MACHINE LEARNING

INTRODUCTION TO MACHINE LEARNING

Universidad Francisco de Vitoria
UFV Madrid

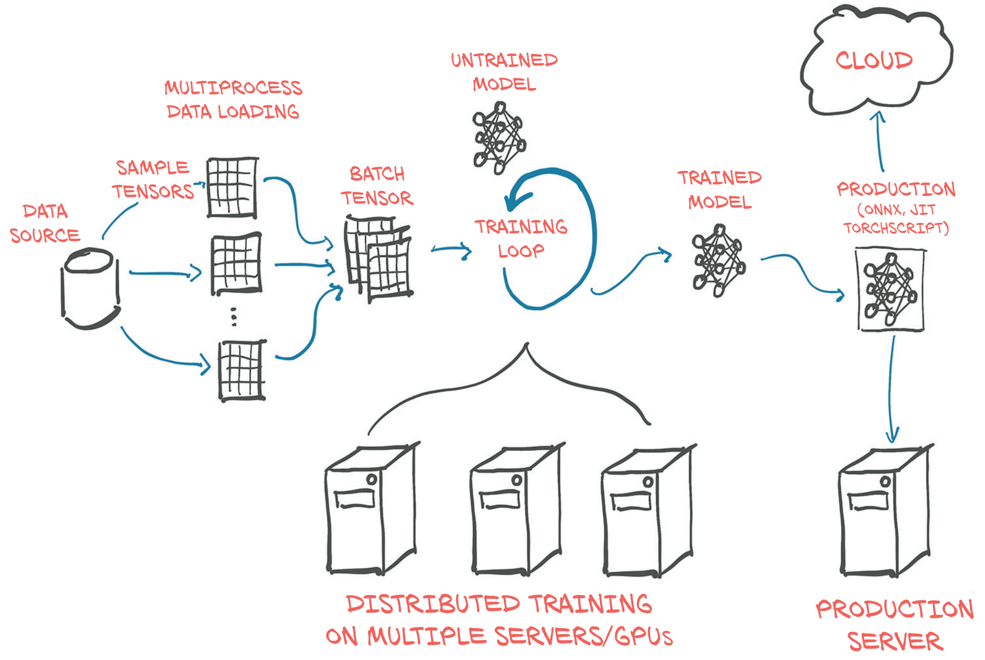
Introduction to ML

- *Overview*
- *Deep supervised, unsupervised and reinforcement learning*
- *Deep learning revolution*
- *Competitive landscape*
- *PyTorch overview*
- *Hardware and software requirements*

Introduction to ML

- *Overview*

What this course is about...



DATA SOURCE
SAMPLE TENSORS
MULTIPROCESS DATA LOADING
BATCH TENSOR
UNTRAINED MODEL
TRAINING LOOP
TRAINED MODEL
CLOUD
PRODUCTION (ONNX, JIT TORCHSCRIPT)
DISTRIBUTED TRAINING ON MULTIPLE SERVERS/GPUs
PRODUCTION SERVER

Data representation as **tensors** (multidimensional arrays)

Computational graphs as a representation of mathematical transformations of tensors.

Training, **loss function** and **optimizers**

Training **data management**

Neural net architecture

System architecture: hardware CPU/GPU/TPU and distributed training

Deployment of trained model

Introduction to ML

- *Overview*
- *Deep supervised, unsupervised and reinforcement learning*

ML: algorithms that improve through experience and data

Machine learning: design and implementation of of computer algorithms that **improve automatically using**:

- **Data;** and/or
- **Experience**: interaction with a real or simulated environment

Three broad **categories**:

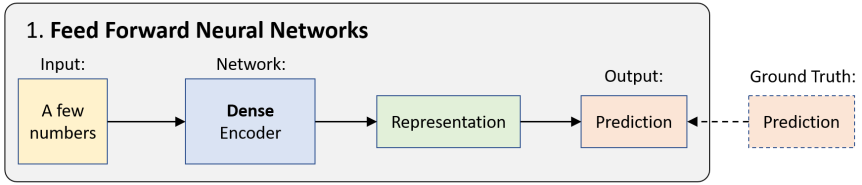
- **Supervised** learning
- **Unsupervised** learning
- **Reinforcement** learning

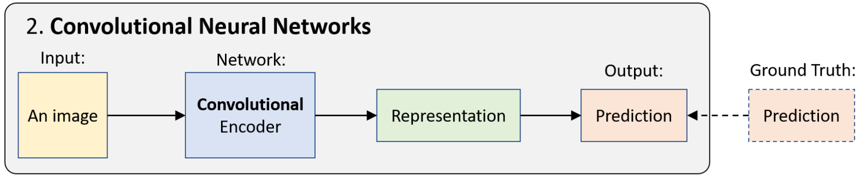
We will see examples of deep learning for each of these categories.

