Project 5: Federated and Distributed Learning

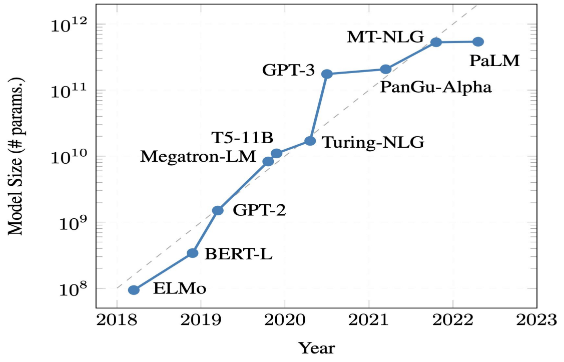
Machine learning and Deep learning, AY 2023/2024

Teaching Assistant: Riccardo Zaccone (riccardo.zaccone@polito.it)

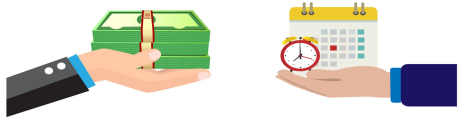
Motivations

Modern success of deep learning models relies on **bigger and bigger models**…

› … but training and serving such models is becoming increasingly more difficult and costly



Evolution of the size of large pre-trained models [Treviso et al., 2022]



› Hardware speedup **scaling laws are not going to cope forever** with the limits of current algorithms and architectures

› The **centralized training** paradigm no longer matches needs of modern applications, which need to learn from **heterogeneous** and **massively decentralized data**

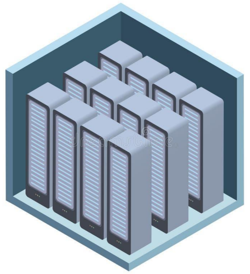
Distributed and Federated Learning

Distributed Learning:

💪 Nodes are powerful servers in a data center

⚡ Interconnections are fast and reliable

🔀 Data can be exchanged between nodes

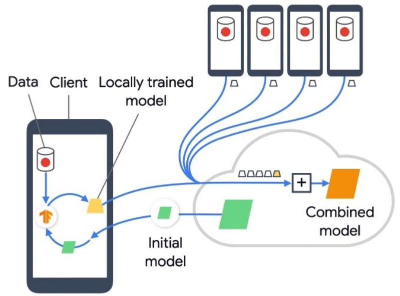


Federated Learning:

📱 Nodes are usually heterogeneous, low-powered and decentralized edge devices

🐢 Interconnections are slow and unreliable

🔒 Additional constraints on privacy and efficiency

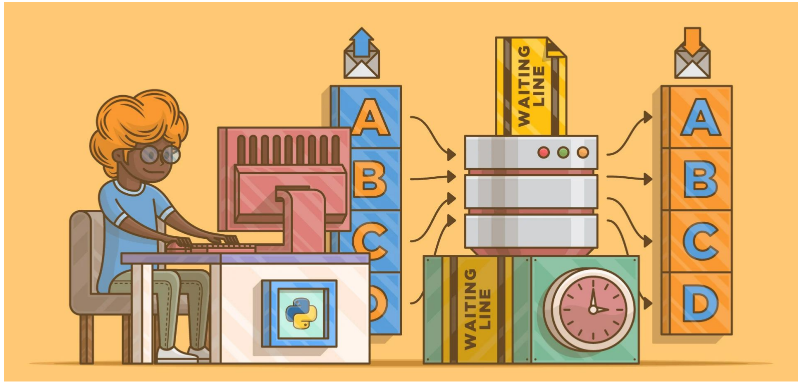


FedAvg = **Fed**erated **Av**era**g**ing

What Distributed Learning is about

Learning at scale:

- 💪 In recent years, models have become so **big** they don't even fit the memory of a **single accelerator**
- ⚡ As fast as powerful it can be, a single accelerator is **still to slow for today's demands**
- 🔀 Models are **overparametrized**, optimization becomes difficult, the **cost rises**



