Tutorial 6: Practical introduction to Federated Learning

Jules Fasquelle, CHUV

Colab Notebooks (hidden until workshope date)

Available on **Github / CHUV-DS / Federated_Learning_Workshop**

(link also provided by email along with the PIMA dataset)

Local deployment (OPTIONAL but more stable)

If you are familiar with Jupyter notebooks or have access to a Unix (Mac/Linux) system, follow these instructions to run a local Jupyter server:

If you are not familiar with pip/conda/Jupyter, or if you are using Windows, I do not recommend installing this. Skip this slide and the next.

https://research.google.com/colaboratory/local-runtimes.html

You can then either run the Colab Notebooks by connecting them to your Jupyter server (see next slide), or open the notebooks directly with the Jupyter viewer.

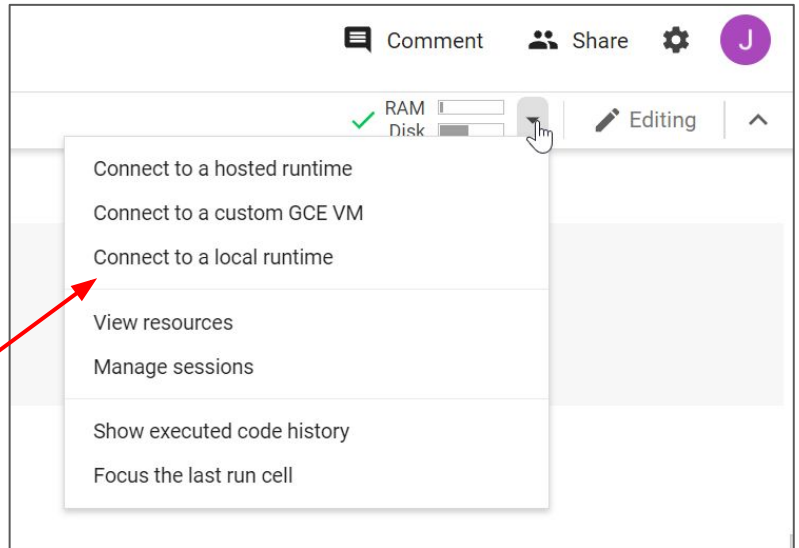
Connecting to your own server will allow you to install the dependencies only once, and to avoid relying on a remote Google server.

Connect Colab Notebooks locally (OPTIONAL, see slide 3)

Once your Jupyter server is running, you can see in your bash something like this:

```
To access the notebook, open this file in a browser:
    file:///home/jules/.local/share/jupyter/runtime/nbserver-78959-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=a73345a9d9b013f4ac76df6fe3aeb710921a3253b6e1fe5b
 or http://127.0.0.1:8888/?token=a73345a9d9b013f4ac76df6fe3aeb710921a3253b6e1fe5b
```

- Copy the URL containing **localhost**
- **Go to your Colab notebook**, in the top right click the menu as shown here
- Select “Connect to a **local** runtime”
- Paste your link, click **Connect**.



Opening the notebooks and loading the dataset

If you are running on a **local environment**, write “loopback=True” in all the `sy.launch_duet` and `sy.join_duet` arguments.

In both **Data Owner** notebooks, load the PIMA dataset (csv file) by using the folder icon on the left bar.