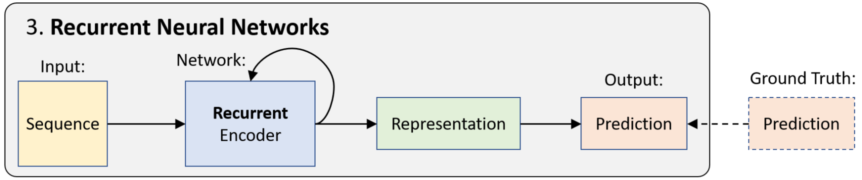
Overview

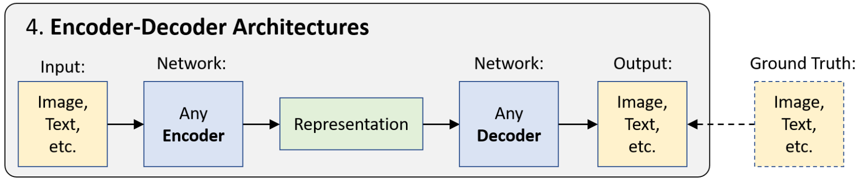
Universidad Francisco de Vitoria
UFV Madrid

Supervised Learning

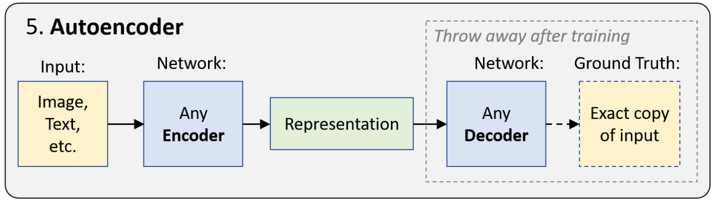


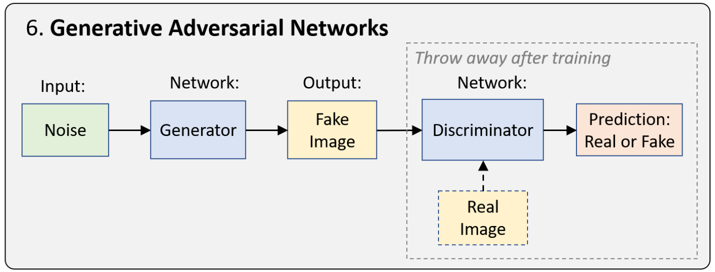




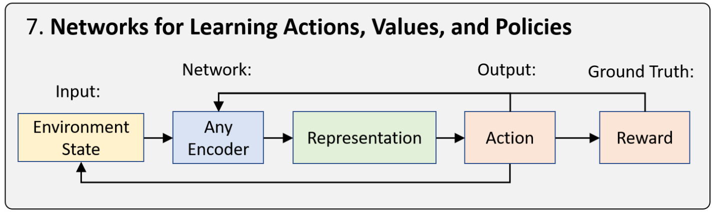


Unsupervised Learning





Reinforcement Learning



Deep supervised learning

Supervised learning: Learning a function that maps input to output based on a labeled dataset.

Trainig examples/observations: input as a vector or real numbers, desired output is also a value (otherwise, we need to develop a proper representation)

- Output is a real value: **regression**
- Output is one among a limited number of classes: **classification**