More with less:

- ➗ **Make every step count:** same or higher performance, with less computation
- ⌛ **I want it now:** reduce the compute time
- 🔀 **Do it locally:** Reduce communication
- 💸 **I'll keep my kidneys:** make AI available to small institutions
- 🌲 **Please stop burning me:** reduce the environmental impact

The current state of data-parallel distributed learning

Current SOTA algorithms for distributed learning

🌏 Large Batch optimizers [17, 18]:

💡 **Intuition:** to use N accelerators, increase the batch size xN and process in parallel

⚙️ **How:** apply layerwise gradient scaling + clever learning rate warmup

🏠 Local methods [19, 20, 21]:

💡 **Intuition:** reduce communication by means of local work

⚙️ **How:** perform more than one optimization steps before model's parameters synchronization

Main challenges in FL

Statistical heterogeneity:

📊 **non-iidness** of local datasets hampers convergence of FedAvg[1]
⌛ **increased number of rounds** to reach a target accuracy [2]

System heterogeneity:

📱 **edge devices** characterized by [3]:
- 🔋 being **transient**
- 🧮 **limited** computing **resources**
- 🚧 **slow**, **expensive**, and **unreliable communication links**

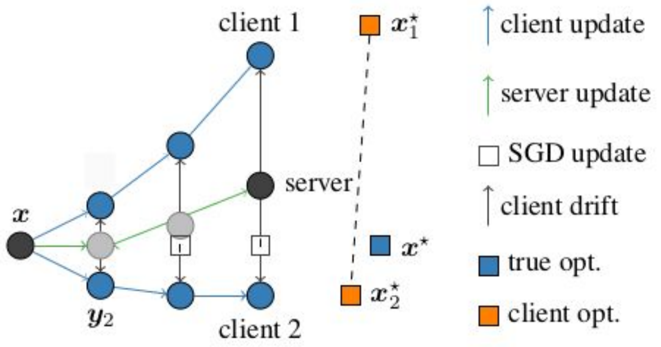


Figure 1: Client-drift in FedAvg illustrated for 2 clients with 3 local steps, from [5]

But there's actually much more

Fairness and bias mitigation

📊 **Unfairness** in ML models often comes from training data: FL training may introduce **bias**

Robustness against attacks and failures

📱 FL may introduce **new attack surfaces** at **training-time**:

- 💣 to **hamper learning** the model
- 👺 **to bias the learned model** to the adversary's preferable inferences

Preserving Privacy

💥 **gradient inversion attacks**, to leak user data

Multi-Task, Personalization & Meta-Learning

👥 see **each client as a separate task (MTL)**

🔨 **Meta-learn** a global model, use it as **starting point** for learning an **adapted** one (MAML)

Projects overview

TRACK A: Distributed Learning

Task: CV (CIFAR-100)

Main objectives:

a. Familiarize with state-of-art **data-parallel** solutions
b. Identify the **problems** in **scaling up** a typical computer vision task

Your contribution (choose one):

a. **Model aggregation**
b. **Asynchronous training**
c. Adaptive learning rates and batch size

TRACK B: Federated Learning

Task: CV (CIFAR-100), NLP (Shakespeare)

Main objectives:

a. Understand the **unique challenges** of Federated Learning
b. Identify **applications** and **opportunities** for FL in **real-world** applications

Your contribution (choose one):

a. **Smart client selection**
b. **Personalized FL**
c. **FL-tailored architectures**

TRACK A: Distributed Learning

1. **Becoming familiar with the state of art:**
   Before starting the project, you should take some time to study and get familiar with the distributed learning, its algorithms, challenges and the main proposed solutions. Here some references:
   a. **Large Batch Training** [11, 12]
   b. **Local SGD** [13, 14]
   c. **Optimizers for distributed learning** [15]

2. **Codebase, Resources & Dataset:**
   a. Start **building a codebase** for **centralized training**, using the model architecture described in [a]. For your experiments you will use the free version of **Google Colab**
   b. Use **CIFAR-100** dataset, which can be downloaded from torchvision

3. Start with the experiments:

1. **Run the centralized baseline:** it is important to have as reference the performance in centralized.

