FedML: A Research Library and Benchmark for Federated Machine Learning

Chaoyang He[*] **Songze Li,** **Jinhyun So,** **Mi Zhang,** **Hongyi Wang,** **Xiaoyang Wang**
USC Stanford USC MSU UW-Madison UIUC
Praneeth Vepakomma, **Abhishek Singh,** **Hang Qiu,** **Li Shen, Peilin Zhao,** **Yan Kang**
MIT MIT USC Tencent AI WeBank
Yang Liu, **Ramesh Raskar,** **Qiang Yang,** **Murali Annavaram**[*] **Salman Avestimehr**[*]
WeBank MIT HKUST USC USC

Abstract

Federated learning is a rapidly growing research field in the machine learning domain. Although considerable research efforts have been made, existing libraries cannot adequately support diverse algorithmic development (e.g., diverse topology and flexible message exchange), and inconsistent dataset and model usage in experiments make fair comparisons difficult. In this work, we introduce FedML, an open research library and benchmark that facilitates the development of new federated learning algorithms and fair performance comparisons. FedML supports three computing paradigms (*distributed training, mobile on-device training, and standalone simulation*) for users to conduct experiments in different system environments. FedML also promotes diverse algorithmic research with *flexible and generic API design and reference baseline implementations*. *A curated and comprehensive benchmark dataset* for the non-I.I.D setting aims at making a fair comparison. We believe FedML can provide an efficient and reproducible means of developing and evaluating algorithms for the federated learning research community. We maintain the source code, documents, and user community at https://FedML.ai.

1 Introduction

Federated learning (FL) is a distributed learning paradigm that aims to train machine learning models from scattered and isolated data [1]. FL differs from data center-based distributed training in three major aspects: *statistical heterogeneity* (non-I.I.D., limited labels, etc.), *system constraints* (communication and computation), and *trustworthiness* (security, privacy, fairness, etc.). Solving these unique challenges calls for efforts from a variety of fields, including machine learning, wireless communication, mobile computing, distributed systems, and information security, making federated learning a truly *interdisciplinary* research area.

In the past few years, considerable efforts have been made to address these unique challenges. To tackle the statistical heterogeneity challenge, distributed optimization methods such as FedMA [2], FedProx [3], Adaptive Federated Optimization [4] and FedNAS [5] have been proposed. To tackle the system constraints challenge, researchers apply compression, sparsification, or quantization techniques to reduce the communication overheads and computation costs during the training process [6, 7, 8, 9, 10, 11, 12]. To tackle the trustworthiness challenge, existing research focuses on developing new adversarial attack and defense techniques to make federated learning robust [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 22], and proposing methods such as differential privacy (DP) and secure multiparty computation (SMPC) to protect privacy [25, 26, 27, 28, 29, 30, 31, 32, 33] (see Table 8 in the appendix for a taxonomy of the federated learning literature).

[*] Corresponding authors. Email: chaoyang.he@usc.edu

Despite these efforts, we observe that existing efforts are confronted with a number of limitations that we argue are critical to FL research:

Lack of support of diverse FL configurations. FL algorithms are diverse in *network topology*, *exchanged information*, and *training procedures*. In terms of network topology, a variety of network typologies such as vertical FL [34, 35, 36, 37, 38, 39, 40], split learning [41, 42], decentralized FL [43, 44, 45, 46], hierarchical FL [47, 48, 49, 50, 51, 52], and meta FL [53, 54, 55] have been proposed. In terms of exchanged information, aside from exchanging gradients and models, recent FL algorithms propose to exchange information such as pseudo labels in semi-supervised FL [56] and architecture parameters in neural architecture search-based FL [5, 57, 58]. In terms of training procedures, the training procedure in federated GAN [59, 60] and transfer learning-based FL [61, 62, 63, 64, 65] is significantly different from the vanilla FedAvg algorithm [66]. However, existing FL libraries are not able to support such diversity.

Lack of support of diverse FL computing paradigms. General distributed training libraries in PyTorch [67], TensorFlow [68], MXNet [69], and distributed training-specialized libraries such as Horovod [70] and BytePS [71] are designed for distributed training in data centers. Although simulation-oriented FL libraries such as TensorFlow-Federated (TFF) [2], PySyft [28], and LEAF [72] are developed, they only support centralized topology-based algorithms like FedAvg [66] or FedProx [73] algorithms with simulation in a single machine, making them unsuitable for algorithms which require the exchange of complex auxiliary information and customization of the training procedure. Production-oriented libraries such as FATE [74] and PaddleFL [75] are released by the industry. However, they are not designed as flexible frameworks that aim at supporting algorithmic innovation for open problems. Industry-led products normally have heavy system design, inflexible APIs, and complicated environmental setup, which is a high learning burden for algorithm researchers who do not have enough expertise in distributed system development.

Lack of standardized FL algorithm implementations. Given the disadvantages of the standard distributed training library, implementations from many publications are in a different programming manner under different frameworks, making the algorithm comparison impractical or inefficient. The non-I.I.D. characteristic of FL makes reproducibility even more challenging [76]: training the same DNN on different non-I.I.D. datasets produces varying model accuracies; one algorithm that achieves higher accuracy than the other algorithms on a specific non-I.I.D. distribution may perform worse on another non-I.I.D. distribution. As such, reference implementations are essential to evaluate the performance of FL algorithms fairly.

Lack of standardized FL benchmarks. A fair comparison with different algorithms is difficult since the non-I.I.D. datasets used in existing work are so diverse. In Table 7 in the appendix, we summarize the non-I.I.D. datasets and models used in existing publications from top tier conferences of the machine learning community (*e.g.* ICML, NeurIPS, CVPR, ICLR, and etc.) during the past two years. Surprisingly, the experimental settings of these published articles are very inconsistent, including settings for the datasets, non-I.I.D. partition methods, models used for training, and the number of clients involved in each round. Any difference in these settings could affect the experimental results.

To address the above problems and facilitate innovations in FL research, in this work, we present `FedML`, an open research library and benchmark for FL. `FedML` provides an end-to-end toolkit for developing and evaluating FL algorithms in diverse configurations and computing paradigms. Moreover, it provides standardized implementations of existing FL algorithms and benchmarks that enable fair performance comparisons. Table 1 summarizes the key differences between `FedML` and existing FL libraries and benchmarks. The highlights of `FedML` are summarized below:

(i) Support of diverse FL configurations. `FedML` introduces a worker (also refer to client or device in the FL literature)-oriented programming interface for flexible topology configuration and arbitrary information exchanging among workers (Section 3). Users can attach any behavior for workers in the FL setting (e.g., training, aggregation, attack, and defense, etc.), customize additional exchange information, and control information flow among workers, making `FedML` more flexible and generic for more advanced algorithm development.

(ii) Support of diverse FL computing paradigms. `FedML` supports three computing paradigms: distributed computing, standalone simulation, and mobile on-device training. Figure 1 shows the

[2] https://www.tensorflow.org/federated

Table 1: Comparison with existing federated learning libraries and benchmarks.

		TFF	FATE	PaddleFL	LEAF	PySyft	FedML
Flexible and Generic API Design	topology customization	✗	✗	✗	✗	✗	✓
	flexible message flow	✗	✗	✗	✗	✗	✓
	exchange message customization	✗	✗	✗	✗	✗	✓
Diversified Computing Paradigms	standalone simulation	✓	✓	✓	✓	✓	✓
	distributed computing	✗	✓	✓	✗	✗	✓
	mobile on-device training	✓	✗	✗	✗	✗	✓
Standardized Algorithm Implementations	FedAvg	✓	✓	✓	✓	✓	✓
	decentralized FL	✗	✗	✗	✗	✗	✓
	FedNAS (beyond gradient/model)	✗	✗	✗	✗	✗	✓
	vertical FL	✗	✓	✓	✗	✗	✓
Standardized Benchmarks	linear models	✓	✓	✓	✓	✓	✓
	shallow NN	✓	✓	✓	✓	✓	✓
	Deep Neural Network	✗	✗	✗	✗	✗	✓
	vertical FL	✗	✓	✗	✗	✗	✓

abstract architecture of FedML. Three computing paradigms can meet various algorithmic and system-level research requirements on different scale models and datasets under different system environments (Section 2).