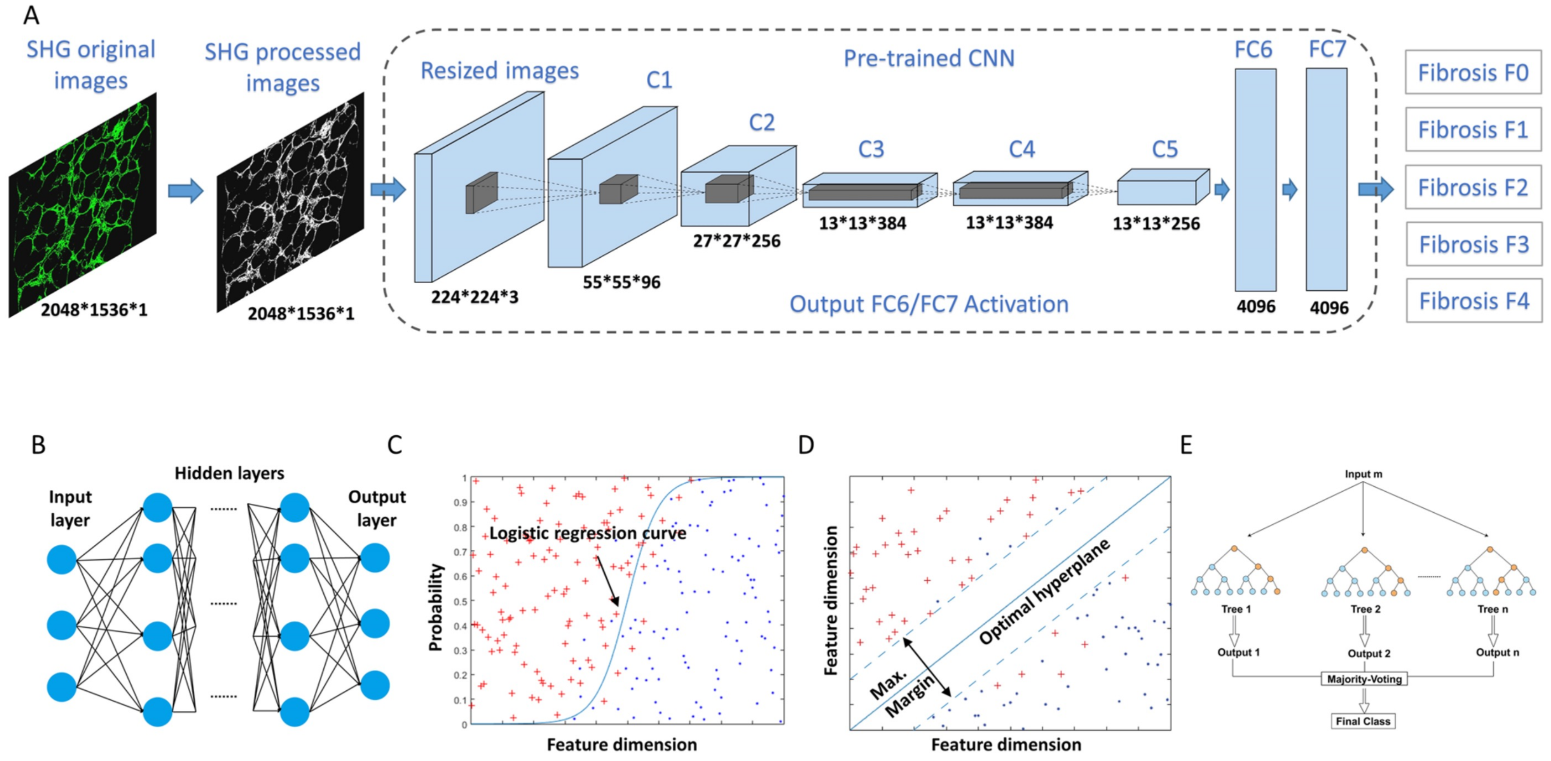
After training can then infer the values for new, unseen examples.

Examples of **deep** supervised learning:

- Natural language processing: e.g., translation, **automatic speech recognition**
- Medical: image processing, e.g., **detecting lung tumoral nodules**

Deep supervised learning

Example: Deep learning for automated scoring of liver fibrosis stages from microscopy images.



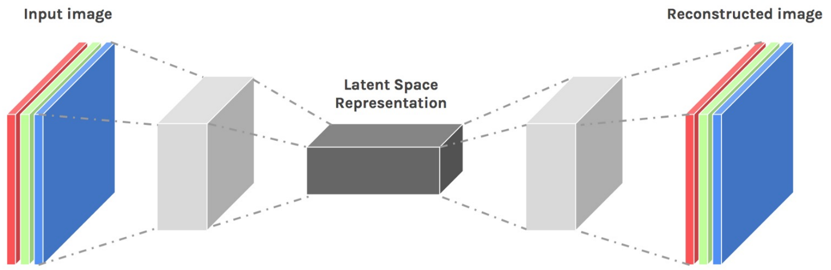
Deep unsupervised learning

Unsupervised learning: Data contains only inputs (not labeled)

- Objective is to find structure in data

Examples of **deep** unsupervised learning:

- **Autoencoder**: efficient data encodings (representations). Given a dataset, learn a low dimensional representation of the data by learning to ignore noise.
- Generative Adversarial Networks (**GAN**): learn to generate new examples ”similar” (i.e., hard to distinguish) from the original dataset. Example: new synthetic features.
- **Anomaly detection**: identify samples that are not “fit” the pattern of the data.



Example of an autoencoder

Deep reinforcement learning

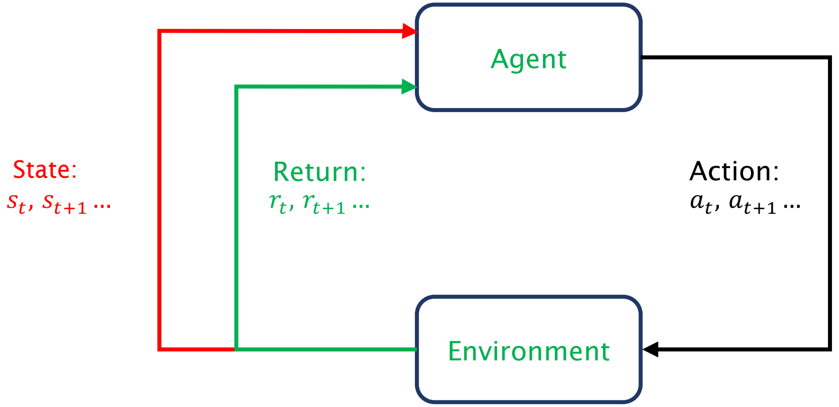
Reinforcement learning: Develop intelligent agents that take actions in an environment in order to maximize the (expected) cumulative reward.