2. **Test Large Batch Optimizers**: replace the standard optimizers with their large batch counterparts [17,18].

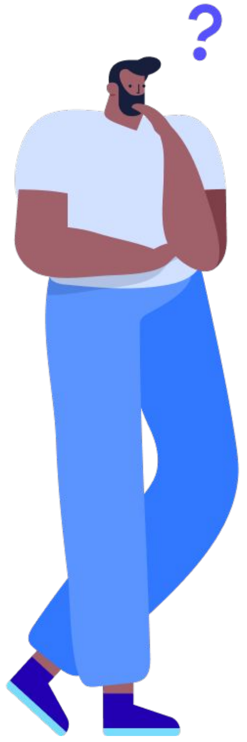
3. **Time to go distributed**: perform data sharding and train using the LocalSGD algorithm [19].

4. **Try other distributed optimizers:** what about optimizing both in the outer and inner loops? Try the strategy of [21]

5. **Analyze your results:** we will understand the impact of computation and communication cost. Which alternative is the best?

4. Time for your personal contribution (choose one):

1. Explore **asynchronous schemes**: there some works on asynchronous methods for distributed learning [22]
2. Can we optimize for the number of steps J in local training methods?
3. Can we adaptively change the batch size during training to reduce the number of computation and enhance final performance?



IMPORTANT:

- During all the phases of experimentation, remind to always apply the **best deep learning practices**
- **Before engaging in phase 4, always discuss your plan with the TA**

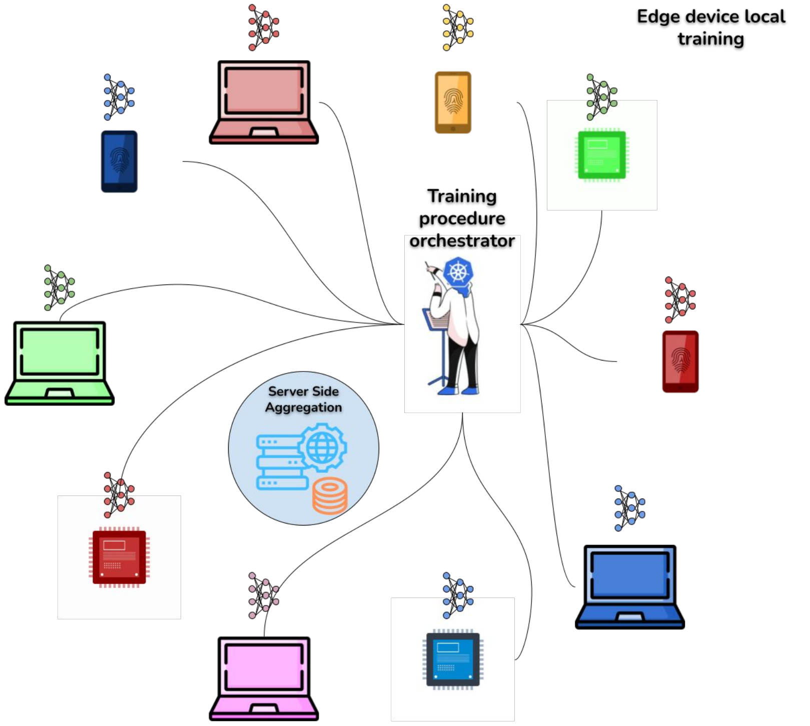
TRACK B: Federated Learning

1. **Becoming familiar with the state of art:**
   Before starting the project, you should take some time to study and get familiar with the federated scenario, its algorithms and challenges. Here some references:
   a. **Understand the FL setting** [2, 3, 10]
   b. **Outlook on issues and state of the art** [7,8,9]
   c. **Statistical and system heterogeneity:** algorithms [1,4,5] and architectural investigations [12, 13, 14]
   d. **Real-world applications** [11]

