CDL systems. This dimension is important when considering what kind of data owners participate in use cases. If data owners use mobile devices with unreliable internet connections, data owners are more likely to drop out during the execution of DL stages. In CDL systems that require frequent interaction with data owners, dropouts of data owners may lead to aborted or delayed execution of DL stages. Hence, CDL systems that require less frequent communication with data owners seem more suitable for such use cases. The data owner interactivity dimension comprises two characteristics. CDL systems in which data owners communicate in at most two communication rounds with other parties (i.e., other data owners or servers) are commonly referred to as *non-interactive*. In a first communication round, data owners transmit inputs to DL stages (i.e., training data or inference data) to other parties. In a second communication round, other parties transmit outputs of DL stages (i.e., NNs after training or predictions after inference) to data owners. Conversely, data owners in *interactive* CDL systems communicate in more than two communication rounds. In interactive CDL systems, data owners commonly exchange intermediate results of computations required for DL stages with other parties.

Deep Learning Stages. The DL stages dimension describes whether a CDL system supports the training of NNs, the use of NNs for inference, or both. Parties commonly intend to execute specific DL stages when using CDL systems. When selecting a CDL system for use cases, it must support the intended DL stages. The DL stages dimension comprises three characteristics. Some CDL systems support only the *training* stage. After training, trained NNs are commonly made available to (a subset of) data owners. Similarly, some CDL systems support only the *inference* stage. Such CDL systems commonly use pre-trained NNs (i.e., NNs trained outside the CDL system). Last, some CDL systems support *training and inference* stages. Such CDL systems are commonly used to train and subsequently use NNs for inference, often without revealing trained NNs to data owners.

Main Computing Parties. The main computing parties dimension describes which parties execute predominant parts of the computations required for DL stages. The distribution of computations across parties is relevant when considering the type of parties participating in use cases. For example, CDL systems in which servers execute predominant parts of computations are more suitable when data owners have limited computational resources. The main computing parties dimension comprises three characteristics. In some CDL systems, either only *data owners* or only *servers* execute predominant parts of computations required for DL stages. In other CDL systems, computations are shared more evenly between *data owners and servers*.

4.3 Threats

The three dimensions and nine characteristics of the threats category are related to the conditions under which CDL systems are designed to fulfill their privacy goals.

Colluding Parties. The colluding parties dimension describes which parties a CDL system assumes to potentially collude. In the context of CDL systems, collusion describes secret communication or coordination between a subset of parties with the aim of gaining unauthorized access to private information. Parties not assumed to collude may still misbehave, but only individually without communicating or coordinating with other parties. To meet privacy goals, it is crucial to correctly identify potentially colluding parties. If colluding parties are not considered as such, they may be able to extract private information from training data, inference data, or NNs. The colluding parties dimension comprises three characteristics. CDL systems that assume no collusion have the characteristic of *none*. Some CDL systems only assume collusion between *data owners*, whereas others assume collusion between *data owners and servers*.

Data Owner Corruption. The data owner corruption dimension describes the assumption of CDL systems about how the most severely corrupted data owner may behave in an attempt to extract private information from other parties in the system. If data owners deviate from this assumption, they may be able to extract private information from inference data, training data, or NNs. CDL systems with multiple data owners do not necessarily assume that all data owners are corrupted according to the data owner corruption dimension. In fact, most CDL systems assume that only a subset of data owners is corrupted. The data owner corruption dimension comprises three characteristics. In CDL systems where the data owner corruption is *none*, all data owners are assumed to strictly follow the protocols of the CDL system without attempting to learn private information. With *honest-but-curious* data owner corruptions, data owners are also assumed to strictly follow the protocols of CDL systems. However, (a subset of) data owners may attempt to use the data they receive to extract private information from other parties. In CDL systems in which data owner corruption is *malicious*, (a subset of) data owners may deviate arbitrarily from protocols of CDL systems to extract private information from other parties.

Server Corruption. The server corruption dimension describes the assumption of CDL systems about how the most severely corrupted server may behave in an attempt to learn private information from other parties in the system. If servers deviate from this assumption, they may be able to extract private information from training data, inference data, or NNs. CDL systems with multiple servers often assume only a subset of servers to be corrupted. The server corruption dimension comprises three characteristics. CDL systems in which the server corruption is *none* assume all servers to strictly follow the protocols of CDL systems without attempting to extract private information. CDL systems that do not use servers also have this characteristic. With *honest-but-curious* server corruptions, servers are also assumed to strictly follow the protocols of CDL systems, but may attempt to use data they receive to extract private information from other parties. CDL systems in which servers may deviate arbitrarily from the protocols of CDL systems to extract private information operate under a *malicious* server corruption assumption.

4.4 Topology

The six dimensions and 18 characteristics of the topology category are related to the parties that participate in CDL systems and their relations.

Communication Paths. The communication paths dimension describes which parties communicate in CDL systems. The communication paths required are important when considering how parties in CDL systems are connected. For example, in use cases where data owners are mobile devices with unreliable internet connections, systems without direct interaction among data owners seem more suitable. Direct interaction between data owners over unreliable internet connections may lead to aborts and interruptions of DL stages. The communication paths dimension comprises four characteristics. CDL systems without servers only require *data owner* $\leftrightarrow$ *data owner* communication paths. CDL systems with one server either require only *data owner* $\leftrightarrow$ *server* communication paths (i.e., if data owners do not interact with each other), or *data owner* $\leftrightarrow$ *data owner* and *data owner* $\leftrightarrow$ *server* communication paths (i.e., if data owners interact with each other). CDL systems with multiple servers require *data owner* $\leftrightarrow$ *server* and *server* $\leftrightarrow$ *server* communication paths.

Data Partitioning. CDL systems support varying data partitionings. Data partitionings describe along which axes data is split between data owners. The type of data partitioning is typically predetermined in use cases. Thus, it is important to select CDL systems that support the data partitioning given in use cases. In *horizontal* data partitionings, data is split by data instance. Data owners hold the same features but for different data instances. For example, different hospitals may contribute health records for different patients. A *vertical* data partitioning describes data that is split by feature. Data owners hold different features of the same data instances. For example, hospitals may contribute health records for the same patients but with different types of measurements. CDL systems with the characteristic *arbitrary* support horizontal and vertical data partitionings.

Number of Data Owners. CDL systems support varying numbers of data owners. The number of data owners dimension defines the maximum number of data owners that can participate in the execution of DL stages supported by a CDL system. In practice, the number of data owners can vary substantially between use cases. For instance, CDL systems are used to train NNs across a small number of hospitals [111] or billions of smartphones [45]. The number of data owners dimension comprises three characteristics. Some CDL systems only support *one* data owner. We refer to CDL systems as supporting *few* data owners if they support up to 100 data owners. In contrast, we refer to CDL systems that support more than 100 data owners as supporting *many* data owners.

Number of Servers. CDL systems require varying numbers of servers. When selecting CDL systems for use cases, it must be ensured that the number of servers required is available. For

example, CDL systems often require a certain number of (non-colluding) servers to fulfill privacy goals. The number of servers dimension comprises three characteristics. First, some CDL systems do not use any servers (i.e., the number of servers is *none*). Second, some CDL systems use exactly *one* server. Third, some CDL systems use *multiple* servers.

Scope of Neural Networks. CDL systems vary regarding the scope of NNs that are trained or used for inference. The scope of NNs describes whether all data owners train and use a single NN (e.g., to save computational resources) or whether individual data owners train and use separate NNs (e.g., to achieve better predictive performance on their local data). The NN scope dimension comprises three characteristics. In CDL systems with a *global* scope of NNs, all data owners train and use a single NN. In CDL systems with an *individual* scope of NNs, each data owner trains and uses a separate NN. Moreover, the scope of NNs can also be *split*: parts of NNs (e.g., layers at the start of NNs) are trained and used for inference by data owners individually, whereas other parts (e.g., layers at the end of NNs) are trained and used globally by all data owners.

Trusted Third Party. In addition to data owners and servers, some CDL systems require trusted third parties. Trusted third parties are used, for example, to distribute encryption keys to data owners and servers [43] or to reduce communication with data owners [65]. The trusted third party dimension comprises two characteristics: a trusted third party is either *not required* or *required* by CDL systems.

5 Archetypes of Collaborative Deep Learning Systems

We used our privacy-focused taxonomy of CDL systems to identify four clusters of CDL systems that use PPTs. Based on these clusters, we developed four CDL system archetypes that illustrate prevalent CDL system designs (see Table 3). In this section, we explain which kinds of use cases the CDL system archetypes are suitable for. Moreover, we describe common characteristics of CDL systems that implement each CDL system archetype and detail how CDL systems use PPTs to fulfill training data, inference data, and NN privacy goals.

5.1 Autonomy Archetype

CDL systems of the autonomy archetype are suitable for use cases in which data owners aim at maintaining a high degree of autonomy while collaboratively executing DL stages. Figure 2 shows the topology of an exemplary CDL system that implements the autonomy archetype.

Data owners maintain a high degree of autonomy for two reasons. First, data owners control how their training data is used by keeping it local. Based on local training data, data owners compute intermediate results (e.g., gradients or NN parameters) and exchange these intermediate results with each other. Second, data owners decide how NNs are updated. There is no central party that orchestrates the execution of DL stages. Data owners either keep individual (parts of) NNs and update their (parts of) NNs based on intermediate results from other data owners [58, 71], or they collectively decide how a global NN is updated (e.g., using a voting process). If data owners collectively decide how a global NN is updated, they interact in protocols that update a global NN and provide the updated global NN to all data owners as outputs [123, 138].

5.1.1 Common Characteristics. CDL systems that implement the autonomy archetype are either limited to the *training* stage [7, 58] or support *training and inference* stages [71, 120]. Such CDL systems are commonly designed for *few* data owners and do not use servers (i.e., the number of servers is *none*) [58, 120]. Hence, *data owners* are the main computing parties [58, 71]. Data owners compute intermediate results based on local training data and exchange these intermediate results

Table 3. Overview of the CDL System Archetypes and the Most Frequent Characteristics of CDL Systems in the Corresponding Clusters

Dimension	Autonomy Archetype	Delegation Archetype	Dyadic Archetype	Supervision Archetype
Privacy	Inference Data Privacy Goal	None (6)	Input Privacy (28)	Input Privacy (36)
	Training Data Privacy Goal	Input Privacy (7)	Input Privacy (26)	None (31)
	NN Architecture Privacy Goal	None (9)	From DOs (33)	From DOs (19)
	NN Parameter Privacy Goal	From DOs (7)	From DOs and Servers (34)	From DOs (30)
Processes	Data Owner Interactivity	Interactive (10)	Non-Interactive (34)	Interactive (22)
	DL Stages	Training (6)	Training and Inference (18)	Inference (31)
	Main Computing Parties	DOs (10)	Servers (36)	DOs and Servers (23)
Threats	Colluding Parties	None (7)	None (28)	None (39)
	Data Owner Corruption	Honest-but-Curious (9)	Honest-but-Curious (36)	Honest-but-Curious (35)
	Server Corruption	None (10)	Honest-but-Curious (23)	Honest-but-Curious (35)
Topology	Communication Paths	DO $\leftrightarrow$ DO (10)	DO $\leftrightarrow$ Server and Server $\leftrightarrow$ Server (36)	DO $\leftrightarrow$ Server (38)
	Data Partitioning	Horizontal (6)	Arbitrary (22)	Horizontal (38)
	Number of Data Owners	Few (9)	Many (20)	One (36)
	Number of Servers	None (10)	Multiple (36)	One (39)
	Scope of NNs	Global (6)	Global (36)	Global (31)
	Trusted Third Party	Not Required (10)	Not Required (30)	Not Required (35)
Number of CDL systems in cluster		10	36	39
				64

The numbers in parentheses indicate how many CDL systems in the corresponding cluster share the most frequent characteristics. Data owners are abbreviated as DOs in characteristics.