Usually, the target model of the decision process that represents the environment is not known (or too complex to build a Markov Decision Process). **Learning happens through interaction.**

Key difference with supervised learning: **actions and data are coupled**. Exploration vs. exploitation.

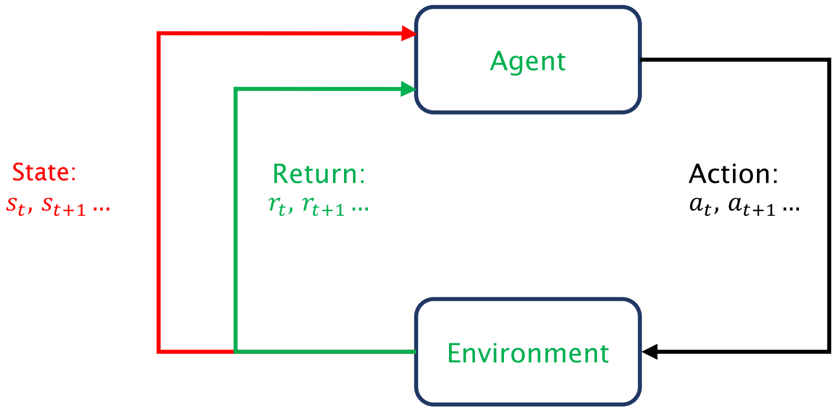
Examples:

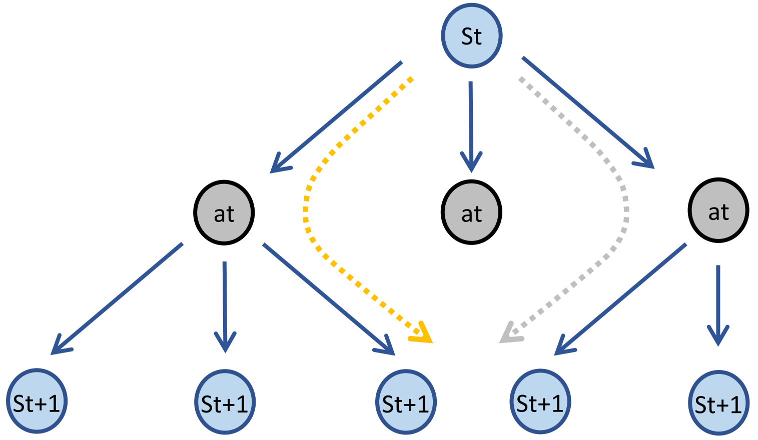
- Games (DOTA2, Go, chess)
- Personalized Recommendations (adapting)
- Robotics
- Traffic light control
- Chemistry (optimization of chemical reactions)


Agent
Environment
State: s_t, s_{t+1} …
Return: r_t, r_{t+1} …
Action: a_t, a_{t+1} …

Deep reinforcement learning

Deep RL: agents incorporate neural nets: for the **prediction of value, or to encode the policy** (mapping states into actions).


Agent
Environment
State: s_t, s_{t+1} …
Return: r_t, r_{t+1} …
Action: a_t, a_{t+1} …



Deep reinforcement learning

Example, DOTA 2

OpenAI Five wins back-to-back games versus Dota 2 world champions OG at Finals, becoming the **first AI to beat the world champions in an esports game**.

https://www.twitch.tv/videos/410533063?t=44m53s

Bill Gates
@BillGates

#AI bots just beat humans at the video game Dota 2. That's a big deal, because their victory required teamwork and collaboration – a huge milestone in advancing artificial intelligence.

