



# Continual Learning

---

Course structure and motivation

Antonio Carta

[antonio.carta@unipi.it](mailto:antonio.carta@unipi.it)

# What is Continual Learning?

Motivations and origins of the field

# Machine Learning Definition



“A computer program is said to **learn** from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.”

Tom Mitchell, Machine Learning

# Offline ML vs Mitchell ML



In practice, up to now you have seen only offline training:

“A computer program is said to **learn** from a dataset **D** sampled **i.i.d.** from a task **T** and performance measure **P**, if its performance at task **T**, as measured by **P**, improves with dataset **D**.”

Actual Machine Learning Methods

**Offline ML**=everything you have seen in ML and ISPR courses

“A computer program is said to **learn** from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.”

Tom Mitchell, Machine Learning

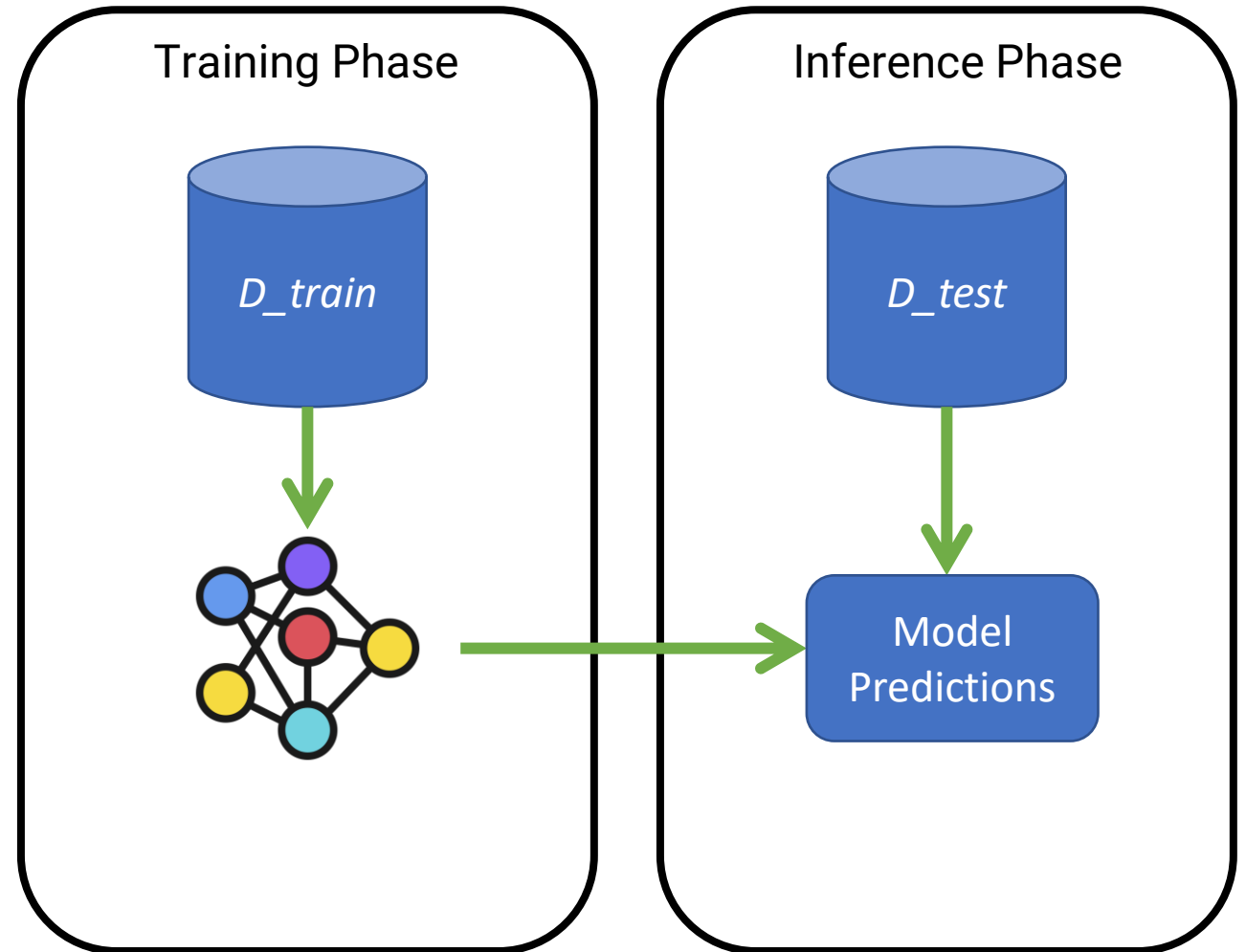
What we will see in this course is much closer to this ideal definition, with classes of tasks and incremental learning over time

# Offline Machine Learning – State of the Art

- Deep learning models today have a high performance in common vision benchmarks
- The key behind their success are:
  - Large models
  - Computational power (M\$)
  - Huge amounts of high quality data for training (again, \$\$\$ and time)
  - Having all the data before starting training (\$\$\$ and time)
  - There are also other more theoretical reasons that we will see during the course



- What happens if I don't have a single dataset?
- How do I use a pretrained model or update a previous model?
- How do I pretrain the model?
- What if my train and test distributions differ?