(iii) Standardized FL algorithm implementations. FedML provides standardized implementations of a number of status quo FL algorithms. These implementations not only help users to familiarize the APIs but also can be used as baselines for comparisons with newly developed FL algorithms (Section 4).

(iv) Standardized FL benchmarks. Lastly, FedML provides a standardized benchmark for the non-I.I.D. setting. To promote fair and solid comparison, FedML presents meaningful metrics and baseline results under fair settings, making sure that all algorithms are evaluated on multiple synthetic and real-world non-I.I.D datasets with reasonably tuned hyperparameters (Section 5).

(v) FedML is fully open and evolving. FL is a research field that evolves at a considerably fast pace. Old algorithms are being replaced by new algorithms with superior performances, and new datasets are collected in many newly explored usage scenarios. This requires FedML to adapt at the same pace. Consequently, we plan to expand FedML to include more APIs, reference implementations, and benchmarks. We hope that the FL research community will also contribute to FedML to make this research library and benchmark more comprehensive and thus further benefiting the community.

2 Architecture Design

The system architecture of FedML is shown in Figure 1. FedML supports three computing paradigms: distributed computing, mobile on-device training, and standalone simulation. At its core are FedML-API and FedML-core, which represent high-level API and low-level API, respectively.

FedML-core separates the communication and the model training into two core components. The first is the communication protocol component (labeled as *distributed* in the figure). It is responsible for low-level communication among different works in the network. The communication backend is based on MPI (message passing interface) [3]. We consider adding more backends as necessary, such as RPC (remote procedure call). Inside the communication protocol component, a TopologyManager supports the flexible topology configuration required by different distributed learning algorithms. The second is the on-device deep learning component, which is built based on the popular deep learning framework PyTorch or TensorFlow. For flexibility, there is no restriction on the framework for this part. Users can implement trainers and coordinators according to their needs. In addition, low-level APIs support security and privacy-related algorithms (introduced in Section 3).

[3] https://pypi.org/project/mpi4py/

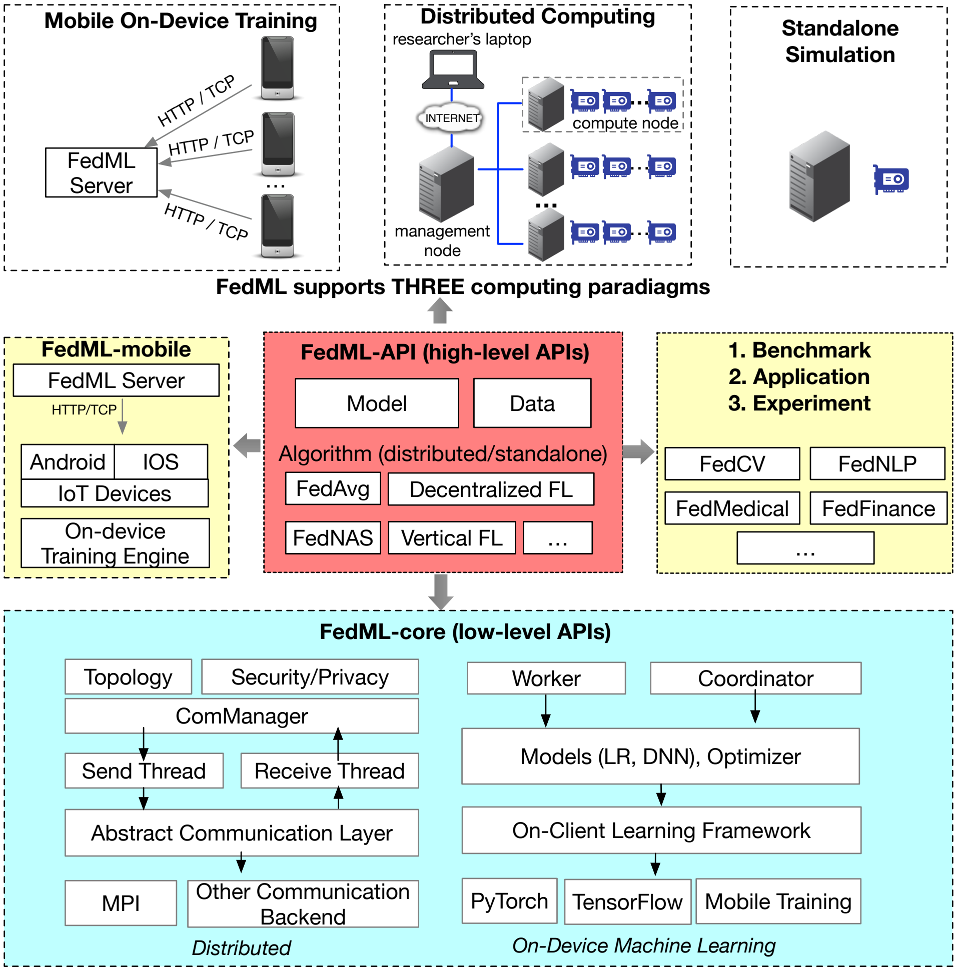


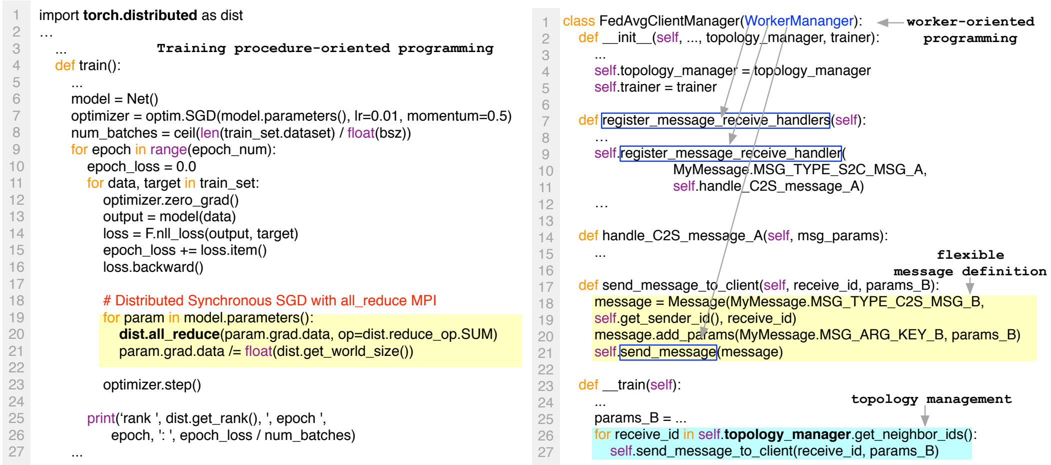
Figure 1: Abstract System Architecture

FedML-API is built based on FedML-core. With the help of FedML-core, new algorithms in distributed version can be easily implemented by adopting the worker-oriented programming interface, which is a novel design pattern for flexible distributed computing (introduced in Section 3). Such a distributed computing paradigm is essential for scenarios in which large DNN training cannot be handled by standalone simulation due to GPU memory and training time constraints. *We specifically point out that this distributed computing design is not only used for FL, but it can also be used for conventional in-cluster large-scale distributed training (e.g., training modern neural architectures like CNNs or transformers)*. FedML-API also suggests a machine learning system practice that separates the implementations of models, datasets, and algorithms. This practice can enable code reuse and also fair comparison, avoiding statistical or system-level gaps among algorithms led by non-trivial implementation differences. Another benefit is that FL applications can develop more models and submit more realistic datasets without the need to understand the details of different distributed optimization algorithms. We hope that researchers in diverse FL applications can contribute more valuable models and realistic datasets to our community. Promising application domains include, but are not limited to, computer vision [77, 78], natural language processing [79, 80, 81, 82, 83], finance [74, 35, 84], transportation [85, 86, 87, 88, 89, 90, 91, 92, 93, 94], digital health [95, 96, 97, 98, 99, 100, 101], recommendation [102, 103, 104, 105, 106, 107], robotics [108, 109], and smart cities [110, 111].

