`FedML`: A Research Library and Benchmark for Federated Machine Learning

Chaoyang He,[*]   **Songze Li,**   **Jinhyun So,**   **Xiao Zeng,**   **Mi Zhang**
USC   Stanford   USC   MSU   MSU

Hongyi Wang,   Xiaoyang Wang,   Praneeth Vepakomma,   Abhishek Singh,   Hang Qiu
UW-Madison   UIUC   MIT   MIT   USC

Xinghua Zhu, Jianzong Wang,   Li Shen, Peilin Zhao,   Yan Kang, Yang Liu
Ping An Tech.   Tencent   WeBank

Ramesh Raskar,   Qiang Yang,   Murali Annavaram,[*]   Salman Avestimehr[*]
MIT   HKUST   USC   USC

Abstract

Federated learning (FL) is a rapidly growing research field in machine learning. However, existing FL libraries cannot adequately support diverse algorithmic development; inconsistent dataset and model usage make fair algorithm comparison challenging. In this work, we introduce `FedML`, an open research library and benchmark to facilitate FL algorithm development and fair performance comparison. `FedML` supports three computing paradigms: on-device training for edge devices, distributed computing, and single-machine simulation. `FedML` also promotes diverse algorithmic research with flexible and generic API design and comprehensive reference baseline implementations (optimizer, models, and datasets). We hope `FedML` could provide an efficient and reproducible means for developing and evaluating FL algorithms that would benefit the FL research community. We maintain the source code, documents, and user community at https://fedml.ai.

FedML

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: 1) statistical heterogeneity, 2) system constraints, and 3) trustworthiness. 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 field.

In the past few years, more and more efforts have been made to address these unique challenges. To tackle the challenge of statistical heterogeneity, distributed optimization methods such as Adaptive Federated Optimizer [2], FedNova [3], FedProx [4], and FedMA [5] have been proposed. To tackle the challenge of system constraints, researchers apply sparsification and quantization techniques to reduce the communication overheads and computation costs during the training process [6, 7, 8, 9, 10, 11, 12]. To tackle the challenge of trustworthiness, existing research focuses on developing new defense techniques for adversarial attacks to make FL 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].

Although a lot of progress has been made, existing efforts are confronted with a number of limitations that we argue are critical to FL research:

Lack of support of diverse FL computing paradigms. Distributed training libraries in PyTorch [34], TensorFlow [35], MXNet [36], and distributed training-specialized libraries such as Horovod

---

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

Preprint.

Table 1: Comparison between `FedML` and existing federated learning libraries and benchmarks.

		TFF	FATE	PaddleFL	LEAF	PySyft	FedML
Diversified Computing Paradigms	standalone simulation	✓	✓	✓	✓	✓	✓
	distributed computing	✓	✓	✓	✗	✓	✓
	on-device training (Mobile, IoT)	✗	✗	✗	✗	✗	✓
Flexible and Generic API Design	topology customization	✗	✗	✗	✗	✓	✓
	flexible message flow	✗	✗	✗	✗	✗	✓
	exchange message customization	✗	✗	✗	✗	✓	✓
Standardized Algorithm Implementations	FedAvg	✓	✓	✓	✓	✓	✓
	decentralized FL	✗	✗	✗	✗	✗	✓
	FedNAS (beyond gradient/model)	✗	✗	✗	✗	✗	✓
	VFL (vertical federated learning)	✗	✓	✓	✗	✗	✓
	SplitNN (split learning)	✗	✗	✓	✗	✓	✓
Standardized Benchmarks	linear models (e.g., Logistic Regression)	✓	✓	✓	✓	✓	✓
	shallow NN (e.g., Bi-LSTM)	✓	✓	✓	✓	✓	✓
	Model DNN (e.g., ResNet)	✗	✗	✗	✗	✗	✓
	vertical FL	✗	✓	✗	✗	✗	✓

[37] and BytePS [38] are designed for distributed training in data centers. Although simulation-oriented FL libraries such as TensorFlow-Federated (TFF) [39], PySyft [28], and LEAF [40] are developed, they only support centralized topology-based FL algorithms like FedAvg [41] or FedProx [4] with simulation in a single machine, making them unsuitable for FL algorithms which require the exchange of complex auxiliary information and customized training procedure. Production-oriented libraries such as FATE [42] and PaddleFL [43] are released by industry. However, they are not designed as flexible frameworks that aim to support algorithmic innovation for open FL problems.