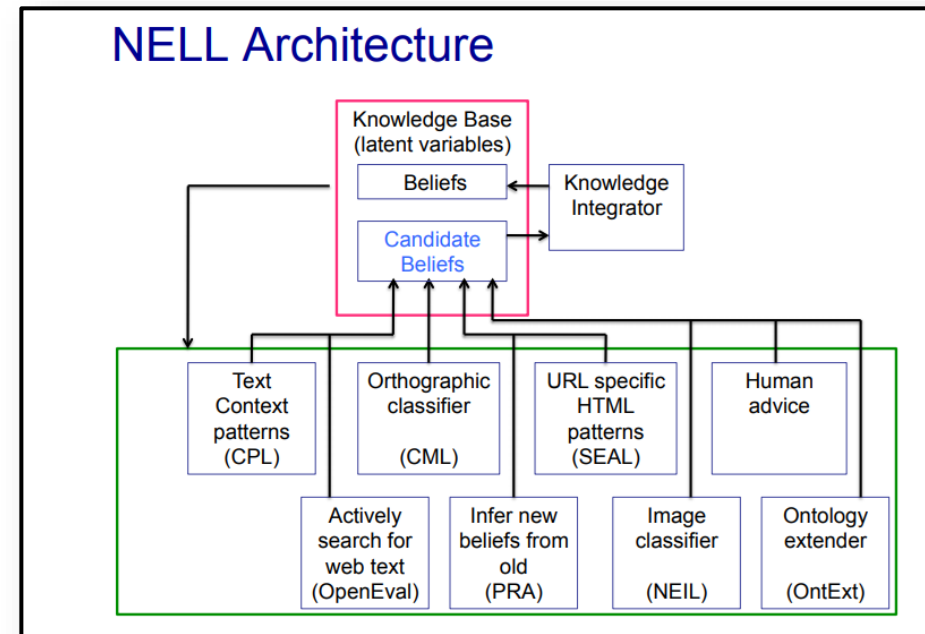
- **Efficiency:** learning shouldn't require burning the entire Amazon forest to train a single image classification model
- **Scalability:** learning algorithms should scale well with increasing model size and amount of data
- **Adaptability:** models should adapt to different train and test distributions
- **Reuse:** we want to be able to start learning from a model trained on related tasks
- **Incremental Learning:** we want to update the model over time

Offline ML does not satisfy any of these desiderata

# Historical Example – NELL



- Semi-Supervised Learning System
  - Semi-supervised = **limited need for labeling**
- Ran 24x7, from January, 2010 to September 2018
  - **Lifelong learning**
- Combination of many learning algorithms (CPL, CML, SEAL, OpenEval, PRA, NEIL)
  - **Modular system**
- Intended as a case-study for a never-ending learning agent