Other components in the Figure 1, such as experiments, benchmarks, and FedML-Mobile, are built based on FedML-API. FedML-Mobile is *a real-world FL on-device training test bed*. It can train neural networks on Android/iOS smartphones. With this testbed in the wireless network environment, researchers can evaluate realistic system performance, such as training time, communication, and computation cost. On the worker side, FedML-mobile adopts DL4J [4] as the on-device training engine. On the server-side of FedML-mobile, we build cloud service using *Python*-based framework, so it can reuse core implementations from FedML-API. Except for the Android/iOS platform, all our codes are implemented in *Python*, which is the primary programming language used by researchers.

3 Programming Interface

The philosophy of the FedML programming interface is to provide the simplest user experience *i.e.* allowing users to build distributed training applications (*e.g.* to design customized message flow and topology definitions) by only focusing on algorithmic implementations while ignoring the low-level communication backend details.


Training procedure-oriented programming
worker-oriented programming
flexible message definition
topology management

```
import torch.distributed as dist
…
   ...
   def train():
      ...
      model = Net()
      optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
      num_batches = ceil(len(train_set.dataset) / float(bsz))
      for epoch in range(epoch_num):
         epoch_loss = 0.0
         for data, target in train_set:
            optimizer.zero_grad()
            output = model(data)
            loss = F.nll_loss(output, target)
            epoch_loss += loss.item()
            loss.backward()

            # Distributed Synchronous SGD with all_reduce MPI
            for param in model.parameters():
               dist.all_reduce(param.grad.data, op=dist.reduce_op.SUM)
               param.grad.data /= float(dist.get_world_size())

            optimizer.step()

         print('rank ', dist.get_rank(), ', epoch ',
              epoch, ': ', epoch_loss / num_batches)
      ...
```

(a) Training procedure-oriented programming

```
class FedAvgClientManager(WorkerMananger):
   def __init__(self, ..., topology_manager, trainer):
      ...
      self.topology_manager = topology_manager
      self.trainer = trainer

   def register_message_receive_handlers(self):
      …
      self.register_message_receive_handler(
                     MyMessage.MSG_TYPE_S2C_MSG_A,
                     self.handle_C2S_message_A)
      …

   def handle_C2S_message_A(self, msg_params):
      ...

   def send_message_to_client(self, receive_id, params_B):
      message = Message(MyMessage.MSG_TYPE_C2S_MSG_B,
      self.get_sender_id(), receive_id)
      message.add_params(MyMessage.MSG_ARG_KEY_B, params_B)
      self.send_message(message)

   def __train(self):
      ...
      params_B = ...
      for receive_id in self.topology_manager.get_neighbor_ids():
         self.send_message_to_client(receive_id, params_B)
```

(b) Worker-oriented programming

Figure 2: A worker-oriented programming design pattern of FedML

Worker-oriented programming. As shown in Figure 2(b), FedML provides the worker-oriented programming design pattern, which can be used to program the worker behavior when participating in training or coordination in the FL algorithm. We describe it as worker-oriented because its counterpart, the standard distributed training library (as the torch.distributed example [5] shown in Figure 2(a)), normally completes distributed training programming by describing the entire training procedure rather than focusing on the behavior of each worker.

With the worker-oriented programming design pattern, the user can customize its own worker in FL network by inheriting the WorkerManager class and utilizing its predefined APIs register_message_receive_handler and send_message to define the receiving and sending messages without considering the underlying communication mechanism (as shown in the highlighted blue box in Figure 2(b)). Conversely, existing distributed training frameworks do not have such flexibility for algorithm innovation. In order to make the comparison clearer, we use the most popular machine learning framework PyTorch as an example. Figure 2(a) illustrates a complete training procedure (distributed synchronous SGD) and aggregates gradients from all other workers with the all_reduce messaging passing interface. Although it supports multiprocessing training, it cannot flexibly customize different messaging flows in any network topology. In PyTorch, another distributed training API, torch.nn.parallel.paraDistributedDataParallel [6], also has such inflexibly.

[4] https://deeplearning4j.org/

[5] More details can be found at https://pytorch.org/tutorials/intermediate/dist_tuto.html

[6] It is recommended to use torch.nn.parallel.paraDistributedDataParallel instead of torch.nn.DataParallel. For more details, please refer to https://pytorch.org/tutorials/intermediate/ddp_tutorial.html and https://pytorch.org/docs/master/notes/cuda.html#cuda-nn-ddp-instead

Note that `torch.distributed.rpc` [7] is a low-level communication back API that can finish any communication theoretically, but it is not user-friendly for federated learning researchers.

Message definition beyond gradient and model. FedML also considers supporting message exchange beyond the gradient or model from the perspective of message flow. This type of auxiliary information may be due to either the need for algorithm design or the need for system-wide configuration delivery. Each worker defines the message type from the perspective of sending. Thus, in the above introduced worker-oriented programming, the `WorkerManager` should handle messages defined by other trainers and also send messages defined by itself. The sending message is normally executed after handling the received message. As shown in Figure 2(b), in the yellow background highlighted code snippet, workers can send any message type and related message parameters during the `train()` function.

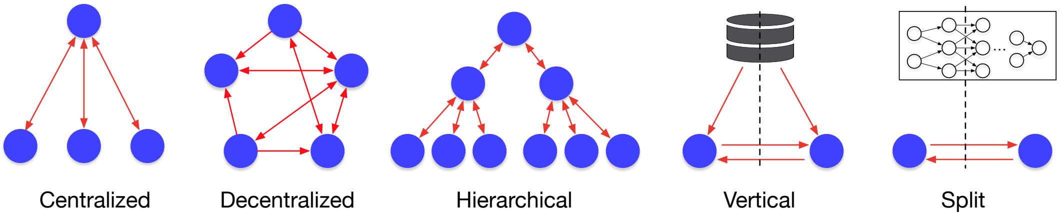


Figure 3: Various Topology Definitions in Federated Learning

Topology management. As demonstrated in Figure 3, FL has various topology definitions, such as vertical FL [34, 35, 36, 37, 38, 39, 40], split learning [41, 42], decentralized FL [43, 44, 45, 46], and Hierarchical FL [47, 48, 49, 50, 51, 52]. In order to meet such diverse requirements, FedML provides TopologyManager to manage the topology and allow users to send messages to arbitrary neighbors during training. Specifically, after the initial setting of TopologyManager is completed, for each trainer in the network, the neighborhood worker ID can be queried through the TopologyManager. In line 26 of Figure 2(b), we see that the trainer can query its neighbor nodes through the TopologyManager before sending its message.

Trainer and coordinator. We also need the coordinator to complete the training (*e.g.*, in the FedAvg algorithm, the central worker is the coordinator while the others are trainers). For the trainer and coordinator, FedML does not over-design. Rather, it gives the implementation completely to the developers, reflecting the flexibility of our framework. The implementation of the trainer and coordinator is similar to the process in Figure 2(a), which is completely consistent with the training implementation of a standalone version training. We provide some reference implementations of different trainers and coordinators in our source code (Section 4).