Lack of support of diverse FL configurations. FL is diverse in network topology, exchanged information, and training procedures. In terms of network topology, a variety of network topologies such as vertical FL [44, 45, 46, 47, 48, 49, 50], split learning [51, 52], decentralized FL [53, 54, 55, 56], hierarchical FL [57, 58, 59, 60, 61, 62], and meta FL [63, 64, 65] have been proposed. In terms of exchanged information, besides exchanging gradients and models, recent FL algorithms propose to exchange information such as pseudo labels in semi-supervised FL [66] and architecture parameters in neural architecture search-based FL [67, 68, 69]. In terms of training procedures, the training procedures in federated GAN [70, 71] and transfer learning-based FL [72, 73, 74, 75, 76] are very different from the vanilla FedAvg algorithm [41]. Unfortunately, such diversity in network topology, exchanged information, and training procedures is not supported in existing FL libraries.

Lack of standardized FL algorithm implementations and benchmarks. The diversity of libraries used for algorithm implementation in existing work makes it difficult to fairly compare their performance. The diversity of benchmarks used in existing work also makes it difficult to fairly compare their performance. The non-I.I.D. characteristic of FL makes such comparison even more challenging [77]: training the same DNN on the same dataset with different non-I.I.D. distributions produces varying model accuracies; one algorithm that achieves higher accuracy on a specific non-I.I.D. distribution than the other algorithms may perform worse on another non-I.I.D. distribution. In Table 8, we summarize the datasets and models used in existing work published at the top tier machine learning conferences such as NeurIPS, ICLR, and ICML in the past two years. We observe that the experimental settings of these work differ in terms of datasets, non-I.I.D. distributions, models, and the number of clients involved in each round. Any difference in these settings could affect the results.

In this work, we present `FedML`, an open research library and benchmark to address the aforementioned limitations and facilitate FL research. `FedML` provides an end-to-end toolkit to facilitate FL algorithm development and fair performance comparison under diverse computing paradigms and configurations. 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 computing paradigms. `FedML` supports three diverse computing paradigms: 1) on-device training for edge devices including smartphones and Internet of Things (IoT), 2) distributed computing, and 3) single-machine simulation to meet algorithmic and system-level research requirements under different system deployment scenarios.

(ii) Support of diverse FL configurations. `FedML` introduces a worker/client-oriented programming interface to enable diverse network topologies, flexible information exchange among workers/clients, and various training procedures.

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

(iv) Standardized FL benchmarks. `FedML` provides standardized benchmarks with well-defined evaluation metrics, multiple synthetic and real-world non-I.I.D. datasets, as well as verified baseline results to facilitate fair performance comparison.