2. **Codebase, Resources & Dataset:**
   a. Start **building a codebase** for **centralized training**, using the model architecture described in [a]. For your experiments you will use the free version of **Google Colab**
   b. Use **CIFAR-100** and **Shakespeare** datasets, which can be downloaded respectively from torchvision and LEAF [15]

3. Start with the experiments:

1. **Centralized baseline:** train your models on both CIFAR-100 and Shakespeare
2. **Test the impact of client participation ratio and local steps**
3. **Simulate heterogeneous distributions**: test the performance of FedAvg [10] on CIFAR-100 and Shakespeare, comparing with the iid and non-iid sharding
4. **Test different architectures:** let's go beyond simple CNN, and test how a ResNet would work in FL!



4. Time for your personal contribution (choose one):

1. **Client selection:** consider a situation in which clients have non-uniform probability of being selected for training at each round.

2. **Personalized FL:** explore state-of-art solutions for personalized FL, identify room for improvement and propose novel variations

3. **FL-tailored architectures:** some works [12,13] have investigated the role of architectural design in FL.

IMPORTANT:

- During all the phases of experimentation, remind to always apply the **best deep learning practices**
- **Before engaging in phase 4, always discuss your plan with the TA**

Rules

APPLYING FOR PROJECTS (TA OR TB)

Deadline: Wednesday, March 20th 23:59

Modality:

a. Groups are made by exactly 3 students (strict rule, no exceptions)
b. Use this form to express your preference

Reminders:

a. For (TA+TB) I can accept **up to 10 groups**, when applying for projects you will be asked to list your preferences.
b. No rush, take your time to think: **the form is not first-come, first-served**
c. For other details, refer to comm rules

ORGANIZATION

Lectures: there will be **introductory lectures** on distributed and federated learning

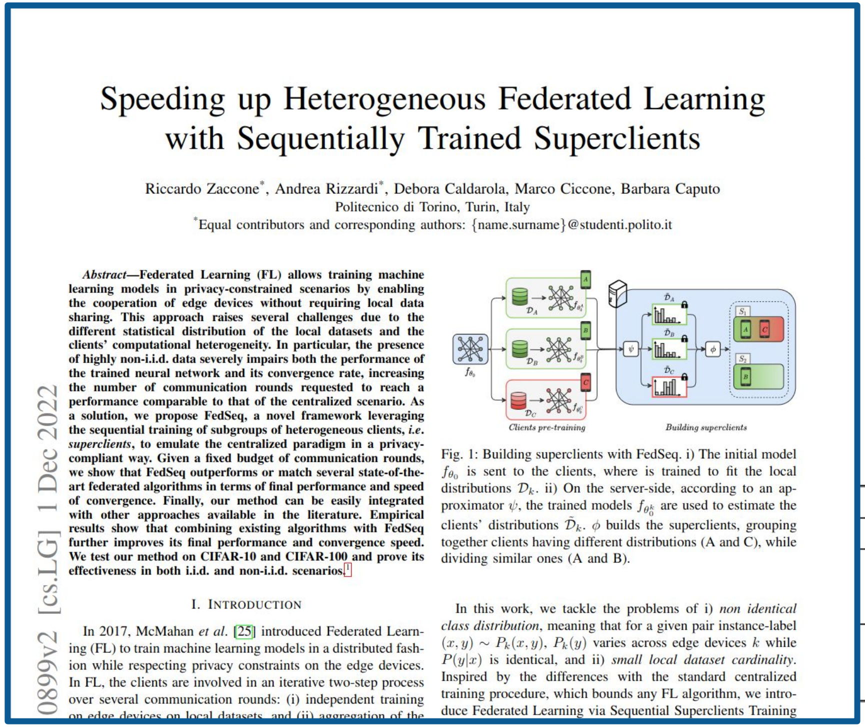
During (only) the trimester:

a. There will be **reading groups** on topics related to T1 & T2
b. There will be **weekly group meetings** with me to discuss the progress of your projects (globally, not one for each group!)