Privacy, security, and robustness. While the FL framework facilitates data privacy [89] by keeping data locally available to the users and only requiring communication for model updates, users may still be concerned about partial leakage of their data which may be inferred from the communicated model (see, *e.g.*, [112]). Aside from protecting the privacy of users' data, another critical security requirement for the FL platform, especially when operating over mobile devices, is the robustness towards user dropouts. Specifically, to accomplish the aforementioned goals of achieving security, privacy, and robustness for FL, various cryptography and coding-theoretic approaches have been proposed to manipulate intermediate model data (see [113, 114]).

To facilitate rapid implementation and evaluation of data manipulation techniques to enhance security, privacy, and robustness, we include low-level APIs that implement common cryptographic primitives such as secrete sharing, key agreement, digital signature, and public key infrastructure. We also plan to include an implementation of `Lagrange Coded Computing` (LCC) [115]. LCC is a recently developed coding technique on data that achieves optimal resiliency, security (against adversarial

[7] https://pytorch.org/tutorials/intermediate/rpc_tutorial.html

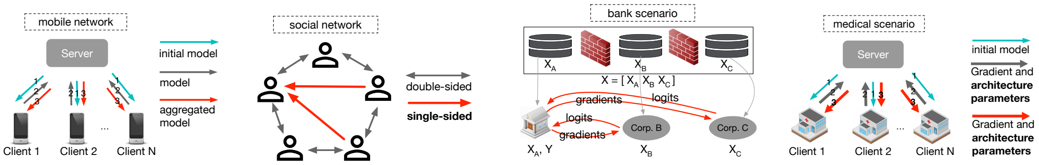
nodes), and privacy for any polynomial evaluations on the data. Finally, we plan to provide a sample implementation of the secure aggregation algorithm [113], using the above APIs.

In standard FL settings, it is assumed that there is no single central authority that owns or verifies the training data or user hardware, and it has been argued by many recent studies that FL lends itself to new adversarial attacks during decentralized model training [20, 23, 18, 116, 117]. Several robust aggregation methods have been proposed to enhance the robustness of FL against adversaries [23, 118, 119].

To accelerate generating benchmark results on new types of adversarial attacks in FL, we include the latest robust aggregation methods presented in literature *i.e.* (i) norm difference clipping [23]; weak differential private (DP) [23]; (ii) RFA (geometric median) [118]; (iii) KRUM and (iv) MULTI-KRUM [119]. Our APIs are easily extendable to support newly developed types of robust aggregation methods. On the attack end, we observe that most of the existing attacks are highly task-specific. Thus, it is challenging to provide general adversarial attack APIs. Our APIs support the *backdoor with model replacement attack* presented in [20] and the *edge-case backdoor attack* presented in [117] to provide a reference for researchers to develop new attacks.

4 Application Examples and Reference Implementations

Our initial release of `FedML` provides five algorithm examples. These examples can also be used as reference implementations. We explain the property of each example and describe how they are developed based on `FedML`. We plan to provide more examples (*e.g.*, Adaptive FL [4], FedProx [3], FedMA [2] and Turbo-Aggregate [114]) in the near future.



(a) Centralized FL (b) Decentralized FL (c) Vertical FL (d) FedNAS

Figure 4: Application Examples that Use `FedML`

- **Federated Averaging (FedAvg).** FedAvg [66] is a standard federated learning algorithm that is normally used as a baseline for advanced algorithm comparison. We summarize the algorithm message flow in Figure 4(a). Each worker trains its local model for several epochs, then updates its local model to the server. The server aggregates the uploaded client models into a global model by weighted coordinate-wise averaging (the weights are determined by the number of data points on each worker locally), and then synchronizes the global model back to all workers. In our `FedML` library, based on the worker-oriented programming, we can implement this algorithm in a distributed computing manner. We suggest that users start from FedAvg to learn using `FedML`.
- **Decentralized FL.** We use [43], a central server free FL algorithm, to demonstrate how `FedML` supports decentralized topology with directed communication. As Figure 4(b) shows, such an algorithm uses a decentralized topology, and more specifically, some workers do not send messages (model) to all of their neighbors. The worker-oriented programming interface can easily meet this requirement since it allows users to define any behavior for each worker.
- **Vertical Federated Learning (VFL).** VFL or feature-partitioned FL [120] is applicable to the cases where all participating parties share the same sample space but differ in the feature space. As illustrated in Figure 4(c), VFL is the process of aggregating different features and computing the training loss and gradients in a privacy-preserving manner to build a model with data from all parties collaboratively [121, 35, 122, 123]. The `FedML` library currently supports the logistic regression model with customizable local feature extractors in the vertical FL setting, and it provides NUS-WIDE [124] and lending club loan [125] datasets for the experiments.

- **Split Learning.** Split learning is computing and memory-efficient variant of FL introduced in [41, 42] where the model is split at a layer and the parts of the model preceding and succeeding this layer are shared across the worker and server, respectively. Only the activations and gradients from a single layer are communicated in split learning, as against that the weights of the entire model are communicated in federated learning. Split learning achieves better communication-efficiency under several settings, as shown in [126]. Applications of this model to wireless edge devices are described in [127, 128]. Split learning also enables matching client-side model components with the best server-side model components for automating model selection as shown in work on ExpertMatcher [129].
- **Federated Neural Architecture Search (FedNAS).** FedNAS [5] is a federated neural architecture search algorithm [130] that can help scattered workers collaboratively searching for a better architecture with higher accuracy in the non-I.I.D. setting. We use this FL algorithm as an example because it differs from other algorithms in that it exchanges information more than the gradient even though it has a centralized topology similar to FedAvg. With `FedML` library, this algorithm can be easily implemented by defining the auxiliary exchanging information such as the architecture parameter in addition to the gradient.

5 Benchmark

Inconsistent usage of datasets, partition methods, and models motivates us to reorganize the benchmark (in Table 7 in the appendix, we summarize the non-I.I.D. datasets and models used in existing publications in top tier conferences of the machine learning community throughout the past two years). To enforce fair comparisons, `FedML` benchmarks not only specify the dataset but also explicitly fix the non-I.I.D. partition method and the model used for experiments. Notably, we divide benchmarking datasets into three categories by three model scales: linear model (convex optimization), shallow neural network (non-convex optimization), and deep neural network.

Table 2: Federated Datasets for Linear Models (Convex Optimization)

Datasets	**# of training samples**	**# of testing samples**	**non-I.I.D. partition method**	**# of clients / devices**	**baseline model**
MNIST	60000	10000	power law	1000	logistic regression
Federated EMINST	671585	77483	realistic partition	3400	logistic regression
Synthetic (α, β)[3]	4305	4672	refer to [3]	30	logistic regression

Federated datasets for linear models (convex optimization). The linear model category is used for convex optimization experiments, such as experiments in [3] and [131]. In this setting, we consider three data sets: MNIST [132], Federated EMINST [4], and Synthetic (α, β) [3], with the logistic regression model as the benchmark. See Table 2 and our source code for details.

Table 3: Federated Datasets for Lightweight Shallow Neural Networks (Non-convex Optimization))

Datasets	**# of training samples**	**# of testing samples**	**partition method**	**# of clients / devices**	**baseline model**
Federated EMINST	671585	77483	realistic partition	3400	CNN (2 Conv + 2 FC)[4]
CIFAR-100	50000	10000	Pachinko Allocation	500	ResNet-18 + group normalization
Shakespeare	16068	2356	realistic partition	715	RNN (2 LSTM + 1 FC)
StackOverflow	135818730	16586035	realistic partition	342477	RNN (1 LSTM + 2 FC)

Federated datasets for lightweight shallow neural networks (non-convex optimization). Due to the resource limitations of the edge devices, shallow neural networks are commonly used for experiments, such as experiments in Adaptive Federated Optimization [4]. In this case, as shown in Table 3, we recommend using the following four datasets as the benchmark:

- **Federated EMNIST**: EMNIST [133] consists of images of digits and upper and lower case English characters, with 62 total classes. The federated version of EMNIST [72] partitions the digits by their author. The dataset has natural heterogeneity stemming from the writing style of each person.