(v) Fully open and evolving. FL is a research field that evolves at a considerably fast pace. This requires `FedML` to adapt at the same pace. We will continuously expand `FedML` to optimize three computing paradigms and support more algorithms (distributed optimizer) and benchmarks (models and datasets) for newly explored usage scenarios. `FedML` is fully open and welcomes contributions from the FL research community as well. We hope researchers in diverse FL applications could contribute more valuable models and realistic datasets to our community. Promising application domains include, but are not limited to, computer vision [78, 79], natural language processing [80, 81, 82, 83, 84], finance [42, 45, 85], transportation [86, 87, 88, 89, 90, 91, 92, 93, 94, 95], digital health [96, 97, 98, 99, 100, 101, 102], recommendation [103, 104, 105, 106, 107, 108], robotics [109, 110], and smart cities [111, 112].
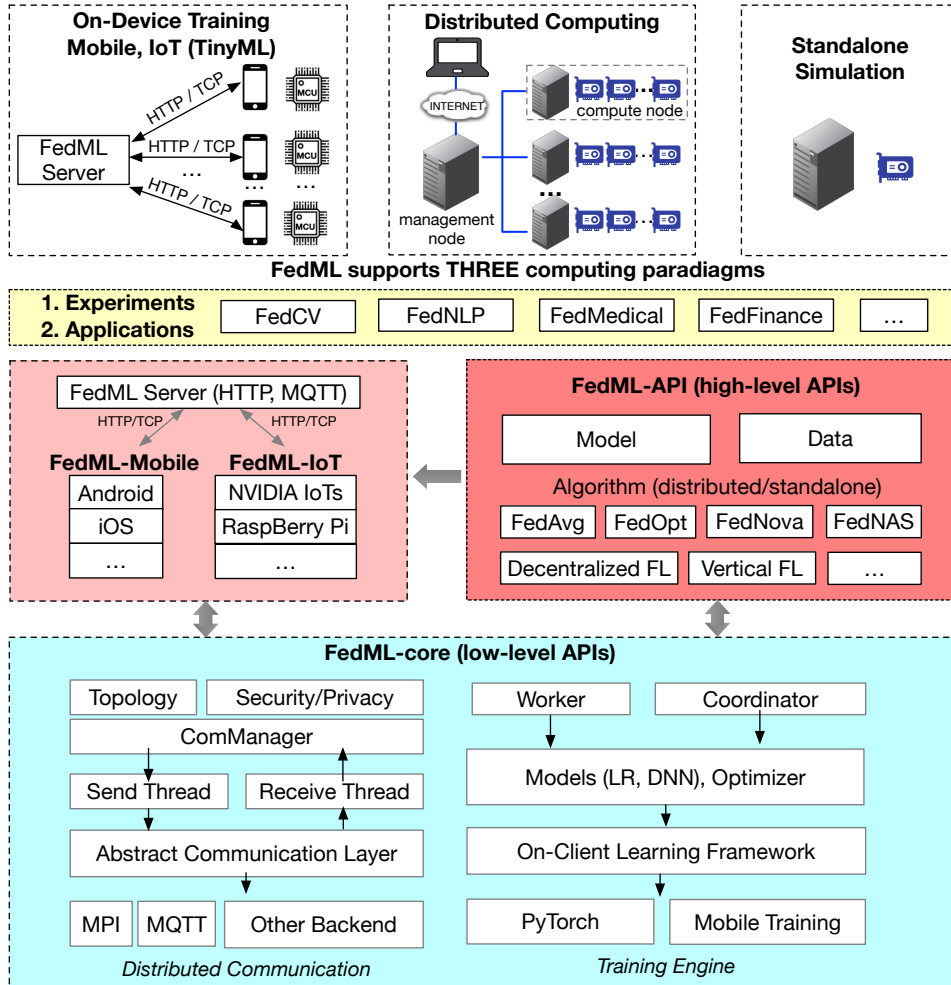

Figure 1: Overview of `FedML` library.

2 `FedML` **Library: Architecture Design**

Figure 1 provides an overview of `FedML` library. The `FedML` library has two key components: `FedML-API` and `FedML-core`, which represents high-level API and low-level API, respectively.

`FedML-core` separates distributed communication and model training into two separate modules. The distributed communication module is responsible for low-level communication among different workers/clients. The communication backend is based on MPI (message passing interface)[2]. Inside the distributed communication module, a `TopologyManager` supports a variety of network topologies that can be used in many existing FL algorithms [53, 54, 55, 56]. In addition, security/privacy-related functions are also supported. The model training module is built upon PyTorch. Users can implement workers (trainers) and coordinators according to their needs.

`FedML-API` is built upon `FedML-core`. With the help of `FedML-core`, new algorithms in distributed version can be easily implemented by adopting the client-oriented programming interface, which is a novel design pattern for flexible distributed computing (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. This distributed computing design is not only used for FL, but it can also be used for conventional in-cluster large-scale distributed training. `FedML-API` also suggests a machine learning system practice that separates the implementations of models, datasets, and algorithms. This practice enables code reuse and 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.