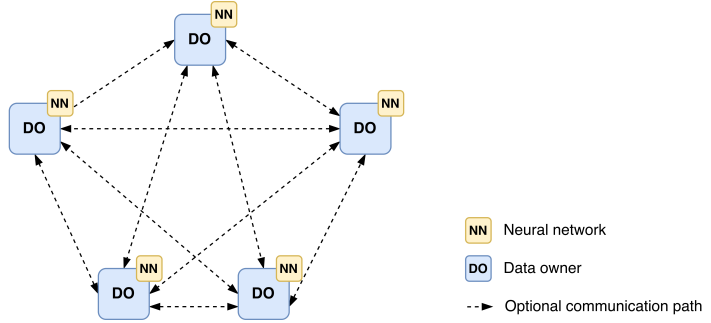


Fig. 2. A visualization of the autonomy archetype.

with each other using *data owner* $\leftrightarrow$ *data owner* communication paths [36, 120]. Communication with data owners is *interactive* because data owners repeatedly exchange intermediate results [71, 123]. Trusted third parties are typically *not required* [7, 120]. CDL systems of the autonomy archetype support varying data partitionings, including *arbitrary* [7], *horizontal* [120], and *vertical* [36] data partitionings. Moreover, such CDL systems have different scopes of NNs, including *global* scopes of NNs [138], *individual* scopes of NNs [58], and *split* scopes of NNs [36].

CDL systems of the autonomy archetype are commonly designed to protect parameters of (parts of) NNs *from data owners*, whereas NN architectures are typically known by all data owners (i.e., the NN architecture privacy goal is *none*) [36, 151]. Moreover, these CDL systems are designed to ensure *input privacy* [36, 71] or *input and output privacy* [58, 123] of training data. CDL systems that support the inference stage are also designed to ensure *input privacy* of inference data [71, 120]. Data owners are assumed to be *honest-but-curious* and not to collude (i.e., colluding parties are *none*) [58, 71].

5.1.2 Privacy-Preserving Techniques. In CDL systems of the autonomy archetype, data owners exchange intermediate results (e.g., gradients or NN parameters) to execute DL stages. Intermediate

Table 4. Exemplary CDL Systems That Implement the Autonomy Archetype, the PPTs They Use, and the Privacy Goals They are Designed to Fulfill

CDL System	PPTs				Privacy Goals			
	DP	HE	MPC	TEE	Inference Data	Training Data	NN Architecture	NN Parameters
Bansal et al. [7]	-	✓	✓	-	None	Input Privacy	None	From DOs
Biscotti [123]	✓	-	✓	-	None	Input Privacy and Output Privacy	None	None
BlindFL [36]	-	✓	✓	-	Input Privacy	Input Privacy	None	From DOs
Liu et al. [71] *	-	✓	-	-	Input Privacy	Input Privacy	None	From DOs
Liu et al. [71] *	-	-	✓	-	Input Privacy	Input Privacy	None	From DOs
POSEIDON [120]	-	✓	-	-	Input Privacy	Input Privacy	None	From DOs
ProxyFL [58]	✓	-	-	-	None	Input Privacy and Output Privacy	From DOs	From DOs
PVD-FL [151]	-	✓	-	-	None	Input Privacy	None	From DOs

Data owners are abbreviated as DOs in privacy goals. References denoted by an asterisk (*) describe multiple CDL systems and appear more than once.

results can reveal private information from training data, inference data, and NNs. To fulfill privacy goals, CDL systems of the autonomy archetype integrate DP, HE, and MPC (see Table 4).

Differential Privacy. CDL systems of the autonomy archetype integrate DP to ensure *input privacy* and *output privacy* of training data [58, 123]. In particular, data owners apply DP mechanisms to add calibrated noise to the intermediate results they transmit to other data owners. DP mechanisms mathematically bound the amount of information that can be learned from intermediate results or trained NNs [58, 123].

Data owners in ProxyFL, for example, simultaneously train two NNs in multiple training rounds: an individual NN that each data owner keeps local and a proxy NN that is exchanged with other data owners [58]. The individual NNs are trained with local training data and with knowledge from proxy NNs. The proxy NNs are also trained based on local training data of data owners, but with a training algorithm that ensures DP (e.g., DP-SGD [1]). At the end of each training round, data owners transmit proxy NNs to other data owners. Data owners aggregate proxy NNs received from other data owners. In subsequent training rounds, data owners use aggregated proxy NNs to incorporate knowledge from training data of other data owners into their individual NNs [58].

Homomorphic Encryption. CDL systems of the autonomy archetype use HE to ensure *input privacy* of training data and inference data [71, 120]. Moreover, CDL systems of this archetype use HE to protect NN parameters *from data owners* [120, 151]. To fulfill these privacy goals, CDL systems of this archetype use single-key or multi-key HE schemes. Single-key HE schemes are commonly sufficient in CDL systems with two data owners. In such systems, data owners only need to protect private information from one party, that is, the other data owner [36, 71]. Data owners hold individual (parts of) NNs [71]. In the training stage, data owner *A* computes intermediate results (e.g., outputs of hidden layers [71]) based on its (part of a) NN and local training data. Then, data owner *A* encrypts its intermediate results with an HE scheme and transmits the encrypted intermediate result to data owner *B*. Data owner *B* uses its local training data and (part of a) NN, as well as the encrypted intermediate results of data owner *A*, to compute further intermediate results (e.g., masked gradients [71]). Data owner *B* transmits the encrypted intermediate result to data owner *A*, who uses its private key to decrypt the intermediate result and update its (part of a) NN [71]. By mirroring this process, data owner *B* also improves its (part of a) NN [36, 71]. Both data owners train their (parts of) NNs without revealing training data or their (parts of) NNs to the other data owners. Computations required for the inference stage are executed with a similar process [36, 71].

CDL systems of the autonomy archetype with more than two data owners use multi-key HE schemes. With multi-key HE schemes, computations can be executed based on intermediate results encrypted by multiple data owners with different keys. For example, CDL systems use multi-key HE schemes to generate individual private keys and a shared public key for all data owners [120]. The parameters of a NN are encrypted with the shared public key. Each data owner executes computations with its local training data and the encrypted NN parameters. Then, each data owner transmits encrypted intermediate results to a selected data owner. The selected data owner aggregates the encrypted intermediate results of the other data owners to update the NN parameters. To decrypt NN parameters or predictions made with a NN, the private keys of all data owners are required. Even if all data owners except one collude, they cannot decrypt NN parameters or predictions made with a NN without the participation of the remaining data owner [120].

Secure Multi-Party Computation. CDL systems of the autonomy archetype use MPC to ensure *input privacy* of training data and inference data [36, 71] and to protect NN parameters *from data owners* [7, 71]. To ensure *input privacy* of training data, data owners interact in MPC protocols to aggregate intermediate results. For example, all data owners hold a NN and use local training data to compute intermediate results, such as gradients [123]. Then, data owners generate secret shares of their intermediate results (e.g., using Shamir’s secret sharing [123]) and distribute these secret shares to a selected set of data owners. The selected data owners aggregate the secret shares received, for example, by summing up gradients. Finally, the selected data owners reconstruct and broadcast aggregated intermediate results to all data owners. Data owners use aggregated intermediate results to update their local NNs [123].

To ensure *input privacy* of training data and inference data while also protecting NN parameters *from data owners*, data owners secret share (e.g., using additive secret sharing) NN parameters of a *global* NN [71] or of parts of a *split* NN (e.g., input layers [36]). In the training stage, data owners use training data and their secret shares of NN parameters to execute forward propagation and backward propagation for their (parts of a) NN. Similarly, in the inference stage, data owners use inference data and their secret shares of NN parameters to execute forward propagation. During forward and backward propagation, data owners exchange intermediate results (e.g., homomorphically encrypted outputs of hidden layers). At the end of the training stage, data owners hold updated secret shares of parameters of (parts of) a NN. At the end of the inference stage, data owners hold secret shares of predictions. Data owners transmit these secret shares to a data owner who reconstructs the prediction. No data owner learns training data (in the training stage) and inference data (in the inference stage) of other data owners, or parameter values of secret-shared (parts of) NNs [36, 71].

5.2 Delegation Archetype

CDL systems of the delegation archetype are suitable for use cases in which data owners aim at outsourcing computations of DL stages to a set of servers. Figure 3 shows the topology of an exemplary CDL system that implements the delegation archetype.

Data owners aim at minimizing the computational resources required locally and interactions with other parties. Therefore, they delegate computations required for DL stages to a set of servers. Data owners upload their training data instances or inference data instances to one of the servers [67] or distribute parts of training data instances or inference data instances to different servers [147]. To execute the computations required for training and inference stages, servers interact with each other. At the end of the training stage, servers keep trained NNs for the inference stage [84] or transmit trained NNs to data owners [75]. At the end of the inference stage, servers transmit predictions to data owners [27].

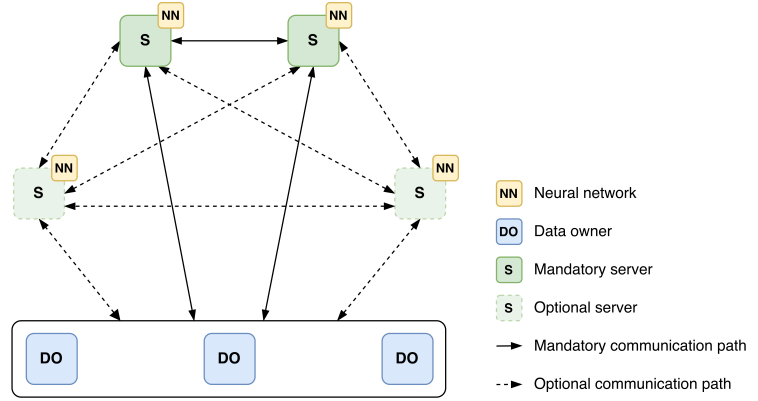


Fig. 3. A visualization of the delegation archetype.

Table 5. Exemplary CDL Systems That Implement the Delegation Archetype, the PPTs They Use, and the Privacy Goals They are Designed to Fulfill

CDL System	PPTs					Privacy Goals		
	DP	HE	MPC	TEE	Inference Data	Training Data	NN Architecture	NN Parameters
ABY ³ [84]	-	-	✓	-	Input Privacy	Input Privacy	From DOs	From DOs and Servers
Chen et al. [19]	-	✓	-	-	Input Privacy	None	From DOs and Servers	From DOs and Servers
FLASH [17]	-	-	✓	-	Input Privacy	Input Privacy	From DOs	From DOs and Servers
Li et al. [64]	-	✓	✓	-	Input Privacy	None	From DOs and Servers	From DOs and Servers
NPMML [67]	-	✓	-	-	None	Input Privacy	From DOs	From DOs
SecureML [85]	-	✓	✓	-	Input Privacy	Input Privacy	From DOs	From DOs and Servers
SecureNN [135]	-	-	✓	-	Input Privacy	Input Privacy	From DOs	From DOs and Servers
Zhang et al. [147]	-	✓	-	-	Input Privacy	Input Privacy	From DOs	From DOs and Servers

Data owners are abbreviated as DOs in privacy goals.