The PIMA dataset

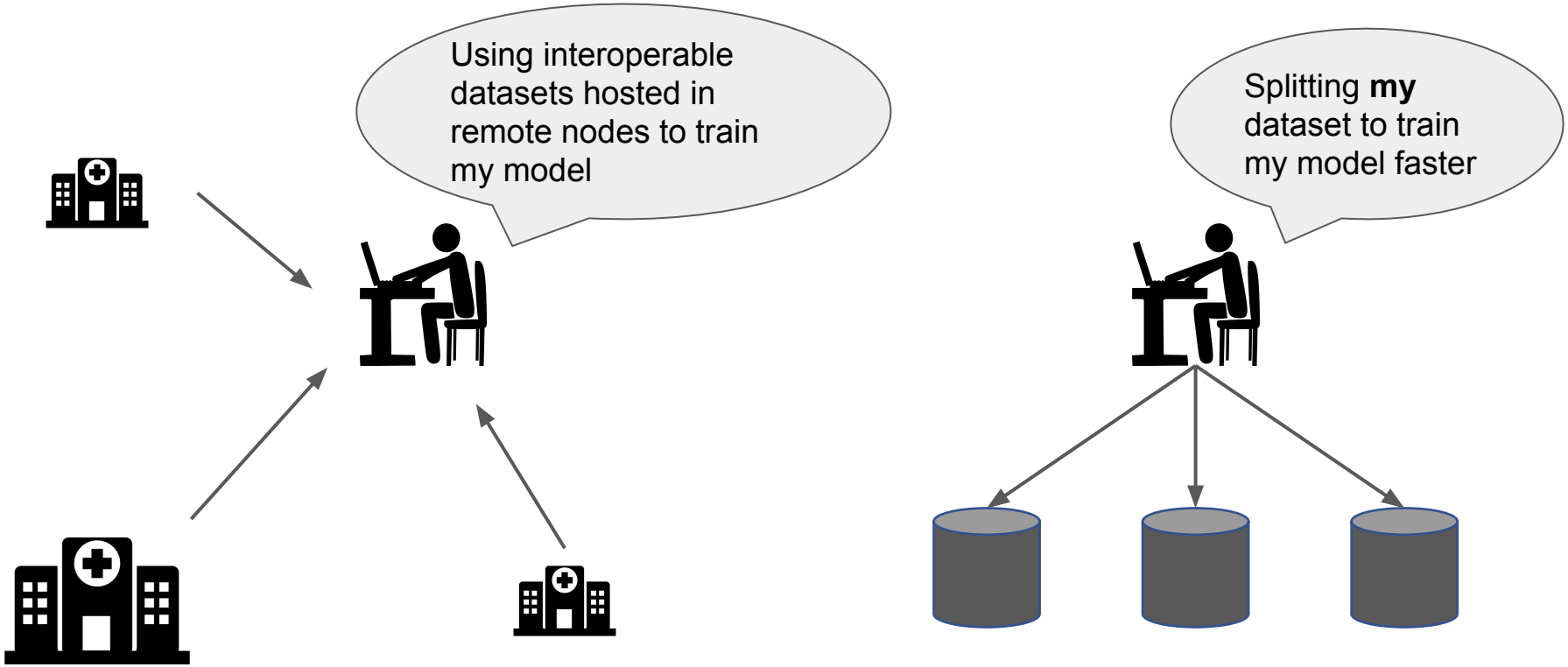
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
379	0	93	100	39	72	43.4	1.021	35	0
380	1	107	72	30	82	30.8	0.821	24	0
381	0	105	68	22	0	20.0	0.236	22	0
382	1	109	60	8	182	25.4	0.947	21	0
383	1	90	62	18	59	25.1	1.268	25	0

384 rows x 9 columns

Goal of this tutorial

- Get an intuition of Federated Learning: why, when, who and how
- Try an open-source tool providing a simple architecture for privacy: explicit requests and authorizations
- Using this framework, train a model remotely with one, then two data shards and compare the results
- Discuss the issues enabled by such a system and review the state of the art

Federated Learning vs Distributed Learning



Benefits of Federated (and distributed) Learning

- Use computational resources of the remote hosts
- Reduce overfitting since training is done separately on each node
- Increase the number of samples to train on (FL)
- Mitigate unbalanced datasets (FL)
- Data does not leave its host (FL)

OpenMined is a community of scholars, developers and cryptographers working on privacy enhancing technologies (PETs), mostly in the domain of federated learning.

