AgroParisTech INRAE

MLOps for Reproducibility and Traceability of Machine Learning Experiments

Structuring Research Projects with UV, Hydra and MLflow

Julian Agudelo and Joseph Allyndrée

MIA
PARIS-SACLAY
EKINOCS

23/04/2026

Outline

1. Motivations
2. MLOps Tools
3. UV
4. Hydra
5. MLflow
6. Hands-on

Motivations

The Reproducibility Challenge

Reproducibility

The ability for a third party to reproduce results by reusing a described experimental setup.

The Reproducibility Challenge

Reproducibility

The ability for a third party to reproduce results by reusing a described experimental setup. This de facto implies different levels:

- Methods (describing the protocols used in detail)
- Data (accessible data)
- Code (data processing)

The Reproducibility Challenge

Reproducibility

The ability for a third party to reproduce results by reusing a described experimental setup. This de facto implies different levels:

- Methods (describing the protocols used in detail)
- Data (accessible data)
- Code (data processing)

Why does it matter that research be reproducible?

The Reproducibility Challenge

The science produced by research must be correct!

- Reproducibility crisis around 2015
- Landmark paper: “Scientists lift the lid on reproducibility”

This undermines the **credibility** of research and prevents science from **building** on past developments!

Baker 2016; Collaboration 2015

The Traceability Challenge in Deep Learning

Classical code is explicitly programmed

- requirement $\rightarrow$ feature $\rightarrow$ code

The Traceability Challenge in Deep Learning

Classical code is explicitly programmed

- requirement $\rightarrow$ feature $\rightarrow$ code

A neural network is implicitly learned

- The final logic is distributed between the code and the learned parameters
- Development process through “trial and error”

The Traceability Challenge in Deep Learning

Classical code is explicitly programmed

- requirement $\rightarrow$ feature $\rightarrow$ code

A neural network is implicitly learned

- The final logic is distributed between the code and the learned parameters
- Development process through “trial and error”
  - Data
  - Preprocessing pipeline
  - Model architecture
  - Hyperparameters
  - Training regime

Aravantinos and Diehl 2019

Artifacts and Traceability

Artifact

Document delivered with the software

- Source code
- Executable code
- Test results

Artifacts and Traceability

Artifact

Document delivered with the software

- Source code
- Executable code
- Test results

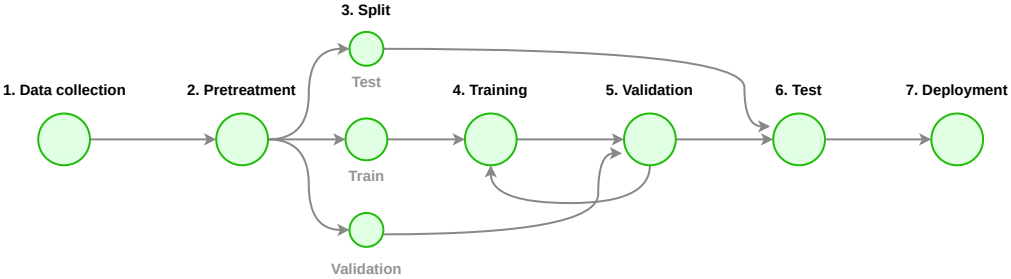
Traceability

Artifacts are linked to one another
e.g.: the executable code is derived from the source code

- **Trace**: relationship between two artifacts
- **Traceability**: set of artifacts and their relationships

In addition, we must consider the challenge of version control! Aravantinos and Diehl 2019

Artifacts in Deep Learning



Training:

1. Collect and preprocess data
   Raw dataset and preprocessing functions

Training:

1. Collect and preprocess data
   Raw dataset and preprocessing functions
2. Split the dataset
   Train, validation and test datasets

Artifacts in Deep Learning

Training:

1. Collect and preprocess data
   Raw dataset and preprocessing functions
2. Split the dataset
   Train, validation and test datasets
3. Design the architecture
   Architecture

Artifacts in Deep Learning

Training:

1. Collect and preprocess data
   Raw dataset and preprocessing functions
2. Split the dataset
   Train, validation and test datasets
3. Design the architecture
   Architecture
4. Define the “training protocol”
   Loss function and training parameters (e.g. hyperparameters, dropout, learning rate, optimizer)

