Twitter Thread by Simone Scardapane

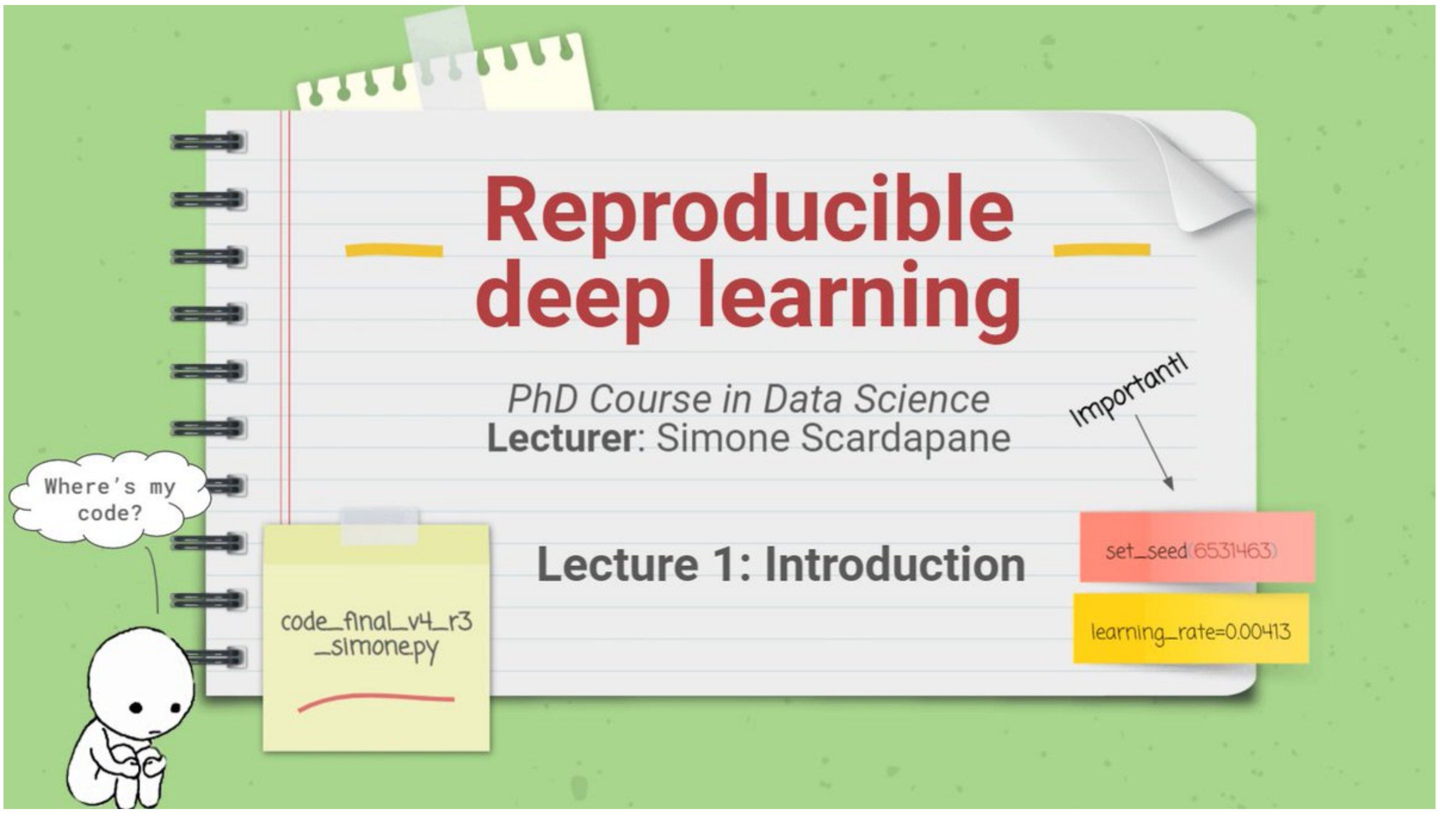
Simone Scardapane
@s_scardapane

Reproducible Deep Learning

The first two exercises are out!