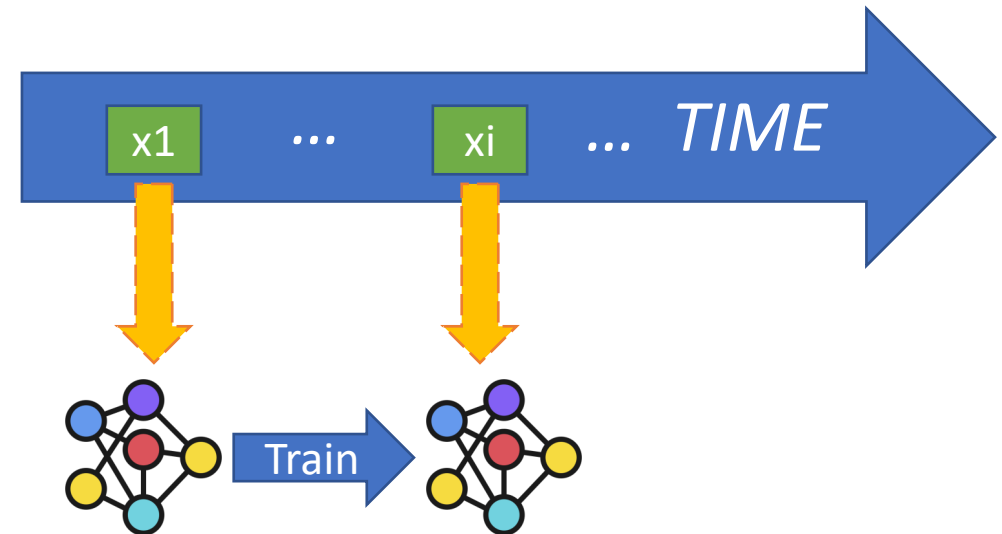




# ML with nonstationary data

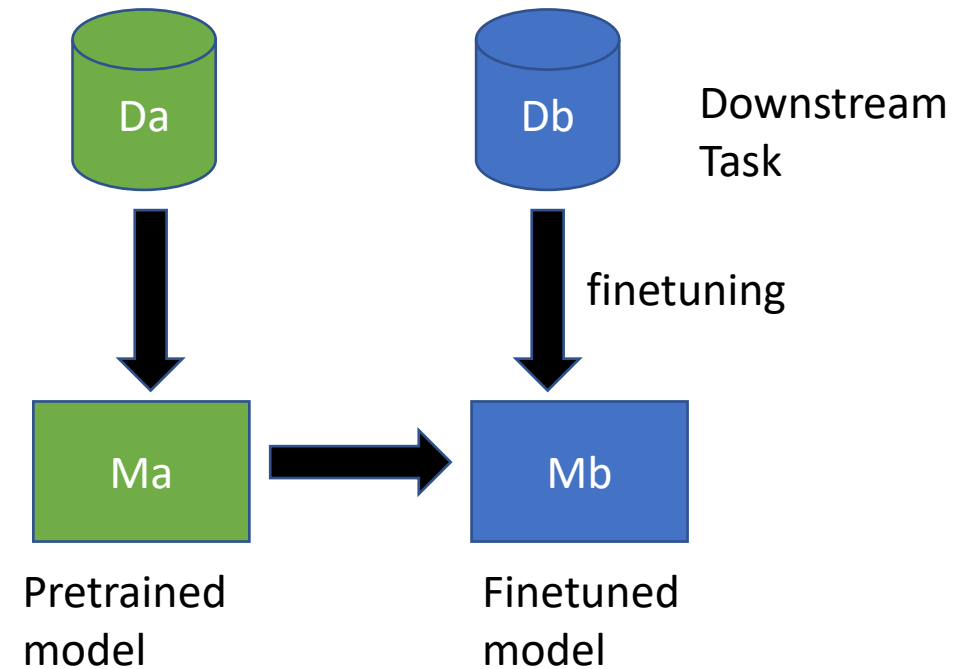
Real-world problems beyond static iid data

- Learning one sample at a time
- Efficiently
- On nonstationary data (not i.i.d.)



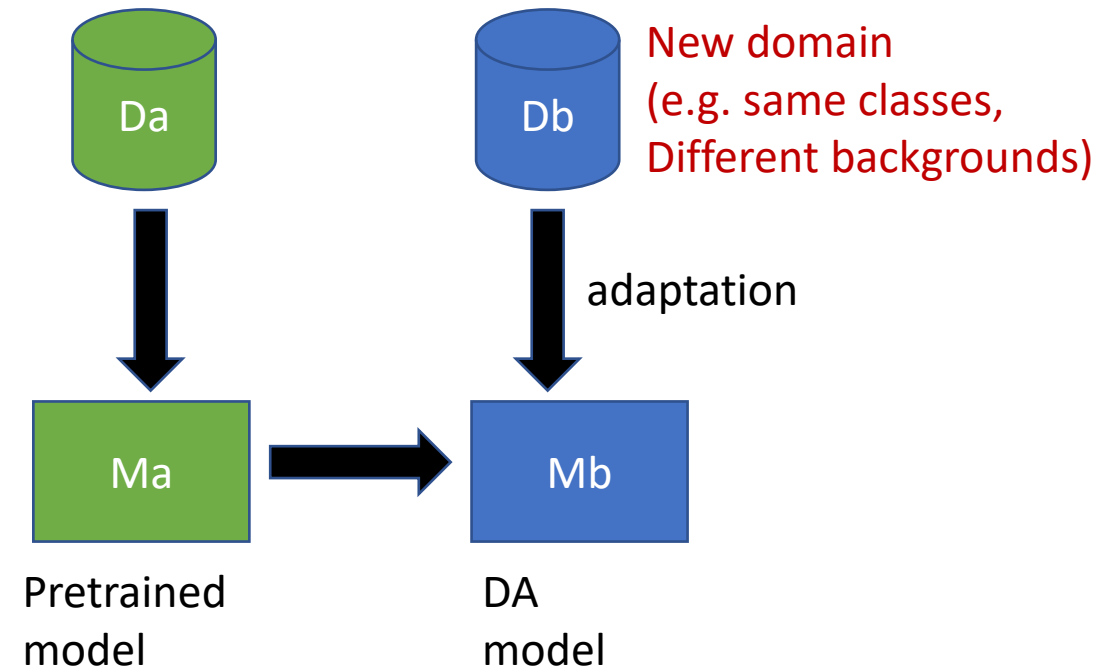
# Transfer Learning

- Given a large pretrained model
  - Trained on a large dataset (\$\$\$)
- Finetune a new model on a novel **task**
  - On a small dataset
- Example: given a language model, train a sentiment analysis model



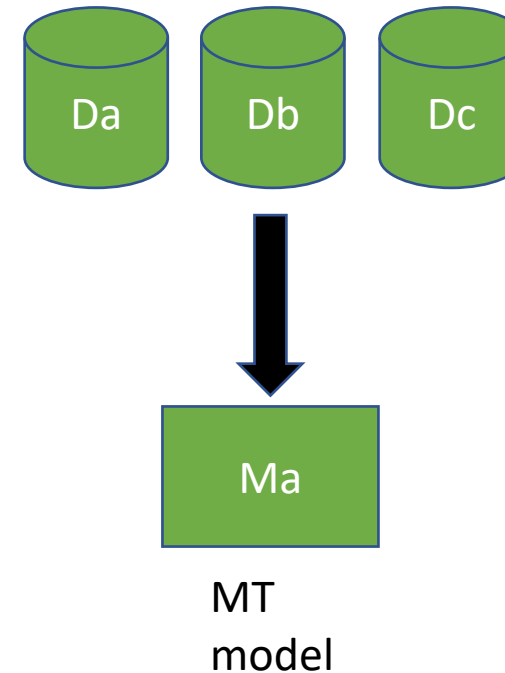
# Domain Adaptation

- Given a large pretrained model
  - Trained on a large dataset (\$\$\$)
- Finetune a new model on a novel **domain**
  - On a small labeled dataset
  - Sometimes, we may have access to unlabeled data
- Example: new images have a different background