Artifacts in Deep Learning

Training:

1. Collect and preprocess data
   Raw dataset and preprocessing functions
2. Split the dataset
   Train, validation and test datasets
3. Design the architecture
   Architecture
4. Define the “training protocol”
   Loss function and training parameters (e.g. hyperparameters, dropout, learning rate, optimizer)
5. Train the model
   Learned parameters

Aravantinos and Diehl 2019

Inference:

1. Post-process the trained network (e.g. remove dropout)
   Inference architecture

Inference:

1. Post-process the trained network (e.g. remove dropout)
   Inference architecture
2. Test the model on the validation set
   Real numbers representing accuracy

Inference:

1. Post-process the trained network (e.g. remove dropout)
   Inference architecture
2. Test the model on the validation set
   Real numbers representing accuracy
3. Modify the architecture or training configuration (3–4) based on the results

Inference:

1. Post-process the trained network (e.g. remove dropout)
   Inference architecture
2. Test the model on the validation set
   Real numbers representing accuracy
3. Modify the architecture or training configuration (3–4) based on the results
4. Evaluate quality using the test set
   Final metrics

Aravantinos and Diehl 2019

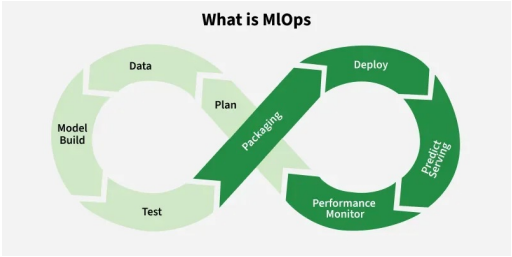
MLOps: Machine Learning Operations

Track, trace and supervise the training and deployment of AI models at scale.

TRACEABILITY

MLOps

- Data changes continuously
- Hyperparameters change
- Model architectures evolve
- Different testing protocols
- Monitoring once deployed



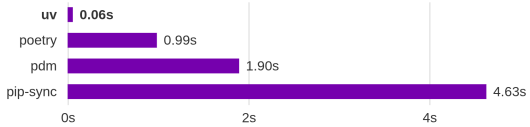
In spaghetti code, the relations between the pieces of code are so tangled that it is nearly impossible to add or change something without unpredictably breaking something somewhere else.

MLOps Tools

MLOps Tools for Managing Scientific Code

UV

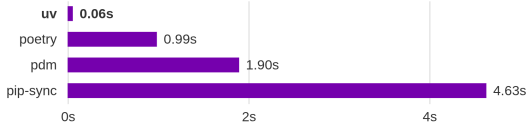
UV, an All-in-One Manager



- Package manager
  `pip, poetry`

Link to uv documentation: https://docs.astral.sh/uv/

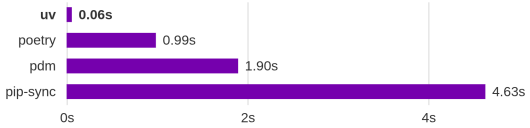
UV, an All-in-One Manager



- Package manager
  `pip, poetry`
- Virtual environment manager
  `venv, conda`

Link to uv documentation: https://docs.astral.sh/uv/

UV, an All-in-One Manager



- Package manager
  `pip, poetry`
- Virtual environment manager
  `venv, conda`
- Dependency resolver
  `pip`

Link to uv documentation: https://docs.astral.sh/uv/

The Philosophy Behind UV

Versioned, Reproducible and Portable

- Specify the exact version of the interpreter
- Declarative project configuration
- Promote reproducible collaboration
- Treat environments as part of the project

Versioned, Reproducible and Portable

UV Compared to Other Reproducibility Tools

Advantages

- Minimal installation and setup
- Extremely fast learning curve
- Starting a new project is just as easy as picking up someone else's

UV Compared to Other Reproducibility Tools

Advantages

- Minimal installation and setup
- Extremely fast learning curve
- Starting a new project is just as easy as picking up someone else's

Disadvantages

- Reproducibility starts at the Python version level
- Partial reproducibility across operating systems

UV vs Docker

	UV	docker
Insulation	Python	Système
Management	Dependencies	Environment
Insulation	Venv	Container OS
Use	Development	Deployment
Performance	Fast	Heavier

UV vs Docker

	UV	docker
Insulation	Python	Système
Management	Dependencies	Environment
Insulation	Venv	Container OS
Use	Development	Deployment
Performance	Fast	Heavier

Currently, UV is used as a Python package manager inside Docker
It is 10 to 100 times faster than pip!

