MLOps for Scientific Applications: Building Reliable, Reproducible, and Transparent AI Workflows

Luca Clissa, Leonardo Plini, Daniele Bonacorsi
luca.clissa2@unibo.it

Outline

- Motivating examples
- MLOps definition
- MLOps pipeline
- Best practice examples
- Conclusions

Disclaimer: just sharing my experience!
Curious to know what other people are doing

The reproducibility crisis

Most ML projects:

- Fail before deployment
- Deteriorate silently in production

Lack of robust engineering pipelines to track, compare, replicate experiments

Mounting technical debt, no reproducibility, lose trust

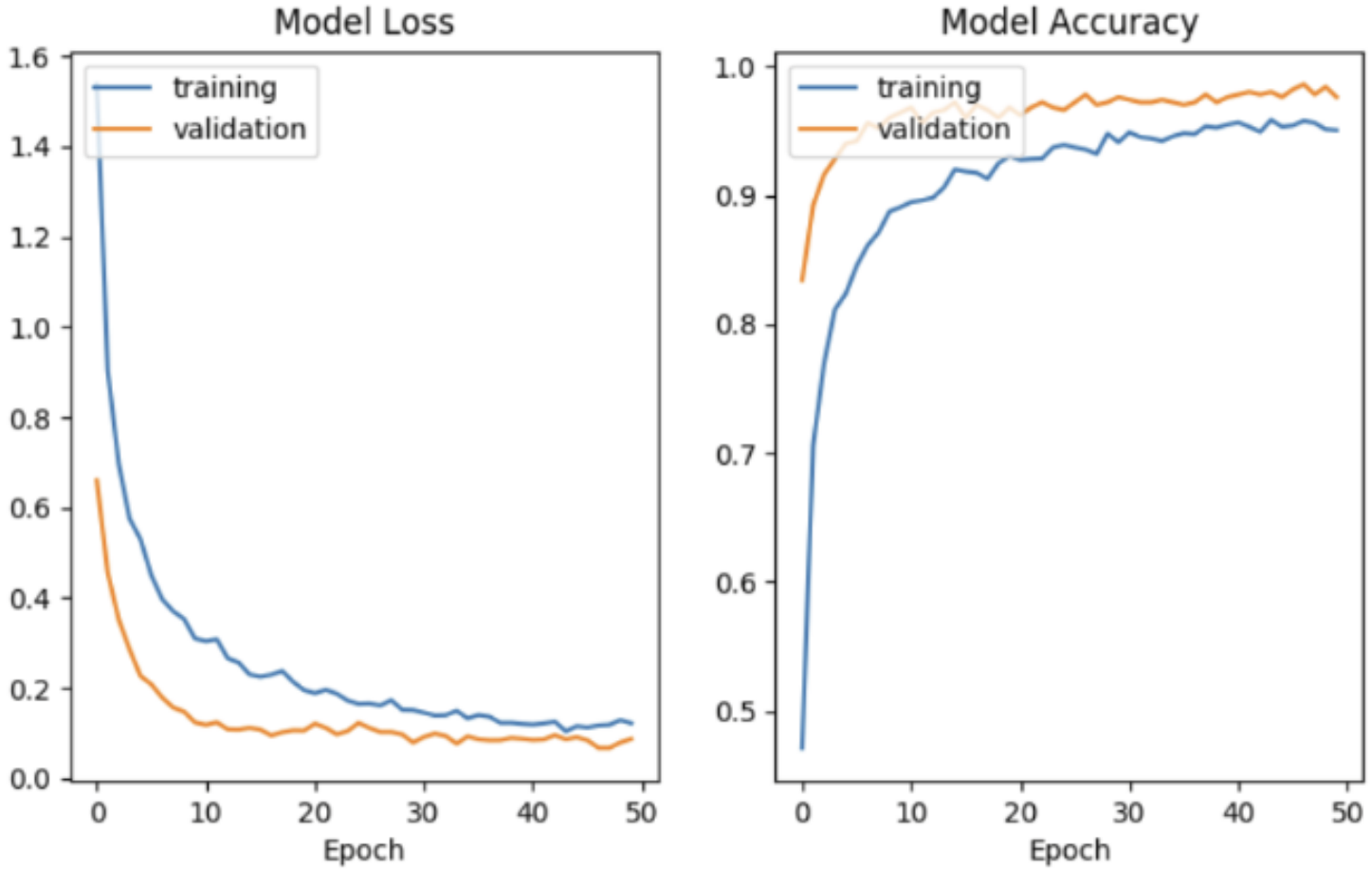
Unstructured development

- We've all been there...
  at least once!
  - Poorly named files
  - Hard-coded parameters
  - Messy code
  - Non-linear execution