One key feature of `FedML` is its support of FL on real-world hardware platforms. Specifically, `FedML` includes `FedML-Mobile` and `FedML-IoT`, which are two on-device FL testbeds built upon real-world hardware platforms. Currently, `FedML-Mobile` supports Android smartphones and `FedML-IoT` supports Raspberry PI 4 and NVIDIA Jetson Nano (see Appendix C for details). With such testbeds built upon real-world hardware platforms, researchers can evaluate realistic system performance, such as training time, communication, and computation cost. To support conducting experiments on those real-world hardware platforms, our `FedML` architecture design can smoothly transplant the distributed computing code to the `FedML-Mobile` and `FedML-IoT` platforms, reusing nearly all algorithmic implementations in the distributed computing paradigm. Moreover, for `FedML-IoT`, researchers only need to program with Python to customize their research experiments without the need to learn new system frameworks or programming languages (e.g., Java, C/C++)[3].

3 `FedML` **Library: Programming Interface**

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

Worker/client-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[4] 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. In order to make the comparison clearer, we use the most popular machine learning

---

[2]https://pypi.org/project/mpi4py/
[3]Please check here for details: https://github.com/FedML-AI/FedML/tree/master/fedml_iot
[4]More details can be found at https://pytorch.org/tutorials/intermediate/dist_tuto.html

(a) Training procedure-oriented programming

```
1  import torch.distributed as dist
2  ...
3      ...            Training procedure-oriented programming
4    def train():
5      ...
6      model = Net()
7      optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
8      num_batches = ceil(len(train_set.dataset) / float(bsz))
9      for epoch in range(epoch_num):
10       epoch_loss = 0.0
11       for data, target in train_set:
12         optimizer.zero_grad()
13         output = model(data)
14         loss = F.nll_loss(output, target)
15         epoch_loss += loss.item()
16         loss.backward()
17
18         # Distributed Synchronous SGD with all_reduce MPI
19         for param in model.parameters():
20           dist.all_reduce(param.grad.data, op=dist.reduce_op.SUM)
21           param.grad.data /= float(dist.get_world_size())
22
23         optimizer.step()
24
25       print('rank ', dist.get_rank(), ', epoch ',
26           epoch, ': ', epoch_loss / num_batches)
27     ...
```

(b) Worker-oriented programming

```
1  class FedAvgClientManager(WorkerManager):          worker-oriented
2    def __init__(self, ..., topology_manager, trainer):   programming
3      ...
4      self.topology_manager = topology_manager
5      self.trainer = trainer
6
7    def register_message_receive_handlers(self):
8      ...
9      self.register_message_receive_handler(
10           MyMessage.MSG_TYPE_S2C_MSG_A,
11           self.handle_C2S_message_A)
12     ...
13
14   def handle_C2S_message_A(self, msg_params):
15     ...                                              flexible
16                                              message definition
17   def send_message_to_client(self, receive_id, params_B):
18     message = Message(MyMessage.MSG_TYPE_C2S_MSG_B,
19       self.get_sender_id(), receive_id)
20     message.add_params(MyMessage.MSG_ARG_KEY_B, params_B)
21     self.send_message(message)
22
23   def __train__(self):
24     ...                                      topology management
25     params_B = ...
26     for receive_id in self.topology_manager.get_neighbor_ids():
27       self.send_message_to_client(receive_id, params_B)
```
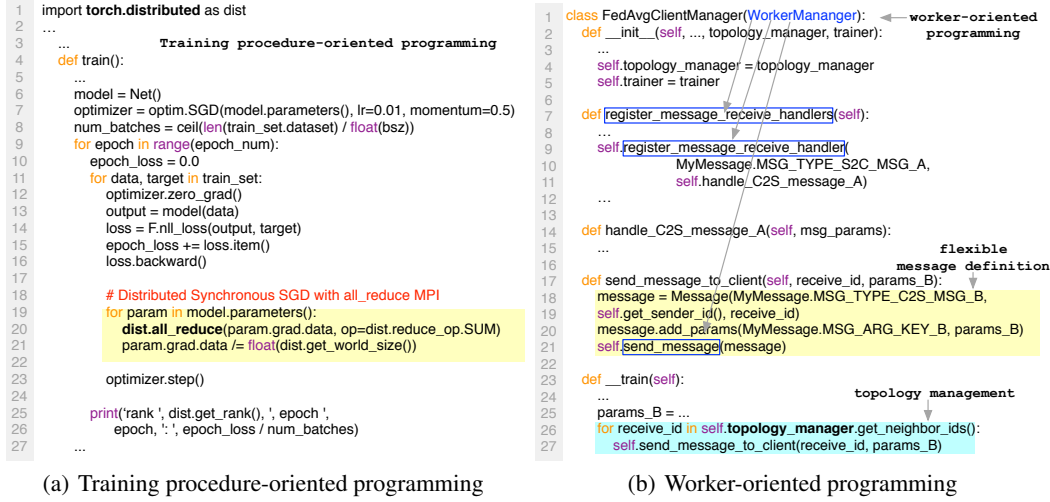
Figure 2: A worker-oriented programming design pattern of FedML.