Hydra

What is Hydra?

Two main features:

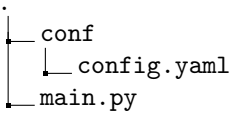
1. Create a hierarchical configuration through composition
2. Override it via configuration files and the command line

Link to Hydra documentation: https://hydra.cc/

Yadan 2019

A Minimal Example of Hydra Usage

1. **A configuration folder**



A Minimal Example of Hydra Usage

1. **A configuration folder**

```
.
├── conf
│   └── config.yaml
└── main.py
```

2. **Configurable parameters**

```
### conf/config.yaml

batch_size: 10
lr: 1e-4
```

A Minimal Example of Hydra Usage

1. A configuration folder

```
.
├── conf
│   └── config.yaml
└── main.py
```

2. Configurable parameters

```
### conf/config.yaml

batch_size: 10
lr: 1e-4
```

3. A script that uses the configuration

```python
import hydra

@hydra.main(config_path="conf", config_name="config")
def func(cfg: DictConfig):
    # Print the parameters, for example
    print(f"The batch size is {cfg.batch_size}")
    print(f"The learning rate is {cfg.lr}")

if __name__ == "__main__":
    func()
```

A Minimal Example of Hydra Usage

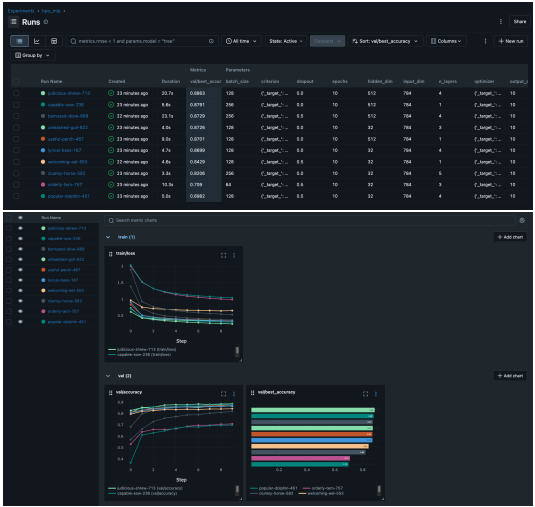
3. A script that uses the configuration

```python
import hydra

@hydra.main(config_path="conf", config_name="config")
def func(cfg: DictConfig):
    # Print the parameters, for example
    print(f"The batch size is {cfg.batch_size}")
    print(f"The learning rate is {cfg.lr}")

if __name__ == "__main__":
    func()
```

4. A simple override syntax

```
uv run script.py batch_size=32
```

MLflow



The cornerstone:

- Explicit logging in code
- Artifact management
- Intuitive graphical interface

Link to MLflow documentation: https://mlflow.org/get-started

Zaharia et al. 2018

A Minimal Example of MLflow Usage

```python
import mlflow

def training_loop():
    mlflow.set_experiment(exp_name) # instantiating mlflow
    with mlflow.start_run(run_id=id): # logging performance related to a run id
        ...
        mlflow.log_metrics(
                {"test/accuracy": float(accuracy),
                "test/macro_f1": float(f1_score), }

        # logging artifacts
        mlflow.log_figure(fig, "confusion_matrix.png")
```