For the exam:

a. Deliver a **report** along with the **code** and data for replicating your results

Projects from last years - Federated Sequential Learning



Speeding up Heterogeneous Federated Learning with Sequentially Trained Superclients

Riccardo Zaccone*, Andrea Rizzardi*, Debora Caldarola, Marco Ciccone, Barbara Caputo
Politecnico di Torino, Turin, Italy
*Equal contributors and corresponding authors: {name.surname}@studenti.polito.it

***Abstract*—Federated Learning (FL) allows training machine learning models in privacy-constrained scenarios by enabling the cooperation of edge devices without requiring local data sharing. This approach raises several challenges due to the different statistical distribution of the local datasets and the clients' computational heterogeneity. In particular, the presence of highly non-i.i.d. data severely impairs both the performance of the trained neural network and its convergence rate, increasing the number of communication rounds requested to reach a performance comparable to that of the centralized scenario. As a solution, we propose FedSeq, a novel framework leveraging the sequential training of subgroups of heterogeneous clients, *i.e.* *superclients*, to emulate the centralized paradigm in a privacy-compliant way. Given a fixed budget of communication rounds, we show that FedSeq outperforms or match several state-of-the-art federated algorithms in terms of final performance and speed of convergence. Finally, our method can be easily integrated with other approaches available in the literature. Empirical results show that combining existing algorithms with FedSeq further improves its final performance and convergence speed. We test our method on CIFAR-10 and CIFAR-100 and prove its effectiveness in both i.i.d. and non-i.i.d. scenarios.**[1]

I. Introduction

In 2017, McMahan *et al.* [25] introduced Federated Learning (FL) to train machine learning models in a distributed fashion while respecting privacy constraints on the edge devices. In FL, the clients are involved in an iterative two-step process over several communication rounds: (i) independent training on edge devices on local datasets, and (ii) aggregation of the

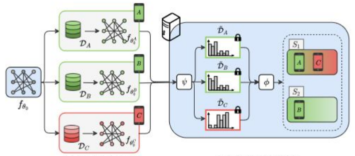
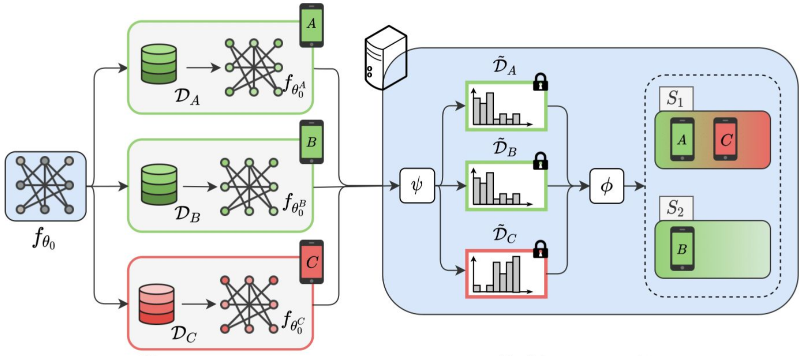


Fig. 1: Building superclients with FedSeq. i) The initial model f_{θ_0} is sent to the clients, where is trained to fit the local distributions $\mathcal{D}_k$. ii) On the server-side, according to an approximator ψ, the trained models $f_{\theta_0^k}$ are used to estimate the clients' distributions $\tilde{\mathcal{D}}_k$. ϕ builds the superclients, grouping together clients having different distributions (A and C), while dividing similar ones (A and B).