They developed PySyft, “A library for answering questions using data you cannot see”, built on top of famous ML framework PyTorch.

This is the library we will be using in this tutorial.

Hands-on tutorial scenario

A **data scientist** wants to train a classifier to predict diabetes in a female population.

Two hospitals (**data owners**) give access to their dataset. Thanks to prior project management, the two datasets are interoperable: same feature set (*horizontal* FL), same authorizations and no bottleneck on computational capacity.

Yet, no data transfer agreement has been signed and no record should leave its owner's host server.

Data Owner 1

Data scientist

Data Owner 2

Hands-on tutorial scenario

The data scientist will first train a model on data from **owner 1 only**. (remote learning) to check everything is setup correctly.

Once it is done, **data owner 2** will join and the actual federated learning will take place.

Results from training on one and two datasets will then be compared.

For the sake of demo simplicity:

1. Both data owners share the same .csv dataset, each one uses half of it.

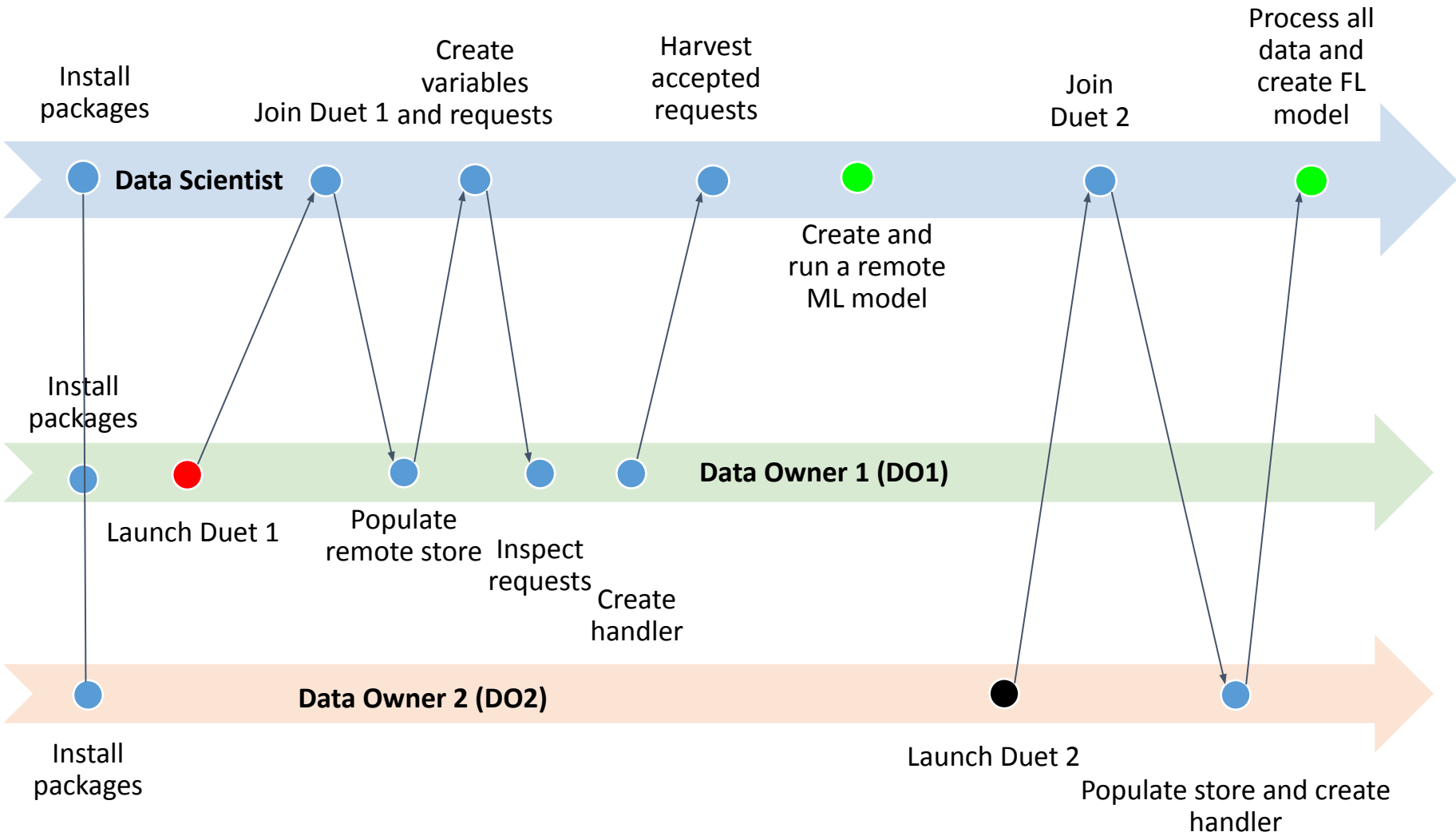
2. If running on a local environment, all notebooks have access to the same files.