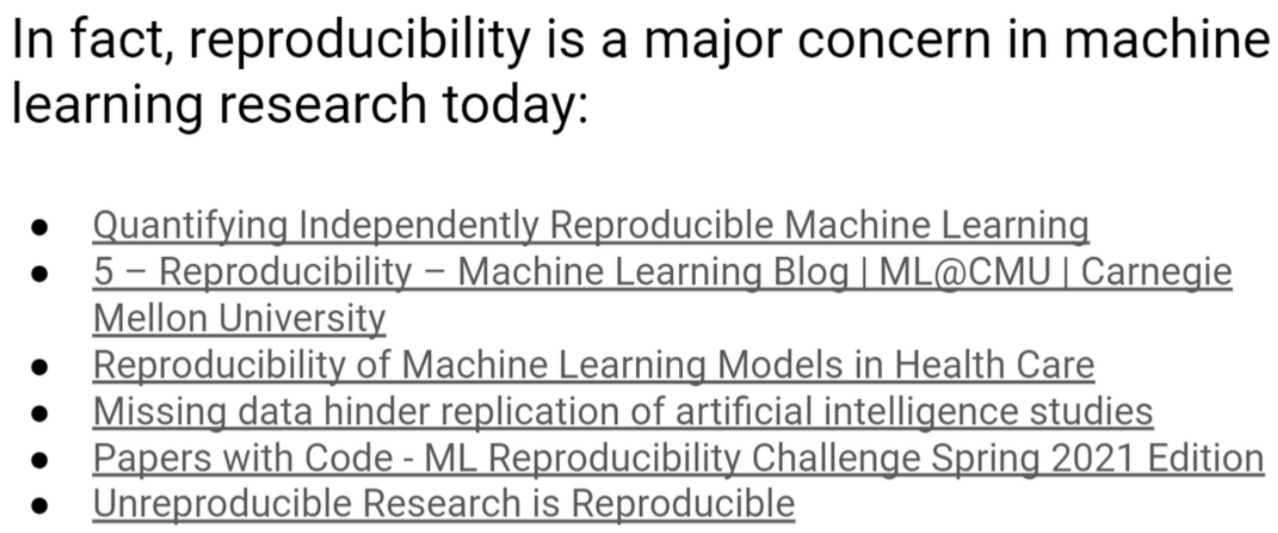
We start quick and easily, with some simple manipulation on Git branches, scripting, audio classification, and configuration with @Hydra_Framework.

Small thread with all information ■ /n


Reproducible deep learning
PhD Course in Data Science
Lecturer: Simone Scardapane
Lecture 1: Introduction
Important!
set_seed(6531463)
learning_rate=0.00413
code_final_v4_r3_simone.py
Where's my code?

Reproducibility is associated to production environments and MLOps, but it is a major concern today also in the research community.

My biased introduction to the issue is here: https://t.co/PqWH6uL5eT


In fact, reproducibility is a major concern in machine learning research today:

- Quantifying Independently Reproducible Machine Learning
- 5 – Reproducibility – Machine Learning Blog | ML@CMU | Carnegie Mellon University
- Reproducibility of Machine Learning Models in Health Care
- Missing data hinder replication of artificial intelligence studies
- Papers with Code - ML Reproducibility Challenge Spring 2021 Edition
- Unreproducible Research is Reproducible