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[5], also has such inflexibly.

Message definition beyond gradient and model.   FedML also supports 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 using the train() function.
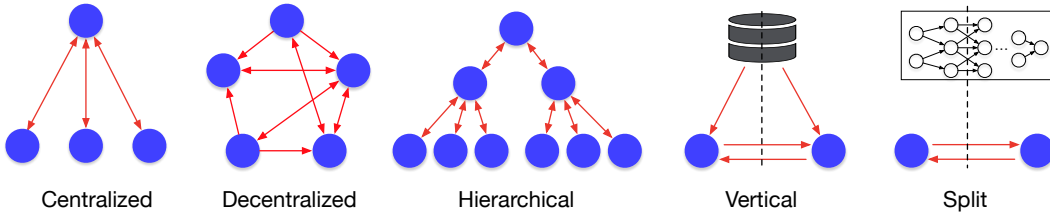

Figure 3: Illustration of various topology definitions in federated learning.

Topology management.   As demonstrated in Figure 3, FL has various topology definitions, such as vertical FL [44, 45, 46, 47, 48, 49, 50], split learning [51, 52], decentralized FL [53, 54, 55, 56], and Hierarchical FL [57, 58, 59, 60, 61, 62]. In order to meet such diverse requirements, FedML provides TopologyManager to manage the topology and allows 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 via 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.

---

[5] 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

Trainer and coordinator. We also need the coordinator to complete the training (e.g., in FedAvg, 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 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 4.1).

Privacy, security, and robustness. While the FL framework facilitates data privacy [90] 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 (e.g., [113]). 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, various cryptography and coding-theoretic approaches have been proposed to manipulate intermediate model data [114, 115].

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) [116]. LCC is a recently developed coding technique on data that achieves optimal resiliency, security (against adversarial nodes), and privacy for any polynomial evaluations on the data. Finally, we plan to provide a sample implementation of the secure aggregation algorithm [114] 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, 117, 118]. Several robust aggregation methods have been proposed to enhance the robustness of FL against adversaries [23, 119, 120].

To accelerate generating benchmark results on new types of adversarial attacks in FL, we include the latest robust aggregation methods presented in literature including (i) norm difference clipping [23]; weak differential private (DP) [23]; (ii) RFA (geometric median) [119]; (iii) KRUM and (iv) MULTI-KRUM [120]. 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 [118] to provide a reference for researchers to develop new attacks.