3. Requests handlers are completely permissive.

Progress

We will be using Duet, a requests/responses tool that requires a lot of back and forths from one client to the other.

The first part of the tutorial requires focus not to get lost in those first steps.


Install packages
Join Duet 1
Create variables and requests
Harvest accepted requests
Create and run a remote ML model
Join Duet 2
Process all data and create FL model
Data Scientist
Install packages
Launch Duet 1
Populate remote store
Inspect requests
Create handler
Data Owner 1 (DO1)
Install packages
Data Owner 2 (DO2)
Launch Duet 2
Populate store and create handler

Strategies for Federated averaging (non-exhaustive)

- Averaging gradients

Algorithm 1 Parameter server with synchronized SGD

Server executes:
Initialize θ_0
for $t = 1$ to T **do**
  for each client k **do**
    $g_t^k \leftarrow$ **ClientUpdate**(θ_{t-1})
  end for
  $\theta_t \leftarrow \theta_{t-1} - \eta \sum_k g_t^k$ ▷ synchronized gradient updates
end for

ClientUpdate(θ):
Select batch b from client's data
return local gradients $\nabla L(b; \theta)$

- Averaging models, weighted

Algorithm 2 Federated learning with model averaging

Server executes:
Initialize θ_0
$m \leftarrow max(C \cdot K, 1)$
for $t = 1$ to T **do**
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **do**
    $\theta_t^k \leftarrow$ **ClientUpdate**(θ_{t-1})
  end for
  $\theta_t \leftarrow \sum_k \frac{n^k}{n} \theta_t^k$ ▷ averaging local models
end for

ClientUpdate(θ):
for each local iteration **do**
  for each batch b in client's split **do**
    $\theta \leftarrow \theta - \eta \nabla L(b; \theta)$
  end for
end for
return local model θ

source (1), (2)