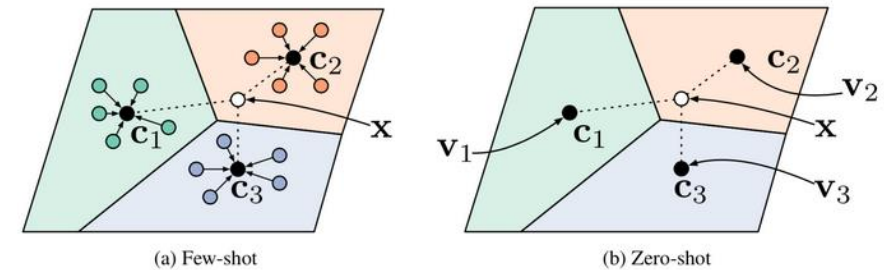


# Multi-Task Learning

- Can we train a deep neural network that is able to solve multiple tasks together?
  - Can we share some layers?
  - How do we split the network into task-specific and general components?
  - How do we train the components?



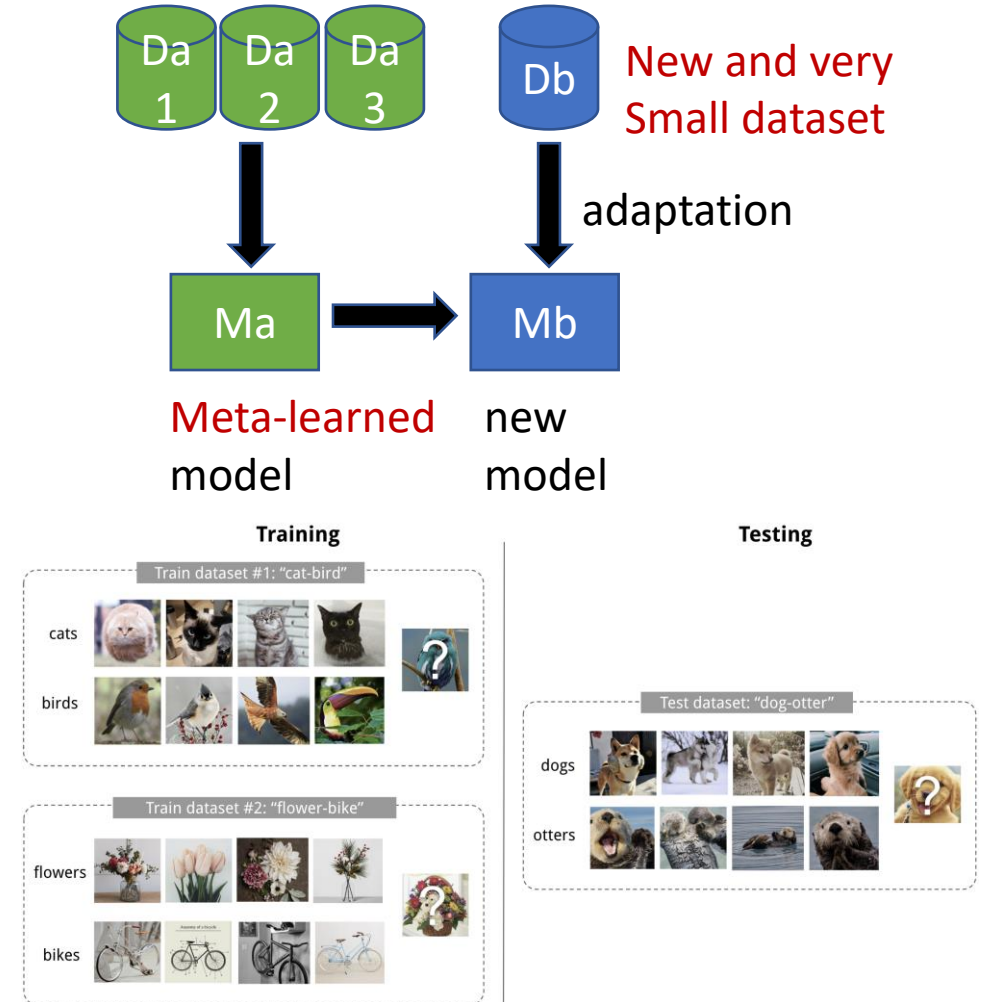
- Meta-learning is the problem of learning the optimization algorithm itself
  - Learning an optimizer
  - Automatic hyperparameter optimization
  - Neural Architecture Search
  - Learning a model that quickly generalize to new tasks
- The underlying assumption is that there is a common structure in the family of tasks that we want to solve