- **CIFAR-100**: Google introduced a federated version of CIFAR-100 [134] by randomly partitioning the training data among 500 clients, with each client receiving 100 examples [4]. The partition method is Pachinko Allocation Method (PAM) [135].
- **Shakespeare**: [66] first introduced this dataset to FL community. It is a dataset built from *The Complete Works of William Shakespeare*. Each speaking role in each play is considered a different device.
- **StackOverflow** [136]: Google TensorFlow Federated (TFF) team maintains this federated dataset, which is derived from the Stack Overflow Data hosted by kaggle.com. We integrate this dataset into our benchmark.

Table 4: Federated Datasets for Deep Neural Networks

Datasets	**# of training samples**	**# of testing samples**	**partition method**	**# of clients / devices**	**baseline model**
CIFAR-10	50,000	10,000	latent Dirichlet allocation	10	ResNet-56, MobileNet
CIFAR-100	50,000	10,000	latent Dirichlet allocation	10	ResNet-56, MobileNet
CINIC-10	90,000	90,000	latent Dirichlet allocation	10	ResNet-56, MobileNet
StackOverflow	135,818,730	10,586,035	realistic partition	342477 (10)	RNN (2 LSTM + 1 FC)
PersonaChat	refer to [137]	refer to [137]	realistic partition	17,568 (16)	GPT2-Small

Federated datasets for deep neural networks. It may be impractical to train a large DNN on resource-constrained edge devices, but it is meaningful to research on large DNN models for the cross-organization FL (also called cross-silo FL). For example, [5] has studied large DNN for FL in the hospital scenario. Another reason that we propose this benchmark within this category is that large DNN models dominate the accuracy in most learning tasks, thus it is more realistic to explore the performance of large DNN in the FL setting. Table 4 shows datasets and models we recommend. The introduction is as follows:

- **CIFAR-10/100.** CIFAR-10/100 [134] consists of 3232 colour images in 10/100 classes. Following [138] and [2], we use latent Dirichlet allocation (LDA) to partition the dataset according to the number of workers involved in training in each round.
- **CINIC-10.** CINIC-10 [139] has 4.5 times as many images as that of CIFAR-10. It is constructed from two different sources: ImageNet and CIFAR-10. It is not guaranteed that the constituent elements are drawn from the same distribution. This characteristic fits for federated learning because we can evaluate how well models cope with samples drawn from similar but not identical distributions.
- **PersonaChat.** Following [140], we use PersonaChat, a chit-chat dataset consisting of conversations between Amazon Mechanical Turk workers, as the benchmark dataset for the large-scale natural language processing task. GPT2-small is a transformer-based model to tackle this task.

6 Experiments

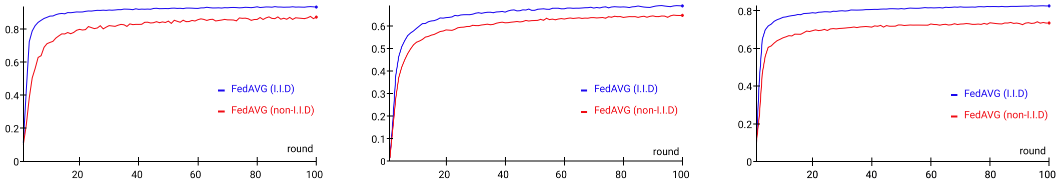
`FedML` provides benchmark experimental results as references for newly developed algorithms and systems. To ensure real-time updates, we maintain benchmark experimental results using `Weight and Bias`[8], which is an online platform that can help to manage and visualize experimental results. The weblink to view the experimental results can found at our GitHub [9].

To demonstrate the capability of FedML, we ran experiments in a real distributed computing environment. We trained large CNN architecture (ResNet and MobileNet) using the standard FedAvg algorithm. Table 5 shows the experiment settings and results, and Figure 5 shows the corresponding curves during training. A common phenomenon is that the accuracy of the non-I.I.D. setting is lower than that of the I.I.D. setting, which is consistent with findings reported in [76].

We also compared the training time of distributed computing with that of standalone simulation. The result in Table 6 reveals that when training large CNNs, the standalone simulation is around 8

[8] `https://www.wandb.com/`

[9] `https://github.com/FedML-AI/FedML/tree/master/benchmark`



(a) ResNet-56 on CIFAR-10 (b) ResNet-56 on CIFAR-100 (c) ResNet-56 on CINIC-10

Figure 5: The Test Accuracy of ResNet-56 (Worker Number = 16)

Table 5: Experimental Results of Training Modern CNNs on Federated Datasets

Dataset	**Non-I.I.D. Partition Method**	**Model**	**Number of Workers**	**Algorithm**	**Acc. on I.I.D**	**Acc. on non-I.I.D**
CIFAR-10	latent Dirichlet allocation	ResNet MobileNet	10	FedAvg FedAvg	93.19 91.12	87.12 ($\downarrow$ 6.07) 86.32 ($\downarrow$ 4.80)
CIFAR-100	latent Dirichlet allocation	ResNet MobileNet	10	FedAvg FedAvg	68.91 55.12	64.70 ($\downarrow$ 4.21) 53.54 ($\downarrow$ 1.58)
CINIC-10	latent Dirichlet allocation	ResNet MobileNet	10	FedAvg FedAvg	82.57 79.95	73.49 ($\downarrow$ 9.08) 71.23 ($\downarrow$ 8.72)

*Note: to reproduce the result, please use the same random seeds we set in the library.

Table 6: FedML's Training Time with FedAvg on Modern CNN architectures (Hardware: 8 x NVIDIA Quadro RTX 5000 GPU (16GB/GPU); RAM: 512G; CPU: Intel Xeon Gold 5220R 2.20GHz).

	ResNet-52	**MobileNet**
number of workers	10	10
single GPU standalone simulation (wall clock time)	> 4 days	> 3 days
multi GPU distributed training (wall clock time)	11 hours	7 hours

*Note that the number of workers can be larger than the number of GPUs because FedML supports multiple processing training in a single GPU.

times slower than distributed computing with 10 parallel workers. Therefore, when training large DNNs, we suggest using FedML's distributed computing paradigm, which is not supported by existing federated learning libraries like PySyft [28], LEAF [72], and TTF [10]. We also see from Table 6 that FedML supports multiprocessing in a single GPU card, which can utilize empty memory in GPUs to support more training workers. This enables FedML accommodate a large number of workers in only a few GPU cards. According to our experiment, when training ResNet on CIFAR-10, FedML can run at most 112 workers in a server with 8 GPUs (the hardware configuration is the same as the configuration described in Table 6).

7 Conclusion

FedML is a research-oriented federated learning library and benchmark. It provides researchers and engineers with an end-to-end toolkit to develop and evaluate their own FL algorithms and fairly compare with existing algorithms. In this work, we describe the system design, new programming interface, application examples, benchmark, dataset, and some experimental results. Its reference implementations, benchmark, and datasets aim to promote the rapid reproduction of baselines and fair comparisons for newly developed algorithms. We accept user feedback and will continuously update our library to support more advanced requirements.

References

[1] Kairouz, P., H. B. McMahan, B. Avent, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[10] https://www.tensorflow.org/federated

[2] Wang, H., M. Yurochkin, Y. Sun, et al. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[3] Li, T., A. K. Sahu, M. Zaheer, et al. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[4] Reddi, S., Z. Charles, M. Zaheer, et al. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[5] He, C., M. Annavaram, S. Avestimehr. Fednas: Federated deep learning via neural architecture search. *arXiv preprint arXiv:2004.08546*, 2020.