4 `FedML` **Benchmark: Algorithms, Models, and Datasets**
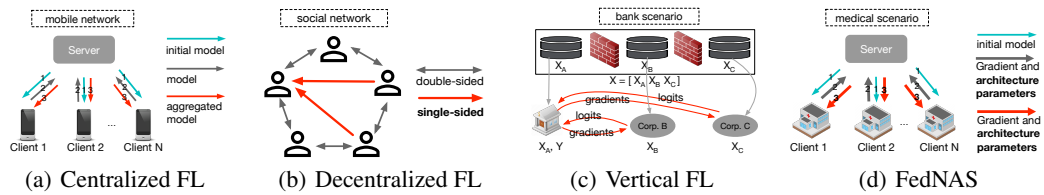
4.1 Algorithms: Federated Optimizer

Figure 4: Supported algorithms that are diverse in network topology, exchanged information, and training procedures.

As shown in Figure 4, `FedML` is capable of supporting FL algorithms that are diverse in network topology, exchanged information, and training procedures. These supported algorithms can be used as implementation examples and baselines to help users develop and evaluate their own algorithms. Currently, `FedML` includes the standard implementations of multiple status quo FL algorithms: Federated Averaging (FedAvg) [41], Decentralized FL [53], Vertical Federated Learning (VFL) [121],

Split learning [51, 52], Federated Neural Architecture Search (FedNAS) [67], and Turbo-Aggregate [115]. Fore more details of these algorithms, please refer to Appendix B.1.

We will keep following the latest algorithm to be published at top-tier machine learning conferences, and will continuously add new FL algorithms such as Adaptive Federated Optimizer [2], FedNova [3], FedProx [4], and FedMA [5] in near future.

4.2 Models and Datasets

Inconsistent usage of datasets, models, and non-I.I.D. partition methods makes it difficult to fairly compare the performance of FL algorithms (in Table 8, we summarize the non-I.I.D. datasets and models used in existing work published at the top tier machine learning venues in the past two years). To enforce fair comparison, FedML benchmark explicitly specifies the combinations of datasets, models, and non-I.I.D. partition methods to be used for experiments. In particular, we divide the benchmark into three categories: 1) linear models (convex optimization), 2) lightweight shallow neural networks (non-convex optimization), and 3) deep neural networks (non-convex optimization).

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 EMNIST	671585	77483	realistic partition	3400	logistic regression
Synthetic (α, β) [122]	4305	4672	refer to [122]	30	logistic regression

Federated datasets for linear models (convex optimization). The linear model category is used for convex optimization experiments such as the ones in [122] and [123]. In this category, we include three datasets (Table 2): MNIST [124], Federated EMNIST [2], and Synthetic (α, β) [122], with the logistic regression as the baseline model.

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 EMNIST	671585	77483	realistic partition	3400	CNN (2 Conv + 2 FC)[2]
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 resource constraints of edge devices, shallow neural networks are commonly used in existing work for experiments. In this category, we include four datasets (Table 3): Federated EMNIST [40], CIFAR-100 [125], Shakespeare [41], and StackOverflow [126]. Please refer to Appendix B.2 for more details.

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)

Federated datasets for deep neural networks (non-convex optimization). Given the resource constraints of edge devices, large DNN models are usually trained under the cross-organization FL (also called cross-silo FL) setting. For example, [67] has studied large DNN models for cross-silo FL in the hospital scenario. However, large DNN models dominate the accuracy in most learning tasks. Pushing FL of large DNN models on edge devices is challenging but a meaningful endeavor, which motivates us to make this benchmark category. For example, [127] has proposed an efficient training algorithm for large CNN models on edge devices. Table 4 shows datasets and models we include in this category. Please refer to Appendix B.2 for more details.

5 Experiments

`FedML` provides benchmark experimental results as references for newly developed FL algorithms. To ensure real-time updates, we maintain benchmark experimental results using `Weight and Bias`[6]. The web link to view all the benchmark experimental results can be found at our GitHub repository.
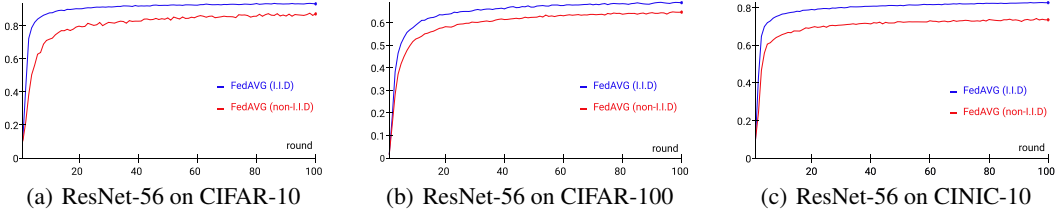

| (a) ResNet-56 on CIFAR-10 | (b) ResNet-56 on CIFAR-100 | (c) ResNet-56 on CINIC-10 |

Figure 5: Test accuracy of ResNet-56 during training.