5.2.1 Common Characteristics. CDL systems of the delegation archetype typically support *training and inference* stages [61, 84]. Such CDL systems require *multiple* servers and support *many* (i.e., more than 100) data owners [61, 136]. Servers are the main computing parties [18, 60]. Communication with data owners is *non-interactive* [19, 62]. CDL systems of this archetype require *data owner* $\leftrightarrow$ *server* and *server* $\leftrightarrow$ *server* communication paths [4, 136]. The scope of NNs is *global* [70, 75], whereas data is partitioned *arbitrarily* [61] or *horizontally* [64] among data owners. Trusted third parties are *not required* by most CDL systems of this archetype [98, 136].

CDL systems of the delegation archetype are designed to protect NN architectures and NN parameters *from data owners* or *from data owners and servers* [19, 75]. Moreover, such CDL systems are also designed to ensure *input privacy* of inference data (for CDL systems that support the inference stage) [62, 84] and of training data (for CDL systems that support the training stage) [60, 97]. CDL systems of the delegation archetype commonly assume no collusion (i.e., colluding parties are *none*) [84, 147]. Such CDL systems assume *honest-but-curious* data owners, whereas servers are assumed to be *honest-but-curious* or *malicious* [27, 84].

5.2.2 Privacy-Preserving Techniques. In CDL systems that implement the delegation archetype, data owners upload training data instances or inference data instances to servers. Moreover, servers transmit trained NNs or predictions to data owners. To avoid leakage of private information from training data, inference data, and NNs, such CDL systems integrate HE and MPC (see Table 5).

Homomorphic Encryption. CDL systems of the delegation archetype integrate HE to ensure *input privacy* of training data and inference data [85, 147]. Moreover, such CDL systems also use HE to protect NN architectures and NN parameters *from data owners* [67] or *from data owners and servers* [19]. In the training stage, data owners encrypt training data with an HE scheme (e.g., Paillier [67]). Then, they either transmit entire encrypted training data instances to one server [67] or distribute parts of encrypted training data instances to distinct servers (e.g., features to one server and labels to another server [147]). Servers execute the main computations required for the training stage. During these computations, servers exchange intermediate results, such as masked output layer weights [67]. At the end of the training stage, one server holds an encrypted NN [67] or multiple servers hold different parts of NNs (e.g., hidden layers and output layers [147]). Servers transmit (their parts of) NNs to data owners or store them for the inference stage.

In the inference stage, servers either use NNs they stored at the end of the training stage [147] or pre-trained NNs, that is, NNs trained outside the CDL system [19]. In the latter case, NN owners encrypt NNs with an HE scheme and transmit the encrypted NNs to servers. Multi-key HE schemes (e.g., multi-key CKKS) enable data owners and NN owners to encrypt their respective inputs with different keys to fulfill privacy goals in case of collusion [19]. Data owners transmit inference data encrypted with an HE scheme (e.g., Paillier) to a server [19, 147]. Servers execute the computations required for the inference stage and exchange intermediate results, such as partially decrypted outputs of layers [147]. At the end of the inference stage, servers transmit encrypted predictions to data owners [19]. Data owners decrypt the predictions [19, 147].

Secure Multi-Party Computation. CDL systems of the delegation archetype use MPC to ensure *input privacy* of training data and inference data [61, 136], to protect NN architectures *from data owners* [84, 98], and to protect NN parameters *from data owners and servers* [84, 98]. In the training stage, data owners distribute secret shares of training data to servers, for example, using additive secret sharing [62, 84] or Boolean secret sharing [17, 98]. Then, servers train a NN by interacting in MPC protocols. After training, servers hold secret shares of NNs. They either retain the secret shares for the inference stage [84] or transmit them to data owners who reconstruct the NN [60]. In the inference stage, servers make predictions using secret-shared NNs. Servers either use a NN they retained from the training stage [136] or an external NN owner distributes secret shares of a NN to servers [70]. Data owners distribute secret shares of inference data to servers. The servers interact in MPC protocols to make predictions with the NN. The resulting predictions exist as secret shares distributed across servers. Servers transmit their secret shares to data owners who reconstruct predictions [84, 136].

In addition to secret sharing, CDL systems of the delegation archetype often use other MPC primitives for NN training and inference, including garbled circuits [4, 64] and oblivious transfer [4, 85]. To improve efficiency in terms of communication and computation cost, some CDL systems of the delegation archetype use different MPC primitives for different kinds of operations [4, 18, 85]. For example, arithmetic secret sharing is generally more efficient for linear operations, such as computing the weighted sum of inputs to neurons. Boolean secret sharing and garbled circuits are more efficient for non-linear operations, such as evaluating ReLU activation functions [84].

5.3 Dyadic Archetype

CDL systems of the dyadic archetype are suitable for use cases in which data owners aim at interacting with a single server in a one-to-one relationship to execute DL stages. Figure 4 shows the topology of an exemplary CDL system that implements the dyadic archetype.

Data owners either aim at obtaining a NN trained based on their training data (in the training stage) [91, 137], or predictions based on their inference data (in the inference stage) [29, 110]. A

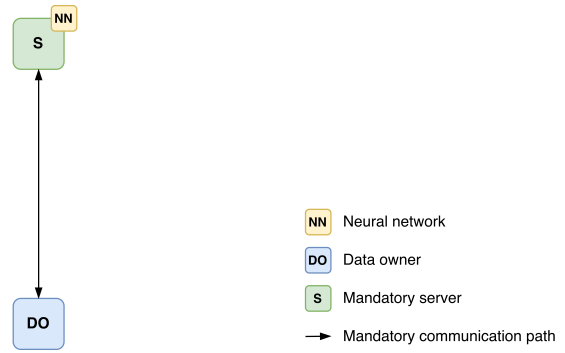


Fig. 4. A visualization of the dyadic archetype.

party operates a server to support data owners by training a NN with training data of the data owner, or by making predictions based on inference data of the data owner with a trained NN. To execute DL stages, data owners either transmit entire training data instances or entire inference data instances to servers [29, 118], or data owners interact with servers in protocols that do not require the transmission of entire training data instances or entire inference data instances to servers [82, 110].

5.3.1 Common Characteristics. Most CDL systems of the dyadic archetype are limited to the *inference* stage [56, 110]. Only two parties are involved in the execution of DL stages: *one* data owner and *one* server [108, 109]. Hence, only *data owner* $\leftrightarrow$ *server* communication paths are required [4, 69]. The main computing parties are either *data owners and servers* [10, 53] or only *servers* [74, 131]. If data owners and servers are the main computing parties, communication with data owners is *interactive* [10, 53]. If servers are the main computing parties, communication with data owners is *non-interactive* [74, 131]. Data owners hold entire data instances. Hence, data partitioning between data owners is *horizontal* [56, 82]. The scope of NNs can be *global* [69], *individual* [49], or *split* [137]. Trusted third parties are commonly *not required* [53, 106].

CDL systems of the dyadic archetype are designed to ensure *input privacy* of inference data [56, 110]. Because most CDL systems of this archetype are limited to the inference stage, the training data privacy goal is typically *none* [48, 108]. CDL systems of this archetype are either designed to protect NN architectures and NN parameters *from data owners* (i.e., if NN owners operate servers) [14, 94] or *from data owners and servers* (i.e., if NN owners upload NNs to servers of third parties) [49, 131]. CDL systems of the dyadic archetype assume *honest-but-curious* or *malicious* data owners and servers [29, 109]. Moreover, such CDL systems assume that data owners and servers do not collude (i.e., colluding parties are *none*) [4, 109].

5.3.2 Privacy-Preserving Techniques. In CDL systems of the dyadic archetype, data owners either upload entire training data instances and entire inference data instances to servers, or they interact with servers in protocols that require the exchange of intermediate results. To avoid leakage of private information from training data, inference data, and NNs, CDL systems of the dyadic archetype integrate HE, MPC, and TEEs (see Table 6).

Homomorphic Encryption. CDL systems of the dyadic archetype that integrate HE ensure *input privacy* of inference data and, if they support the training stage, *input privacy* of training data. Moreover, they protect NN architectures and NN parameters *from data owners* [29, 94] or *from data owners and servers* [6, 49]. Servers are the main computing parties [6, 74], and communication with data owners is *non-interactive* [14, 73]. In CDL systems of the dyadic archetype that support

Table 6. Exemplary CDL Systems that Implement the Dyadic Archetype, the PPTs They Use, and the Privacy Goals They are Designed to Fulfill

CDL System	PPTs				Privacy Goals			
	DP	HE	MPC	TEE	Inference Data	Training Data	NN Architecture	NN Parameters
Chameleon [110]	-	-	✓	-	Input Privacy	None	None	From DOs
Cheetah [53]	-	✓	✓	-	Input Privacy	None	None	From DOs
Chex-Mix [87]	-	✓	-	✓	Input Privacy	None	From DOs	From DOs and Servers
CryptoNets [29]	-	✓	-	-	Input Privacy	None	From DOs	From DOs
DELPHI [82]	-	✓	✓	-	Input Privacy	None	None	From DOs
FALCON [66]	-	✓	✓	-	Input Privacy	None	None	From DOs
Hesamifard et al. [49]	-	✓	-	-	Input Privacy	Input Privacy	From DOs and Servers	From DOs and Servers
Slalom [131]	-	-	-	✓	Input Privacy	None	From DOs	From DOs

Data owners are abbreviated as DOs in privacy goals.

the training stage, data owners encrypt their training data with an HE scheme (e.g., CKKS [6]) and upload the encrypted training data to a server [6, 49]. The server uses encrypted training data to train a NN. After training, servers either transmit NNs to data owners who decrypt them for inference [6] or servers keep NNs for subsequent use in the inference stage [49].

In the inference stage, a server uses a NN to make predictions. The server either uses a NN kept from the training stage [6, 49], or a NN owner uploads a pre-trained NN specifically for the inference stage [29, 108]. If a CDL system only protects NN architectures and NN parameters *from data owners*, servers hold NNs in plain form, that is, without encryption [29, 73]. Conversely, if a CDL system protects NN architectures and NN parameters *from data owners and servers*, NNs are encrypted with an HE scheme [6, 49]. Data owners encrypt inference data with an HE scheme (e.g., BFV [108] or TFHE [118]) and upload encrypted inference data to servers [29, 48]. Servers use the NN to make predictions and transmit the encrypted prediction to the data owner. Finally, the data owner uses its private key to decrypt the prediction [29, 48].

Secure Multi-Party Computation. CDL systems of the dyadic archetype that use MPC are typically limited to the *inference* stage [110, 114]. *Data owners and servers* are the main computing parties [4, 53]. Such CDL systems are commonly designed to ensure *input privacy* of inference data [56, 82] and to protect NN parameters *from data owners* [53, 69]. Some CDL systems of this archetype are also designed to protect NN architectures *from data owners* [4, 115]. To execute the inference stage, data owners and servers interact in MPC protocols. CDL systems of the dyadic archetype that use MPC are *interactive* or *non-interactive*. Garbled circuits can be used to create non-interactive CDL systems [109, 114]. In such CDL systems, data owners typically take the role of the garbler and create a garbled circuit, which the server evaluates. CDL systems that use secret sharing require additional communication rounds [69, 110]. In such CDL systems, inference data and NN parameters are secret-shared between data owners and servers. CDL systems of the dyadic archetype often convert between different MPC primitives to improve efficiency in terms of communication and computation cost. For example, they use arithmetic secret sharing for linear operations (e.g., computing the weighted sum of inputs to neurons), and Boolean secret sharing or garbled circuits for non-linear operations (e.g., evaluating ReLU activation functions) [4, 110].