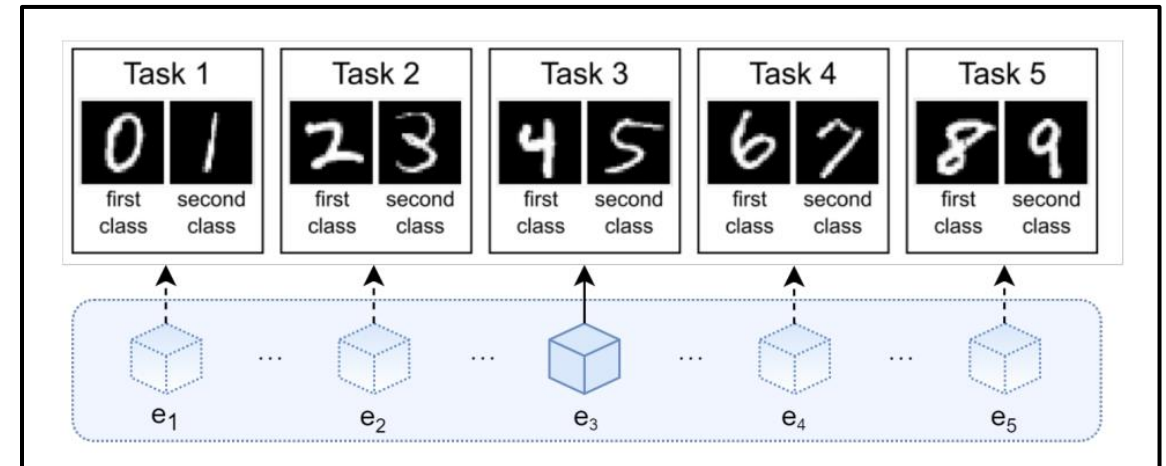
**Example:** learn an embedding space  
That generalize quickly to small  
novel tasks

# Few-shot learning

- What if the datasets are very small? E.g. 5 instances per class?
- We need to learn a generic model that can be **quickly adapted**
  - How do we learn the meta-model?
  - How do we adapt it?

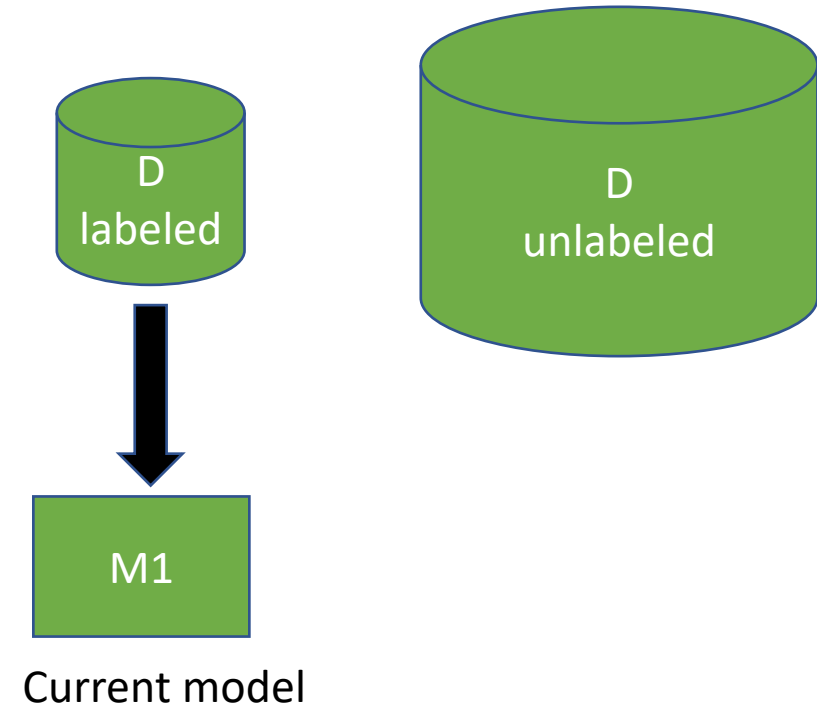


- How do we incrementally train a deep neural network without forgetting the previous data?
  - Adaptation
  - Efficiency
  - forgetting





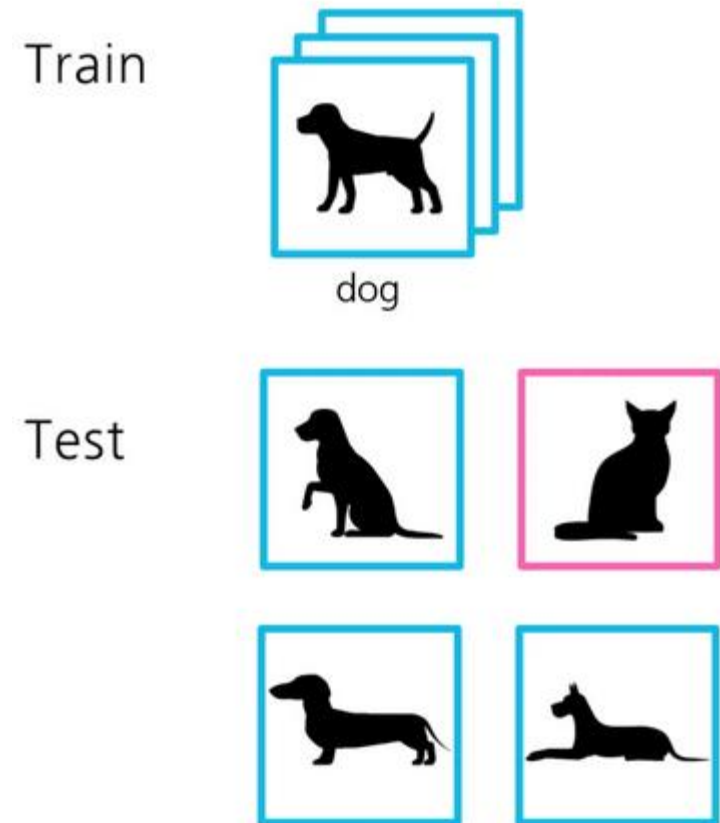
- We have a trained model, a labeled dataset, and a large unlabeled dataset
- Labeling is expensive
- How do we choose the best samples to label?