Table 5: Experimental results of training modern CNNs.

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-56 MobileNet	10	FedAvg FedAvg	93.19 91.12	87.12 ($\downarrow$ 6.07) 86.32 ($\downarrow$ 4.80)
CIFAR-100	Latent Dirichlet Allocation	ResNet-56 MobileNet	10	FedAvg FedAvg	68.91 55.12	64.70 ($\downarrow$ 4.21) 53.54 ($\downarrow$ 1.58)
CINIC-10	Latent Dirichlet Allocation	ResNet-56 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.

To demonstrate the capability of FedML, we ran experiments in a real distributed computing environment. We trained two CNNs (ResNet-56 and MobileNet) using the standard FedAvg algorithm. Table 5 shows the experimental results, and Figure 5 shows the corresponding test accuracy during training. As shown, 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 prior work [77].

Table 6: Training time with FedAvg on modern CNNs (Hardware: 8 x NVIDIA Quadro RTX 5000 GPU (16GB/GPU); RAM: 512G; CPU: Intel Xeon Gold 5220R 2.20GHz).

	ResNet-56	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.

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 about 8 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 FL libraries such as PySyft [28], LEAF [40], and TTF [39]. Moreover, `FedML` supports multiprocessing in a single GPU card which enables `FedML` to run a large number of training workers by using only a few GPU cards. As an example, when training ResNet on CIFAR-10, `FedML` can run 112 workers in a server with 8 GPUs.

6 Conclusion

`FedML` is a research-oriented federated learning library and benchmark. We hope it could provide researchers and engineers with an end-to-end toolkit to facilitate developing FL algorithms and fairly comparing with existing algorithms. We welcome any useful feedback from the readers, and will continuously update `FedML` to support the research of the federated learning community.

---

[6]https://www.wandb.com/

References

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

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

[3] Wang, J., Q. Liu, H. Liang, et al. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

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

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

[35] 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.

[36] 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.

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

[38] 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.

[39] Ingerman, A., K. Ostrowski. *TensorFlow Federated*, 2019.

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

[41] 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.

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

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

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

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

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

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

[52] 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.

[53] 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.

[54] 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.

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

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

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

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

[59] 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.

[60] 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.

[61] 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.

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

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

[64] 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.

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

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

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

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

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

[70] 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.

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

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

[73] 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.

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

[75] 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.

[76] 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.

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

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

[79] 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.

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

[81] 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.

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

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

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

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

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

[87] 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.

[88] 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.

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

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

[91] 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.

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

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

[94] 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.

[95] 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.

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

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

[98] 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.

[99] 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.

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

[101] 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.

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

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

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

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

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

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

[108] 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.

[109] 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.

[110] 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.

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

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

[113] 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.

[114] 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.

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

[116] 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.

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

[118] 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.

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

[120] 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.

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

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

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

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

[125] Krizhevsky, A., G. Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

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

[127] He, C., S. Avestimehr, M. Annavaram. Group knowledge transfer: Collaborative training of large cnns on the edge. *arXiv preprint arXiv:2007.14513*, 2020.

[128] 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.

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

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

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

[132] Kaggle. *Lending Club Loan Data*. https://www.kaggle.com/wendykan/lending-club-loan-data.

[133] 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.

[134] 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.

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

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

[137] 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.

[138] 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.

[139] 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.

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

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

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

[143] Rothchild, D., A. Panda, E. Ullah, et al. Fetchsgd: Communication-efficient federated learning with sketching. *arXiv preprint arXiv:2007.07682*, page 12, 2020.