Trusted Execution Environments. CDL systems of the dyadic archetype use TEEs to ensure *input privacy* of training data and inference data and to protect NN architectures and NN parameters *from data owners and servers*. Data owners generate a secret encryption key and encrypt their training data or inference data. They establish a secure channel with the TEE and use remote attestation to verify the integrity of code inside the TEE [93, 131]. Then, they upload the encrypted training data or inference data, along with their secret key, to the TEE via the secure channel. Inside the

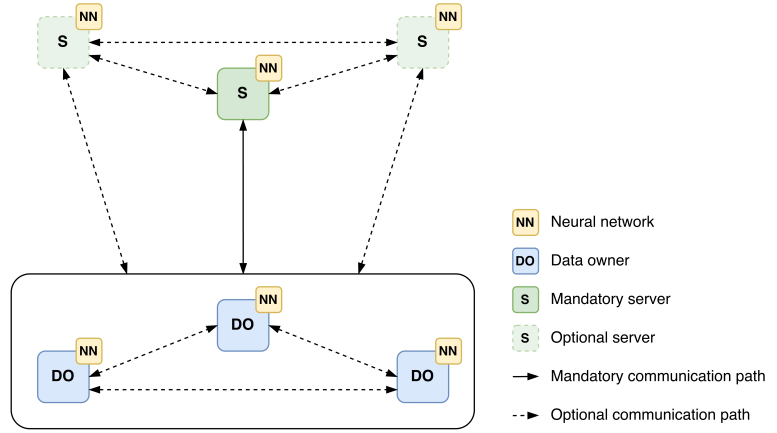


Fig. 5. A visualization of the supervision archetype.

TEE, training data or inference data is decrypted. In the training stage, servers train a NN. The server either encrypts and transmits the trained NN to data owners who decrypt and use it for inference locally [93], or it stores the trained NN for the inference stage. In the inference stage, servers use a trained NN to make predictions. If the NN was not trained with the CDL system, a NN owner encrypts a pre-trained NN and uploads it to the TEE via a secure channel. Like data owners, external NN owners use remote attestation to verify the integrity of the code inside the TEE beforehand [131]. At the end of the inference stage, predictions are encrypted with the secret key of data owners and transmitted to data owners. Data owners use their secret key to decrypt predictions [131].

5.4 Supervision Archetype

CDL systems of the supervision archetype are suitable for use cases in which data owners aim at participating in the training of NNs under the supervision of central servers. Figure 5 shows the topology of an exemplary CDL system that implements the supervision archetype.

Data owners are willing to provide training data and computational resources to train NNs, but cannot or do not want to transmit entire training data instances to other parties. One or multiple central servers orchestrate the training stage, for example, by implementing the concept of federated learning [79]. Central servers broadcast a global NN to data owners. Data owners compute intermediate results (e.g., gradients or NN parameters) based on their local training data. Then, they transmit these intermediate results back to central servers. Central servers aggregate the intermediate results from data owners and update the global NN. This process is repeated for several training rounds [57]. After training, data owners use NNs for inference locally.

5.4.1 Common Characteristics. CDL systems of the supervision archetype are limited to the training stage [43, 133]. Data owners collaboratively train a *global* NN that they use for inference without further interaction with other parties [3, 34]. CDL systems of this archetype support *few* (i.e., up to 100) [119, 133] or *many* (i.e., more than 100) data owners [11, 42]. Such CDL systems use *one* or *multiple* servers that orchestrate the training stage [79, 107]. Training data is typically partitioned *horizontally* between data owners [79, 119]. Communication with data owners is *interactive* [107, 148]. *Data owners* are the main computing parties: they compute intermediate results based on local training data and exchange their intermediate results with a server (i.e.,

Table 7. Exemplary CDL Systems That Implement the Supervision Archetype, the PPTs They Use, and the Privacy Goals They are Designed to Fulfill

CDL System	PPTs				Privacy Goals			
	DP	HE	MPC	TEE	Inference Data	Training Data	NN Architecture	NN Parameters
Bonawitz et al. [11]	-	-	✓	-	None	Input Privacy	None	None
Dong et al. [28]	-	-	✓	-	None	Input Privacy	None	From Servers
FedOpt [5]	✓	✓	-	-	None	Input Privacy and Output Privacy	None	None
Hercules [141]	-	✓	-	-	None	Input Privacy	From DOs	From DOs
LDP-Fed [133]	✓	-	-	-	None	Input Privacy and Output Privacy	None	None
Mo et al. [83]	-	-	-	✓	None	Input Privacy	None	None
Xu et al. [143]	-	✓	✓	-	None	Input Privacy	From Servers	From Servers
Zheng et al. [154]	-	-	-	✓	None	Input Privacy	None	None

Data owners are abbreviated as DOs in privacy goals.

using *data owner* $\leftrightarrow$ *server* communication paths [119, 133]), with other data owners and a server (i.e., using *data owner* $\leftrightarrow$ *data owner* and *data owner* $\leftrightarrow$ *server* communication paths) [128, 148], or with multiple servers (i.e., using *data owner* $\leftrightarrow$ *server* and *server* $\leftrightarrow$ *server* communication paths) [107, 143]. Trusted third parties are typically *not required* [34, 79].

CDL systems of the supervision archetype are designed to ensure *input privacy* [11], *output privacy* [139], or *input privacy and output privacy* [152] of training data. Because CDL systems of the supervision archetype are limited to the training stage, the inference data privacy goal is *none* [107, 119]. NN architectures and NN parameters are commonly known by all parties (i.e., the NN architecture privacy goal and the NN parameter privacy goal are *none*) [79, 133] or protected *from servers* [143, 148]. Data owners and servers are assumed to be *honest-but-curious* [11, 153] or *malicious* [42, 125]. Moreover, CDL systems of this archetype vary regarding the parties they assume to collude. Colluding parties can be *none* [52], *data owners* [153], or *data owners and servers* [11].

5.4.2 Privacy-Preserving Techniques. In CDL systems of the supervision archetype, data owners transmit intermediate results to servers. Intermediate results can reveal private information from training data and NNs. To fulfill privacy goals, CDL systems of the supervision archetype integrate DP, HE, MPC, and TEEs (see Table 7).

Differential Privacy. CDL systems of the supervision archetype use DP to ensure *output privacy* or *input and output privacy* of training data. DP mechanisms can be applied by servers [139] or by data owners [133] to add calibrated noise to intermediate results or NN parameters. Servers use DP mechanisms to ensure *output privacy* of training data. Typically, servers apply DP mechanisms when aggregating intermediate results to ensure that updated global NNs do not reveal private information from training data [139].

In contrast, data owners use DP mechanisms to ensure *input and output privacy* of training data [72, 133]. Data owners commonly apply DP mechanisms during the computation of intermediate results. For example, they use the DP-SGD algorithm when training a NN with their local training data [72]. Data owners use DP mechanisms to ensure that neither intermediate results nor updated global NNs reveal private information from training data.

Homomorphic Encryption. CDL systems of the supervision archetype use HE to ensure *input privacy* of training data, and to protect NN architectures and NN parameters *from data owners* or *from servers*. To ensure *input privacy* of training data and to protect NN architectures and NN parameters *from servers*, data owners encrypt intermediate results (e.g., gradients [43, 99] or NN parameters [76, 96]) with an HE scheme. Then, data owners transmit encrypted intermediate results to a central server. The central server aggregates the received intermediate results (e.g., by summing gradients [43, 99] or NN parameters [76]). Then, the server transmits the encrypted, aggregated

intermediate results to data owners. Data owners decrypt aggregated intermediate results and update their local NNs. To protect against collusion between *data owners and servers*, data owners use multi-key HE schemes [76, 141], HE schemes with support for partial decryption [77, 96], or a second layer of encryption [43]. These approaches ensure that the central server cannot access intermediate results of individual data owners even if a subset of data owners colludes with the central server [43, 77, 96].

To ensure *input privacy* of training data and to protect NN architectures and NN parameters *from data owners*, servers transmit an encrypted global NN to data owners. Data owners use local training data to compute intermediate results (e.g., NN parameters [141]) based on the encrypted NN. Then, data owners transmit encrypted intermediate results to the server. The server aggregates the intermediate results of data owners and updates the global NN [141]. This process is repeated for several training rounds. At the end of the training stage, data owners use key switching to convert the encrypted NN to a NN encrypted with a public key of the server. Finally, the server decrypts the resulting NN [141].

Secure Multi-Party Computation. MPC is integrated into CDL systems of the supervision archetype to ensure *input privacy* of training data and to protect NN parameters *from servers*. CDL systems of this archetype that integrate MPC require *one* or *multiple* servers.

CDL systems of the supervision archetype with *one* server commonly use masking techniques and secret sharing (e.g., Shamir's secret sharing or additive secret sharing) [11, 126]. The server distributes a global NN to data owners. Data owners compute intermediate results (e.g., gradients [42] or NN parameters [11]) based on local training data. Then, data owners mask the intermediate results and transmit them to the central server [11, 42]. A common masking technique is to generate pairwise masks that cancel out during aggregation if all data owners, or a subset of a predefined size, submit their masked intermediate results [11, 42, 143]. To enable aggregation if data owners drop out before the aggregation is completed, data owners generate n secret shares of their masks with an (n, t) -threshold secret sharing scheme. Data owners distribute these secret shares to n other data owners. If at least t data owners survive, they can reconstruct the masks of dropped-out data owners. If fewer than t data owners survive, aggregation aborts without revealing the masks of surviving data owners [11, 42, 143]. The central server aggregates the intermediate results received [11, 42, 143] and unmask the aggregated intermediate results [126, 128]. Then, the central server distributes the aggregated intermediate results to the data owners, who update their NNs.

CDL systems with *multiple* servers use secret sharing and additional MPC primitives to ensure *input privacy* of training data [44, 107] and to protect NN parameters *from servers* [28, 143]. Servers distribute a global NN to data owners. Data owners compute intermediate results based on local training data. Then, data owners generate secret shares of intermediate results and distribute these secret shares to servers [28, 34, 107], or data owners encrypt intermediate results with an HE scheme and upload them to one of the servers [44, 142]. The servers interact in MPC protocols (e.g., garbled circuits [142] and oblivious transfer [44, 107]) to aggregate the intermediate results of data owners. The aggregated intermediate results are revealed to servers (in CDL systems with a NN parameter privacy goal of *none*) [34] or data owners (in CDL systems with a NN parameter privacy goal of *from servers*) [28].

Trusted Execution Environments. CDL systems of the supervision archetype incorporate TEEs to ensure *input privacy* of training data. Such CDL systems use *one* server with a TEE [83, 154]. Data owners establish a secure channel to the TEE of the server. The server transmits a global NN to data owners. Data owners compute intermediate results (e.g., gradients or NN parameters) based on local training data. Then, data owners upload intermediate results to the TEE of the server. The server collects and aggregates the intermediate results of data owners and updates the global NN.

By executing these computations in a TEE, intermediate results of data owners are not revealed to the server [83, 154].

6 Discussion

6.1 Principal Findings

In this article, we systematically developed a privacy-focused CDL system taxonomy, comprising 4 categories (i.e., privacy, processes, threats, and topology) with 16 dimensions and 49 characteristics. Using a taxonomy-based clustering approach, we developed four CDL system archetypes: the *autonomy archetype*, the *delegation archetype*, the *dyadic archetype*, and the *supervision archetype*. For each CDL system archetype, we described how PPTs (i.e., DP, HE, MPC, and TEEs) can be integrated to fulfill privacy goals for training data, inference data, and NNs. The privacy-focused CDL system taxonomy and the CDL system archetypes illustrate the diversity of CDL system designs and their use of PPTs. No single CDL system can be applied to every use case.