# Out-of-Distribution Detection

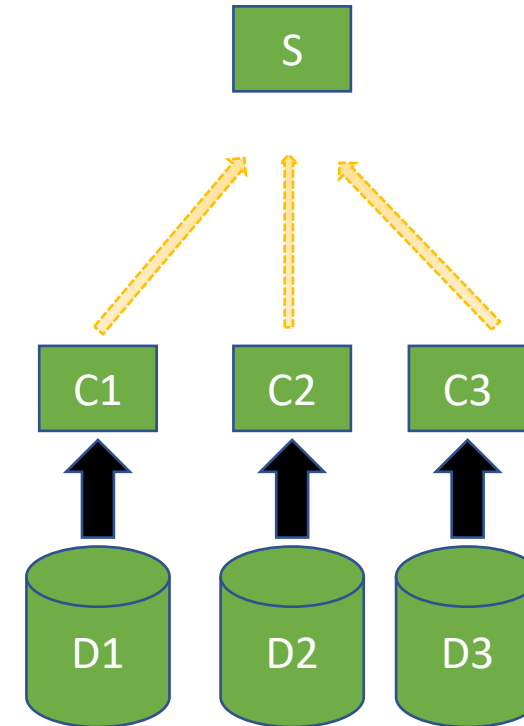
- Deep neural networks are overconfident
- How do we detect examples that are outside the model's known domain?
- Can we compute confidence intervals for the network's output?

## Image Classification



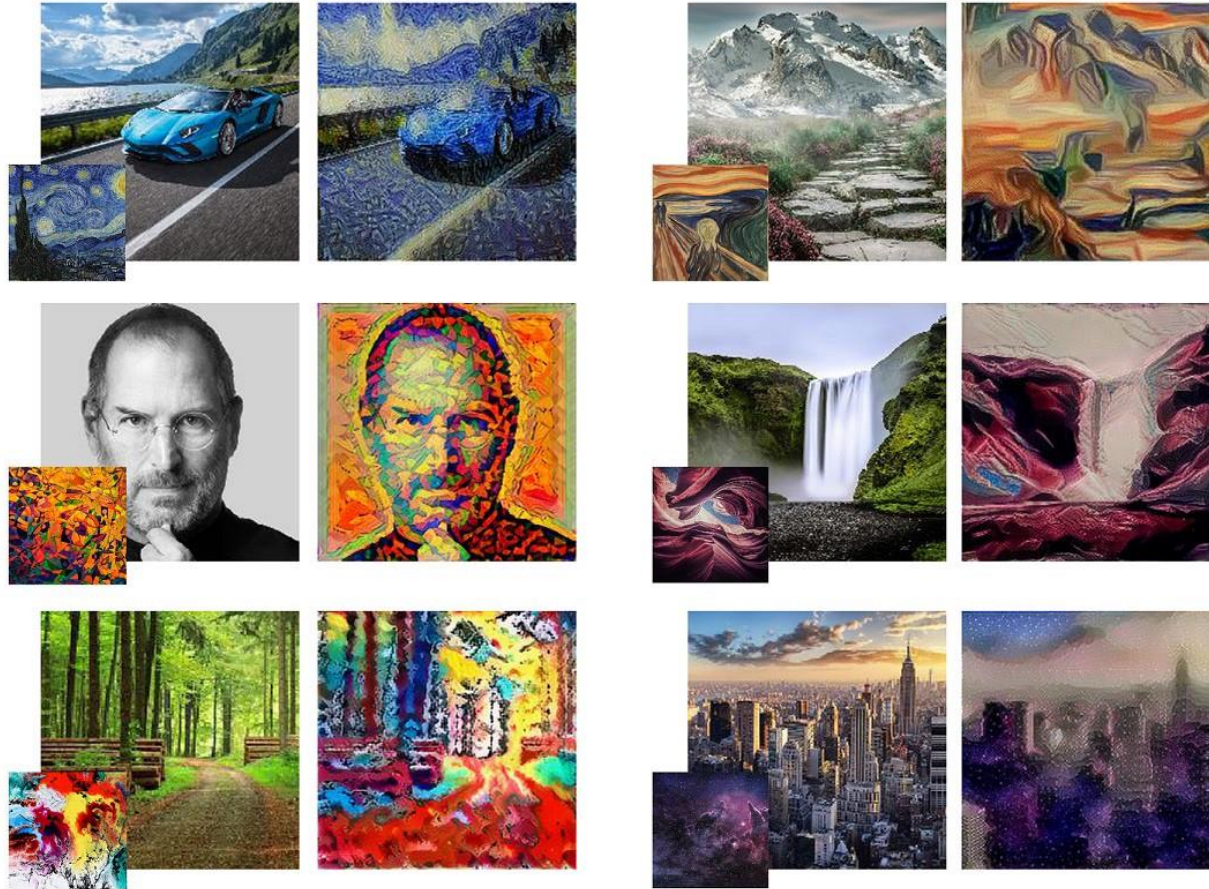
# Federated Learning

- We have multiple devices
  - Each device is small
  - It sees only a small biased dataset
  - Data is private and it cannot be shared
- How do we train a single model without sharing the data?
  - We want to minimize the communication overhead too



# Examples

# Style Transfer



# Example

- From <https://lambdalabs.com/blog/how-to-fine-tune-stable-diffusion-how-we-made-the-text-to-pokemon-model-at-lambda>
  - Starting from stable-diffusion v1.5
  - 2xA6000 GPUs, 15,000 steps, ~6 hours of training, at ~\$10
  - almost a thousand images, automatically captioned



