

# Decentralized Learning Made Easy With DecentralizePy

Akash Dhasade

Anne-Marie Kermarrec

Rafael Pires

**Rishi Sharma**

Milos Vujasinovic

Scalable Computing Systems Laboratory  
EPFL

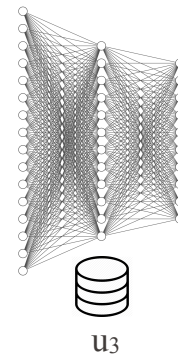
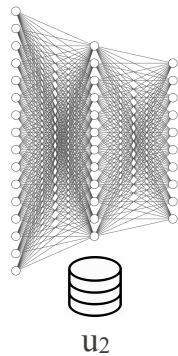
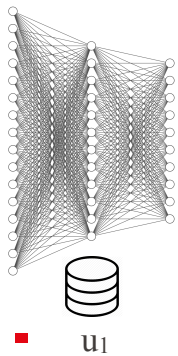
A **framework** for designing and studying **decentralized learning systems**.

Rapid development

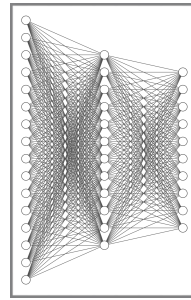
Scalability

▪

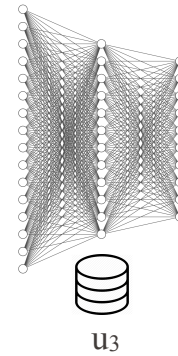
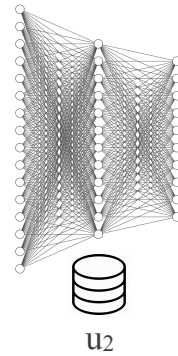
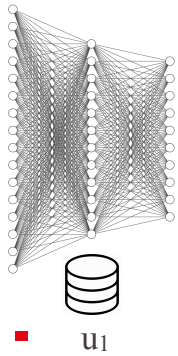




# EPFL Federated learning

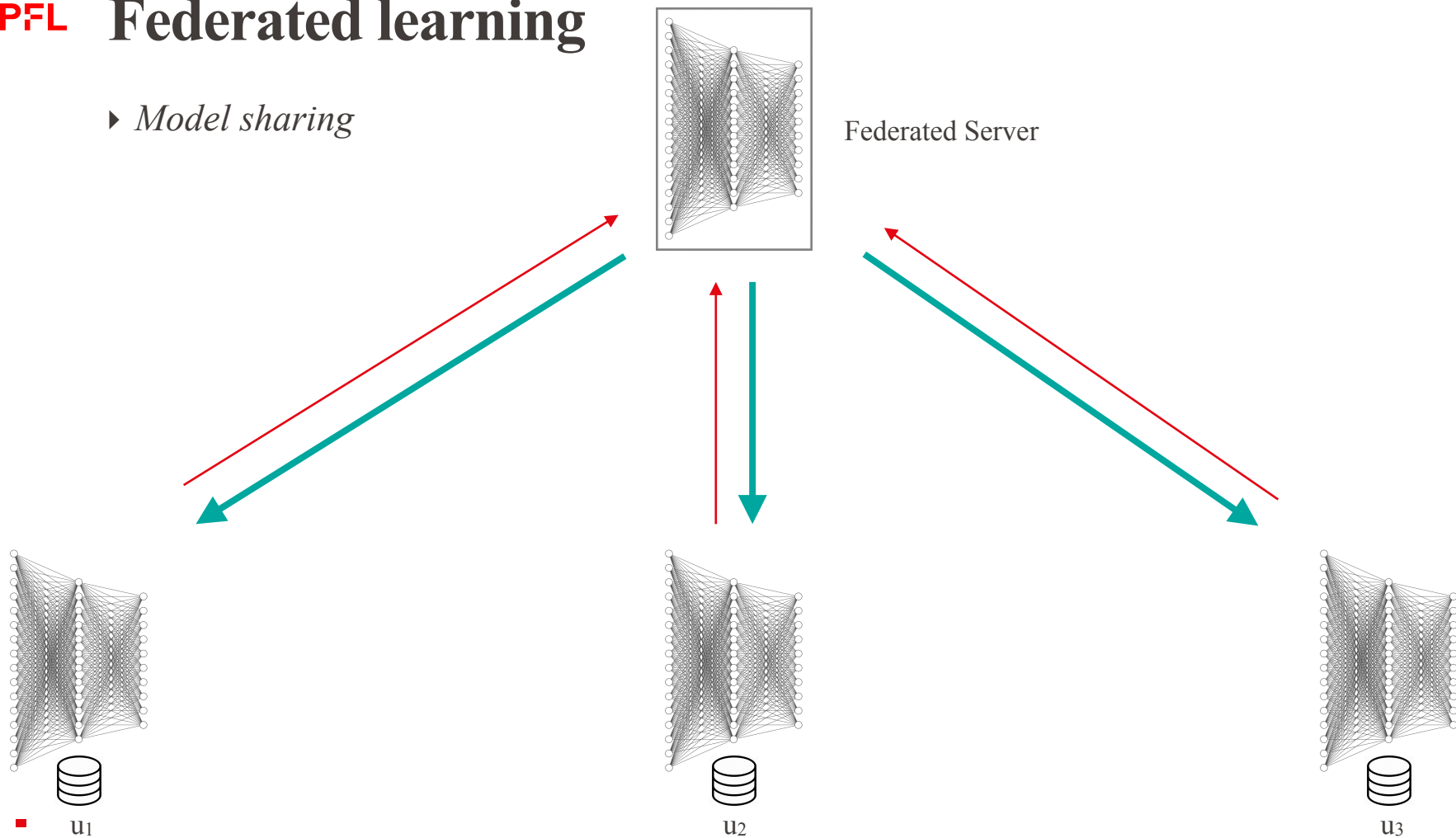


Federated Server



# EPFL Federated learning

► *Model sharing*



# EPFL So many frameworks...



## FedScale

A scalable and extensible federated learning engine and benchmark

GET STARTED

FedScale is a scalable and extensible [open-source](#) federated learning (FL) engine. It provides high-level APIs to implement FL algorithms, deploy and evaluate them at scale across diverse hardware and software backends. FedScale also includes the largest [FL benchmark](#) that contains FL tasks ranging from image classification and object detection to language modeling and speech recognition. Moreover, it includes datasets to faithfully emulate [FL runtime environments](#) where FL solutions will realistically be deployed.