The local setup is on the repository: https://t.co/9mhtZoJhE9

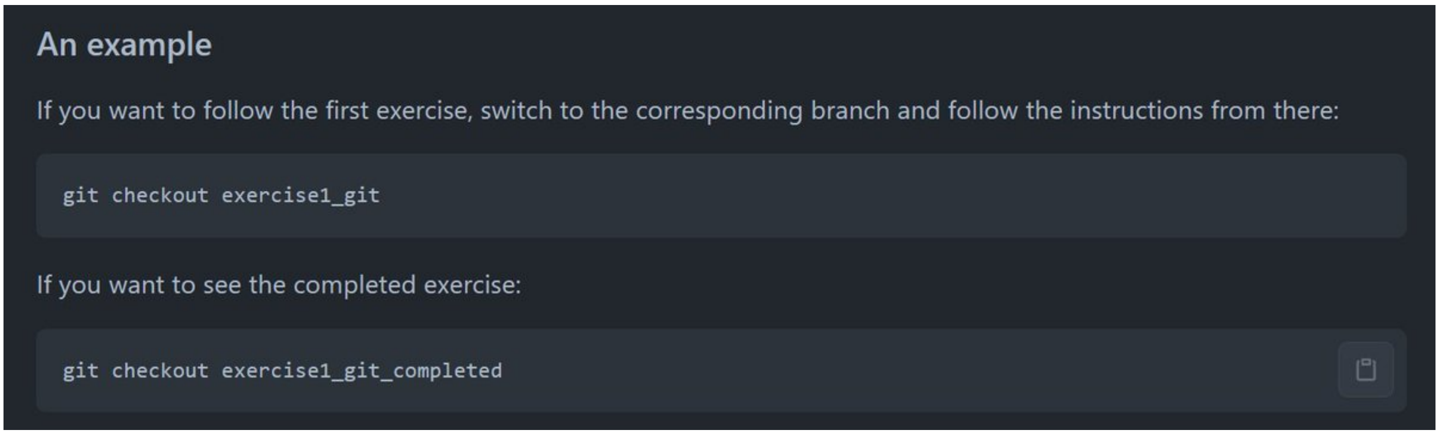
The use case for the course is a small audio classification model trained on event detection with the awesome @PyTorchLightnin library.

Feel free to check the notebook if you are unfamiliar with the task. /n

I spent some time understanding how to make the course as modular and "reproducible" as possible.

My solution is to split each exercise into a separate Git branch containing all the instructions, and a separate branch with the solution.

Two branches for now (Git and Hydra). /n


An example

If you want to follow the first exercise, switch to the corresponding branch and follow the instructions from there:

```
git checkout exercise1_git
```

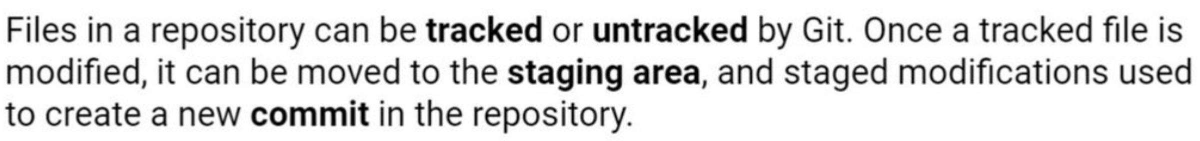
If you want to see the completed exercise:

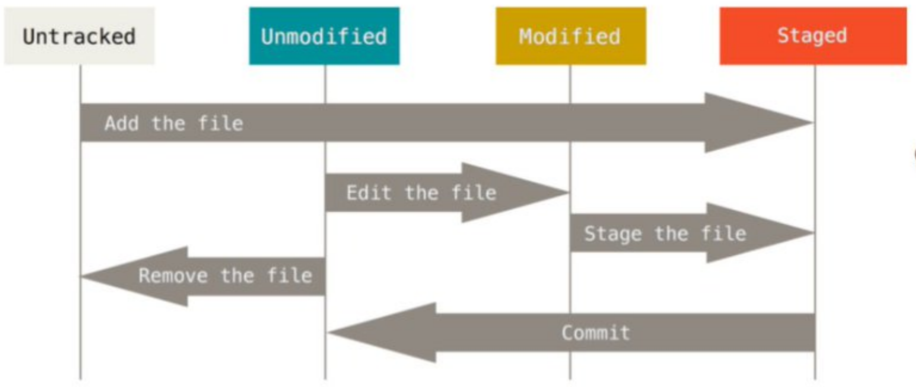
```
git checkout exercise1_git_completed
```