*Girl with a pearl earring, Cute Obama creature, Donald Trump,  
Boris Johnson, Totoro, Hello Kitty*

# Textual Inversion

Input samples  $\xrightarrow{\text{invert}}$  “ $S_*$ ”

→

“An oil painting of  $S_*$ ”

“App icon of  $S_*$ ”

“Elmo sitting in the same pose as  $S_*$ ”

“Crochet  $S_*$ ”

Input samples  $\xrightarrow{\text{invert}}$  “ $S_*$ ”

→

“Painting of two  $S_*$  fishing on a boat”

“A  $S_*$  backpack”

“Banksy art of  $S_*$ ”

“A  $S_*$  themed lunchbox”

# DreamBooth – Subject-Driven Generation



Figure 1. With just a few images (typically 3-5) of a subject (left), *DreamBooth*—our AI-powered photo booth—can generate a myriad of images of the subject in different contexts (right), using the guidance of a text prompt. The results exhibit natural interactions with the environment, as well as novel articulations and variation in lighting conditions, all while maintaining high fidelity to the key visual features of the subject.



- Learning from small videos
- Non-iid minibatches
- Training on-device
- Can learn new classes over time



# AI Dungeon



You are Mario, a cyborg living in the the futuristic city of Zail. You have a bionic arm and a holoband. You're walking down the dark city streets while neon lights flash brightly above you. As you pass an alley you see a strange man standing behind a dumpster. He's holding a large yellow brick.

You step closer and the man quickly runs away. He's afraid of you! That's hilarious.

> You laugh and run towards him.

As you run past him, he throws the yellow brick at your head. It misses you but buries itself into the pavement.

> You collect the yellow brick.

"Ha ha!" You shout. "That was a good try, but it doesn't work on me." The brick lays right next to the dumpster.



Do ... 

# Hugging Face Models



**Hugging Face** Search models, datasets, users... Models Datasets S

**Tasks** Libraries Datasets Languages Licenses Other

Filter Tasks by name

**Multimodal**

- Feature Extraction
- Text-to-Image
- Image-to-Text
- Visual Question Answering
- Document Question Answering
- Graph Machine Learning

**Computer Vision**

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Image-to-Image
- Unconditional Image Generation
- Video Classification
- Zero-Shot Image Classification

**Natural Language Processing**

- Text Classification
- Token Classification

**Models** 135,622 Filter by name

- bert-base-uncased**  
Updated Nov 16, 2022 • ↓ 32.3M • ♥ 517
- gpt2**  
Updated Dec 16, 2022 • ↓ 19.9M • ♥ 584
- xlm-roberta-large**  
Updated Jun 27, 2022 • ↓ 10.2M • ♥ 85
- openai/clip-vit-large-patch14**  
Updated Oct 4, 2022 • ↓ 8.91M • ♥ 190
- bert-base-cased**  
Updated Nov 16, 2022 • ↓ 6.67M • ♥ 74
- microsoft/layoutlmv3-base**  
Updated Dec 13, 2022 • ↓ 5.76M • ♥ 72

# Course Structure

## **Learn how to train ML models with streaming data:**

- Nomenclature, definitions, problem settings
- Understanding the challenges of continual learning
- General solutions
- Practical applications of transfer/continual learning

## **Focus on:**

- 1D time series data (first module)
- DNN for Computer vision

## Methodology:

- Gain an intuitive understanding of the different types of domain shifts and the problems they cause
- Understand effective solutions to train on non-stationary data
- Ability to code such solutions (e.g. with PyTorch)

## Applications:

- Ability to reuse pretrained models for different applications. Examples:
  - Image classification -> style transfer
  - GPT-language-modeling -> Dungeon master
- Knowledge about efficient models that can be trained on-device

- Knowledge of **Machine Learning** fundamentals: Followed ML course
- Basic knowledge of **deep learning**:
  - You have trained a simple DL model at least once (e.g. CNN on MNIST with Keras)
  - Suggested: ISPR course.
- Basic programming experience in Python
- **Math**: anything that you needed for the ML/ISPR courses

None of these requirements are strict but they help a lot. If in doubt just ask me.

- **You know how to train a machine learning model given a dataset. Most practical problems don't fit this simple scenario.**
  - Define problems that go beyond «offline training»
  - Highlight the challenges
  - Show the methods to solve them
- We will start with simple models and build on top of them incrementally (from linear models to ChatGPT, almost)
- Focus on practical approaches, using the theory to complement and explain the empirical findings and methodologies



## Four Modules:

- Online Machine Learning
- Knowledge Transfer and Adaptation
- Deep Continual Learning
- Application and Frontiers of Continual Learning

Each module builds on top of the previous ones and focuses on different aspects of learning with nonstationary data.