We are actively developing FedScale, and welcome contributions from community. [Join our slack](#) to keep up to date.

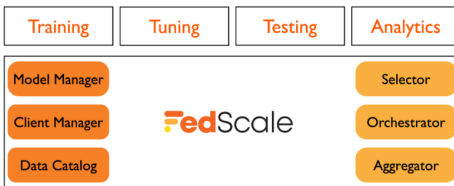
What's new? [Flower Next Pilot Program](#) >

## Flower A Friendly Federated Learning Framework

A unified approach to federated learning, analytics, and evaluation. Federate any workload, any ML framework, and any programming language.

Take the tutorial

to learn federated learning



## TensorFlow Federated: Machine Learning on Decentralized Data

TensorFlow Federated (TFF) is an open-source framework for machine learning and other computations on decentralized data. TFF has been developed to facilitate open research and experimentation with [Federated Learning \(FL\)](#), an approach to machine learning where a shared global model is trained across many participating clients that keep their training data locally. For example, FL has been used to train [prediction models for mobile keyboards](#) without uploading sensitive typing data to servers.

TFF enables developers to simulate the included federated learning algorithms on their models and data, as well as to experiment with novel algorithms. Researchers will find [starting points and complete examples](#) for many kinds of research. The building blocks provided by TFF can also be used to implement non-learning computations, such as [federated analytics](#). TFF's interfaces are organized in two main layers:

- > **Federated Learning (FL) API**  
 This layer offers a set of high-level interfaces that allow developers to apply the included implementations of federated training and evaluation to their existing TensorFlow models.
- > **Federated Core (FC) API**  
 At the core of the system is a set of lower-level interfaces for concisely expressing novel federated algorithms by combining TensorFlow with distributed communication operators within a strongly-typed functional programming environment. This layer also serves as the foundation upon which we've built Federated Learning.

TFF enables developers to declaratively express federated computations, so they could be deployed to diverse runtime environments. Included with TFF is a performant multi-machine simulation runtime for experiments. Please visit the [tutorials](#) and try it out yourself!

For questions and support, find us at the [tensorflow-federated](#) tag on StackOverflow.

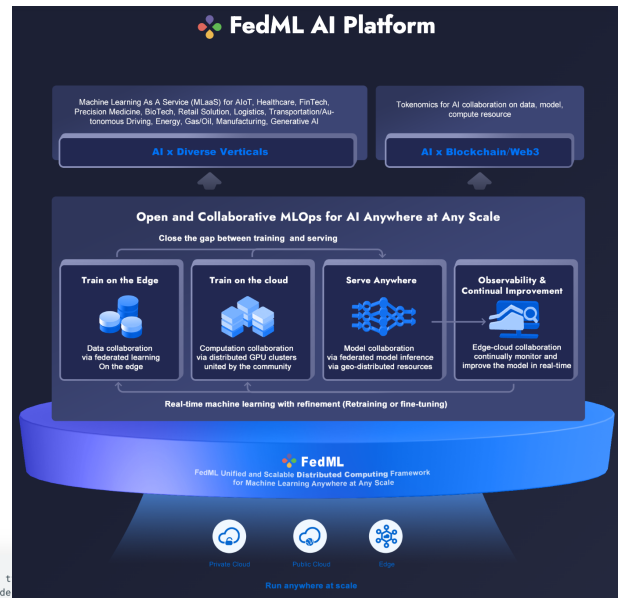
```
import tensorflow as tf
import tensorflow_federated as tff

# Load simulation data.
source, _ = tff.simulation.datasets.emnist.load_data()
def client_data(n):
    return source.create_tf_dataset_for_client(source.client_ids[n]).map(
        lambda e: (tf.reshape(e['pixels'], [-1]), e['label'])
    ).repeat(10).batch(20)

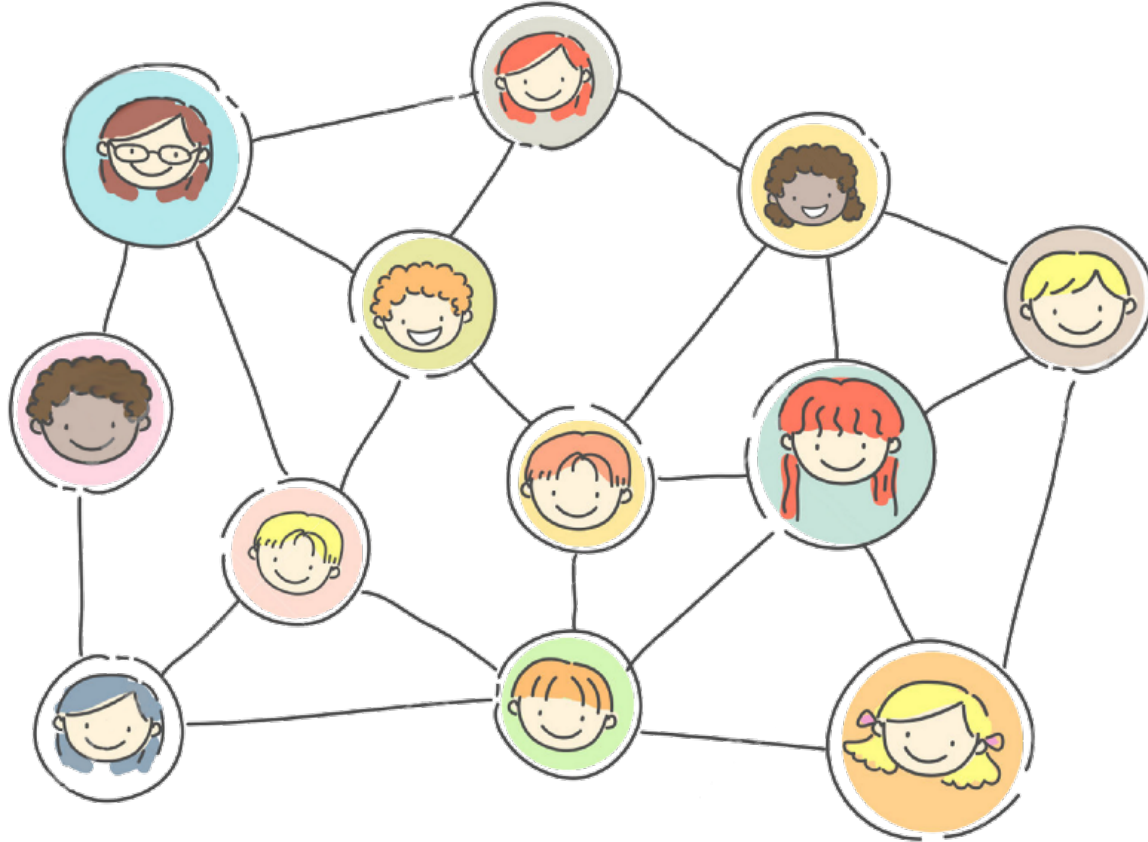
# Pick a subset of client devices to participate in training.
train_data = [client_data(n) for n in range(3)]

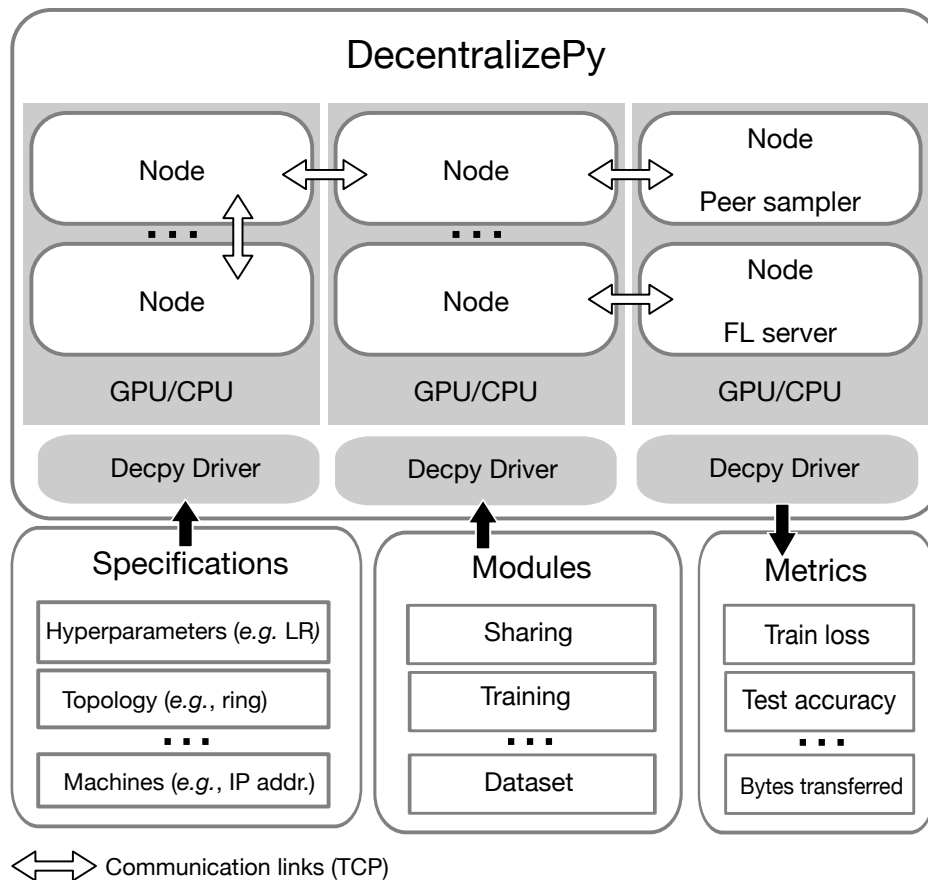
# Wrap a Keras model for use with TFF.
def model_fn():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(10, tf.nn.softmax, input_shape=(784,),
            kernel_initializer='zeros')
    ])
    return tff.learning.models.from_keras_model(
        model,
        input_spec=train_data[0].element_spec,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(),
        metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])

# Simulate a few rounds of training with the selected client devices.
trainer = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn,
    client_optimizer_fn=lambda: tf.keras.optimizers.SGD(0.1))
state = trainer.initialize()
for _ in range(5):
```