Hands-on

Getting Started with UV

Installation

```
# For Unix based systems
$ curl -LsSf https://astral.sh/uv/install.sh | sh

# For Windows
$ powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

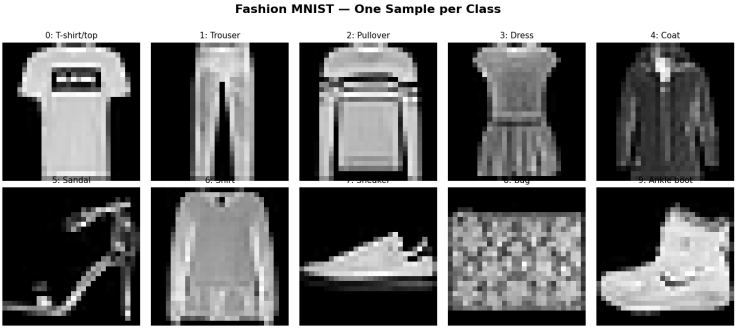
A few commands to get familiar with uv:

1. `uv init`
2. `uv add {package_name}`
3. `uv remove {package_name}`
4. `uv sync`
5. `uv run {script.py}`

Hands-on Session Structure

1. Showcase of a complete Logistic Regression learning project
2. Hands-on with a project using an MLP and a CNN
3. Competitive hands-on on Fashion-MNIST

Fashion-MNIST Dataset Presentation



10 classes; 28x28x1 (grayscale: from 0 (black) to 255 (white)) images!

Link to the Fashion-MNIST dataset: Fashion MNIST

Xiao, Rasul, and Vollgraf 2017

Hands-on – Logistic Regression

Objectifs

1. Identify the structural building blocks (project architecture) in the code
2. Identify the code required to integrate Hydra and MLFlow
3. Launch multiple experiments from the terminal using Hydra's syntax
4. Launch the MLflow graphical interface, explore the experiments and make comparisons

Definition: Logistic Regression

Assuming the target y_i takes values in $\{0, 1\}$ for data point i. Once fitted, a LR predicts the probability $P(y_i = 1|X_i)$ of the positive class as:

$$\hat{p}(X_i) = \frac{1}{1 + \exp(-X_i w - w_0)}$$

Definition: Logistic Regression

Assuming the target y_i takes values in $\{0, 1\}$ for data point i. Once fitted, a LR predicts the probability $P(y_i = 1|X_i)$ of the positive class as:

$$\hat{p}(X_i) = \frac{1}{1 + \exp(-X_i w - w_0)}$$

Optimization Problem

As an optimization problem, binary LR with a regularization term $r(w)$ and inverse regularization strength C minimizes the following cost function:

$$\min_w \sum_{i=1}^{n} (-y_i \log(\hat{p}(X_i)) - (1 - y_i) \log(1 - \hat{p}(X_i))) + \frac{r(w)}{C}$$

Hands-on – Logistic Regression

The code for the logistic regression example is available here:

Github

https://github.com/J-ally/Deep_repro_LR

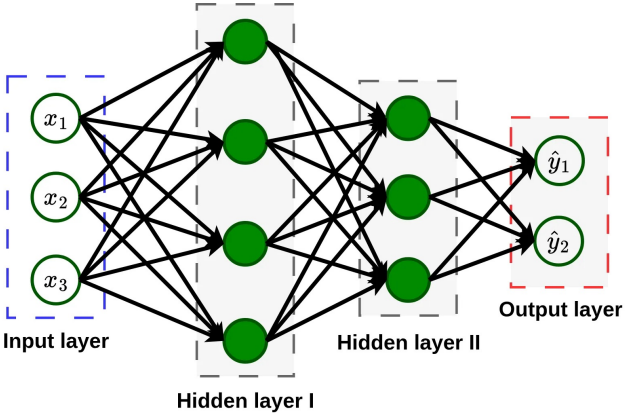
```
$ git clone https://github.com/J-ally/Deep_repro_LR.git
```

Hands-on – Neural Architectures

Objectifs

1. Complete the “##TODO##” code stubs
2. Run model training experiments
3. Verify experiments with MLflow
   inspect training artifacts and performance metrics
4. Evaluate the trained models

Multi-Layer Perceptron (MLP)



Convolutional Neural Network (CNN)

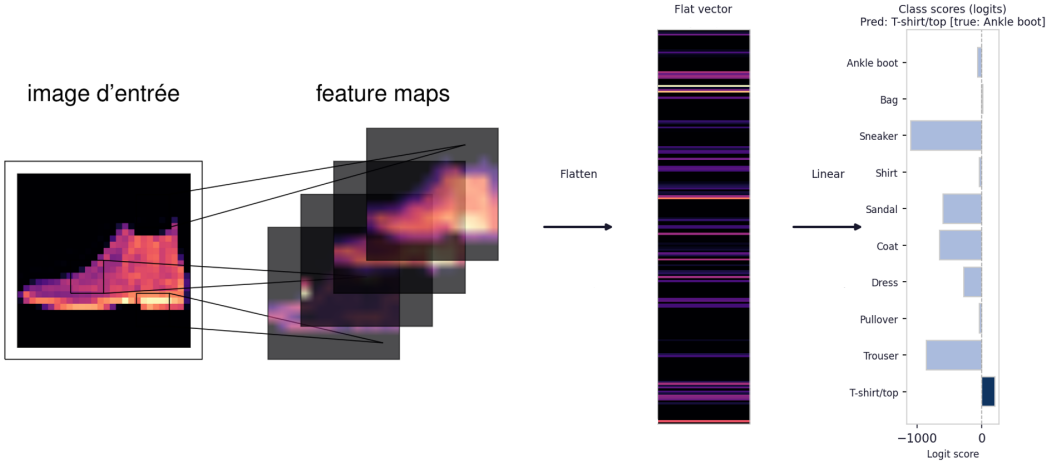


Illustration of a single convolution for multi-class prediction

Hands-on – Neural Architectures

The code for the MLP and CNN neural architecture examples is available here:

Github

https://github.com/J-ally/Deep_repro