*How do we learn a simple machine learning one sample at a time efficiently?*

- Learning one sample at a time
  - 1D/small-dimensional time series data
  - Simple models: linear classifier, knn, decision trees...
- Concept drift
- Online classification models
- Ensemble methods
- Practical examples with River

*How can we reuse a pretrained DNN for a novel task?*

- Deep learning with PyTorch
- Multi-Task Learning
- Transfer Learning and Domain Adaptation
- Self-supervised learning and large pretrained models
- Meta-learning, metric learning, few-shot learning
- Practical labs with PyTorch

*How can we retrain a DNN on a stream of data without forgetting?*

- Continual Learning: problem definition, settings, evaluation
- Catastrophic forgetting
- Methodologies
  - Replay
  - Regularization methods
  - Architectural strategies
- Practical labs with Avalanche

- Active Learning and Curriculum Learning
- Open set Recognition and out-of-distribution detection
- Federated Learning (guest lecture by Valerio De Caro)
  - Possibly with a lab
- Large Language Models and Continual Pretraining
- Invited seminars about research and applications in CL

Antonio Carta

- Email: [antonio.carta@unipi.it](mailto:antonio.carta@unipi.it)
- Office: Room 293
- Office hours: send me an email



# Course Schedule

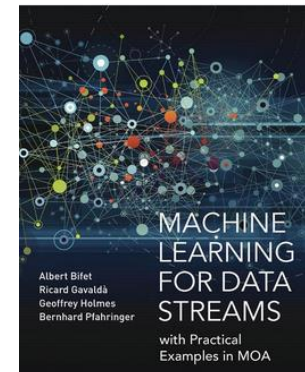
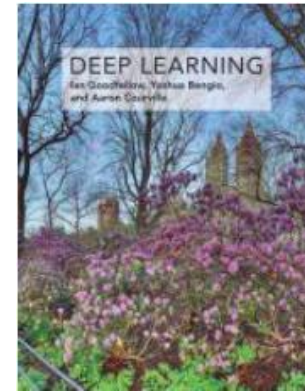


- Available on the master course website
  - Monday 11-13: I-Lab
  - Thursday 11-13: I-Lab
- 24 lectures
- In-person lectures

- The reference page is on Moodle:  
<https://elearning.di.unipi.it/course/view.php?id=1038>
- Course information
  - Slides
  - Articles and references
- Teams is also available but we will use it only for sharing onenote notebooks



- There is no single book
  - Deep learning preliminary knowledge:
    - Deep learning book, Goodfellow et al. ([free pdf](#))
  - OML Module:
    - Machine Learning for Data Streams with Practical Examples in MOA. Bifet et al.
      - <https://direct.mit.edu/books/book/4475/Machine-Learning-for-Data-Streamswith-Practical>
  - I will give you articles and references for the other modules
  - Some lecture notes are also in preparation
- All the books and references used are open access and available for free
- Detailed references at the end of each lecture



- ContinualAI is a non-profit organization with several resources about continual learning
- Course by Vincenzo Lomonaco:  
<https://course.continualai.org/>
- Avalanche: PyTorch-based library for continual learning
- Seminars and lectures on the ContinualAI youtube channel
- Others:
  - ContinualAI Wiki: a shared and collaboratively maintained knowledge base for Continual Learning: tutorials, workshops, demos, tutorials, courses, etc.
  - Continual Learning Papers: curated list of CL papers & books with meta-data by ContinualAI
  - ContinualAI Forum + Slack: discussions / Q&As about Continual Learning
  - ContinualAI Research Consortium: networks of Top CL Labs across the world.



- A python library for online machine learning
- API similar to scikit-learn
- We will use it for the first module
- <https://riverml.xyz/0.14.0/>

*River*

# Software Library – PyTorch



- Deep Learning library
- We will use it for the deep learning code



# Software Library – Avalanche



- PyTorch-based library for continual learning with deep neural networks
- We will use it in the Deep Continual Learning module
- Developed by ContinualAI and some people here at UNIPi, so feel free to ask any questions
- Docs: <https://avalanche.continualai.org/>
- Baseline experiments: <https://github.com/ContinualAI/continual-learning-baselines>



*powered by*



- **Final Project + Oral Exam**

- Final project: either a **written report** on a course-related topic or a **coding project**
  - Oral exam: a discussion of the final project and an examination on the course program.
- 
- There will be **no midterm**. If you want to finish the project earlier, you can start the project before the end of the course.
  - The **final vote** will be an average of the project and oral vote.
  - More detailed **rules** on the evaluation will be published on moodle in the following weeks.

- A **non-trivial implementation of a method** relevant for the course
  - From the suggested list or proposed by you
- **Python/PyTorch/Avalanche are suggested** but not mandatory
- **You can reuse code** from public libraries and open repositories
  - Ask for permission when you submit the project proposal
  - Write explicitly what codebases you used as reference, if any
  - You should understand the code you are using and be able to explain it
  - Don't trust random repositories on github
- **Evaluated on:**
  - Correctness of the implementation
  - Your own understanding of the code and ability to explain it
  - Experiments: you should provide a description and motivations for the experimental choices, such as benchmark used, hyperparameters, and comparison against the expected results from the literature

- A **written report/mini-review** on a course-related topic
  - List of **suggested topics** on moodle (coming soon)
  - You can **propose your own topic**, subject to my approval
  - Max 8 pages
  - More or less 5 papers not covered during lectures
- **Evaluated on:**
  - **Clarity** of the written report
  - **References:** you should add references that are not cited/explained during the course
  - **Understanding:** you should understand the reference material,
    - Can you identify the main points of the paper? Find connection between them?  
Open problems and limitations?



- First part: final project discussion:
  - Software: brief description of the codebase and experimental results.
  - Report: presentation of the report
- Second part: open questions on the course material

# Next Lecture



- Start of the Online ML module
  - What is OML
  - Prequential evaluation
- Notebook with example in River