Performance improvements in Federated Learning

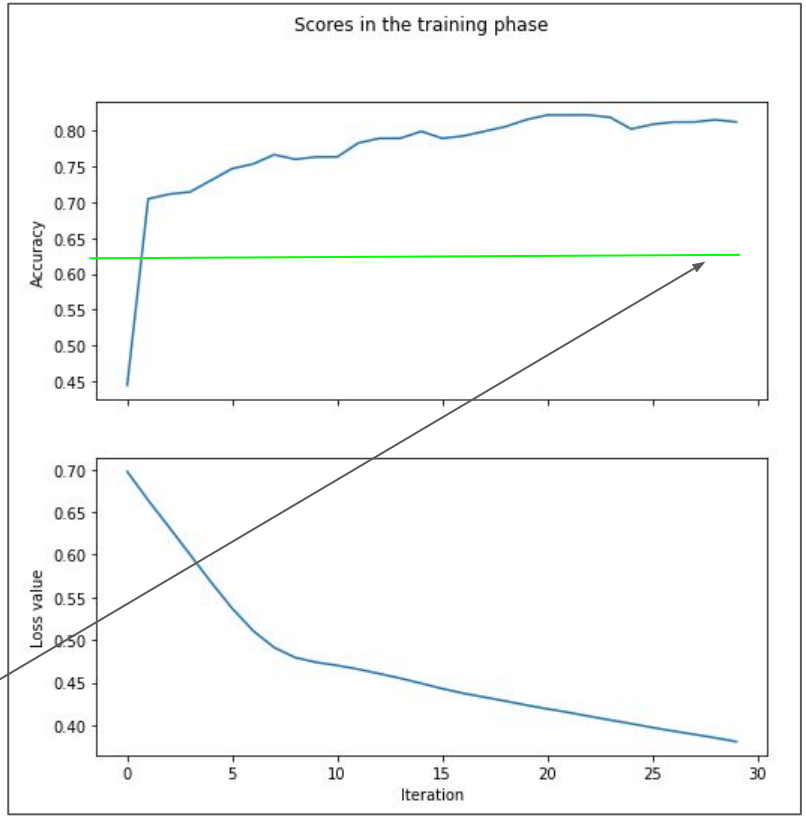
- In general, it does not make a lot of sense to start each round with the same model on all sides. The point is only to integrate contributions from the other nodes.
- Do not necessarily train on every data shard at each iteration (mitigating poisoning or deviating data)
- Take into account previous global and local averages
- As in non-federated learning, tune the hyperparameters

Results of remote training (one data shard)

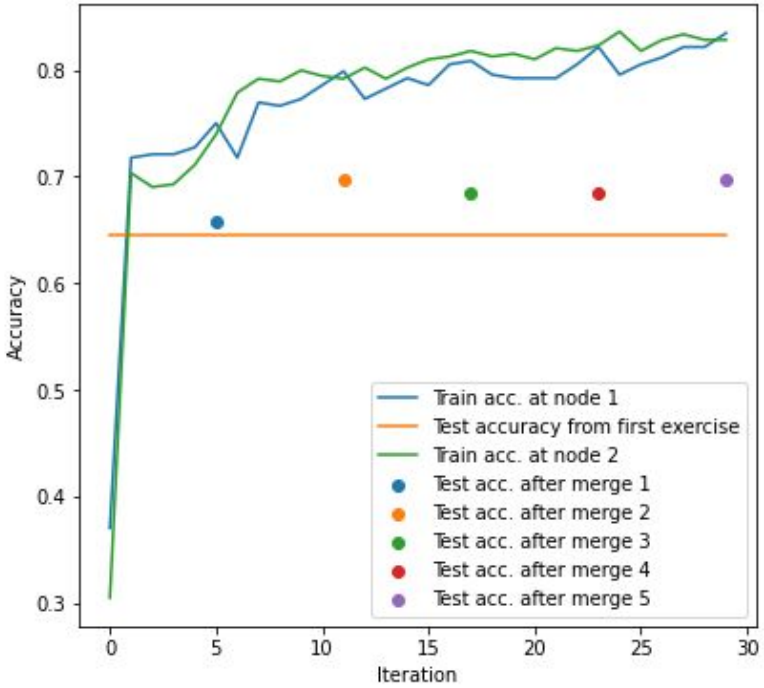
On the right are the loss and accuracy scores in the training phase of 30 epochs.
Training accuracy goes up to 81%, which denotes a bit of overfitting.