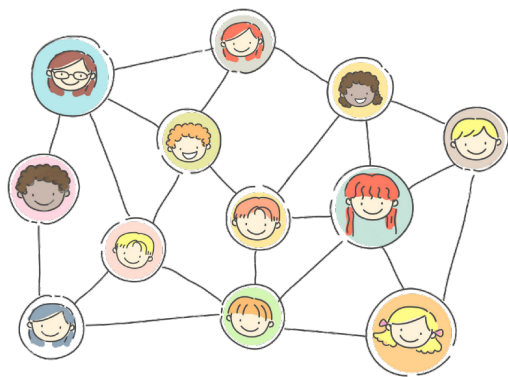




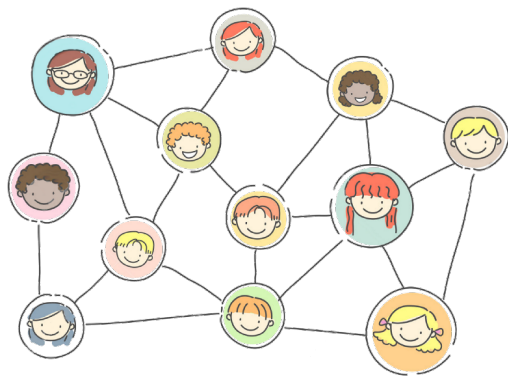








Topology

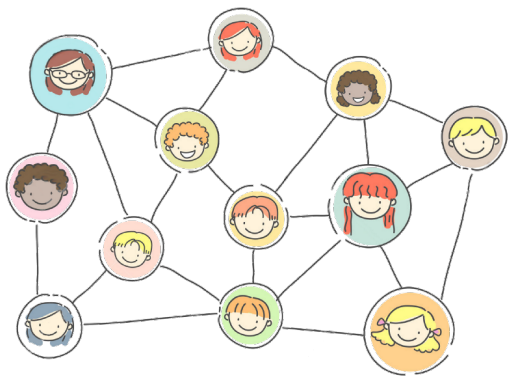


Topology



Communication

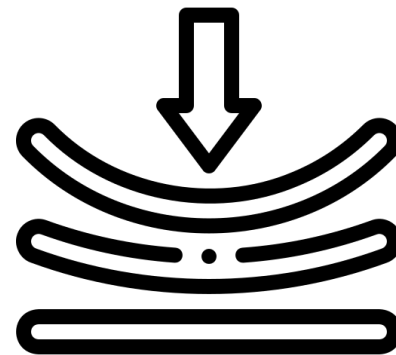
# EPFL Building decentralized learning systems