In the analyzed studies, we noticed a shift in research focus over time. Earlier studies predominantly focused on centralized CDL systems in which entire training data instances and entire inference data instances are transmitted to central servers (i.e., CDL systems of the delegation archetype and the dyadic archetype). Recently, studies increasingly focus on distributed CDL systems in which data owners keep their training data local (i.e., CDL systems of the autonomy archetype and the supervision archetype). A reason for this shift in research focus could be, for example, that the amount of training data used to train contemporary NN models continuously increases. Distributed CDL systems allow using computational resources of data owners for training and inference stages. Therefore, distributed CDL systems can scale to process large amounts of training data without proportionally increasing the computational resources of servers. Despite this shift in research focus, we expect centralized CDL systems to remain relevant because they are more suitable for use cases where data owners aim at outsourcing computations to servers.

We also recognized a prevalence of CDL systems that assume honest-but-curious corruptions. Out of the 149 analyzed CDL systems, 117 assume honest-but-curious data owners and 107 assume honest-but-curious servers. The honest-but-curious corruption model is useful if parties are assumed to follow protocols correctly but may attempt to learn private information from data they receive. CDL systems with this corruption model seem suitable for use cases where all parties have incentives to follow protocols correctly. For example, studies often assume that servers are operated by cloud service providers who will not violate protocols because of risks to their reputation [97]. For use cases in which assumptions like these are not feasible, CDL systems with malicious corruption models are required. Yet, only 38 CDL systems in the analyzed literature assumed a malicious corruption model for data owners or servers.

6.2 Contributions to Research and Practice

Our study has three principal contributions to research and practice. First, our systematically developed, privacy-focused CDL system taxonomy facilitates a thorough understanding of CDL systems that integrate PPTs. By synthesizing knowledge and contributions from diverse research streams (e.g., DL, security, and statistics), our CDL system taxonomy enables researchers and practitioners to more consistently classify, compare, and describe CDL systems that integrate PPTs.

Second, the four CDL system archetypes we derived based on our privacy-focused CDL system taxonomy (i.e., the *autonomy archetype*, the *delegation archetype*, the *dyadic archetype*, and the *supervision archetype*) reveal insights into prevalent CDL system designs. By explaining which kinds of use cases CDL systems of each archetype are suitable for, we support practitioners in selecting CDL systems for use cases. By describing common characteristics of CDL systems of each archetype, we highlight similarities and differences between prevalent CDL system designs.

Third, our analysis of how CDL systems implement the four CDL system archetypes using PPTs provides a foundation for understanding how privacy goals can be fulfilled using different PPTs. This understanding is useful to support researchers and practitioners in comparing CDL systems that integrate different PPTs. Moreover, it aids researchers and practitioners in extending CDL systems with additional PPTs and in transferring PPTs to CDL systems of other archetypes.

6.3 Limitations

Our study is subject to three main limitations. First, although we analyzed a large set of CDL systems, we did not analyze all papers that describe CDL systems. Doing so is infeasible due to the vast amount of literature on CDL systems. Instead, we opted for a snowballing approach to identify a set of relevant papers that describe CDL systems. We used a start set of highly cited papers and applied backward and forward searches to identify additional papers. In total, we analyzed 122 papers describing 149 CDL systems. We think that our approach yielded a representative sample of CDL systems from literature. Considering that we did not need to change our taxonomy in the last iteration of taxonomy development, we are confident that we developed a robust taxonomy that covers a broad range of CDL system designs based on different PPTs.

Second, the taxonomy development method by Nickerson et al. [90] is inherently subjective. It requires applying subjective ending conditions (i.e., that the developed taxonomy is comprehensive, concise, explanatory, extendable, and robust). Even when applying the same method with the same meta-characteristic to the same domain, different researchers will likely develop different taxonomies [90]. Also, interpreting the meaningfulness of clustering solutions for different numbers of clusters is subjective. To reduce subjectivity, we discussed the categories, dimensions, and characteristics of our taxonomy within the author team in each iteration. At the end of each iteration, we validated the current version of our taxonomy with the entire set of CDL systems from prior iterations. Moreover, we discussed the meaningfulness of clustering solutions for different numbers of clusters within the author team.

Third, our privacy-focused CDL system taxonomy does not explain every nuance of CDL systems that integrate PPTs. CDL systems that integrate different PPTs could be classified and described in even more detail with dedicated taxonomies for each PPT. The identified dimensions and characteristics could be dissected further to describe CDL systems more extensively. However, as Nickerson et al. [90] argue, taxonomies should not describe every possible detail of objects but provide useful explanations of the nature of objects. Because our privacy-focused CDL system taxonomy proved useful for developing and describing four CDL system archetypes, we believe it fulfills this goal. Moreover, our taxonomy can be extended to describe specific types of CDL systems in more detail or to accommodate future developments in research and practice.

6.4 Future Research

Based on our analysis of literature on CDL systems, we identified three promising directions for future research, which we describe in the following.

Develop Benchmark Frameworks for CDL Systems. Our CDL system taxonomy provides a starting point for selecting CDL systems for use cases. In particular, researchers and practitioners may use our CDL system taxonomy to qualitatively compare CDL systems that use PPTs. However, selecting CDL systems for use cases goes beyond qualitative comparisons. After excluding unsuitable CDL systems based on qualitative criteria, practitioners may still be left with several potentially suitable CDL systems. To ultimately decide on a CDL system, practitioners may need additional support, for example, with benchmark frameworks. Benchmark frameworks aid in making CDL systems comparable by providing a standardized way to measure quantitative metrics of CDL

systems. Benchmark frameworks already exist for individual PPTs, such as HE [41] or MPC [59]. Yet, comparing CDL systems based on different PPTs is more difficult. Developing benchmark frameworks that support quantitative comparison of CDL systems based on different PPTs thus seems to be a promising direction for future research.

Extend CDL Systems to Complete DL Lifecycles. CDL systems are commonly limited to the training and inference stages of DL. While these are the two principal stages of DL, complete DL lifecycles comprise additional activities. For example, before being used to train NNs, data must undergo pre-processing activities, including data cleaning, data labeling, and feature engineering [121]. Moreover, suitable NN architectures and hyperparameters for DL tasks are commonly unknown. Activities like neural architecture search [33] and hyperparameter optimization [144] are conducted to identify NN architectures and hyperparameters suitable for DL tasks. Research on how these activities can be executed in CDL systems is scarce [155]. Yet, these activities are crucial to develop NNs with adequate predictive performance [33, 144]. Literature on CDL systems often seems to implicitly assume that NN architectures and training data are readily available for the training of NNs. Future research may thus benefit from extending CDL systems to cover complete DL lifecycles.

Improve Explainability of NNs in CDL Systems. A key issue with NNs is their lack of human interpretability. While some machine learning models (e.g., linear regression models) tend to be easily understandable, NNs are more difficult to interpret [8, 130]. Explainability methods aim at making NNs more interpretable by providing insights into how NNs make predictions. However, explainability methods can be at odds with privacy goals in CDL systems because they reveal additional information. This limits the applicability of explainability methods. For instance, example-based explanations present training data instances that are similar to inference data instances to explain NN outputs [134]. By disclosing training data instances, this explainability method conflicts with training data privacy goals of CDL systems. Research on explainability methods that do not compromise privacy goals may thus be worthwhile.

7 Conclusion

CDL systems are a promising solution for data owners who lack the resources to train NNs with adequate predictive performance on their own. In this study, we systematically developed a privacy-focused CDL system taxonomy comprising 4 categories with 16 dimensions and 49 characteristics. Based on the taxonomy, we developed four CDL system archetypes and analyzed how prevalent CDL system designs integrate PPTs to implement these archetypes. Our CDL system taxonomy and archetypes synthesize a fragmented body of research and provide researchers and practitioners with a foundation for consistently describing, comparing, and selecting CDL systems for use cases.

References

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Vienna, Austria. DOI: <https://doi.org/10.1145/2976749.2978318>
- [2] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *Comput. Surveys* 51, 4 (2018), 1–35. DOI: <https://doi.org/10.1145/3214303>
- [3] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The Skellam mechanism for differentially private federated learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*.
- [4] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J. Kusner, and Adrià Gascón. 2019. QUOTIENT: Two-party secure neural network training and prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, London, United Kingdom. DOI: <https://doi.org/10.1145/3319535.3339819>
- [5] Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. 2020. FedOpt: Towards communication efficiency and privacy preservation in federated learning. *Applied Sciences* 10, 8 (2020). DOI: <https://doi.org/10.3390/app10082864>

- [6] Ahmad Al Badawi, Louie Hoang, Chan Fook Mun, Kim Laine, and Khin Mi Mi Aung. 2020. PrivFT: Private and fast text classification with homomorphic encryption. *IEEE Access* 8 (2020), 226544–226556. DOI : <https://doi.org/10.1109/ACCESS.2020.3045465>
- [7] Ankur Bansal, Tingting Chen, and Sheng Zhong. 2011. Privacy preserving back-propagation neural network learning over arbitrarily partitioned data. *Neural Computing and Applications* 20, 1 (2011), 143–150. DOI : <https://doi.org/10.1007/s00521-010-0346-z>
- [8] Alejandro Barredo Arrieta, Natalia Diaz-Rodríguez, Javier Del Ser, Adrien Bannetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, et al. 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115. DOI : <https://doi.org/10.1016/j.inffus.2019.12.012>
- [9] Donald Beaver. 1992. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology – CRYPTO ’91*. Springer, Berlin, Heidelberg, 420–432. DOI : https://doi.org/10.1007/3-540-46766-1_34
- [10] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. 2020. MP2ML: A mixed-protocol machine learning framework for private inference. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ACM, Virtual Event, Ireland, 1–10. DOI : <https://doi.org/10.1145/3407023.3407045>
- [11] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Dallas, TX. DOI : <https://doi.org/10.1145/3133956.3133982>
- [12] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, and Joe Kilian (Eds.), Vol. 3378. Springer, Berlin, Heidelberg, 325–341. DOI : https://doi.org/10.1007/978-3-540-30576-7_18
- [13] Shyam Boriah, Varun Chandola, and Vipin Kumar. 2008. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 243–254. DOI : <https://doi.org/10.1137/1.9781611972788.22>
- [14] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. 2018. Fast homomorphic evaluation of deep discretized neural networks. In *Advances in Cryptology – CRYPTO 2018*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10993. Springer International Publishing, Cham, 483–512. DOI : https://doi.org/10.1007/978-3-319-96878-0_17
- [15] Gilles Brassard, Claude Crepeau, and Jean-Marc Robert. 1986. All-or-nothing disclosure of secrets. In *Advances in Cryptology – CRYPTO’ 86*, Andrew M. Odlyzko (Ed.), Vol. 263. Springer, Berlin, 234–238. DOI : https://doi.org/10.1007/3-540-47721-7_17
- [16] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography*, Martin Hirt and Adam Smith (Eds.), Vol. 9985. Springer, Berlin, 635–658. DOI : https://doi.org/10.1007/978-3-662-53641-4_24
- [17] Megha Byali, Harsh Chaudhari, Arpita Patra, and Ajith Suresh. 2020. FLASH: Fast and robust framework for privacy-preserving machine learning. *Proceedings on Privacy Enhancing Technologies* 2020, 2 (2020), 459–480. DOI : <https://doi.org/10.2478/popets-2020-0036>
- [18] Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. 2020. Trident: Efficient 4PC framework for privacy preserving machine learning. In *Proceedings 2020 Network and Distributed System Security Symposium*. San Diego, CA, USA. DOI : <https://doi.org/10.14722/ndss.2020.23005>
- [19] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. 2019. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural Network Inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, London, United Kingdom. DOI : <https://doi.org/10.1145/3319535.3363207>
- [20] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*. Tsuyoshi Takagi and Thomas Peyrin (Eds.), Vol. 10624. Springer International Publishing, Cham, 409–437. DOI : https://doi.org/10.1007/978-3-319-70694-8_15
- [21] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology* 33, 1 (2020), 34–91. DOI : <https://doi.org/10.1007/s00145-019-09319-x>
- [22] Christopher A. Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. 2021. CaPC Learning: Confidential and private collaborative learning. In *9th International Conference on Learning Representations*.
- [23] Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139. PMLR.