```
$ git clone https://github.com/J-ally/Deep_repro.git
```

Once familiar with the code:

Experiment with the code to achieve the best results on the task!

All results will be logged to a shared MLflow instance displaying participants' performance in real time.

Once familiar with the code:

Experiment with the code to achieve the best results on the task!

All results will be logged to a shared MLflow instance displaying participants' performance in real time.

To improve performance:

- Change the hyperparameters
- Change the training parameters
- Try other optimizers
- For the more ambitious: propose new model architectures

Thank you

References I

- Akiba, Takuya et al. (2019). “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Aravantinos, Vincent and Frederik Diehl (May 2019). *Traceability of Deep Neural Networks*. en. arXiv:1812.06744 [cs.LG]. DOI: 10.48550/arXiv.1812.06744. URL: http://arxiv.org/abs/1812.06744 (visited on 05/26/2026).
- Baker, Monya (May 2016). “1,500 scientists lift the lid on reproducibility”. In: *Nature* 533.7604, pp. 452–454. ISSN: 1476-4687. DOI: 10.1038/533452a. URL: https://doi.org/10.1038/533452a.
- Collaboration, Open Science (2015). “Estimating the reproducibility of psychological science”. In: *Science* 349.6251, aac4716. DOI: 10.1126/science.aac4716. URL: https://www.science.org/doi/abs/10.1126/science.aac4716.
- Sohl-Dickstein, Jascha (2024). “The boundary of neural network trainability is fractal”. In: *arXiv preprint arXiv:2402.06184*.

References II

- Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747*.
- Yadan, Omry (2019). *Hydra - A framework for elegantly configuring complex applications*. Github. URL: https://github.com/facebookresearch/hydra.
- Zaharia, Matei A. et al. (2018). “Accelerating the Machine Learning Lifecycle with MLflow”. In: *IEEE Data Eng. Bull.* 41, pp. 39–45. URL: https://api.semanticscholar.org/CorpusID:83459546.

Optuna

Hyperparameter Optimization

- Hyperparameters govern the neural network itself
- Combinations of hyperparameters form a complex search space

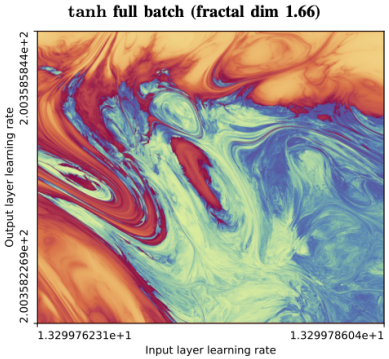


Figure: From Sohl-Dickstein 2024: Two-dimensional grid search over hyperparameters

Optuna

- Bayesian hyperparameter search
- Iterative search with pruning and early stopping
- TPE and CMA-ES

Link to Optuna documentation: https://optuna.org/

Akiba et al. 2019