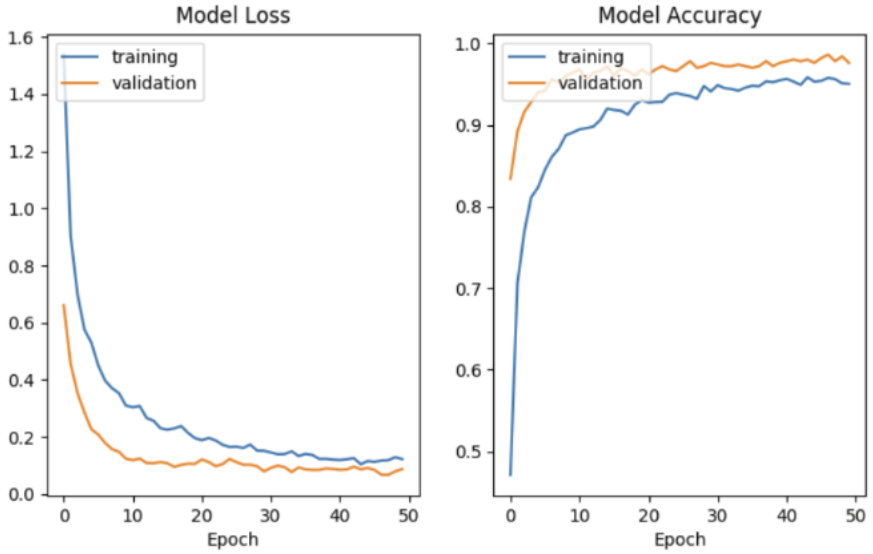
Untitled 7.ipynb

```
[4] df_data['age'].fillna(50, inplace=True)
    df_data['Income] = df_data'['Income] / 1000
    learning_rate = 0.001
    epochs = 20

[2] df_data = pd.read_csv("./data.csv")

[7] model.fit(X_train, y_train, lr=0.001, epochs=15)
    training_accuray = 0.970
    validation_accuracy = 0.965

[5] data_df.head()

ValueError. Cell rexeH lask call lio :>)
> 1_data_df.head()
ValueError: data_df not found
```

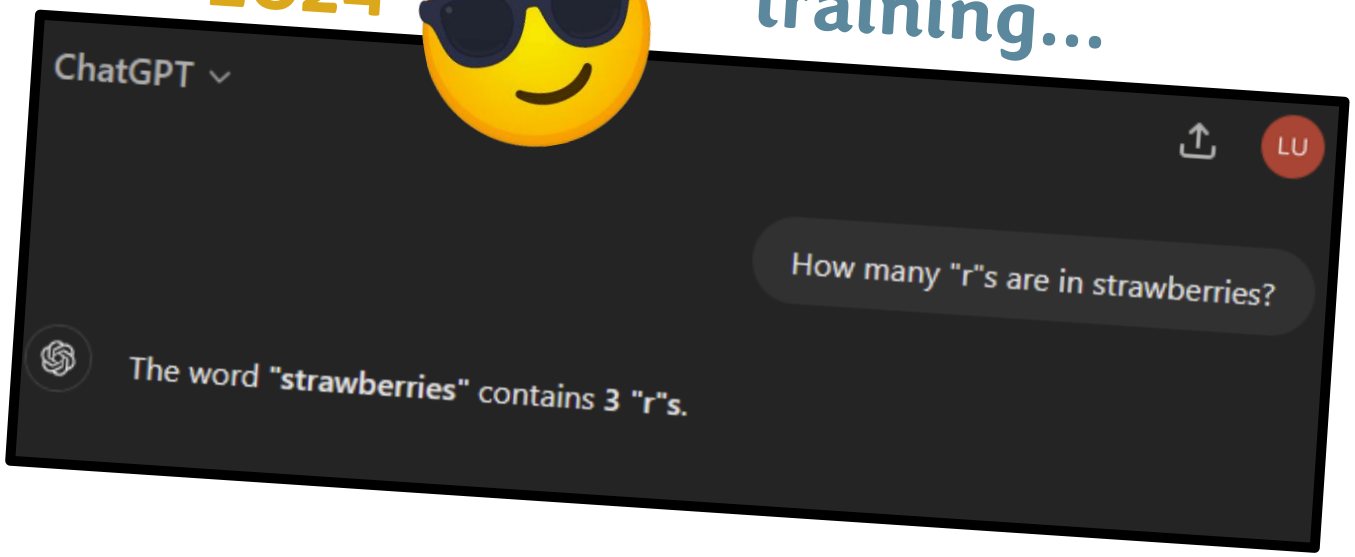
The problem of generalization



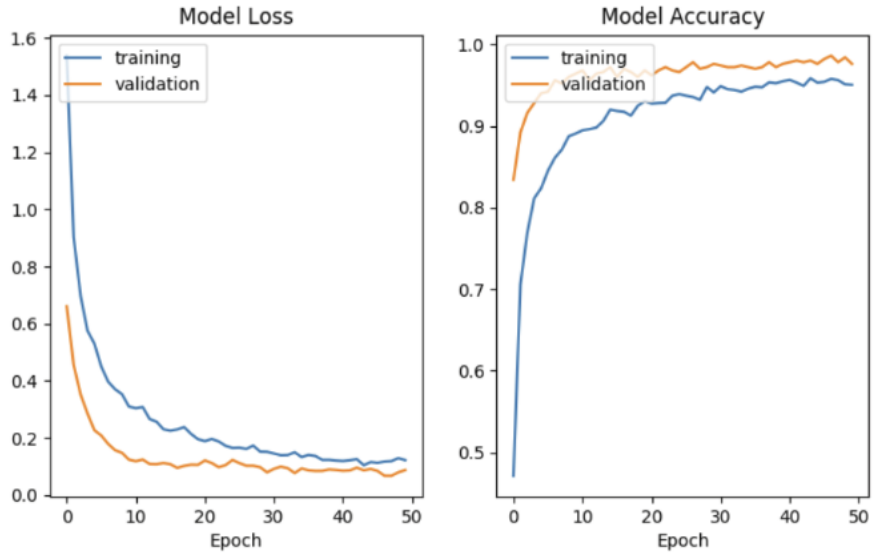
The problem of generalization



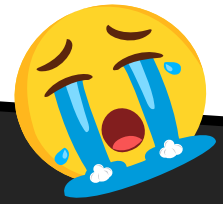
2024 😎 training...