- [24] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. 1985. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual Symposium on Foundations of Computer Science*. Portland, OR, USA, 383–395. DOI : <https://doi.org/10.1109/SFCS.1985.64>
- [25] Michael Clear and Ciarán McGoldrick. 2015. Multi-identity and multi-key leveled FHE from learning with errors. In *Advances in Cryptology – CRYPTO 2015*, Matthew Robshaw and Rosario Gennaro (Eds.), Vol. 9216. Springer, Berlin, Heidelberg, 630–656. DOI : https://doi.org/10.1007/978-3-662-48000-7_31
- [26] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. Retrieved from <https://eprint.iacr.org/2016/086>
- [27] Anders Dalskov, Daniel Escudero, and Marcel Keller. 2020. Secure evaluation of quantized neural networks. *Proceedings on Privacy Enhancing Technologies* 2020, 4 (2020), 355–375. DOI : <https://doi.org/10.2478/popets-2020-0077>
- [28] Caiqin Dong, Jian Weng, Ming Li, Jia-Nan Liu, Zhiquan Liu, Yudan Cheng, and Shui Yu. 2023. Privacy-preserving and Byzantine-robust federated learning. *IEEE Transactions on Dependable and Secure Computing* 21, 2 (2023), 889–904. DOI : <https://doi.org/10.1109/TDSC.2023.3264697>
- [29] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48. PMLR.
- [30] Cynthia Dwork. 2006. Differential privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer, Berlin, 1–12. DOI : https://doi.org/10.1007/11787006_1
- [31] Cynthia Dwork, Krishnamurthy Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology – EUROCRYPT 2006*, Serge Vaudenay (Ed.). Springer, Berlin. DOI : https://doi.org/10.1007/11761679_29
- [32] Cynthia Dwork and Guy N. Rothblum. 2016. Concentrated Differential Privacy. DOI : <https://doi.org/10.48550/arXiv.1603.01887>
- [33] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21.
- [34] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Mollering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. 2021. SAFElearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops*. IEEE, San Francisco, CA, USA, 56–62. DOI : <https://doi.org/10.1109/SPW53761.2021.00017>
- [35] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, Denver, CO, USA. DOI : <https://doi.org/10.1145/2810103.2813677>
- [36] Fangcheng Fu, Huanran Xue, Yong Cheng, Yangyu Tao, and Bin Cui. 2022. BlindFL: Vertical federated machine learning without peeking into your data. In *Proceedings of the 2022 International Conference on Management of Data*. ACM, Philadelphia, PA, USA, 1316–1330. DOI : <https://doi.org/10.1145/3514221.3526127>
- [37] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Toronto, Canada. DOI : <https://doi.org/10.1145/3243734.3243834>
- [38] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. ACM, Bethesda, MD, 169–178. DOI : <https://doi.org/10.1145/1536414.1536440>
- [39] O. Goldreich, S. Micali, and A. Wigderson. 1987. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Conference on Theory of Computing - STOC '87*. ACM, New York, NY, 218–229. DOI : <https://doi.org/10.1145/28395.28420>
- [40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA, USA. Retrieved from <http://www.deeplearningbook.org>
- [41] Charles Gouert, Dimitris Mouris, and Nektarios Tsoutsos. 2023. SoK: New insights into fully homomorphic encryption libraries via standardized benchmarks. *Proceedings on Privacy Enhancing Technologies* 2023, 3 (2023), 154–172. DOI : <https://doi.org/10.56553/popets-2023-0075>
- [42] Changhee Hahn, Hodong Kim, Minjae Kim, and Junbeom Hur. 2023. VerSA: Verifiable secure aggregation for cross-device federated learning. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 36–52. DOI : <https://doi.org/10.1109/TDSC.2021.3126323>
- [43] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. 2020. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics* 16, 10 (2020), 6532–6542. DOI : <https://doi.org/10.1109/TII.2019.2945367>
- [44] Meng Hao, Hongwei Li, Guowen Xu, Hanxiao Chen, and Tianwei Zhang. 2021. Efficient, private and robust federated learning. In *Proceedings of the 37th Annual Computer Security Applications Conference*. ACM, Virtual Event, USA. DOI : <https://doi.org/10.1145/3485832.3488014>

- [45] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2019. Federated Learning for Mobile Keyboard Prediction. DOI : <https://doi.org/10.48550/arXiv.1811.03604>
- [46] Zecheng He, Tianwei Zhang, and Ruby B. Lee. 2019. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*. ACM, San Juan, PR, USA, 148–162. DOI : <https://doi.org/10.1145/3359789.3359824>
- [47] István Hegedűs, Gábor Danner, and Márk Jelasity. 2019. Gossip learning as a decentralized alternative to federated learning. In *Distributed Applications and Interoperable Systems*. José Pereira and Laura Ricci (Eds.). Springer International Publishing, Cham, 74–90. DOI : https://doi.org/10.1007/978-3-030-22496-7_5
- [48] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2019. Deep neural networks classification over encrypted data. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*. ACM, Richardson, TX, USA, 97–108. DOI : <https://doi.org/10.1145/3292006.3300044>
- [49] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N. Wright. 2018. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 123–142. DOI : <https://doi.org/10.1515/popets-2018-0024>
- [50] Joel Hestness, Newsha Ardalani, and Gregory Diamos. 2019. Beyond human-level accuracy: Computational challenges in deep learning. In *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*. ACM, Washington, D.C., USA, 1–14. DOI : <https://doi.org/10.1145/3293883.3295710>
- [51] Maximilian Heumann, Tobias Kraschewski, Oliver Werth, and Michael H. Breitner. 2024. Reassessing taxonomy-based data clustering: Unveiling insights and guidelines for application. *Decision Support Systems* 187 (2024), 114344. DOI : <https://doi.org/10.1016/j.dss.2024.114344>
- [52] Jiahui Hu, Zhibo Wang, Yongsheng Shen, Bohan Lin, Peng Sun, Xiaoyi Pang, Jian Liu, and Kui Ren. 2023. Shield against gradient leakage attacks: Adaptive privacy-preserving federated learning. *IEEE/ACM Transactions on Networking* (2023), 1–16. DOI : <https://doi.org/10.1109/TNET.2023.3317870>
- [53] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and fast secure two-party deep neural network inference. In *Proceedings of the 31st USENIX Security Symposium*. Boston, MA, USA.
- [54] Dino Ienco, Ruggero G. Pensa, and Rosa Meo. 2012. From context to distance: Learning dissimilarity for categorical data clustering. *ACM Transactions on Knowledge Discovery from Data* 6, 1 (2012), 1–25. DOI : <https://doi.org/10.1145/2133360.2133361>
- [55] David Jin, Niclas Kannengießer, Sascha Rank, and Ali Sunyaev. 2025. Collaborative distributed machine learning. *Comput. Surveys* 57, 4 (2025), 1–36. DOI : <https://doi.org/10.1145/3704807>
- [56] Chirag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A low latency framework for secure neural network inference. In *Proceedings of the 27th USENIX Security Symposium*. Baltimore, MD, USA.
- [57] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1-2 (2021), 1–210. DOI : <https://doi.org/10.1561/22000000083>
- [58] Shivam Kalra, Junfeng Wen, Jesse C. Cresswell, Maksims Volkovs, and H. R. Tizhoosh. 2023. Decentralized federated learning through proxy model sharing. *Nature Communications* 14, 2899 (2023). DOI : <https://doi.org/10.1038/s41467-023-38569-4>
- [59] Marcel Keller. 2020. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event, USA. DOI : <https://doi.org/10.1145/3372297.3417872>
- [60] Marcel Keller and Ke Sun. 2022. Secure quantized training for deep learning. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 162 (2022), 10912–10938. Available at: proceedings.mlr.press/v162/keller22a.html
- [61] Nishat Koti, Mahak Pancholi, Ajith Suresh, and Arpita Patra. 2021. SWIFT: Super-fast and robust privacy-preserving machine learning. In *Proceedings of the 30th USENIX Security Symposium*.
- [62] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. CryptFlow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 336–353. DOI : <https://doi.org/10.1109/SP40000.2020.00092>
- [63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444. DOI : <https://doi.org/10.1038/nature14539>
- [64] Minghui Li, Sherman S. M. Chow, Shengshan Hu, Yuejing Yan, Chao Shen, and Qian Wang. 2022. Optimizing privacy-preserving outsourced convolutional neural network predictions. *IEEE Transactions on Dependable and Secure Computing* 19, 3 (2022), 1592–1604. DOI : <https://doi.org/10.1109/TDSC.2020.3029899>
- [65] Ping Li, Jin Li, Zhengan Huang, Tong Li, Chong-Zhi Gao, Siu-Ming Yiu, and Kai Chen. 2017. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems* 74 (2017), 76–85. DOI : <https://doi.org/10.1016/j.future.2017.02.006>