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

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

[146] 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.

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

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

A The Taxonomy of Research Areas and a Comprehensive Publication List

Table 7: 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

B Benchmark

B.1 Details of Supported Algorithms

Federated Averaging (FedAvg). FedAvg [41] 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 [53], 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 [121] 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 [128, 45, 129, 130]. The `FedML` library currently supports the logistic regression model with customizable local feature extractors in the vertical FL setting, and it provides NUS-WIDE [131] and lending club loan [132] datasets for the experiments.

Split Learning. Split learning is computing and memory-efficient variant of FL introduced in [51, 52] 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 [133]. Applications of this model to wireless edge devices are described in [134, 135]. 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 [136].

Federated Neural Architecture Search (FedNAS). FedNAS [67] is a federated neural architecture search algorithm [137] that enables scattered clients to collaboratively search for a neural architecture. FedNAS differs from other FL algorithms in that it exchanges information beyond gradient even though it has a centralized topology similar to FedAvg.

B.2 Details of Datasets

Federated EMNIST: EMNIST [138] consists of images of digits and upper and lower case English characters, with 62 total classes. The federated version of EMNIST [40] 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 [125] by randomly partitioning the training data among 500 clients, with each client receiving 100 examples [2]. The partition method is Pachinko Allocation Method (PAM) [139].

Shakespeare: [41] 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 [126]: 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.

CIFAR-10 and CIFAR-100. CIFAR-10 and CIFAR-100 [125] both consists of 32×32 colorr images. CIFAR-10 has 10 classes, while CIFAR-100 has 100 classes. Following [140] and [5], 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 [141] 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.

B.3 Lack of Fair Comparison: Diverse Non-I.I.D. Datasets and Models

Table 8: various datasets and models used in latest publications from the machine learning community

Conference	Paper Title	dataset	partition method	model	worker/device number
ICML 2019	Analyzing Federated Learning through an Adversarial Lens [18]	Fashion-MNIST	natural non-IID	3 layer CNNs	10
		UCI Adult Census datase	-	fully connected neural network	10
ICML 2019	Agnostic Federated Learning [142]	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 [140]	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 [2]	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 [143]	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 [144]	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[145]	EMNIST	1 class / 1 client	Fully connected network	-
ICML 2020	From Local SGD to Local Fixed-Point Methods for Federated Learning[146]	a9a(LIBSVM)	-	Logistic Regression	-
		a9a(LIBSVM)	-	Logistic Regression	-
ICML 2020	Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization[147]	a5a	-	logistic regression	-
		mushrooms	-	logistic regression	-
		a9a	-	logistic regression	-
		w6a LIBSVM	-	logistic regression	-
ICLR 2020	Federated Learning with Matched Averaging [5]	CIFAR-10	-	VGG-9	16
		Shakespheare	sampling 66 clients	1-layer LSTM	66
ICLR 2020	Fair Resource Allocation in Federated Learning [148]	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[123]	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[117]	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[4]	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.

C   IoT Devices

Currently, we support two IoT devices: Raspberry PI 4 (Edge CPU Computing) and NVIDIA Jetson Nano (Edge GPU Computing).

C.1   Raspberry Pi 4 (Edge CPU Computing - ARMv7l)

Raspberry Pi 4 Desktop kit is supplied with:

- Raspberry Pi 4 Model B (2GB, 4GB or 8GB version)
- Raspberry Pi Keyboard and Mouse
- 2 × micro HDMI to Standard HDMI (A/M) 1m Cables
- Raspberry Pi 15.3W USB-C Power Supply
- 16GB NOOBS with Raspberry Pi OS microSD card

For more details, please check this link: https://www.raspberrypi.org/products/raspberry-pi-4-desktop-kit.

C.2   NVIDIA Jetson Nano (Edge GPU Computing)

NVIDIA® Jetson Nano™ Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts.

For more details, please check this link: https://developer.nvidia.com/embedded/jetson-nano-developer-kit.