[6] Lin, Y., S. Han, H. Mao, et al. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[7] Tang, H., S. Gan, C. Zhang, et al. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems*, pages 7652–7662. 2018.

[8] Tang, H., X. Lian, S. Qiu, et al. Deepsqueeze: Decentralization meets error-compensated compression. *arXiv*, pages arXiv–1907, 2019.

[9] Philippenko, C., A. Dieuleveut. Artemis: tight convergence guarantees for bidirectional compression in federated learning. *arXiv preprint arXiv:2006.14591*, 2020.

[10] Amiri, M. M., D. Gunduz, S. R. Kulkarni, et al. Federated learning with quantized global model updates. *arXiv preprint arXiv:2006.10672*, 2020.

[11] Haddadpour, F., M. M. Kamani, A. Mokhtari, et al. Federated learning with compression: Unified analysis and sharp guarantees. *arXiv preprint arXiv:2007.01154*, 2020.

[12] Tang, Z., S. Shi, X. Chu. Communication-efficient decentralized learning with sparsification and adaptive peer selection. *arXiv preprint arXiv:2002.09692*, 2020.

[13] Hitaj, B., G. Ateniese, F. Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618. 2017.

[14] Yin, D., Y. Chen, K. Ramchandran, et al. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

[15] Zhu, L., Z. Liu, S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, pages 14774–14784. 2019.

[16] Nasr, M., R. Shokri, A. Houmansadr. 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)*, pages 739–753. IEEE, 2019.

[17] Wang, Z., M. Song, Z. Zhang, et al. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2512–2520. IEEE, 2019.

[18] Bhagoji, A. N., S. Chakraborty, P. Mittal, et al. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. 2019.

[19] Fung, C., C. J. Yoon, I. Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[20] Bagdasaryan, E., A. Veit, Y. Hua, et al. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. 2020.

[21] Wei, W., L. Liu, M. Loper, et al. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

[22] Chen, C.-L., L. Golubchik, M. Paolieri. Backdoor attacks on federated meta-learning. *arXiv preprint arXiv:2006.07026*, 2020.

[23] Sun, Z., P. Kairouz, A. T. Suresh, et al. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[24] Enthoven, D., Z. Al-Ars. An overview of federated deep learning privacy attacks and defensive strategies. *arXiv preprint arXiv:2004.04676*, 2020.

[25] Bonawitz, K., V. Ivanov, B. Kreuter, et al. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[26] Geyer, R. C., T. Klein, M. Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[27] Orekondy, T., S. J. Oh, Y. Zhang, et al. Gradient-leaks: Understanding and controlling deanonymization in federated learning. *arXiv preprint arXiv:1805.05838*, 2018.

[28] Ryffel, T., A. Trask, M. Dahl, et al. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017*, 2018.

[29] Melis, L., C. Song, E. De Cristofaro, et al. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[30] Truex, S., N. Baracaldo, A. Anwar, et al. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11. 2019.

[31] Triastcyn, A., B. Faltings. Federated learning with bayesian differential privacy. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2587–2596. IEEE, 2019.

[32] Xu, R., N. Baracaldo, Y. Zhou, et al. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 13–23. 2019.

[33] Triastcyn, A., B. Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 2020.

[34] Hardy, S., W. Henecka, H. Ivey-Law, et al. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[35] Cheng, K., T. Fan, Y. Jin, et al. Secureboost: A lossless federated learning framework. *arXiv preprint arXiv:1901.08755*, 2019.

[36] Yang, S., B. Ren, X. Zhou, et al. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019.

[37] Yang, K., T. Fan, T. Chen, et al. A quasi-newton method based vertical federated learning framework for logistic regression. *arXiv preprint arXiv:1912.00513*, 2019.

[38] Nock, R., S. Hardy, W. Henecka, et al. Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035*, 2018.

[39] Feng, H., Siwei Yu. Multi-participant multi-class vertical federated learning. *arXiv preprint arXiv:2001.11154*, 2020.

[40] Liu, Y., X. Zhang, L. Wang. Asymmetrically vertical federated learning. *arXiv preprint arXiv:2004.07427*, 2020.

[41] Gupta, O., R. Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[42] Vepakomma, P., O. Gupta, T. Swedish, et al. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[43] He, C., C. Tan, H. Tang, et al. Central server free federated learning over single-sided trust social networks. *arXiv preprint arXiv:1910.04956*, 2019.

[44] Lian, X., C. Zhang, H. Zhang, et al. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340. 2017.

[45] Ye, H., L. Luo, Z. Zhou, et al. Multi-consensus decentralized accelerated gradient descent. *arXiv preprint arXiv:2005.00797*, 2020.

[46] Lalitha, A., X. Wang, O. Kilinc, et al. Decentralized bayesian learning over graphs. *arXiv preprint arXiv:1905.10466*, 2019.

[47] Wainakh, A., A. S. Guinea, T. Grube, et al. Enhancing privacy via hierarchical federated learning. *arXiv preprint arXiv:2004.11361*, 2020.

[48] Liao, F., H. H. Zhuo, X. Huang, et al. Federated hierarchical hybrid networks for clickbait detection. *arXiv preprint arXiv:1906.00638*, 2019.

[49] Briggs, C., Z. Fan, P. Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. *arXiv preprint arXiv:2004.11791*, 2020.

[50] Abad, M. S. H., E. Ozfatura, D. Gunduz, et al. Hierarchical federated learning across heterogeneous cellular networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8866–8870. IEEE, 2020.

[51] Luo, S., X. Chen, Q. Wu, et al. Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *arXiv preprint arXiv:2002.11343*, 2020.

[52] Liu, L., J. Zhang, S. Song, et al. Client-edge-cloud hierarchical federated learning. *arXiv preprint arXiv:1905.06641*, 2019.

[53] Jiang, Y., J. Konečnỳ, K. Rush, et al. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[54] Khodak, M., M.-F. F. Balcan, A. S. Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pages 5917–5928. 2019.

[55] Fallah, A., A. Mokhtari, A. Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[56] Jeong, W., J. Yoon, E. Yang, et al. Federated semi-supervised learning with inter-client consistency. *arXiv preprint arXiv:2006.12097*, 2020.

[57] Singh, I., H. Zhou, K. Yang, et al. Differentially-private federated neural architecture search. *arXiv preprint arXiv:2006.10559*, 2020.

[58] Xu, M., Y. Zhao, K. Bian, et al. Neural architecture search over decentralized data. *arXiv preprint arXiv:2002.06352*, 2020.

[59] Hardy, C., E. Le Merrer, B. Sericola. Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 866–877. IEEE, 2019.

[60] Augenstein, S., H. B. McMahan, D. Ramage, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

[61] qiang Liu, Y., Y. Kang, C. Xing, et al. A secure federated transfer learning framework. *The Missouri Review*, pages 1–1, 2020.

[62] Jeong, E., S. Oh, H. Kim, et al. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[63] Li, D., J. Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[64] Sharma, S., C. Xing, Y. Liu, et al. Secure and efficient federated transfer learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2569–2576. IEEE, 2019.

[65] Ahn, J.-H., O. Simeone, J. Kang. Wireless federated distillation for distributed edge learning with heterogeneous data. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6. IEEE, 2019.

[66] McMahan, B., E. Moore, D. Ramage, et al. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. 2017.

[67] Paszke, A., S. Gross, F. Massa, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. 2019.

[68] Abadi, M., P. Barham, J. Chen, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283. 2016.

[69] Chen, T., M. Li, Y. Li, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[70] Sergeev, A., M. Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[71] Peng, Y., Y. Zhu, Y. Chen, et al. A generic communication scheduler for distributed dnn training acceleration. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 16–29. 2019.