https://openai.com/blog/openai-five/

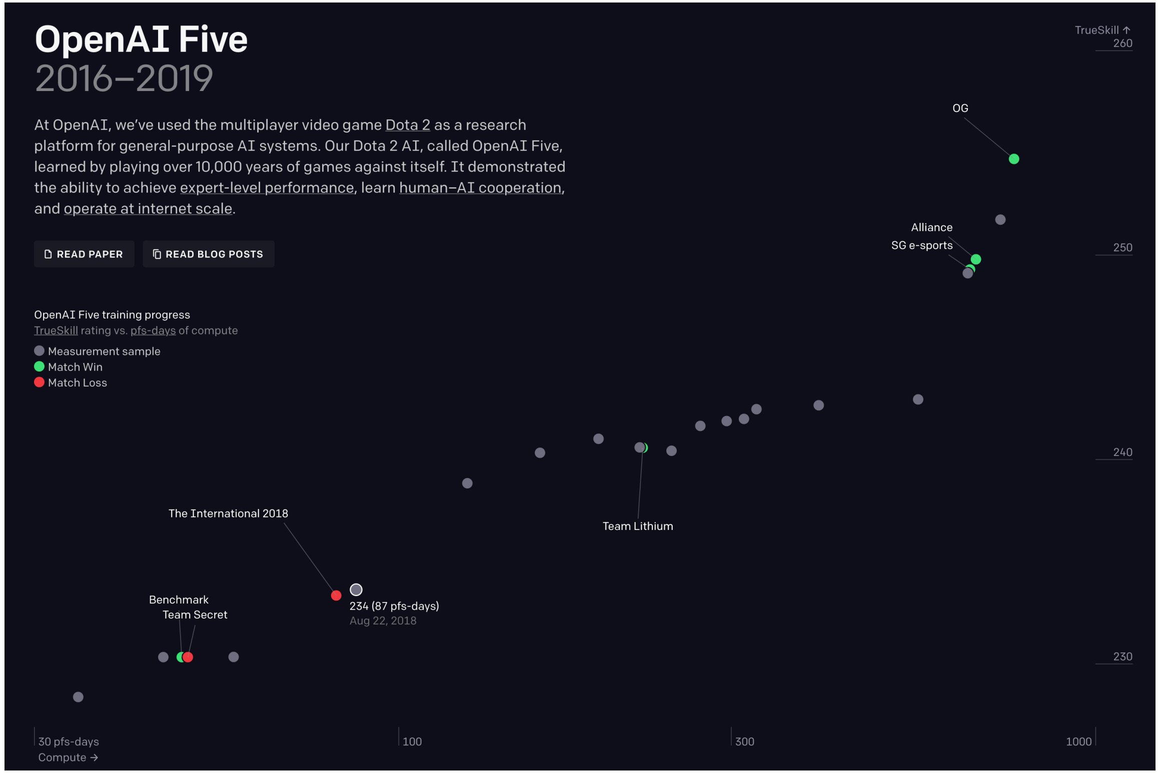
Dota 2 is a multiplayer online battle arena (MOBA) video game developed and published by Valve

Deep reinforcement learning

OpenAI Five

2016–2019

https://openai.com/five/



OpenAI Five
2016–2019

At OpenAI, we've used the multiplayer video game Dota 2 as a research platform for general-purpose AI systems. Our Dota 2 AI, called OpenAI Five, learned by playing over 10,000 years of games against itself. It demonstrated the ability to achieve expert-level performance, learn human–AI cooperation, and operate at internet scale.

Dota 2 is a multiplayer online battle arena (MOBA) video game developed and published by Valve

INTRODUCTION TO ML

- *Overview*
- *Deep supervised, unsupervised and reinforcement learning*
- *Deep learning revolution*

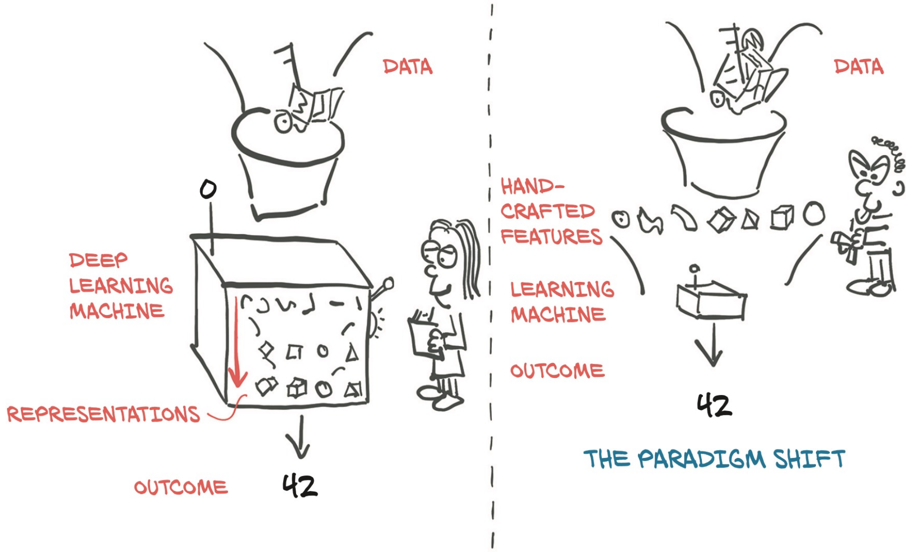
The deep learning revolution

Until the last decade, the broader class of systems that fell under the label ML relied heavily on **feature engineering**.