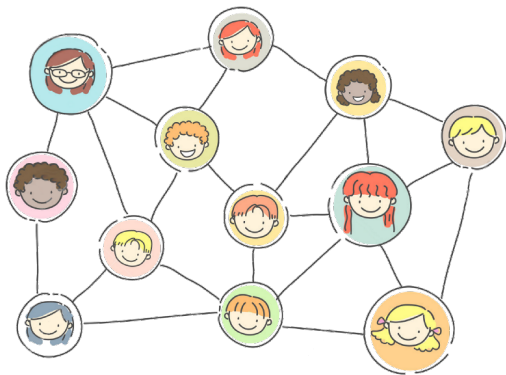
Topology



Communication



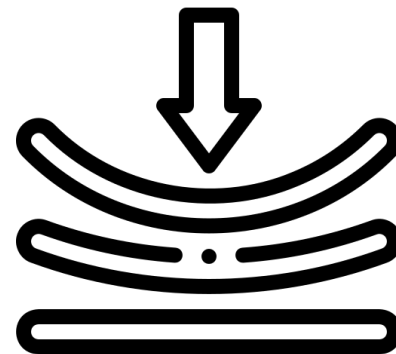
Compression



Topology



Communication

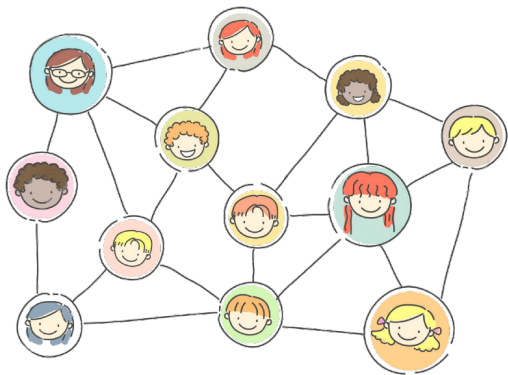


Compression



Roles

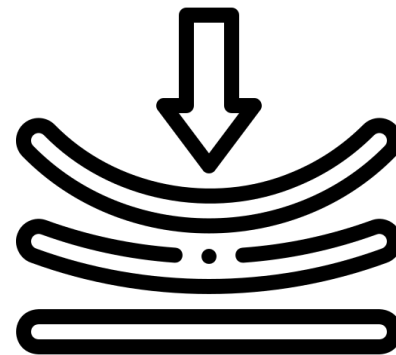




Topology



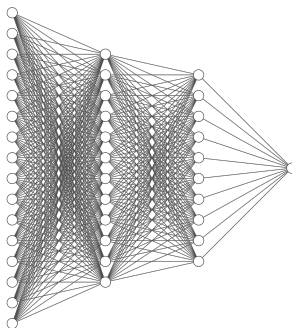
Communication



Compression



Roles

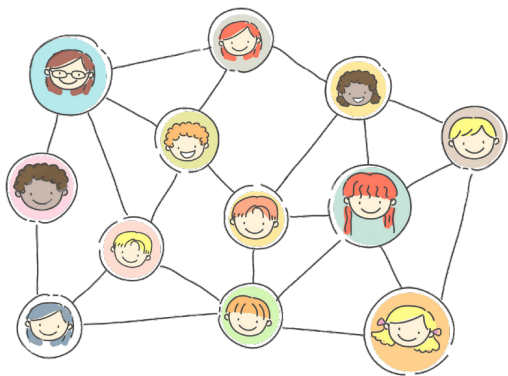


Models





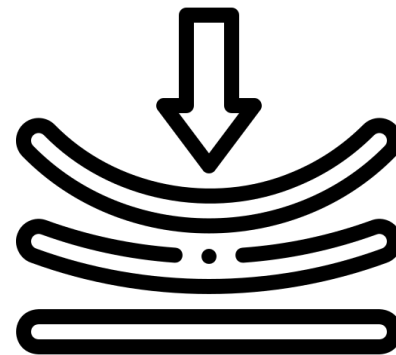
# EPFL Building decentralized learning systems



Topology



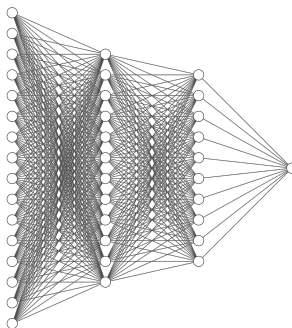
Communication



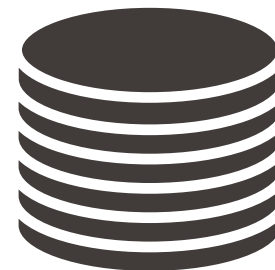
Compression



Roles

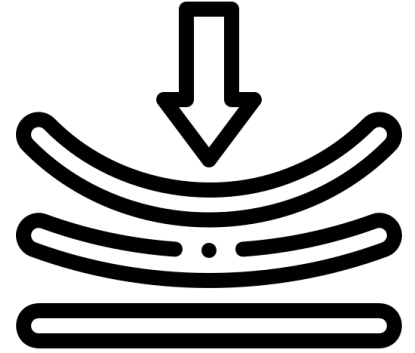
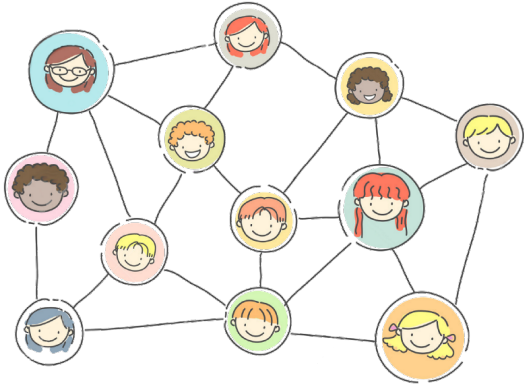