The problem of generalization



2024



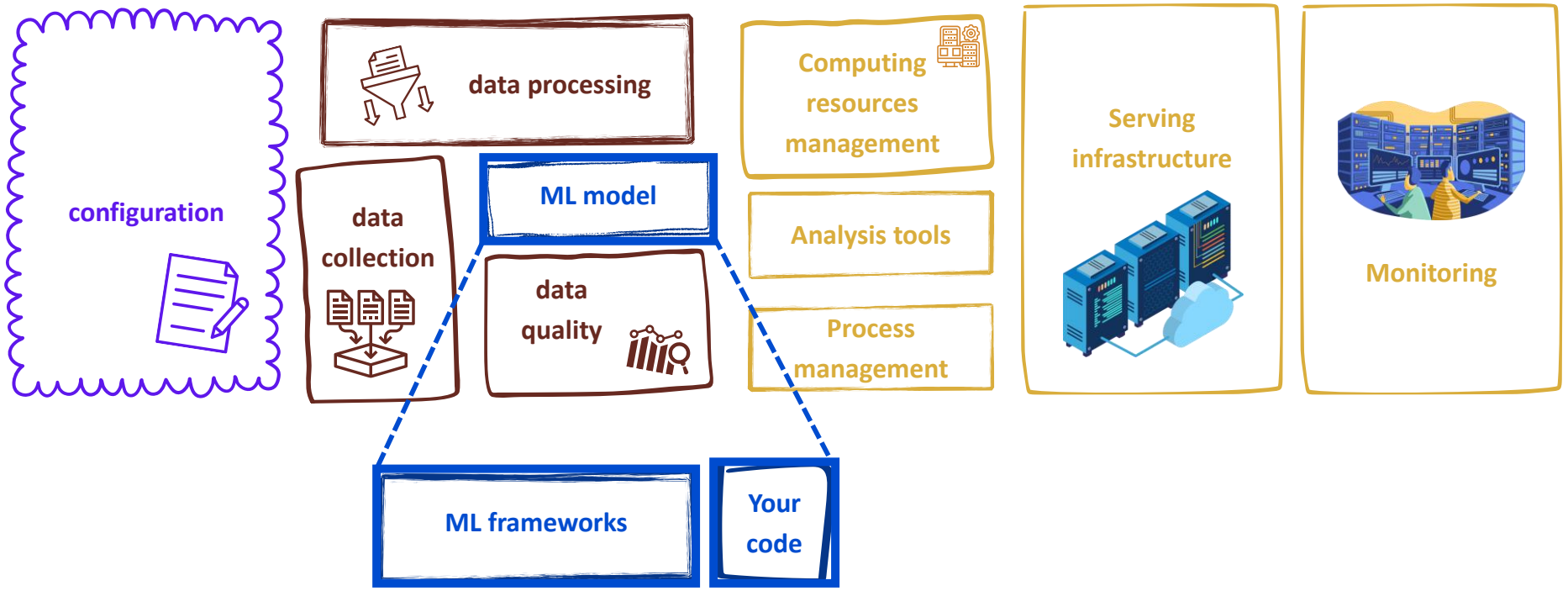
production...

What can we do about it?

Machine Learning Operations (MLOps)

MLOps: one definition

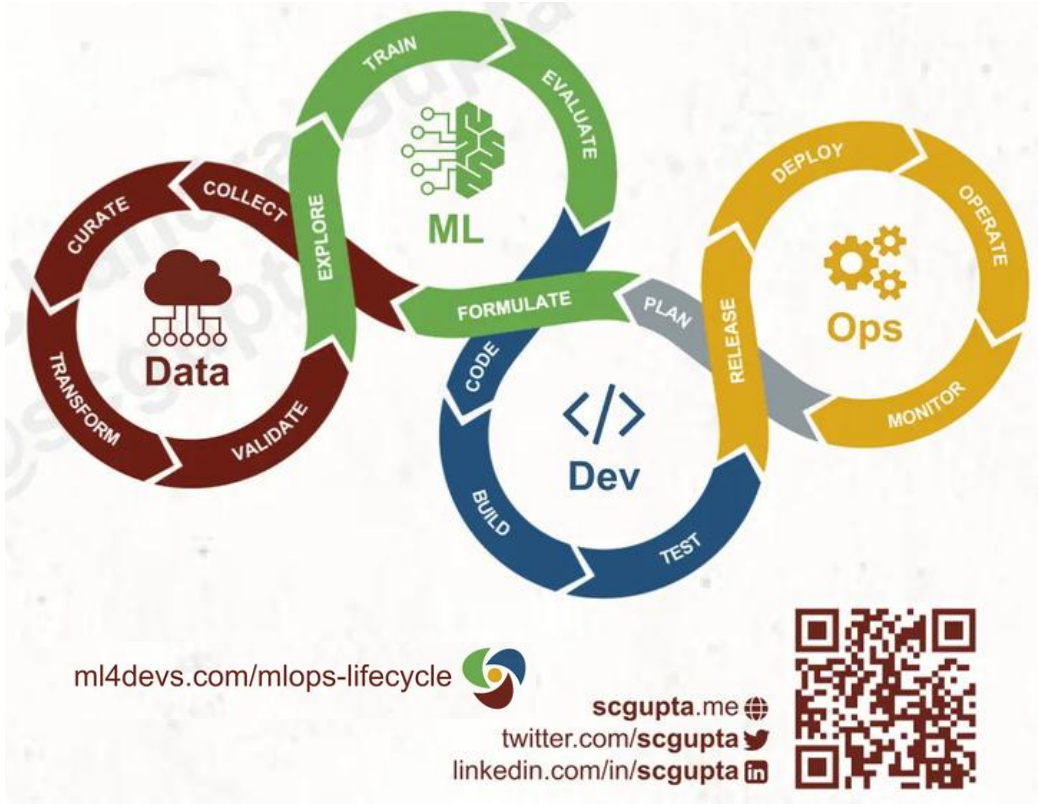
MLOps = **ML lifecycle management** + **Software Engineering best practice (DevOps)**