- Features are transformations on input data that facilitate a downstream algorithm, like a classifier, to produce correct outcomes on new data.

DL **finds the representations automatically**, from raw data

DL exchanges the need to handcraft features for an **increase in data and computational requirements**.



What do we mean by training?

We want to obtain useful representations and make the machine produce desired outputs

During training, we use a **criterion**, **a real-valued function of model outputs and reference data**, to provide a numerical score for the **discrepancy between the desired and actual output of our model**

- by convention, a lower score is typically better, and we use the term **loss**

Training consists of **driving the criterion toward lower and lower scores by incrementally modifying our deep learning machine until it achieves low scores, even on data not seen during training.**

Introduction to ML

- *Overview*
- *Deep supervised, unsupervised and reinforcement learning*
- *Deep learning revolution*
- *Competitive landscape*

The deep learning competitive landscape

Language:

- **Python**, by far (no doubt). Interpreted but PyTorch is written in C++ and CUDA, a C++-like language from NVIDIA. So heavy-lifting happens outside Python, and in specialized hardware.
- C++, not first choice. Only when running things very close to the hardware, like new code directly on GPUs. For inference, it is possible to export a python trained model to a C++ runtime.
- Java, R (small on purpose)

Libraries on Python:

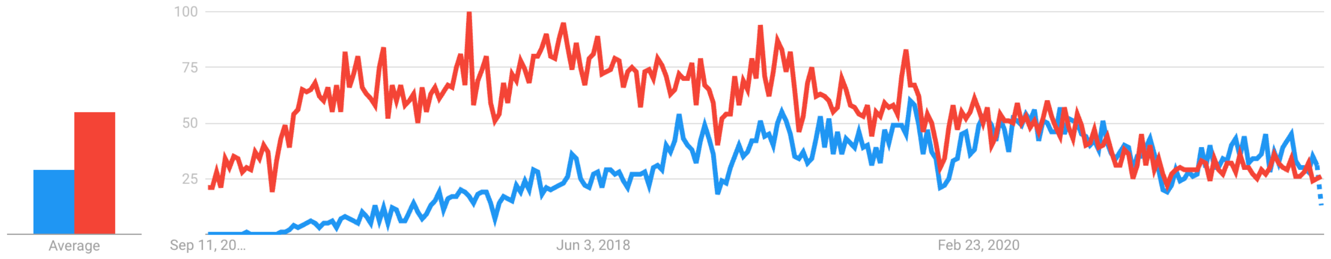
By 2021, the community largely consolidated behind either PyTorch or TensorFlow

- **TensorFlow** has a robust pipeline to production, an extensive industry-wide community, and massive mindshare.
- **PyTorch** initially, research and teaching communities, thanks to its ease of use. Now great momentum into industry.

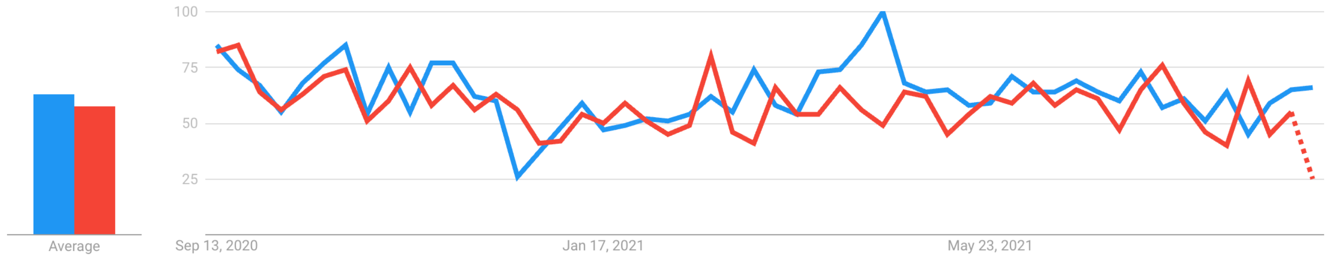
Search interest (red TensorFlow, blue PyTorch), last year, 2021