Models

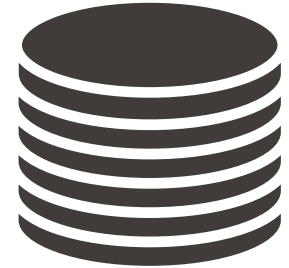
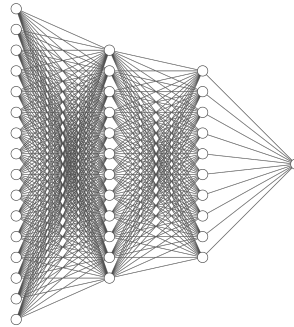
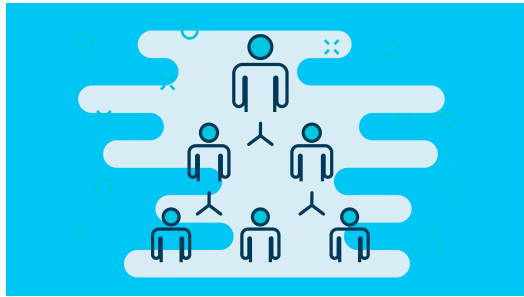


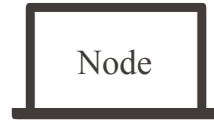
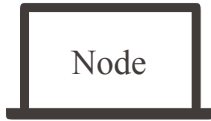
Datasets

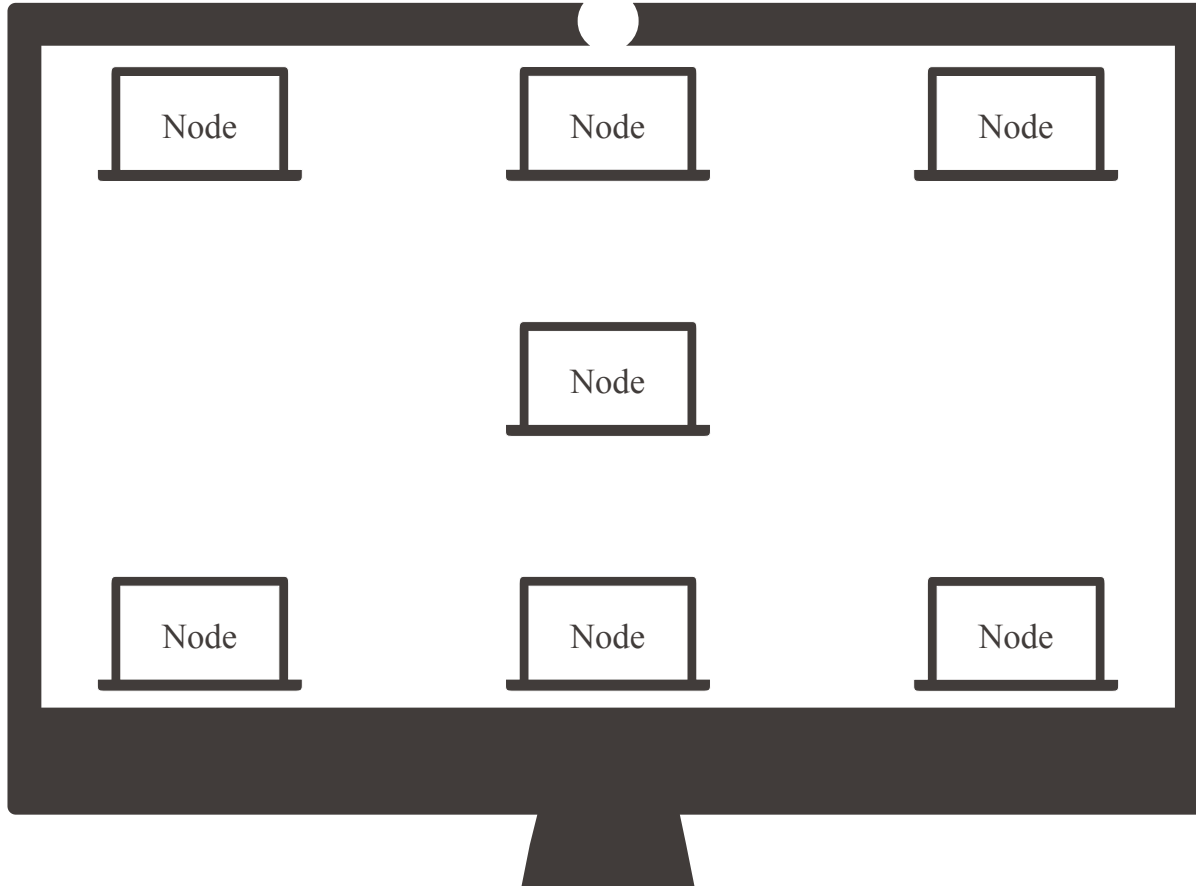




**(DecentralizePy Modules)**







```
1 from decentralizepy.node.Node import Node
2
3 class DLNode(Node):
4     def run(self, iterations, training, dataset,
5             sharing, graph, communication):
6         for round in range(iterations):
7             training.train(dataset)
8             msg = sharing.get_message()
9             neighbors = graph.get_neighbors()
10            communication.send(neighbors, msg)
11            rcv = communication.receive_from_all()
12            sharing.average(rcv)
13            dataset.test()
```

```
1 from decentralizepy.node.Node import Node
2
3 class DLNode(Node):
4     def run(self, iterations, training, dataset,
5             sharing, graph, communication):
6         for round in range(iterations):
7             training.train(dataset)
8             msg = sharing.get_message()
9             neighbors = graph.get_neighbors()
10            communication.send(neighbors, msg)
11            rcv = communication.receive_from_all()
12            sharing.average(rcv)
13            dataset.test()
```

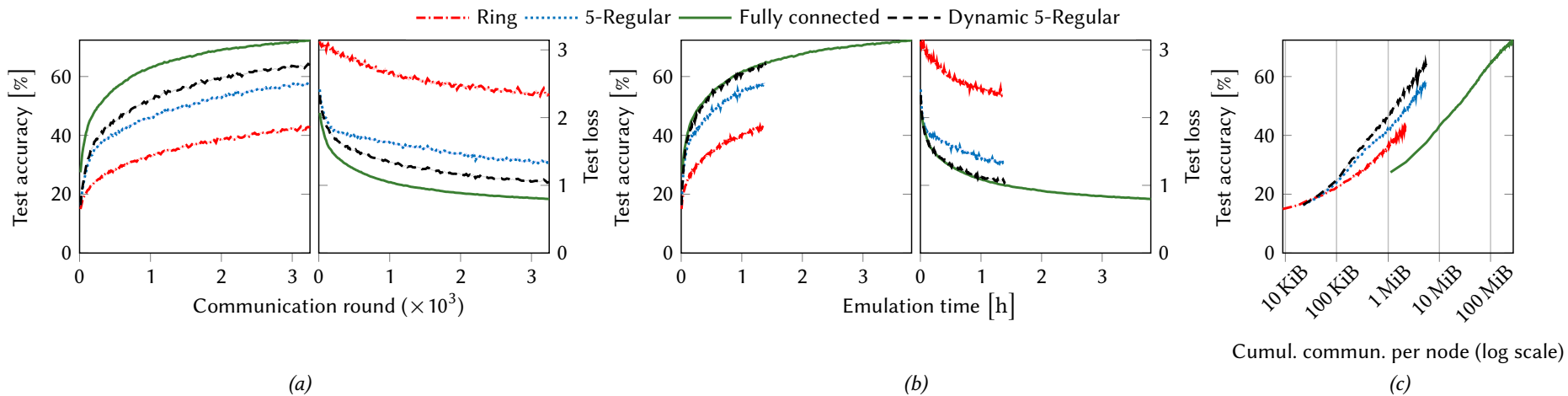
DecentralizePy already contains reference implementations of well-known algorithms.