In this work, we tackle the problems of i) *non identical class distribution*, meaning that for a given pair instance-label $(x, y) \sim P_k(x, y)$, $P_k(y)$ varies across edge devices k while $P(y|x)$ is identical, and ii) *small local dataset cardinality*. Inspired by the differences with the standard centralized training procedure, which bounds any FL algorithm, we introduce Federated Learning via Sequential Superclients Training

90%
1682 (–)
1682 (–)
1113 (–)
3436 (–)
3033 (–)
2014 (–)



Clients pre-training

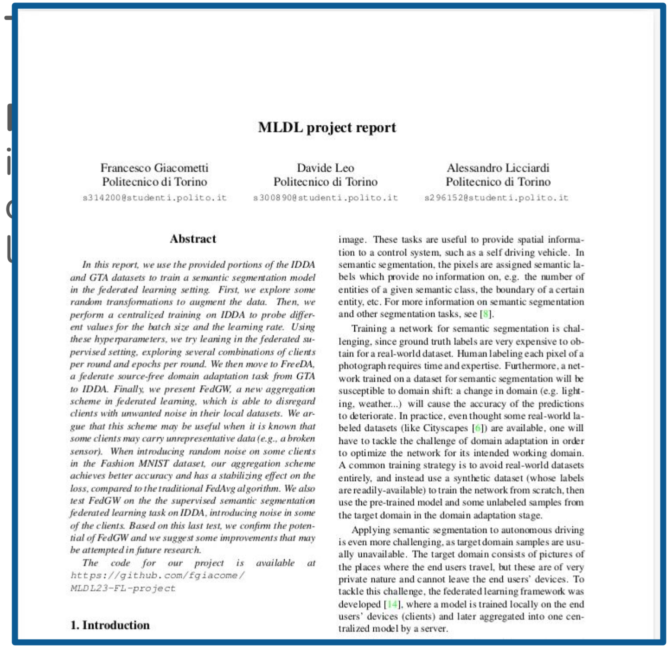
Building superclients

Results:

- state-of-art performance with **improved convergence speed**
- Robustness to suboptimal solutions of the grouping problem
- FedSeq can be combined with existing algorithms

Projects from last years - Federated Sequential Learning

Key points:



MLDL project report

Francesco Giacometti
Politecnico di Torino
s314200@studenti.polito.it

Davide Leo
Politecnico di Torino
s300890@studenti.polito.it

Alessandro Licciardi
Politecnico di Torino
s296152@studenti.polito.it

Abstract

In this report, we use the provided portions of the IDDA and GTA datasets to train a semantic segmentation model in the federated learning setting. First, we explore some random transformations to augment the data. Then, we perform a centralized training on IDDA to probe different values for the batch size and the learning rate. Using these hyperparameters, we try leaning in the federated supervised setting, exploring several combinations of clients per round and epochs per round. We then move to FreeDA, a federate source-free domain adaptation task from GTA to IDDA. Finally, we present FedGW, a new aggregation scheme in federated learning, which is able to disregard clients with unwanted noise in their local datasets. We argue that this scheme may be useful when it is known that some clients may carry unrepresentative data (e.g., a broken sensor). When introducing random noise on some clients in the Fashion MNIST dataset, our aggregation scheme achieves better accuracy and has a stabilizing effect on the loss, compared to the traditional FedAvg algorithm. We also test FedGW on the the supervised semantic segmentation federated learning task on IDDA, introducing noise in some of the clients. Based on this last test, we confirm the potential of FedGW and we suggest some improvements that may be attempted in future research.

The code for our project is available at `https://github.com/fgiacome/MLDL23-FL-project`

1. Introduction

image. These tasks are useful to provide spatial information to a control system, such as a self driving vehicle. In semantic segmentation, the pixels are assigned semantic labels which provide no information on, e.g. the number of entities of a given semantic class, the boundary of a certain entity, etc. For more information on semantic segmentation and other segmentation tasks, see [8].

Training a network for semantic segmentation is challenging, since ground truth labels are very expensive to obtain for a real-world dataset. Human labeling each pixel of a photograph requires time and expertise. Furthermore, a network trained on a dataset for semantic segmentation will be susceptible to domain shift: a change in domain (e.g. lighting, weather...) will cause the accuracy of the predictions to deteriorate. In practice, even thought some real-world labeled datasets (like Cityscapes [6]) are available, one will have to tackle the challenge of domain adaptation in order to optimize the network for its intended working domain. A common training strategy is to avoid real-world datasets entirely, and instead use a synthetic dataset (whose labels are readily-available) to train the network from scratch, then use the pre-trained model and some unlabeled samples from the target domain in the domain adaptation stage.