- [66] Shaohua Li, Kaiping Xue, Bin Zhu, Chenkai Ding, Xindi Gao, David Wei, and Tao Wan. 2020. FALCON: A fourier transform based approach for fast and secure convolutional neural network predictions. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 8702–8711. DOI : <https://doi.org/10.1109/CVPR42600.2020.00873>
- [67] Tong Li, Jin Li, Xiaofeng Chen, Zheli Liu, Wenjing Lou, and Thomas Hou. 2020. NPMML: A framework for non-interactive privacy-preserving multi-party machine learning. *IEEE Transactions on Dependable and Secure Computing* 18, 6 (2020), 2969 – 2982. DOI : <https://doi.org/10.1109/TDSC.2020.2971598>
- [68] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. 2022. When machine learning meets privacy: A survey and outlook. *Comput. Surveys* 54, 2 (2022). DOI : <https://doi.org/10.1145/3436755>
- [69] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. 2017. Oblivious neural network predictions via MiniONN transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Dallas, TX, USA. DOI : <https://doi.org/10.1145/3133956.3134056>
- [70] Xiaoning Liu, Yifeng Zheng, Xingliang Yuan, and Xun Yi. 2023. Securely outsourcing neural network inference to the cloud with lightweight techniques. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 620–636. DOI : <https://doi.org/10.1109/TDSC.2022.3141391>
- [71] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. 2020. A secure federated transfer learning framework. *IEEE Intelligent Systems* 35, 4 (2020), 70–82. DOI : <https://doi.org/10.1109/MIS.2020.2988525>
- [72] Ziyu Liu, Shengyuan Hu, Zhiwei Steven Wu, and Virginia Smith. 2022. On privacy and personalization in cross-silo federated learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.
- [73] Qian Lou and Lei Jiang. 2019. SHE: A fast and accurate deep neural network for encrypted data. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*.
- [74] Qian Lou, Wen-jie Lu, Cheng Hong, and Lei Jiang. 2020. Falcon: Fast spectral inference on encrypted data. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.
- [75] Xindi Ma, Jianfeng Ma, Hui Li, Qi Jiang, and Sheng Gao. 2021. PDLN: Privacy-preserving deep learning model on cloud with multiple keys. *IEEE Transactions on Services Computing* 14, 4 (2021), 1251–1263. DOI : <https://doi.org/10.1109/TSC.2018.2868750>
- [76] Xu Ma, Xiaoqian Sun, Yuduo Wu, Zheli Liu, Xiaofeng Chen, and Changyu Dong. 2022. Differentially private Byzantine-robust federated learning. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (2022), 3690–3701. DOI : <https://doi.org/10.1109/TPDS.2022.3167434>
- [77] Xu Ma, Fangguo Zhang, Xiaofeng Chen, and Jian Shen. 2018. Privacy preserving multi-party computation delegation for deep learning in cloud computing. *Information Sciences* 459 (2018), 103–116. DOI : <https://doi.org/10.1016/j.ins.2018.05.005>
- [78] Saeed Mahloujifar, Esha Ghosh, and Melissa Chase. 2022. Property inference from poisoning. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 1120–1137. DOI : <https://doi.org/10.1109/SP46214.2022.9833623>
- [79] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Seth Hampson. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Vol. 54. JMLR, Fort Lauderdale, FL, USA.
- [80] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 691–706. DOI : <https://doi.org/10.1109/SP.2019.00029>
- [81] Ilya Mironov. 2017. Rényi differential privacy. In *IEEE 30th Computer Security Foundations Symposium (CSF)*. 263–275. DOI : <https://doi.org/10.1109/CSF.2017.11>
- [82] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. DELPHI: A cryptographic inference service for neural networks. In *Proceedings of the 29th USENIX Security Symposium*.
- [83] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. 2021. PPFL: Privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, New York, NY, USA, 94–108. DOI : <https://doi.org/10.1145/3458864.3466628>
- [84] Payman Mohassel and Peter Rindal. 2018. ABY³: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Toronto, Canada. DOI : <https://doi.org/10.1145/3243734.3243760>
- [85] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Jose, CA, USA, 19–38. DOI : <https://doi.org/10.1109/SP.2017.12>
- [86] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 739–753. DOI : <https://doi.org/10.1109/SP.2019.00065>

- [87] Deepika Natarajan, Andrew Loveless, Wei Dai, and Ronald Dreslinski. 2023. Chex-Mix: Combining homomorphic encryption with trusted execution environments for oblivious inference in the cloud. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, Delft, Netherlands, 73–91. DOI : <https://doi.org/10.1109/EuroSP57164.2023.00014>
- [88] Michael Ng, Mark Li, Joshua Huang, and Zengyou He. 2007. On the impact of dissimilarity measure in k-modes clustering algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 3 (2007), 503–507. DOI : <https://doi.org/10.1109/TPAMI.2007.53>
- [89] Thu-Hien Thi Nguyen and Van-Nam Huynh. 2016. A k-means-like algorithm for clustering categorical data using an information theoretic-based dissimilarity measure. In *Foundations of Information and Knowledge Systems*. Marc Gyssens and Guillermo Simari (Eds.). Vol. 9616. Springer International Publishing, Cham, 115–130. DOI : https://doi.org/10.1007/978-3-319-30024-5_7
- [90] Robert C. Nickerson, Upkar Varshney, and Jan Muntermann. 2013. A method for taxonomy development and its application in information systems. *European Journal of Information Systems* 22, 3 (2013), 336–359. DOI : <https://doi.org/10.1057/ejis.2012.26>
- [91] Yue Niu, Ramy E. Ali, and Salman Avestimehr. 2022. 3LegRace: Privacy-preserving DNN training over TEEs and GPUs. *Proceedings on Privacy Enhancing Technologies* 2022, 4 (2022), 183–203. DOI : <https://doi.org/10.56553/popets-2022-0105>
- [92] Maxence Noble, Aurelien Bellet, and Aymeric Dieuleveut. 2022. Differentially private federated learning on heterogeneous data. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Vol. 151. PMLR, Valencia, Spain.
- [93] Olga Ohrimenko, Felix Schuster, Cedric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. 2016. Oblivious multi-party machine learning on trusted processors. In *Proceedings of the 25th USENIX Security Symposium*. Austin, TX, USA.
- [94] C. Orlandi, A. Piva, and M. Barni. 2007. Oblivious neural network computing via homomorphic encryption. *EURASIP Journal on Information Security* 2007 (2007), 1–11. DOI : <https://doi.org/10.1155/2007/37343>
- [95] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT ’99*, Jacques Stern (Ed.), Vol. 1592. Springer, Berlin. DOI : https://doi.org/10.1007/3-540-48910-X_16
- [96] Jaehyoung Park and Hyuk Lim. 2022. Privacy-preserving federated learning using homomorphic encryption. *Applied Sciences* 12, 2 (2022), 734. DOI : <https://doi.org/10.3390/app12020734>
- [97] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2021. ABY2.0: Improved mixed-protocol secure two-party computation. In *Proceedings of the 30th USENIX Security Symposium*.
- [98] Arpita Patra and Ajith Suresh. 2020. BLAZE: Blazing fast privacy-preserving machine learning. In *Proceedings 2020 Network and Distributed System Security Symposium*. San Diego, CA, USA. DOI : <https://doi.org/10.14722/ndss.2020.24202>
- [99] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2018. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security* 13, 5 (May 2018), 1333–1345. DOI : <https://doi.org/10.1109/TIFS.2017.2787987>
- [100] Sandro Pinto and Nuno Santos. 2019. Demystifying ARM TrustZone: A comprehensive survey. *Comput. Surveys* 51, 6 (2019), 1–36. DOI : <https://doi.org/10.1145/3291047>
- [101] Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H. Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Guha Thakurta. 2023. How to DP-fy ML: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research* 77 (2023), 1113–1201. DOI : <https://doi.org/10.1613/jair.1.14649>
- [102] Michael O. Rabin. 1981. *How To Exchange Secrets with Oblivious Transfer*. Technical Report 81. Harvard University. Retrieved from <https://eprint.iacr.org/2005/187>
- [103] Atiqur Rahman, Tanzila Rahman, Robert Laganieri, Noman Mohammed, and Yang Wang. 2018. Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy* 11, 1 (2018), 61–79.
- [104] Sascha Rank, Florian Leiser, Scott Thiebes, and Ali Sunyaev. 2024. Inter-organizational collaboration for machine learning: Motivating and discouraging factors in the automotive industry. In *Proceedings of the European Conference on Information Systems (ECIS’24)*. Available at: https://aisel.aisnet.org/ecis2024/track03_ai/track03_ai/14
- [105] Sascha Rank, Scott Thiebes, and Ali Sunyaev. 2026. Inter-organizational collaborative machine learning: A problem space exploration. In *Proceedings of the 59th Hawaii International Conference on System Sciences*. Maui, Hawaii. Retrieved from <https://hdl.handle.net/10125/111456>
- [106] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. CryptFlow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event, USA. DOI : <https://doi.org/10.1145/3372297.3417274>
- [107] Mayank Rathee, Conghao Shen, Sameer Wagh, and Raluca Ada Popa. 2023. ELSA: Secure aggregation for federated learning with malicious actors. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 1961–1979. DOI : <https://doi.org/10.1109/SP46215.2023.10179468>

- [108] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T. Lee, Hsien-Hsin S. Lee, Gu-Yeon Wei, and David Brooks. 2021. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, Seoul, Republic of Korea, 26–39. DOI : <https://doi.org/10.1109/HPCA51647.2021.00013>
- [109] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. 2019. In *Proceedings of the 28th USENIX Security Symposium*. Available at: <https://www.usenix.org/conference/usenixsecurity19/presentation/riazi>
- [110] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, Incheon, Republic of Korea, 707–721. DOI : <https://doi.org/10.1145/3196494.3196522>
- [111] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus Maier-Hein, et al. 2020. The future of digital health with federated learning. *npj Digital Medicine* 3, 1 (2020). DOI : <https://doi.org/10.1038/s41746-020-00323-1>
- [112] Maria Rigaki and Sebastian Garcia. 2024. A survey of privacy attacks in machine learning. *Comput. Surveys* 56, 4 (2024), 1–34. DOI : <https://doi.org/10.1145/3624010>
- [113] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126. DOI : <https://doi.org/10.1145/359340.359342>
- [114] Bitu Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. 2018. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the 55th Annual Design Automation Conference*. ACM, San Francisco California, 1–6. DOI : <https://doi.org/10.1145/3195970.3196023>
- [115] Théo Ryffel, Pierre Tholoniat, David Pointcheval, and Francis Bach. 2022. ARIANN: Low-interaction privacy-preserving deep learning via function secret sharing. *Proceedings on Privacy Enhancing Technologies* 2022, 1 (2022), 291–316. DOI : <https://doi.org/10.2478/popets-2022-0015>
- [116] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted execution environment: What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. 57–64. DOI : <https://doi.org/10.1109/Trustcom.2015.357>
- [117] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. 2020. Updates-leak: Data set inference and reconstruction attacks in online learning. In *Proceedings of the 29th USENIX Security Symposium*.
- [118] Amartya Sanyal, Matt J. Kusner, Adrià Gascón, and Varun Kanade. 2018. TAPAS: Tricks to accelerate (encrypted) prediction as a service. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR.
- [119] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (Aug. 2021), 3710–3722. DOI : <https://doi.org/10.1109/TNNLS.2020.3015958>
- [120] Sinem Sav, Apostolos Pyrgelis, Juan R. Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. 2021. POSEIDON: Privacy-preserving federated neural network learning. In *Proceedings 2021 Network and Distributed System Security Symposium*. Virtual Event.
- [121] Marius Schlegel and Kai-Uwe Sattler. 2023. Management of machine learning lifecycle artifacts: A survey. *ACM SIGMOD Record* 51, 4 (2023), 18–35. DOI : <https://doi.org/10.1145/3582302.3582306>
- [122] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613. DOI : <https://doi.org/10.1145/359168.359176>
- [123] Muhammad Shayan, Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2021. Biscotti: A blockchain system for private and secure federated learning. *IEEE Transactions on Parallel and Distributed Systems* 32, 7 (2021), 1513–1525. DOI : <https://doi.org/10.1109/TPDS.2020.3044223>
- [124] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. San Jose, CA, USA, 3–18. DOI : <https://doi.org/10.1109/SP.2017.41>
- [125] Jinhyun So, Basak Guler, and A. Salman Avestimehr. 2021. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications* 39, 7 (2021), 2168–2181. DOI : <https://doi.org/10.1109/JSAC.2020.3041404>
- [126] Jinhyun So, Basak Guler, and A. Salman Avestimehr. 2021. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory* 2, 1 (2021), 479–489. DOI : <https://doi.org/10.1109/JSAIT.2021.3054610>
- [127] Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Virtual Event, USA. DOI : <https://doi.org/10.1145/3372297.3417270>
- [128] Timothy Stevens, Christian Skalka, Christelle Vincent, and Samuel Clark. 2022. Efficient differentially private secure aggregation for federated learning via hardness of learning with errors. In *Proceedings of the 31st USENIX Security Symposium*. Boston, MA, USA.