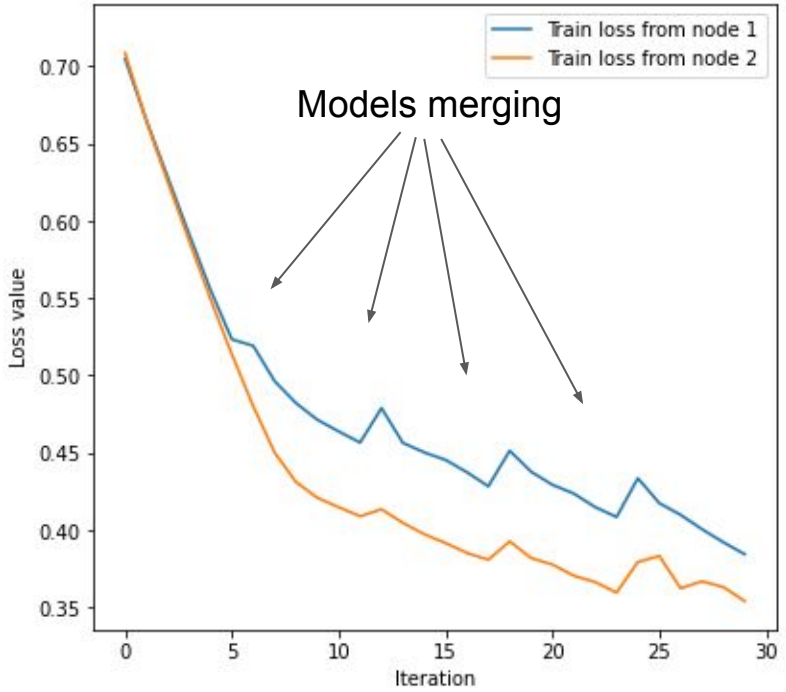
Below is the output of the accuracy on the test set, with a test ratio of 20% (meaning 20% of the whole data shard is reserved for testing). It is 62% in this case.

```
Accuracy of final classifier on test set is  0.62
```