[72] Caldas, S., P. Wu, T. Li, et al. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[73] Sahu, A. K., T. Li, M. Sanjabi, et al. On the convergence of federated optimization in heterogeneous networks. *ArXiv*, abs/1812.06127, 2018.

[74] Yang, Q., Y. Liu, Y. Cheng, et al. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[75] Ma, Y., D. Yu, T. Wu, et al. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105–115, 2019.

[76] Hsieh, K., A. Phanishayee, O. Mutlu, et al. The non-iid data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

[77] Hsu, T.-M. H., H. Qi, M. Brown. Federated visual classification with real-world data distribution. *arXiv preprint arXiv:2003.08082*, 2020.

[78] Liu, Y., A. Huang, Y. Luo, et al. Fedvision: An online visual object detection platform powered by federated learning. In *AAAI*, pages 13172–13179. 2020.

[79] Hard, A., K. Rao, R. Mathews, et al. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[80] Leroy, D., A. Coucke, T. Lavril, et al. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6341–6345. IEEE, 2019.

[81] Ge, S., F. Wu, C. Wu, et al. Fedner: Medical named entity recognition with federated learning. *arXiv preprint arXiv:2003.09288*, 2020.

[82] Chen, M., A. T. Suresh, R. Mathews, et al. Federated learning of n-gram language models. *arXiv preprint arXiv:1910.03432*, 2019.

[83] Liu, D., T. Miller. Federated pretraining and fine tuning of bert using clinical notes from multiple silos. *arXiv preprint arXiv:2002.08562*, 2020.

[84] Liu, Y., S. Sun, Z. Ai, et al. Fedcoin: A peer-to-peer payment system for federated learning. *arXiv preprint arXiv:2002.11711*, 2020.

[85] Elbir, A. M., S. Coleri. Federated learning for vehicular networks. *arXiv preprint arXiv:2006.01412*, 2020.

[86] Lim, W. Y. B., J. Huang, Z. Xiong, et al. Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach. *arXiv preprint arXiv:2004.03877*, 2020.

[87] Saputra, Y. M., D. N. Nguyen, D. T. Hoang, et al. Federated learning meets contract theory: Energy-efficient framework for electric vehicle networks. *arXiv preprint arXiv:2004.01828*, 2020.

[88] Liu, Y., J. James, J. Kang, et al. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal*, 2020.

[89] Mirshghallah, F., M. Taram, P. Vepakomma, et al. Privacy in deep learning: A survey. *arXiv preprint arXiv:2004.12254*, 2020.

[90] Yin, F., Z. Lin, Y. Xu, et al. Fedloc: Federated learning framework for data-driven cooperative localization and location data processing. *arXiv preprint arXiv:2003.03697*, 2020.

[91] Chen, C., B. Wu, W. Fang, et al. Practical privacy preserving poi recommendation. *arXiv preprint arXiv:2003.02834*, 2020.

[92] Liang, X., Y. Liu, T. Chen, et al. Federated transfer reinforcement learning for autonomous driving. *arXiv preprint arXiv:1910.06001*, 2019.

[93] Saputra, Y. M., D. T. Hoang, D. N. Nguyen, et al. Energy demand prediction with federated learning for electric vehicle networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[94] Anastasiou, C., J. Lin, C. He, et al. Admsv2: A modern architecture for transportation data management and analysis. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Advances on Resilient and Intelligent Cities*, pages 25–28. 2019.

[95] Rieke, N., J. Hancox, W. Li, et al. The future of digital health with federated learning. *arXiv preprint arXiv:2003.08119*, 2020.

[96] Liu, D., T. Miller, R. Sayeed, et al. Fadl: Federated-autonomous deep learning for distributed electronic health record. *arXiv preprint arXiv:1811.11400*, 2018.

[97] Sheller, M. J., G. A. Reina, B. Edwards, et al. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*, pages 92–104. Springer, 2018.

[98] Ju, C., R. Zhao, J. Sun, et al. Privacy-preserving technology to help millions of people: Federated prediction model for stroke prevention. *arXiv preprint arXiv:2006.10517*, 2020.

[99] Ju, C., D. Gao, R. Mane, et al. Federated transfer learning for eeg signal classification. *arXiv preprint arXiv:2004.12321*, 2020.

[100] Li, W., F. Milletarì, D. Xu, et al. Privacy-preserving federated brain tumour segmentation. In *International Workshop on Machine Learning in Medical Imaging*, pages 133–141. Springer, 2019.

[101] Chen, Y., X. Qin, J. Wang, et al. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 2020.

[102] Flanagan, A., W. Oyomno, A. Grigorievskiy, et al. Federated multi-view matrix factorization for personalized recommendations. *arXiv preprint arXiv:2004.04256*, 2020.

[103] Chen, C., J. Zhang, A. K. Tung, et al. Robust federated recommendation system. *arXiv preprint arXiv:2006.08259*, 2020.

[104] Li, T., L. Song, C. Fragouli. Federated recommendation system via differential privacy. *arXiv preprint arXiv:2005.06670*, 2020.

[105] Qi, T., F. Wu, C. Wu, et al. Fedrec: Privacy-preserving news recommendation with federated learning. *arXiv*, pages arXiv–2003, 2020.

[106] Ribero, M., J. Henderson, S. Williamson, et al. Federating recommendations using differentially private prototypes. *arXiv preprint arXiv:2003.00602*, 2020.

[107] Ammad-Ud-Din, M., E. Ivannikova, S. A. Khan, et al. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

[108] Liu, B., L. Wang, M. Liu, et al. Federated imitation learning: A privacy considered imitation learning framework for cloud robotic systems with heterogeneous sensor data. *arXiv preprint arXiv:1909.00895*, 2019.

[109] Liu, B., L. Wang, M. Liu. Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. *IEEE Robotics and Automation Letters*, 4(4):4555–4562, 2019.

[110] Wang, Z., Y. Yang, Y. Liu, et al. Cloud-based federated boosting for mobile crowdsensing. *arXiv preprint arXiv:2005.05304*, 2020.

[111] Albaseer, A., B. S. Ciftler, M. Abdallah, et al. Exploiting unlabeled data in smart cities using federated learning. *arXiv preprint arXiv:2001.04030*, 2020.

[112] Fredrikson, M., S. Jha, T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. 2015.

[113] Bonawitz, K., V. Ivanov, B. Kreuter, et al. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. 2017.

[114] So, J., B. Guler, A. S. Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *arXiv preprint arXiv:2002.04156*, 2020.

[115] Yu, Q., S. Li, N. Raviv, et al. Lagrange coded computing: Optimal design for resiliency, security, and privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1215–1225. 2019.

[116] Xie, C., K. Huang, P.-Y. Chen, et al. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*. 2019.

[117] Wang, H., K. Sreenivasan, S. Rajput, et al. Attack of the tails: Yes, you really can backdoor federated learning. *arXiv preprint arXiv:2007.05084*, 2020.

[118] Pillutla, K., S. M. Kakade, Z. Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[119] Blanchard, P., R. Guerraoui, J. Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129. 2017.

[120] Yang, Q., Y. Liu, T. Chen, et al. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), 2019.

[121] Hardy, S., W. Henecka, H. Ivey-Law, et al. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.

[122] Liu, Y., Z. Yi, T. Chen. Backdoor attacks and defenses in feature-partitioned collaborative learning. *arXiv e-prints*, arXiv:2007.03608, 2020.

[123] Liu, Y., Y. Kang, X. Zhang, et al. A Communication Efficient Collaborative Learning Framework for Distributed Features. *arXiv e-prints*, arXiv:1912.11187, 2019.

[124] Chua, T.-S., J. Tang, R. Hong, et al. NUS-WIDE: A real-world web image database from National University of Singapore. 2009.

[125] Club, L. Lending club loan data.