We use DecentralizePy as a **catalyst** for DL research in our lab.

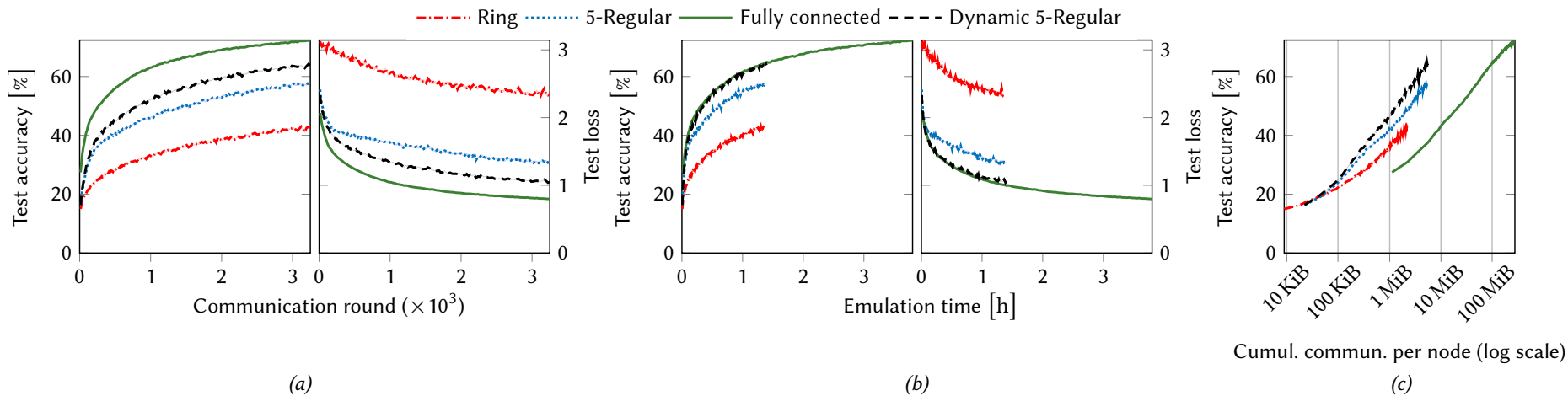
- ◆ CIFAR-10 (Non-IID) with GN-LeNet
- ◆ 256 and 1024 DL nodes
- ◆ Emulation on 16 machines
- ◆ D-PSGD with Metropolis Hastings



(256-nodes)



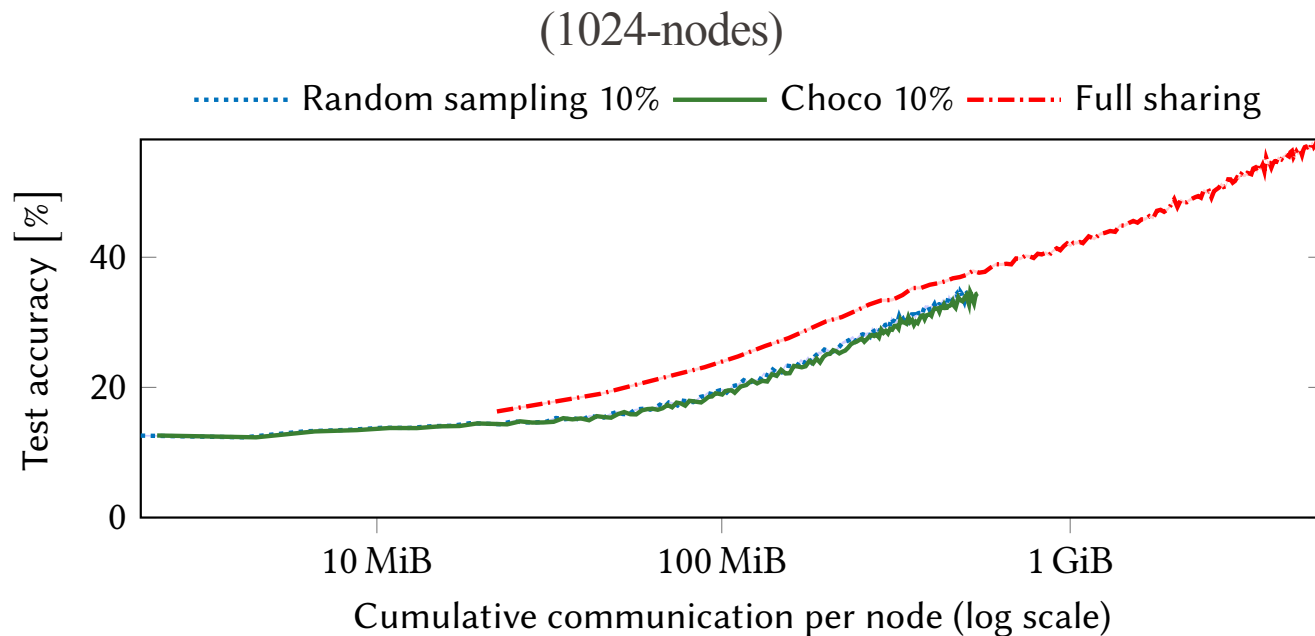
(256-nodes)