MLOps pipeline

MLOps is a multi-stage, iterative process:

- Data processing
- Modelling
- Deployment
- Monitoring



Data processing

Data processing is key to ML success

- Quality control: garbage in, garbage out
- Exploratory Data Analysis (EDA)
  - Understanding data content and challenges
- Preprocessing
  - Validation, cleaning, feature engineering
- Versioning
  - **Lineage**: **where** data come from? (relationship)
  - **Provenance**: **how** were data created? (history)

Paradigm shift: model-centric to data-centric AI [2]

	Model-driven ML	Data-driven ML
Fixed component	Dataset	Model Architecture
Variable component	Model Architecture	Dataset
Objective	High accuracy	Fairness, low bias
Explainability	Limited	Possible

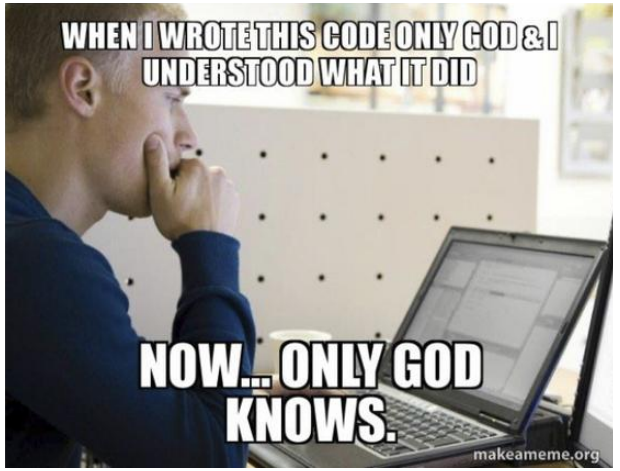
Modelling

- Data splitting:
  - make sure no leakage
  - Representative of target data
  - Balanced sampling can help!

Consider a binary classification problem with a dataset composed of 200 entries.
There are 160 negative examples (no failure) and 40 positive ones (failure).

	Training	Validation	Test
Expected:	75% (120 + 30)	15% (24+6)	10% (16+4)
Random:	75% (131 + 19)	15% (19+11)	10% (10+10)

- EDA is key to understand training requirements & challenges
  - E.g. class imbalance, rare events, metrics
- Experiment tracking
  - Track and document all design choices for reproducibility!
  - ...and of course the outputs!



Modelling: error analysis

- Error analysis is key for model development
  - E.g. looking at wrong predictions help debugging where training fails
- Performance:
  - Carefully consider which metric to optimize
  - Looking at more metrics can give more thorough assessment

Deployment & monitoring

- When model is mature, we can finally deploy it!
  - Very specific to application and infrastructure
  - Typically relies on model file + inference pipeline

- However, not the final step → MLOps is a cycle!

CI/CD

- production is only a checkpoint between development stages
- may want to keep improving the model
- new configs/architectures
- re-train as new data comes in

Monitoring

- always keep an eye on performance, as data shift may impact your application
- model metrics
- infrastructure metrics
- errors, resource utilization, ...

How does it look in practice?

Use case: Point Cloud Deep Learning for particle flow in ATLAS

Tools: several frameworks exist

How does it look in practice?

Use case: Point Cloud Deep Learning for particle flow in ATLAS

Tools: several frameworks exist

Demo use **Weights & Biases**

clissa/mlops-handson

wandb sessions

Every ML experiment should be **traceable, reproducible, and comparable**

- `wandb.init()` creates structured experiment runs

- Use `job_type` to group runs
  - data_prep — dataset creation, EDA, preproc
  - training — model training
  - eval — validation/metrics
  - debug: — diagnostics

  → **choose your tags, just be meaningful!**

```
#!pip install wandb -y
# wandb.login()
wandb.init(
    project="mlops-ISGC2026", # set your project name here
    entity="<user/team>", # set under
    name="basic logging", # automatically set if not specified
    job_type="data_prep", # optional: useful to filter different kind of runs in the UI
    # (e.g. data logging, data splitting, EDA, training, tests...)
    config={
        'DATA_PATH': DATA_PATH
    },
    notes="My first run with wandb" # optional: we can add descriptions as metadata
)
```

- Automatically tracks experiment code and config
  - You can also add additional `notes`

Experiment tracking

Logging metrics, parameters and outputs

- `wandb.log` tracks configs and results
- You can log:
  - Metrics
  - Hyperparams/configs
  - Plots/images/tables