- [129] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Venice, Italy, 843–852. DOI : <https://doi.org/10.1109/ICCV.2017.97>
- [130] Philipp A. Toussaint, Florian Leiser, Scott Thiebes, Matthias Schlesner, Benedikt Brors, and Ali Sunyaev. 2023. Explainable artificial intelligence for omics data: a systematic mapping study. *Briefings in Bioinformatics* 25, 1 (2023). DOI : <https://doi.org/10.1093/bib/bbad453>
- [131] Florian Tramèr and Dan Boneh. 2019. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *7th International Conference on Learning Representations*.
- [132] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In *Proceedings of the 25th USENIX Security Symposium*. Austin, TX, USA.
- [133] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursay, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. ACM, Heraklion, Greece, 61–66. DOI : <https://doi.org/10.1145/3378679.3394533>
- [134] Jasper Van Der Waa, Elisabeth Nieuwburg, Anita Cremers, and Mark Neerincx. 2021. Evaluating XAI: A comparison of rule-based and example-based explanations. *Artificial Intelligence* 291 (2021). DOI : <https://doi.org/10.1016/j.artint.2020.103404>
- [135] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-Party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 26–49. DOI : <https://doi.org/10.2478/popets-2019-0035>
- [136] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2021. FALCON: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies* 2021, 1 (2021), 188–208. DOI : <https://doi.org/10.2478/popets-2021-0011>
- [137] Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S. Yu. 2018. Not just privacy: Improving performance of private deep learning in mobile cloud. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, London, United Kingdom, 2407–2416. DOI : <https://doi.org/10.1145/3219819.3220106>
- [138] Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, Sathyanarayanan Manamohan, Saikat Mukherjee, Vishesh Garg, Ravi Sarveswara, Kristian Händler, Peter Pickkers, N. Ahmad Aziz, et al. 2021. Swarm learning for decentralized and confidential clinical machine learning. *Nature* 594 (2021), 265–270. DOI : <https://doi.org/10.1038/s41586-021-03583-3>
- [139] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469. DOI : <https://doi.org/10.1109/TIFS.2020.2988575>
- [140] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, London, United Kingdom, 1–10. DOI : <https://doi.org/10.1145/2601248.2601268>
- [141] Guowen Xu, Xingshuo Han, Shengmin Xu, Tianwei Zhang, Hongwei Li, Xinyi Huang, and Robert H. Deng. 2023. Hercules: Boosting the performance of privacy-preserving federated learning. *IEEE Transactions on Dependable and Secure Computing* 20, 5 (2023), 4418–4433. DOI : <https://doi.org/10.1109/TDSC.2022.3218793>
- [142] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. 2020. VerifyNet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security* 15 (2020), 911–926. DOI : <https://doi.org/10.1109/TIFS.2019.2929409>
- [143] Guowen Xu, Hongwei Li, Yun Zhang, Shengmin Xu, Jianting Ning, and Robert Deng. 2020. Privacy-preserving federated deep learning with irregular users. *IEEE Transactions on Dependable and Secure Computing* (2020). DOI : <https://doi.org/10.1109/TDSC.2020.3005909>
- [144] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316. DOI : <https://doi.org/10.1016/j.neucom.2020.07.061>
- [145] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science*. 162–167. DOI : <https://doi.org/10.1109/SFCS.1986.25>
- [146] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. 2020. CloudLeak: Large-scale deep learning models stealing through adversarial examples. In *Proceedings 2020 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA, USA. DOI : <https://doi.org/10.14722/ndss.2020.24178>
- [147] Chuan Zhang, Chenfei Hu, Tong Wu, Liehuang Zhu, and Ximeng Liu. 2023. Achieving efficient and privacy-preserving neural network training and prediction in cloud environments. *IEEE Transactions on Dependable and Secure Computing* 20, 5 (2023), 4245–4257. DOI : <https://doi.org/10.1109/TDSC.2022.3208706>
- [148] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning. In *Proceedings of the 2020 USENIX Annual Technical Conference*.

- [149] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. 2020. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 253–261.
- [150] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. 2019. Secure multi-party computation: Theory, practice and applications. *Information Sciences* 476 (2019), 357–372. DOI : <https://doi.org/10.1016/j.ins.2018.10.024>
- [151] Jiaqi Zhao, Hui Zhu, Fengwei Wang, Rongxing Lu, Zhe Liu, and Hui Li. 2022. PVD-FL: A privacy-preserving and verifiable decentralized federated learning framework. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2059–2073. DOI : <https://doi.org/10.1109/TIFS.2022.3176191>
- [152] Lingchen Zhao, Qian Wang, Qin Zou, Yan Zhang, and Yanjiao Chen. 2020. Privacy-preserving collaborative deep learning with unreliable participants. *IEEE Transactions on Information Forensics and Security* 15 (2020), 1486–1500. DOI : <https://doi.org/10.1109/TIFS.2019.2939713>
- [153] Ping Zhao, Zhikui Cao, Jin Jiang, and Fei Gao. 2023. Practical private aggregation in federated learning against inference attack. *IEEE Internet of Things Journal* 10, 1 (2023), 318–329. DOI : <https://doi.org/10.1109/JIOT.2022.3201231>
- [154] Yifeng Zheng, Shangqi Lai, Yi Liu, Xingliang Yuan, Xun Yi, and Cong Wang. 2023. Aggregation service for federated learning: An efficient, secure, and more resilient realization. *IEEE Transactions on Dependable and Secure Computing* 20, 2 (2023), 988–1001. DOI : <https://doi.org/10.1109/TDSC.2022.3146448>
- [155] Hangyu Zhu, Haoyu Zhang, and Yaochu Jin. 2021. From federated learning to federated neural architecture search: a survey. *Complex & Intelligent Systems* 7, 2 (2021), 639–657. DOI : <https://doi.org/10.1007/s40747-020-00247-z>

A Appendix

A.1 Snowballing Literature Search

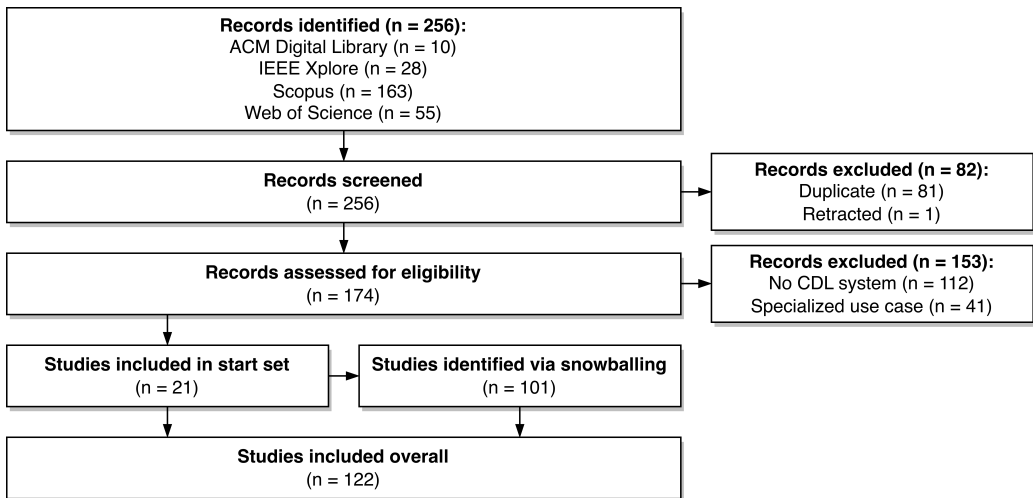


Fig. 6. Identification of relevant literature on CDL systems using a snowballing literature search.

A.2 Taxonomy Development Iterations

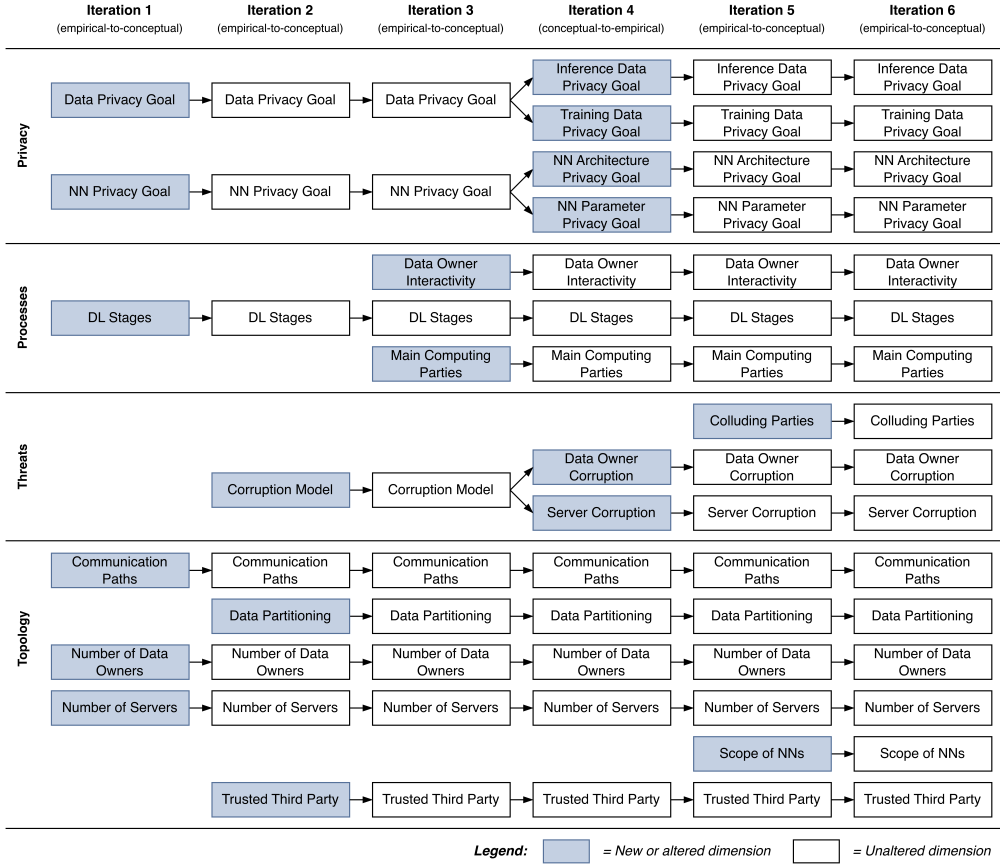


Fig. 7. Overview of the taxonomy development iterations.

A.3 Clustering Results

Table 8. Average Silhouette Scores for Selected Clustering Algorithms and Numbers of Clusters k

k	CD-clustering	k-medoids	k-modes	k-PbC	k-SCC	LSHFk-Centers	StepMix
3	0.4380	0.4438	0.4374	0.3743	0.4450	0.3829	0.4170
4	0.4555	0.3911	0.3974	0.3517	0.4525	0.4593	0.4129
5	0.4492	0.3844	0.4057	0.3253	0.3964	0.4108	0.3933
6	0.3735	0.3637	0.4442	0.2153	0.3511	0.3361	0.3823
7	0.3974	0.3441	0.4327	0.3278	0.3659	0.3492	0.3723
8	0.4015	0.3260	0.4114	0.3312	0.3838	0.3490	0.3699
9	0.3702	0.3604	0.4031	0.3991	0.3883	0.3679	0.3675
10	0.3828	0.3862	0.4157	0.3898	0.3894	0.3831	0.3680

Higher silhouette scores indicate better cluster cohesion and separation. The highest silhouette score for each number of clusters is highlighted in bold. The row with the highest silhouette score overall is highlighted in blue.

Received 19 March 2025; revised 19 March 2025; accepted 3 March 2026