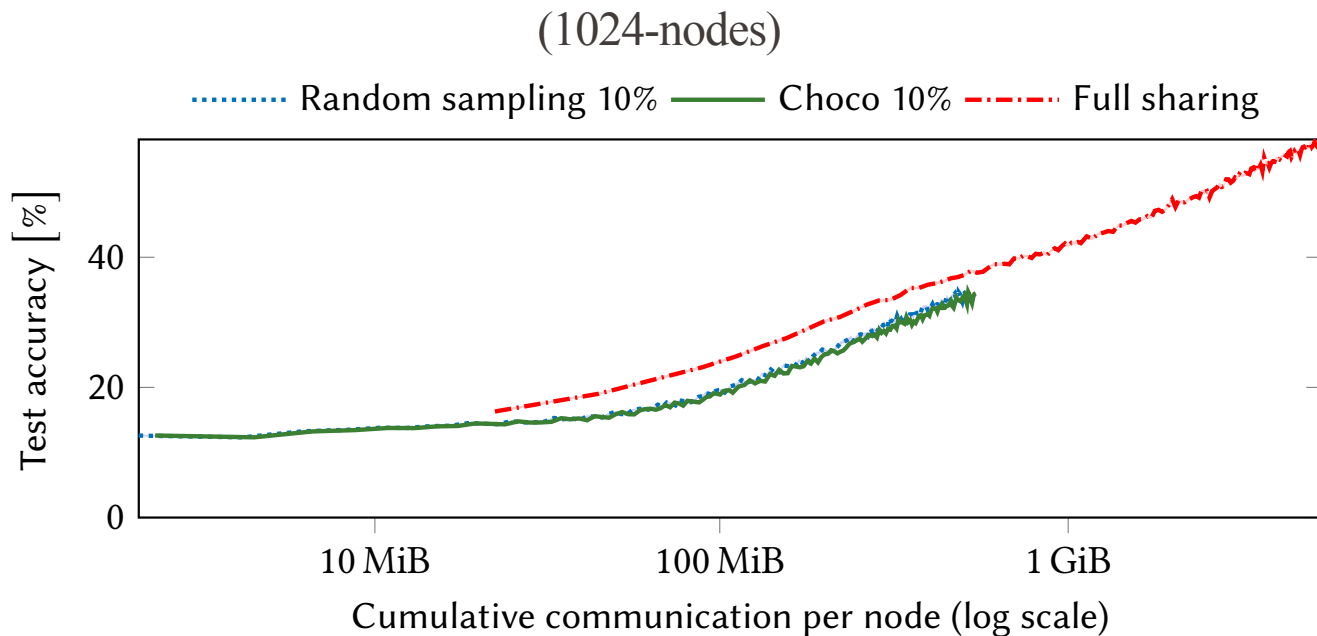


Information spreads faster through the network with dynamic topologies.



```
1 from decentralizepy.node.Node import Node
2
3 class DLNode(Node):
4     def run(self, iterations, training, dataset,
5             sharing, graph, communication):
6         for round in range(iterations):
7             training.train(dataset)
8             msg = sharing.get_message()
9             neighbors = graph.get_neighbors()
10            communication.send(neighbors, msg)
11            rcv = communication.receive_from_all()
12            sharing.average(rcv)
13            dataset.test()
```





The loss of information due to compression dramatically effects the convergence in non-IID settings at scale.

```
1 from decentralizepy.node.Node import Node
2
3 class DLNode(Node):
4     def run(self, iterations, training, dataset,
5             sharing, graph, communication):
6         for round in range(iterations):
7             training.train(dataset)
8             msg = sharing.get_message()
9             neighbors = graph.get_neighbors()
10            communication.send(neighbors, msg)
11            rcv = communication.receive_from_all()
12            sharing.average(rcv)
13            dataset.test()
```



<https://github.com/sacs-epfl/decentralizepy>

- ◆ Open source
- ◆ Already being used for a number of projects
- ◆ Adding new algorithms



<https://github.com/sacs-epfl/decentralizepy>



- ◆ Open source
- ◆ Already being used for a number of projects
- ◆ Adding new algorithms
- ◆ Realistic network emulations
- ◆ Peer-sampling and availability traces



<https://github.com/sacs-epfl/decentralizepy>

The screenshot shows the GitHub repository for DecentralizePy. At the top, the repository name 'sacs-epfl / decentralizepy' is displayed, along with navigation options like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The repository is public and has 192 commits, 8 stars, 0 watchers, and 2 forks. The main content area shows a file browser with a list of files and folders, including 'eval', 'src/decentralizepy', 'tutorial', '.gitignore', '.isort.cfg', 'LICENSE', 'README.rst', 'download\_dataset.py', 'generate\_graph.py', 'install\_nMachines.sh', 'pyproject.toml', 'requirements.txt', 'setup.cfg', 'setup.py', and 'split\_into\_files.py'. Each file entry includes a description of the file's purpose and the time since it was last updated. Below the file browser, the 'README.rst' file is open, showing the project's description: 'decentralizepy is a framework for running distributed applications (particularly ML) on top of arbitrary topologies (decentralized, federated, parameter server). It was primarily conceived for assessing scientific ideas on several aspects of distributed learning (communication efficiency, privacy, data heterogeneity etc.).' The EPFL logo is visible in the top right corner of the README. On the right side of the repository page, there are sections for 'About', 'Releases', 'Packages', 'Contributors', and 'Languages'. The 'About' section describes it as a 'decentralized learning research framework'. The 'Releases' section indicates that no releases have been published. The 'Packages' section indicates that no packages have been published. The 'Contributors' section lists four contributors: rishi-s8, rafaelpires, sissiki, and mvujas. The 'Languages' section shows a bar chart indicating that the repository is primarily Python (88.2%) with some Shell (11.8%).

sacs-epfl / decentralizepy Public

Edit Pins Watch 0 Fork 2 Starred 8

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

rishi-s8 Add script to generate graph 8ae8221 4 hours ago 192 commits

eval	Add dataset download   Update tutorial	20 hours ago
src/decentralizepy	Add script to generate graph	4 hours ago
tutorial	Add script to generate graph	4 hours ago
.gitignore	Add dataset download   Update tutorial	20 hours ago
.isort.cfg	Initial Commit	2 years ago
LICENSE	Add license	3 months ago
README.rst	Add script to generate graph	4 hours ago
download_dataset.py	Add dataset download   Update tutorial	20 hours ago
generate_graph.py	Add script to generate graph	4 hours ago
install_nMachines.sh	6 machine, move to eval	2 years ago
pyproject.toml	Initial Commit	2 years ago
requirements.txt	Initial Commit	2 years ago
setup.cfg	Add peer sampler, refactor everything	10 months ago
setup.py	Modify Data and Dataset, add barebone Node, structure config.ini	2 years ago
split_into_files.py	Reddit	last year

README.rst

## decentralizepy

decentralizepy is a framework for running distributed applications (particularly ML) on top of arbitrary topologies (decentralized, federated, parameter server). It was primarily conceived for assessing scientific ideas on several aspects of distributed learning (communication efficiency, privacy, data heterogeneity etc.).