Applying semantic segmentation to autonomous driving is even more challenging, as target domain samples are usually unavailable. The target domain consists of pictures of the places where the end users travel, but these are of very private nature and cannot leave the end users' devices. To tackle this challenge, the federated learning framework was developed [14], where a model is trained locally on the end users' devices (clients) and later aggregated into one centralized model by a server.

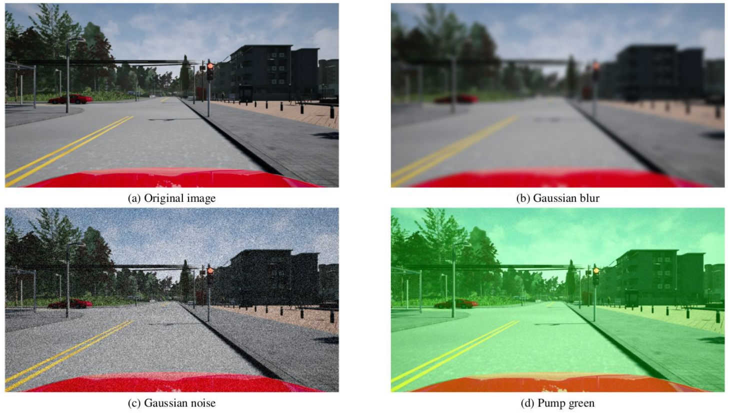


Figure 13. Examples of noise applied on an image from the IDDA dataset.

Results:

- better accuracy and has a stabilizing effect on the loss, compared to the traditional FedAvg algorithm

Questions?

References

[1] Hsu et al., Measuring the effects of non-identical data distribution for federated visual classification, 2019.
[2] Reddi et al., Adaptive federated optimization. ICLR 2021.
[3] Kairouz et al., Advances and open problems in federated learning, Found. Trends Mach. Learn, 2021
[4] Li et al., Federated optimization in heterogeneous networks. PMLR, 2018
[5] Karimireddy et al., Scaffold: Stochastic controlled averaging for federated learning, PLMR 2020
[6] Kairouz et al., Advances and Open Problems in Federated Learning, Foundations and Trends in Machine Learning, 2021
[7] Li at al., A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection, 2019
[8] Chen et al., Privacy and Fairness in Federated Learning: On the Perspective of Tradeoff, ACM computing surveys 2023
[9] Pfeiffer et al., Federated Learning for Computationally Constrained Heterogeneous Devices: A Survey, ACM computing surveys 2023
[10] McMahan et al., Communication-Efficient Learning of Deep Networks from Decentralized Data, AISTATS 2017
[11] Hsu et al., Federated Visual Classification with Real-World Data Distribution, ECCV 2020
[12] Qu et al., Rethinking Architecture Design for Tackling Data Heterogeneity in Federated Learning, CVPR 2022
[13] Pieri et al., Handling Data Heterogeneity via Architectural Design for Federated Visual Recognition, NeurIPS 2023
[14] Li et. al., Fedbn: Federated learning on non-iid features via local batch normalization. ICLR 2021
[15] Caldas et al. "Leaf: A benchmark for federated settings." Workshop on Federated Learning for Data Privacy and Confidentiality 2019
[16] McCandlish, An empirical model of large-batch training, arXiv 2018
[17] You et. Al, Large Batch Training of Convolutional Networks, arXiv 2017
[18] You et. Al, Large Batch Optimization for Deep Learning: Training BERT in 76 minutes, ICLR 2020
[19] Stich et al., Don't Use Large Mini-Batches, Use Local SGD, ICLR 2020
[20] Yu et. Al, On the Linear Speedup Analysis of Communication Efficient Momentum SGD for Distributed Non-Convex Optimization, ICML 2019
[21] Wang et. Al, SlowMo: Improving Communication-Efficient Distributed SGD with Slow Momentum, ICLR 2020
[22] Nabli et al., A2CiD2: Accelerating Asynchronous Communication in Decentralized Deep Learning, NeurIPS 2023