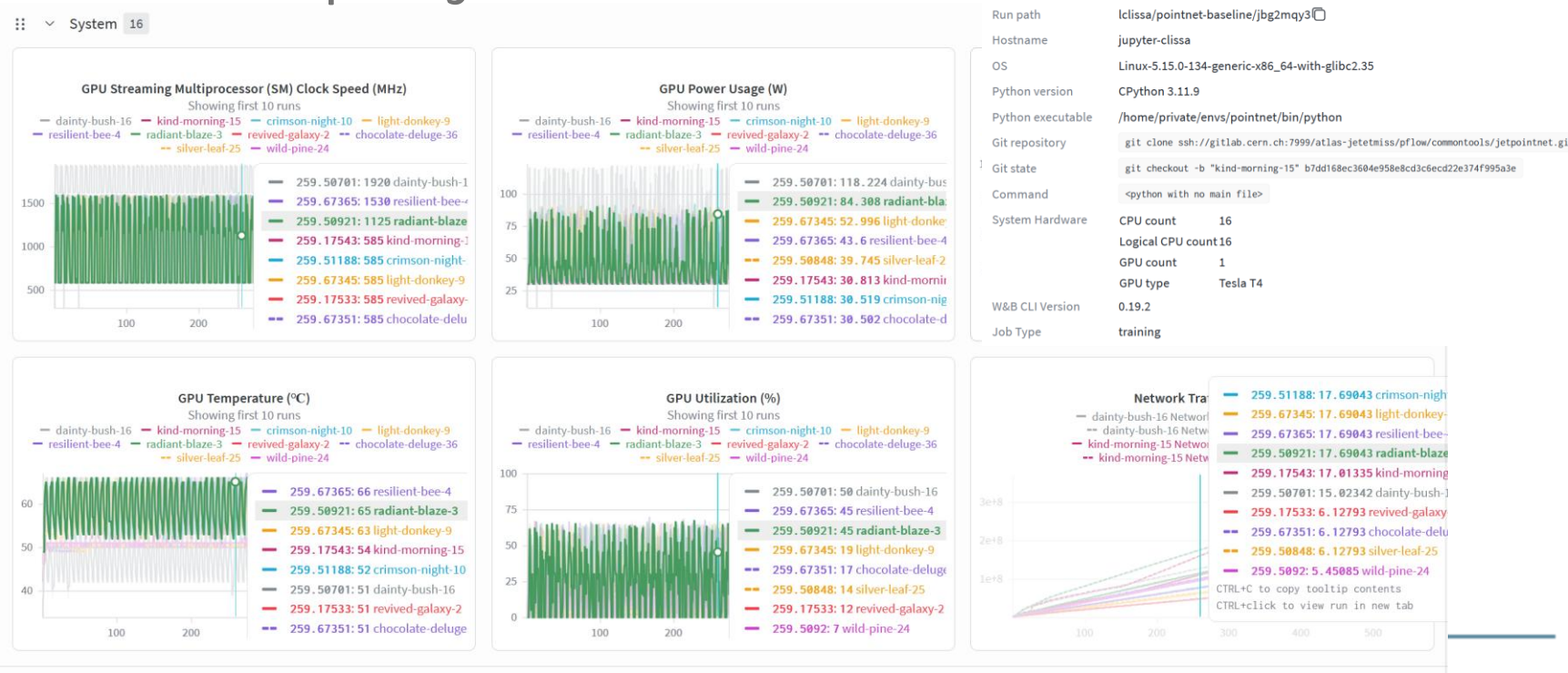
```python
# manually logging history
metrics = history.history
ep=1
for loss, val_loss, acc, val_acc in zip(
    metrics['loss'], metrics['val_loss'],
    metrics['accuracy'], metrics['val_accuracy']):
    run.log({
        'epoch': ep,
        'loss': loss, 'val_loss': val_loss,
        'accuracy': acc, 'val_accuracy': val_acc
    })
    ep +=1
```

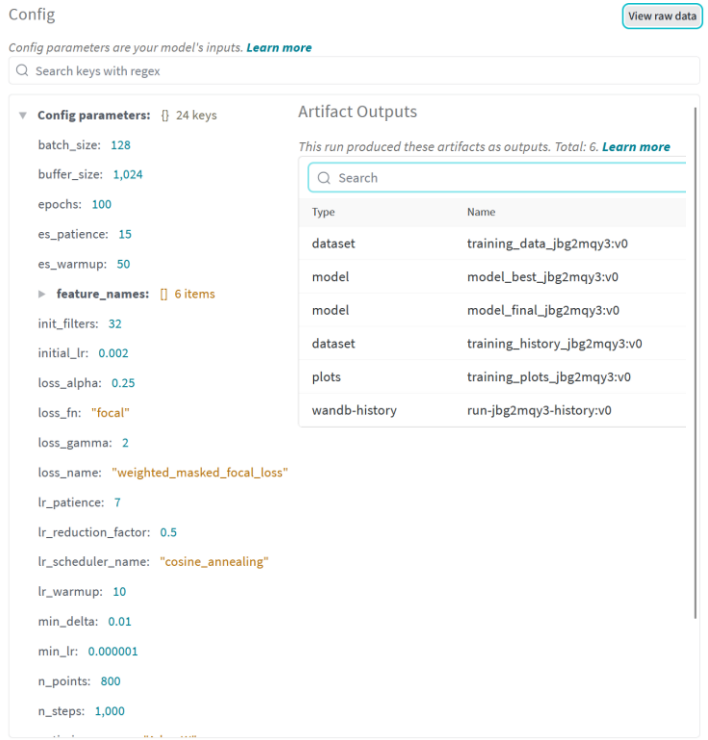
```python
wandb.log({
    "roc_curve": wandb.plot.roc_curve(y_true, y_pred)
})
```

```python
# log top 5 samples with highest loss in a table
columns = [
    "top rank", "sample id", "loss", "visualization"]
validation_table = wandb.Table(columns=columns)
```