Universidad Francisco de Vitoria
UFV Madrid

Last 5 years



Last 12 months

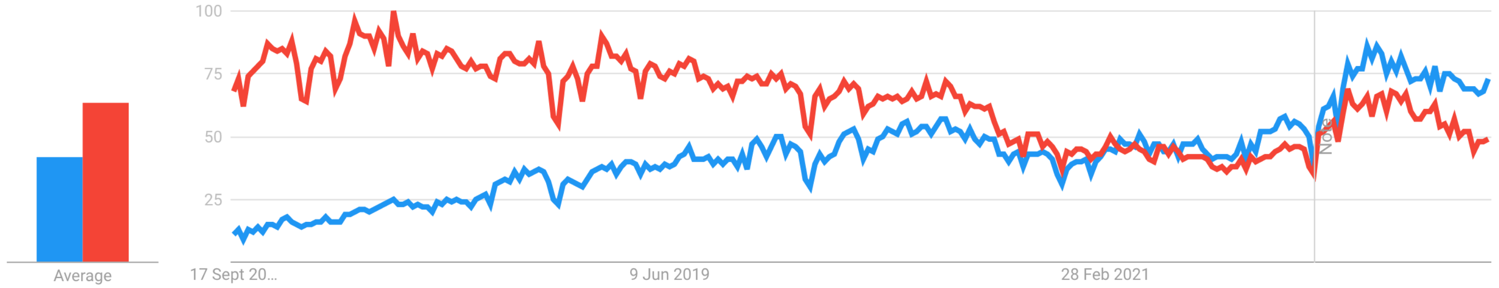


Search interest, this year, 2022

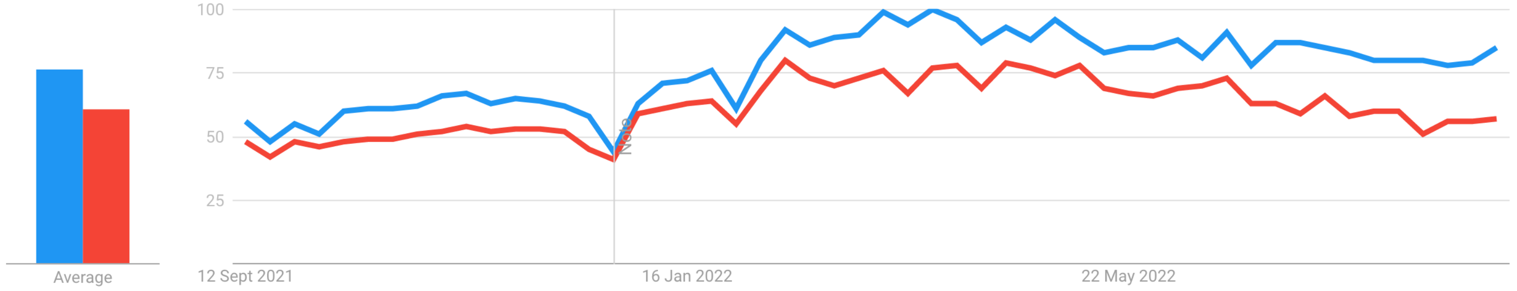
Universidad Francisco de Vitoria
UFV Madrid

● PyTorch ● TensorFlow

Last 5 years



Last 12 months

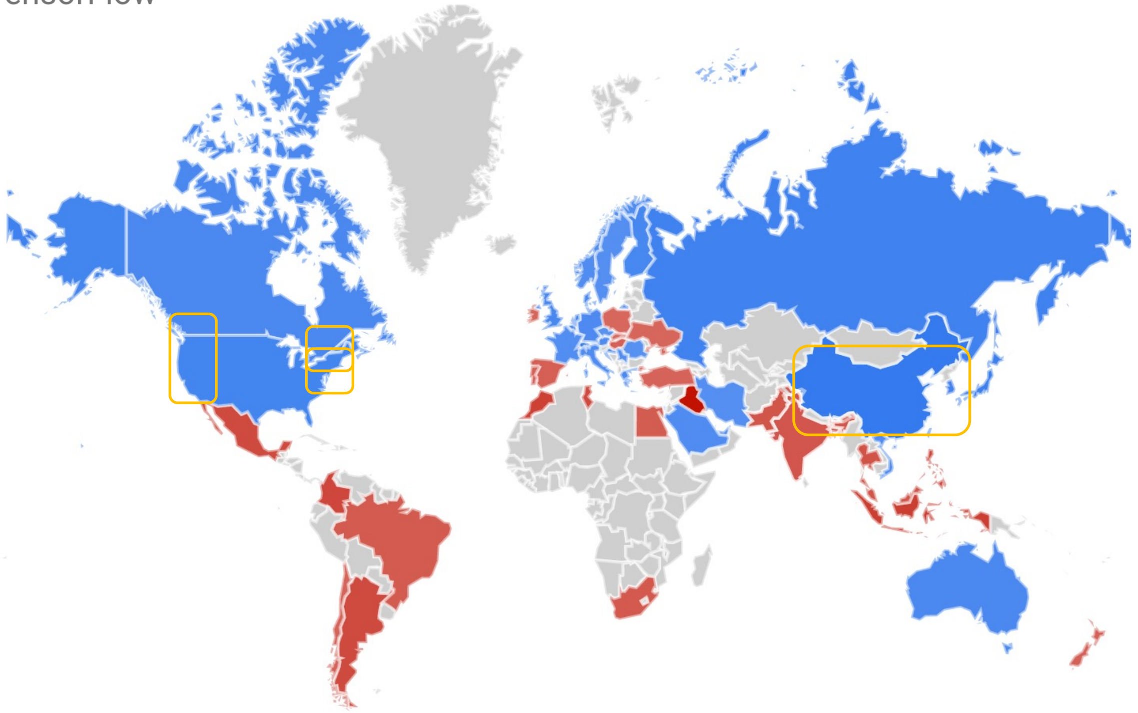


Search interest (September 2021 to September 2022)

Universidad Francisco de Vitoria

UFV Madrid

● PyTorch ● TensorFlow



Why PyTorch

Simplicity

Pythonic

History:

- Static (graph) execution vs. Dynamic execution. Flexibility vs. Speed.
- Now both operation modes are possible with TF and PyTorch.