[126] Singh, A., P. Vepakomma, O. Gupta, et al. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*, 2019.

[127] Koda, Y., J. Park, M. Bennis, et al. Communication-efficient multimodal split learning for mmwave received power prediction. *IEEE Communications Letters*, 24(6):1284–1288, 2020.

[128] Park, J., S. Samarakoon, M. Bennis, et al. Wireless network intelligence at the edge. *Proceedings of the IEEE*, 107(11):2204–2239, 2019.

[129] Sharma, V., P. Vepakomma, T. Swedish, et al. Expertmatcher: Automating ml model selection for clients using hidden representations. *arXiv preprint arXiv:1910.03731*, 2019.

[130] He, C., H. Ye, L. Shen, et al. Milenas: Efficient neural architecture search via mixed-level reformulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11993–12002. 2020.

[131] Li, X., K. Huang, W. Yang, et al. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[132] LeCun, Y., L. Bottou, Y. Bengio, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[133] Cohen, G., S. Afshar, J. Tapson, et al. Emnist: an extension of mnist to handwritten letters. arxiv e-prints. *arXiv preprint arXiv:1702.05373*, 2017.

[134] Krizhevsky, A., G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[135] Li, W., A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. 2006.

[136] Authors, T. T. F. *TensorFlow Federated Stack Overflow dataset*, 2019.

[137] Wolf, T. *How to build a state-of-the-art conversational ai with transfer learning*, 2020. https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313.

[138] Yurochkin, M., M. Agarwal, S. Ghosh, et al. Bayesian nonparametric federated learning of neural networks. *arXiv preprint arXiv:1905.12022*, 2019.

[139] Darlow, L. N., E. J. Crowley, A. Antoniou, et al. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[140] Rothchild, D., A. Panda, E. Ullah, et al. FetchSGD: Communication-Eficient Federated Learning with Sketching. page 12, 2020.

[141] Mohri, M., G. Sivek, A. T. Suresh. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

[142] Yu, F. X., A. S. Rawat, A. K. Menon, et al. Federated learning with only positive labels. *arXiv preprint arXiv:2004.10342*, 2020.

[143] Karimireddy, S. P., S. Kale, M. Mohri, et al. Scaffold: Stochastic controlled averaging for federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

[144] Malinovsky, G., D. Kovalev, E. Gasanov, et al. From local sgd to local fixed point methods for federated learning. *arXiv preprint arXiv:2004.01442*, 2020.

[145] Li, Z., D. Kovalev, X. Qian, et al. Acceleration for compressed gradient descent in distributed and federated optimization. *arXiv preprint arXiv:2002.11364*, 2020.

[146] Li, T., M. Sanjabi, A. Beirami, et al. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

A Benchmark

A.1 Lack of Fair Comparison: Diverse Non-I.I.D. Datasets and Models

Table 7: various datasets and models used in latest publications from the machine learning community

Conference number	Paper Title	dataset	partition method	model	worker/device
ICML 2019	Analyzing Federated Learning through an Adversarial Lens[18]. IBM.	Fashion-MNIST	natural non-IID	3 layer CNNs	10
		UCI Adult Census datase	-	fully connected neural network	10
ICML 2019	Agnostic Federated Learning[141]. Google.	UCI Adult Census datase	-	logistic regression	10
		Fashion-MNIST	-	logistic regression	10
		Cornell movie dataset	-	two-layer LSTM mode	10
		Penn TreeBank (PTB) dataset	-	two-layer LSTM mode	10
ICML 2019	Bayesian Nonparametric Federated Learning of Neural Networks[138]. IBM.	MNIST	Dir(0.5)	1 hidden layer neural networks	10
		CIFAR10	Dir(0.5)	1 hidden layer neural networks	10
ICML 2020	Adaptive Federated Optimization[4]. Google.	CIFAR-100	Pachinko Allocation Method	ResNet-18	10
		FEMNIST	natural non-IID	CNN (2xconv)	10
		FEMNIST	natural non-IID	Auto Encoder	10
		Shakespeare	natural non-IID	RNN	10
		StackOverflow	natural non-IID	logistic regression	10
		StackOverflow	natural non-IID	1 RNN LSTM	10
ICML 2020	FetchSGD: Communication-Efficient Federated Learning with Sketching[140]. UC Berkeley.	CIFAR-10/100	1 class / 1 client	ResNet-9	-
		FEMNIST	natural non-IID	ResNet-101	-
		PersonaChat	natural non-IID	GPT2-small	-
ICML 2020	Federated Learning with Only Positive Labels[142]. Google.	CIFAR-10	1 class / client	ResNet-8/32	-
		CIFAR-100	1 class / client	ResNet-56	-
		AmazonCAT	1 class / client	Fully Connected Nets	-
		WikiLSHTC	1 class / client	-	-
		Amazon670K	1 class / client	-	-
ICML 2020	SCAFFOLD: Stochastic Controlled Averaging for Federated Learning[143]. EPFL.	EMNIST	1 class / 1 client	Fully connected network	-
ICML 2020	From Local SGD to Local Fixed-Point Methods for Federated Learning[144]. KAUST.	a9a(LIBSVM)	-	Logistic Regression	-
		a9a(LIBSVM)	-	Logistic Regression	-
ICML 2020	Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization[145]. KAUST.	a5a	-	logistic regression	-
		mushrooms	-	logistic regression	-
		a9a	-	logistic regression	-
		w6a LIBSVM	-	logistic regression	-
ICLR 2020	Federated Learning with Matched Averaging[2]. UWM & IBM.	CIFAR-10	-	VGG-9	16
		Shakespheare	sampling 66 clients	1-layer LSTM	66
ICLR 2020	Fair Resource Allocation in Federated Learning[146]. CMU.	Synthetic dataset use LR	natural non-IID	multinomial logistic regression	10
		Vehicle	natural non-IID	SVM for binary classification	10
		Shakespeare	natural non-IID	RNN	10
		Sent140	natural non-IID	RNN	10
ICLR 2020	On the Convergence of FedAvg on Non-IID Data[131]. PKU.	MNIST	natural non-IID	logistic regression	10
		Synthetic dataset use LR	natural non-IID	logistic regression	10
ICLR 2020	DBA: Distributed Backdoor Attacks against Federated Learning[116]. UIUC.	Lending Club Loan Data	-	3 FC	10
		MNIST	-	2 conv and 2 fc	10
		CIFAR-10	-	lightweight Resnet-18	10
		Tiny-imagenet	-	Resnet-18	10
MLSys2020	Federated Optimization in Heterogeneous Networks[73]. CMU.	MNIST	natural non-IID	multinomial logistic regression	10
		FEMNIST	natural non-IID	multinomial logistic regression	10
		Shakespeare	natural non-IID	RNN	10
		Sent140	natural non-IID	RNN	10

*Note: we will update this list once new publications are released.

Table 8: The taxonomy of research areas in federated learning and related publication statistics

Research Areas	Approaches or Sub-problems (# of Papers)	Subtotal
Statistical Challenges	Distributed Optimization (56), Non-IID and Model Personalization (49), Vertical FL (8), Decentralized FL (3), Hierarchical FL (7), Neural Architecture Search (4), Transfer Learning (11), Semi-Supervised Learning (3), Meta Learning (3)	144
Trustworthiness	Preserving Privacy (35), Adversarial Attack (43), Fairness (4), Incentive Mechanism (5)	87
System Challenges	Communication-Efficiency (27), Computation Efficiency (17), Wireless Communication and Cloud Computing (71), FL System Design (19)	134
Models and Applications	Models (22), Natural language Processing (15), Computer Vision (3), Health Care (27), Transportation (13), Other (21)	101
Common	Benchmark and Dataset (20), Survey (7)	27

From a comprehensive FL publication list: https://github.com/chaoyanghe/Awesome-Federated-Learning