Experiment tracking (bonus)

- By default, wandb also tracks:
  - Configs
  - Computing resources





Artifacts: data and model versioning

Artifacts enable **versioned datasets and models (not only!)**

- `wandb.Artifact` native object

- Automatically manage data/model objects
  - Customizable metadata tracking
  - Versioning
    - v0, v1, ..., `latest`
    - use custom aliases for readable stages

- Artifact types:
  - dataset v0, v1, ..., latest
  - model baseline, candidate, staging, production

```python
with wandb.init(project="mlops-ISGC2026", name="data-creation",
                job_type="data_prep", config={'DATA_PATH': DATA_PATH,'seed': SEED},
                notes="Playing with Artifacts ...") as run:

    # create artifact for raw data
    raw_data_artifact = wandb.Artifact(name="raw_data", type="dataset",
                                       description="MC simulation of rho -> pions decays (full data)"
    )
    raw_data_artifact.add_file(local_path = str(DATA_PATH), name="rho_small.npz")
    wandb.log_artifact(raw_data_artifact)
```

```python
artifact = wandb.Artifact(
    "pointnet-v0.1",
    type="model",
    aliases=["baseline", "v0.1"]
)

artifact.add_file("model.pkl")
wandb.log_artifact(artifact)
```

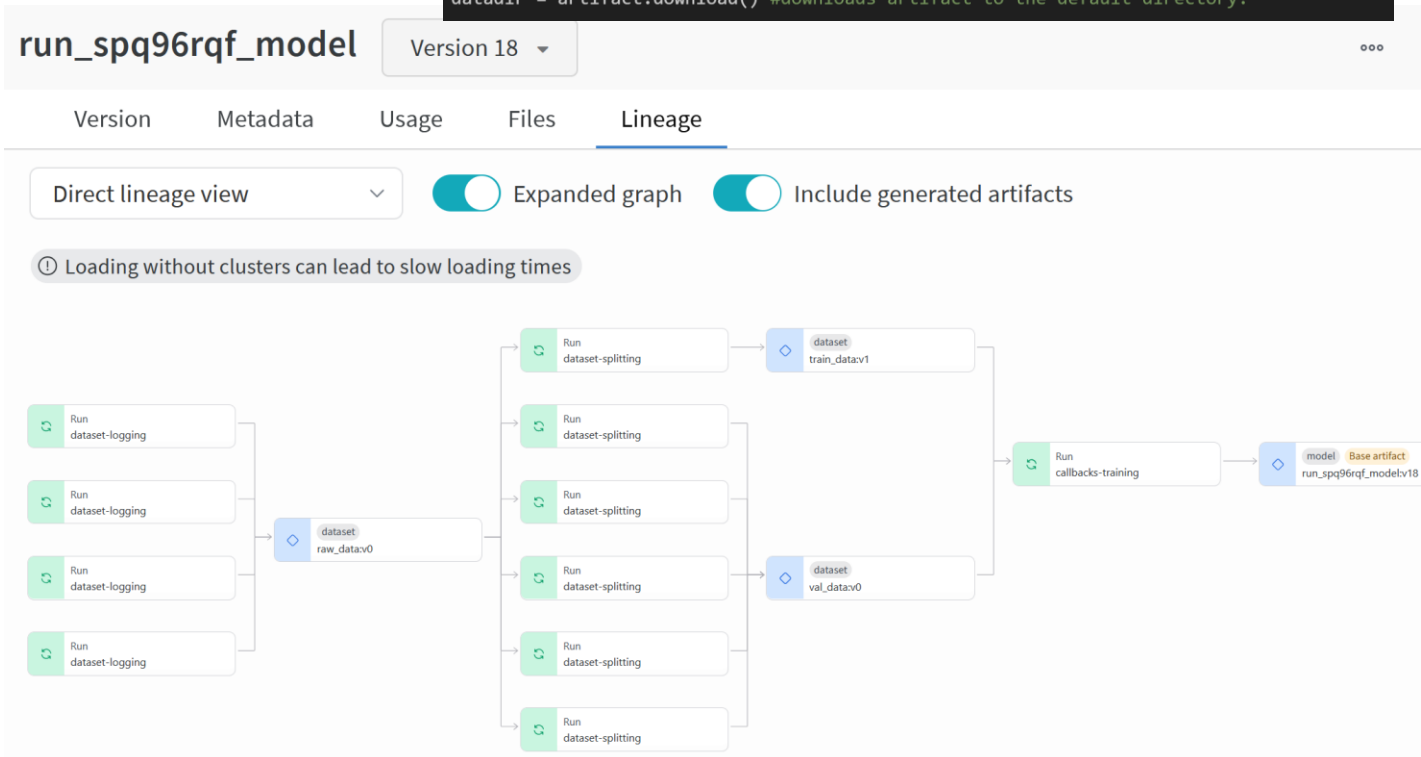
Lineage and provenance

Lineage: where data come from? (relationship)
Provenance: how were data created? (history)

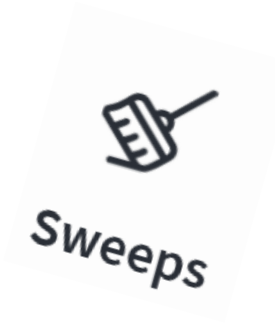
- We can declare what Artifacts are used: `run.use_artifact("my-dataset:latest")`
- This enables wandb to automatically build dependency graphs