Project 5: Federated and Distributed Learning

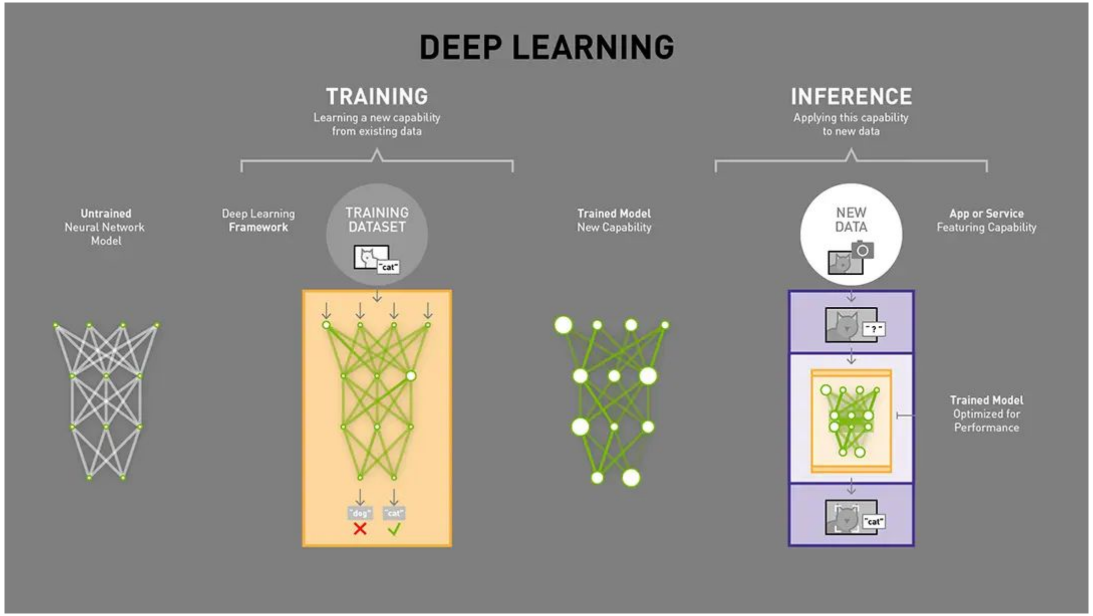
Machine learning and Deep learning, AY 2023/2024

Teaching Assistant:
Riccardo Zaccone (riccardo.zaccone@polito.it)

More on who we are:
The research group I work in: **http://vandal.polito.it/**
My personal page: **linkedin.com/in/riccardozacc/**

License

Anatomy of a classic centralized training


DEEP LEARNING
TRAINING
Learning a new capability from existing data
INFERENCE
Applying this capability to new data
Untrained Neural Network Model
Deep Learning Framework
TRAINING DATASET
"cat"
"dog"
"cat"
Trained Model New Capability
NEW DATA
"?"
"cat"
App or Service Featuring Capability
Trained Model Optimized for Performance

From NVIDIA blog

Main assumptions:

- Training data are **drawn from a distribution** (that we're aiming to learn)
- The test data comes from the **same distribution**
- The learner has **unconstrained access** to the training data

Anatomy of a classic centralized training

```python
def train_loop(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    for batch, (X, y) in enumerate(dataloader):
        # Compute prediction and loss
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch % 100 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")
```

In practice:

- **Stochastically** sample **small subsets** of training data (called *minibatches*)
- Compute the **output of the model**, and calculate its **error** (called the *loss*)
- **Optimize for the error**, using an optimization algorithm (the *optimizer*)
- **Repeat** the process (i.e. the train_loop) until the model **converges**