EPFL

### About

A decentralized learning research framework

- Readme
- MIT license
- 8 stars
- 0 watching
- 2 forks

Report repository

### Releases

No releases published  
[Create a new release](#)

### Packages

No packages published  
[Publish your first package](#)

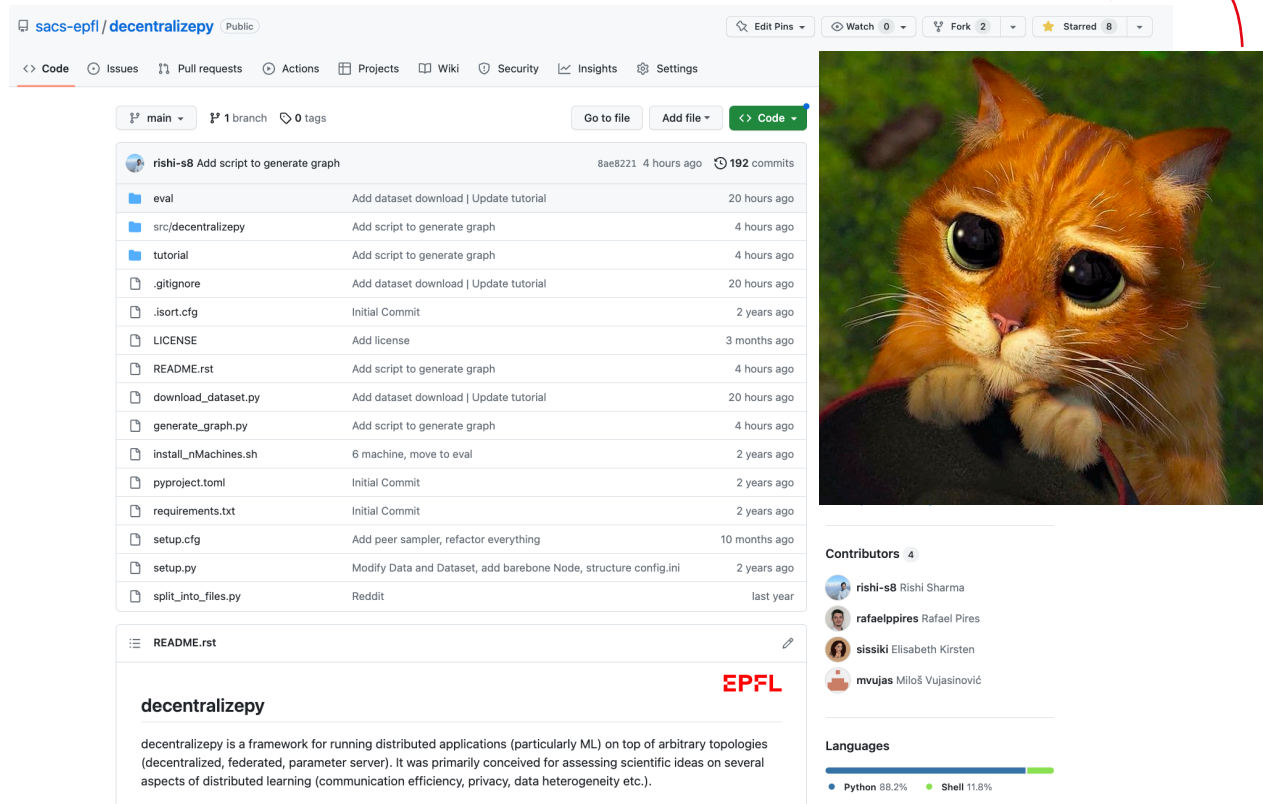
### Contributors 4

- rishi-s8 Rishi Sharma
- rafaelpires Rafael Pires
- sissiki Elisabeth Kirsten
- mvujas Miloš Vujaninović

### Languages

Python 88.2% Shell 11.8%

Please use DecentralizePy if you are working with decentralized learning and help us improve the framework.



sacs-epfl / decentralizepy Public

Edit Pins Watch 0 Fork 2 Starred 8

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

rishi-s8 Add script to generate graph 8ae8221 4 hours ago 192 commits

eval	Add dataset download   Update tutorial	20 hours ago
src/decentralizepy	Add script to generate graph	4 hours ago
tutorial	Add script to generate graph	4 hours ago
.gitignore	Add dataset download   Update tutorial	20 hours ago
.isort.cfg	Initial Commit	2 years ago
LICENSE	Add license	3 months ago
README.rst	Add script to generate graph	4 hours ago
download_dataset.py	Add dataset download   Update tutorial	20 hours ago
generate_graph.py	Add script to generate graph	4 hours ago
install_nMachines.sh	6 machine, move to eval	2 years ago
pyproject.toml	Initial Commit	2 years ago
requirements.txt	Initial Commit	2 years ago
setup.cfg	Add peer sampler, refactor everything	10 months ago
setup.py	Modify Data and Dataset, add barebone Node, structure config.ini	2 years ago
split_into_files.py	Reddit	last year

README.rst

## decentralizepy

decentralizepy is a framework for running distributed applications (particularly ML) on top of arbitrary topologies (decentralized, federated, parameter server). It was primarily conceived for assessing scientific ideas on several aspects of distributed learning (communication efficiency, privacy, data heterogeneity etc.).

**EPFL**

### Contributors 4

- rishi-s8 Rishi Sharma
- rafaelpires Rafael Pires
- sissiki Elisabeth Kirsten
- mvujas Miloš Vujašinović

### Languages

Python 88.2% Shell 11.8%

Please use DecentralizePy if you are working with decentralized learning and help us improve the framework.

- ◆ Open source
- ◆ Already being used for a number of projects
- ◆ Adding new algorithms
- ◆ Realistic network emulations
- ◆ Peer-sampling and availability traces



<https://github.com/sacs-epfl/decentralizepy>