→ **Lineage and provenance tracked automatically!**

```
# reference dataset version used for this experiment
artifact = run.use_artifact("raw_data:latest") #returns a run object using the "my_data"
artifact

# actually download the data
datadir = artifact.download() #downloads artifact to the default directory.
```



Hyperparameter sweeps



W&B supports automated hyperparameter search

- configurable as dict / YAML
- `method` enables 3 search strategies
  - grid, random, bayes
- `metric` defines target metric
- `parameters` defines search space
  - Specify values or distributions
- Run from script or terminal

```python
# 2: Define the search space
sweep_configuration = {
    "method": "bayes",
    "metric": {"goal": "minimize", "name": "epoch/val_loss"},
    "parameters": {
        "init_lr": {'min': 1e-4, 'max': 1e-2,
                    'distribution': 'log_uniform' },
        "batch_size": {"values": [16, 32]},
        "init_size": {"values": [8, 16]},
    },
}
```

PY

YAML

```yaml
method: bayes

metric:
  goal: minimize
  name: epoch/val_loss

parameters:
  init_lr:
    min: 1e-4
    max: 1e-2
    distribution: log_uniform

  batch_size:
    values: [16, 32]

  init_size:
    values: [8, 16]
```

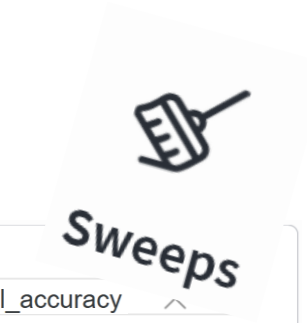
YAML

```python
# 3: Start the sweep
sweep_id = wandb.sweep(sweep=sweep_configuration, project="mlops-ISGC2026", entity="<user/team>")

wandb.agent(sweep_id, function=train_wrapper, count=60)
```

Bash

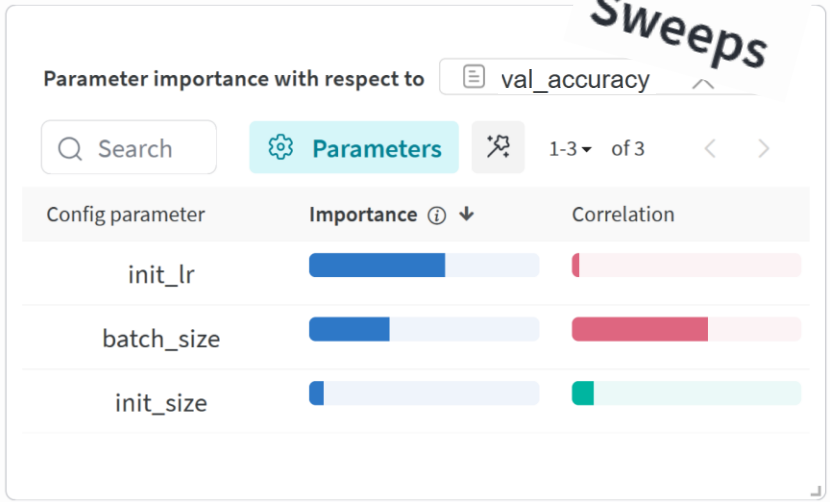
```bash
SWEEP_ID=$(wandb sweep sweep.yaml | tee /dev/stderr | grep "Create sweep with ID
wandb agent atlas-ml/mlops-ISGC2026/$SWEEP_ID
```

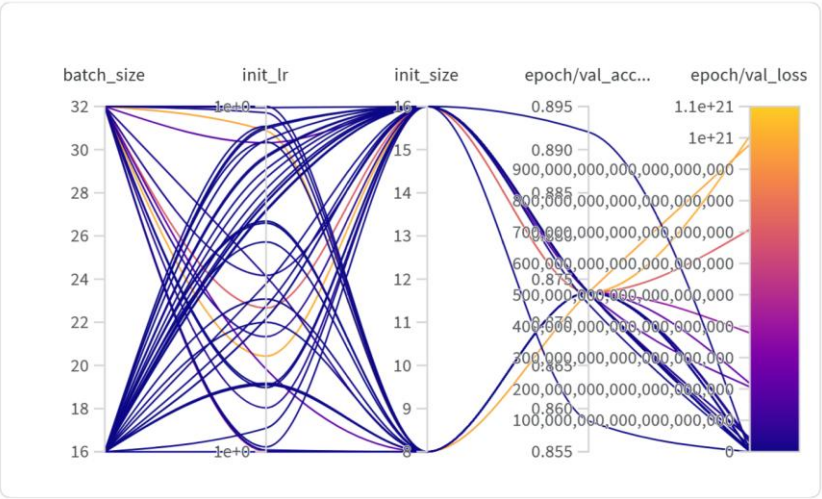
Sweeps visualizations



W&B sweeps provide great default vizs to compare configs

- Parameter importance
  - Evaluate impact of params on target metric
- **Parallel coordinates** plot
  - Great to compare configs based on target metric





Reports

Wandb also supports reporting

- Easily integrate run results with formatted comments
- Interactive
- Useful to:
  - Document
  - Debug
  - Share

[demo report]