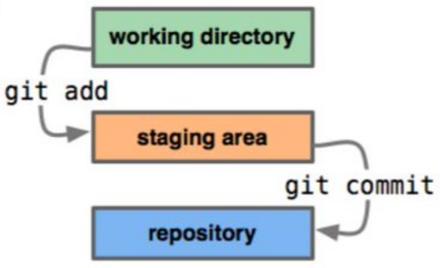
How well do you *really* know Git? The more I learn, the more I find it incredible.

I summarized most of the information on a separate set of slides: https://t.co/6dSmK3IfWB

Be sure to check them out before continuing! /n


Files in a repository can be **tracked** or **untracked** by Git. Once a tracked file is modified, it can be moved to the **staging area**, and staged modifications used to create a new **commit** in the repository.





2.2 Git Basics - Recording Changes to the Repository

Exercise 1 is a simple example of turning a notebook into a working script.

To make things more interesting, you have to complete the exercise while working on a separate Git branch!

https://t.co/35y9NQaVtz

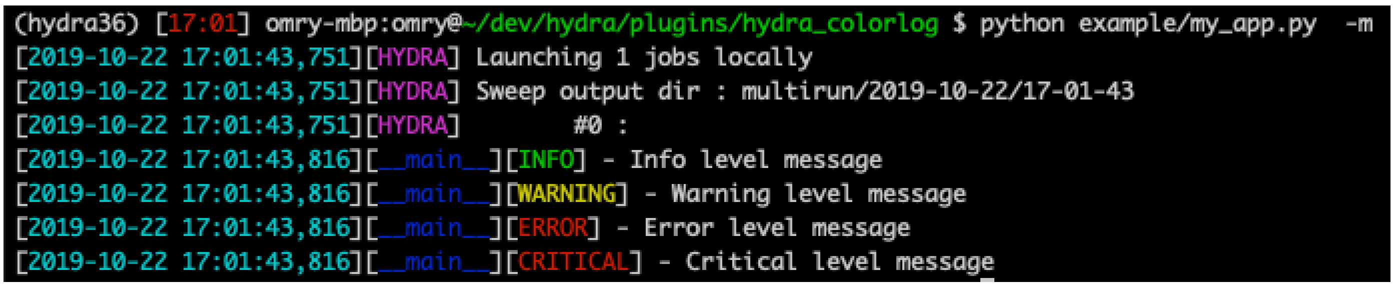
Nothing incredible, but it is always good to start small /n

Once you have a working training script, it is time to add some "bell and whistles"!

My must-have is some external configuration w/ @Hydra_Framework. Exercise 2 guides you in all the required steps.

Plus side: colored logging!

https://t.co/WyAIMXASRl


```
(hydra36) [17:01] omry-mbp:omry@~/dev/hydra/plugins/hydra_colorlog $ python example/my_app.py  -m
[2019-10-22 17:01:43,751][HYDRA] Launching 1 jobs locally
[2019-10-22 17:01:43,751][HYDRA] Sweep output dir : multirun/2019-10-22/17-01-43
[2019-10-22 17:01:43,751][HYDRA]        #0 :
[2019-10-22 17:01:43,816][__main__][INFO] - Info level message
[2019-10-22 17:01:43,816][__main__][WARNING] - Warning level message
[2019-10-22 17:01:43,816][__main__][ERROR] - Error level message
[2019-10-22 17:01:43,816][__main__][CRITICAL] - Critical level message
```

That is all for the moment. The next exercises will explore having external data versioning with @DVCorg and complete isolation with @Docker.

You can follow by starring the official repository, or here on Twitter. ■

https://t.co/9mhtZoJhE9