Features of PyTorch and TF have mostly converged.

INTRODUCTION TO ML

Computational graphs, automatic differentiation, optimization

PyTorch at its core, is all about tensors:

- Tensors as **Data**:
  - *multidimensional arrays*, or *tensors* and an extensive library of operations on them
  - Both tensors and the operations on them can be used on the CPU or the GPU/TPU.
- Tensors as part of a **Computational Graph**:
  - ability of tensors to keep track of the operations performed on them, building a CG
- Automatic differentiation and numerical optimization:
  - **compute derivatives of an output** of a computation with respect to any of its inputs.
  - used for **numerical optimization**, and it is provided natively through *autograd* engine under the hood.

Building neural nets with PyTorch

Neural net architectural components

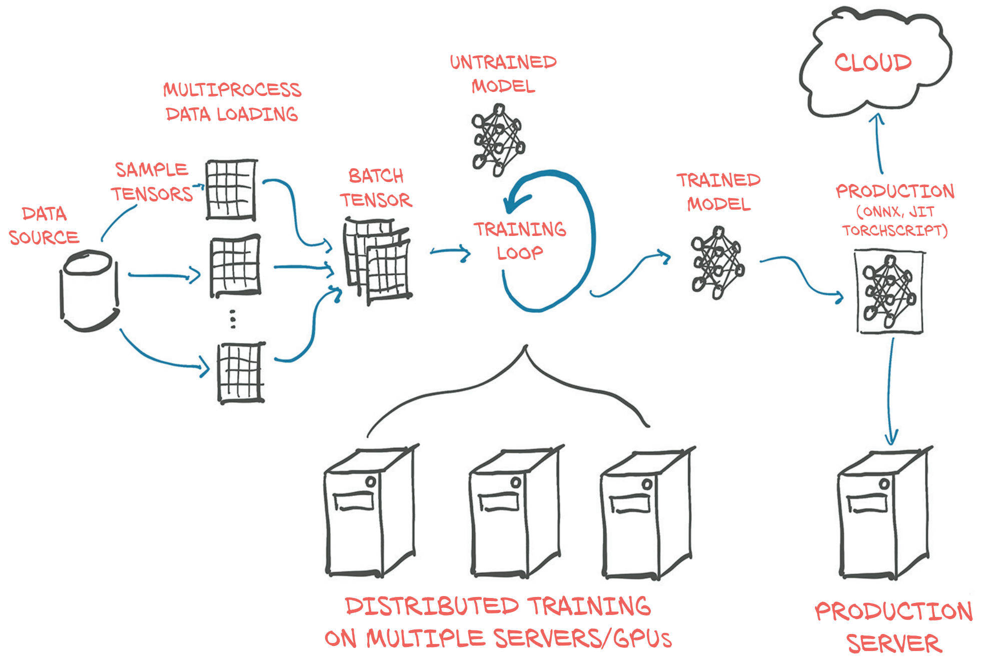
- Modules for building neural networks are located in `torch.nn`, which provides common neural network layers and other architectural components.
- Fully connected layers, convolutional layers, activation functions, and loss functions can all be found here

Training data management

Optimizers

Transfer of data between hardware

Overview


DATA SOURCE
SAMPLE TENSORS
MULTIPROCESS DATA LOADING
BATCH TENSOR
UNTRAINED MODEL
TRAINING LOOP
TRAINED MODEL
PRODUCTION (ONNX, JIT TORCHSCRIPT)
CLOUD
DISTRIBUTED TRAINING ON MULTIPLE SERVERS/GPUs
PRODUCTION SERVER

Universidad Francisco de Vitoria
UFV Madrid

Introduction to ML

- *Overview*
- *Deep supervised, unsupervised and reinforcement learning*
- *Deep learning revolution*
- *Competitive landscape*
- *PyTorch overview*
- *Hardware and software requirements*

Hardware and software requirements

Training:

Simple models we will use in class, any recent laptop or personal computer

For the more advanced models (optional):

- 2 x GPU 6-8 GB RAM each one (for example, GTX 1660)
- 200 GB disk

Inference:

Any recent laptop or personal computer

Google Colaboratory (https://colab.research.google.com)