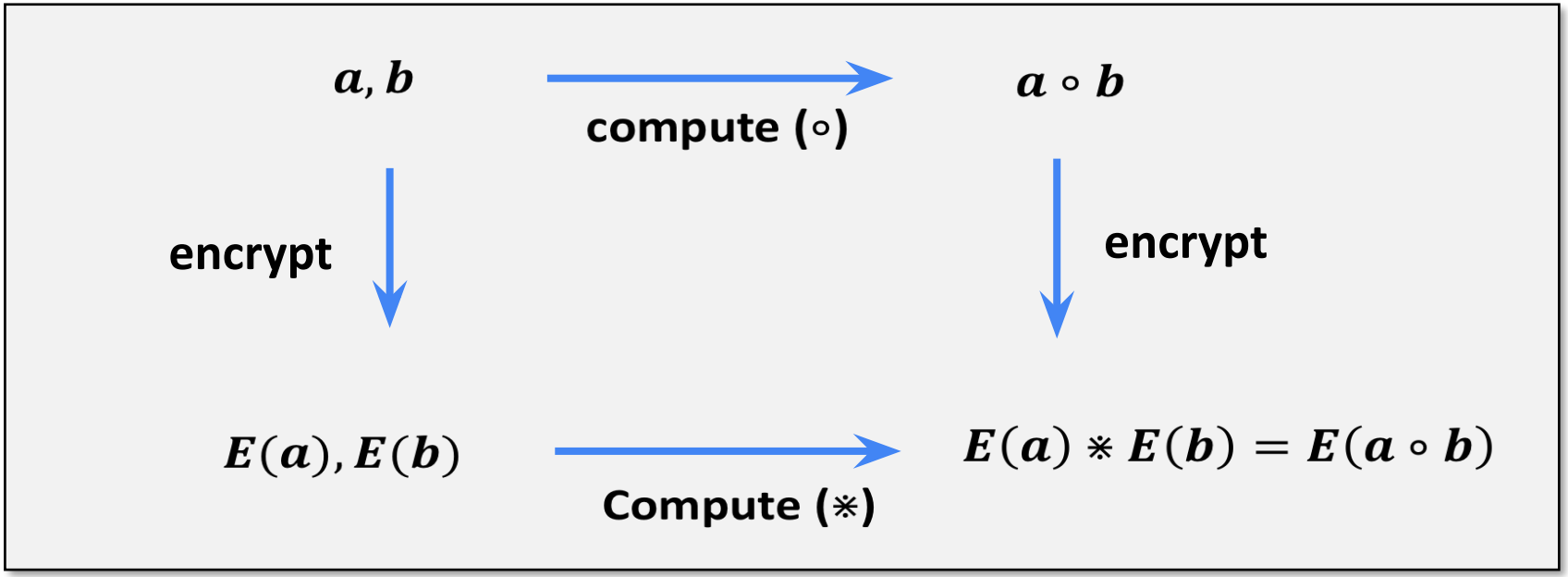


Results of federated learning (2 data shards)





Reminder: Homomorphic Encryption



Homomorphic encryption enables computations directly on encrypted data: "compute on the data without seeing the data"

Reminder: Differential Privacy

- **Basic philosophy:** instead of the real answer to a query, output a random answer, such that by a small change in the database (someone joins or leaves), the distribution of the answer does not change much.
- **A new privacy goal:** minimize the increased risk incurred by an individual when joining (or leaving) a given database.
- **Motivation:** A privacy guarantee that limits risk incurred by joining, therefore encourages participation in the dataset, increasing social utility.

- Differential privacy is a privacy notion, not a mechanism

→ We use mechanisms to *achieve* differential privacy

Programming libraries references

PyMPC (augmentation of Syft with SMPC): https://github.com/kamathhrishi/SyMPC

AutoDP in Syft: https://github.com/OpenMined/PySyft/tree/dev/packages/syft/src/syft/core/adp

FATE federated AI: https://fate.fedai.org/

OPACUS (Differential privacy into PyTorch): https://github.com/pytorch/opacus

The MedCo project for homomorphic-based clinical cohort exploration: https://medco.epfl.ch/ [4]

Academic work references

[1]

H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” *arXiv:1602.05629 [cs]*, Feb. 2017, Accessed: Oct. 27, 2021. [Online]. Available: http://arxiv.org/abs/1602.05629

[2]

R. Shokri and V. Shmatikov, “Privacy-Preserving Deep Learning,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver Colorado USA, Oct. 2015, pp. 1310–1321. doi: 10.1145/2810103.2813687.

[3]

D. Froelicher *et al.*, “Scalable Privacy-Preserving Distributed Learning,” *arXiv:2005.09532 [cs]*, Jul. 2021, Accessed: Oct. 27, 2021. [Online]. Available: http://arxiv.org/abs/2005.09532

[4]

D. Froelicher, M. Misbach, l, J. R. Troncoso-Pastoriza, J. L. Raisaro, and J.-P. Hubaux, “MedCo2: Privacy-Preserving Cohort Exploration and Analysis,” *Digital Personalized Health and Medicine*, pp. 317–321, 2020, doi: 10.3233/SHTI200174.

Notebook references:

[5]

J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting Gradients -- How easy is it to break privacy in federated learning?,” *arXiv:2003.14053 [cs]*, Sep. 2020, Accessed: Oct. 13, 2021. [Online]. Available: http://arxiv.org/abs/2003.14053

[6]

J. Scheibner *et al.*, “Revolutionizing Medical Data Sharing Using Advanced Privacy-Enhancing Technologies: Technical, Legal, and Ethical Synthesis,” *J Med Internet Res*, vol. 23, no. 2, p. e25120, Feb. 2021, doi: 10.2196/25120.

[7]

G. Kaissis *et al.*, “End-to-end privacy preserving deep learning on multi-institutional medical imaging,” *Nat Mach Intell*, vol. 3, no. 6, pp. 473–484, Jun. 2021, doi: 10.1038/s42256-021-00337-8.

A lot of resources, including introduction to differential privacy, homomorphic encryption, and federated learning are available on the OpenMined blog https://blog.openmined.org/