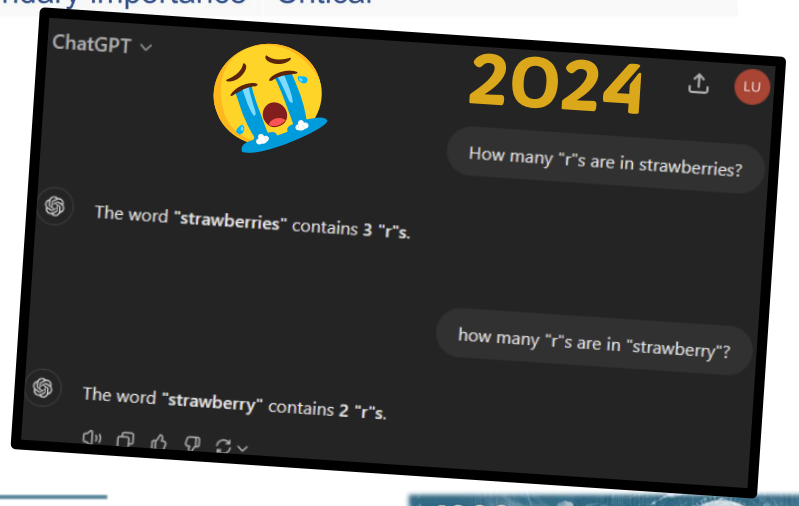
Conclusions

- Reproducibility and stability are key for trustworthy scientific research
- Development and production have different needs

	Development ML	Production ML
Objective	High-accuracy model	Efficiency of the overall system
Dataset	Fixed	Evolving
Code quality	Secondary importance	Critical
Model training	Optimal tuning	Fast turn-arounds
Reproducibility	Secondary importance	Critical
Traceability	Secondary importance	Critical

- MLOps can help:
  - mitigate development challenges
  - speed up experimentation

→**it seems overdoing at first**



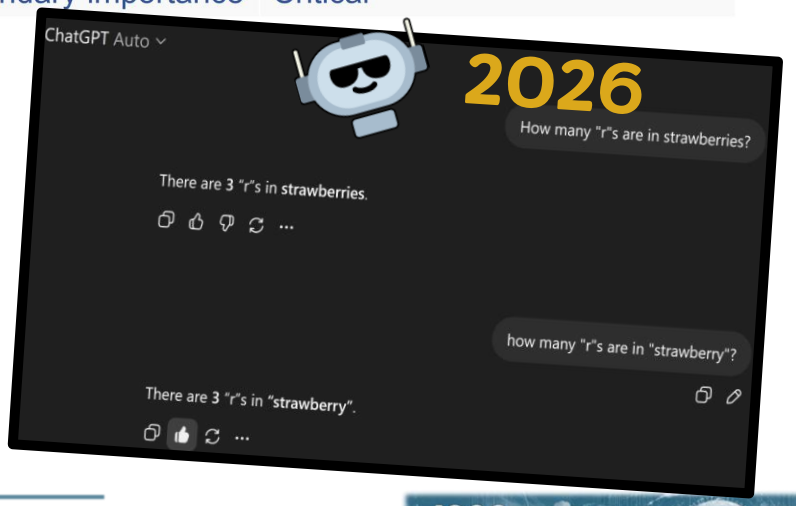
Conclusions

- Reproducibility and stability are key for trustworthy scientific research
- Development and production have different needs

	Development ML	Production ML
Objective	High-accuracy model	Efficiency of the overall system
Dataset	Fixed	Evolving
Code quality	Secondary importance	Critical
Model training	Optimal tuning	Fast turn-arounds
Reproducibility	Secondary importance	Critical
Traceability	Secondary importance	Critical

- MLOps can help:
  - mitigate development challenges
  - speed up experimentation

→**it seems overdoing at first**
...but it pays in the long term

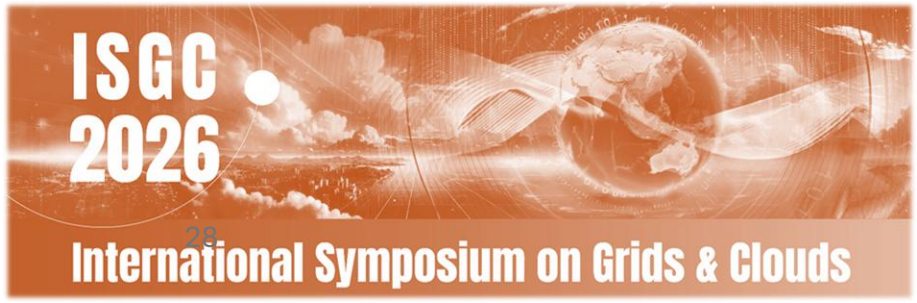


- Many tools, no universal best...

→ **Pick one and get proficient**
...but leave a door open to exploration when you get stuck

Questions?

Contacts: luca.clissa2@unibo.it



References

[1] Sculley, David, et al. "Hidden technical debt in machine learning systems." *Advances in neural information processing systems* 28 (2015).

[2] Testi, M., Clissa, L., Ballabio, M., Ricciardi, S., Baldo, F., Frontoni, E., ... & Vessio, G. (2025). Enhancing Cell Counting through MLOps: A Structured Approach for Automated Cell Analysis. *arXiv preprint arXiv:2504.20126*.

[3] Wazir, S., Kashyap, G. S., & Saxena, P. (2023). MLOps: a review. *arXiv preprint arXiv:2308.10908*.

[4] Amrit, C., & Narayanappa, A. K. (2025). An analysis of the challenges in the adoption of MLOps. *Journal of Innovation & Knowledge*, *10*(1), 100637.

[5] Clissa, L. (2024). An introduction to MLOps. 1st AI_INFN Advanced Hackathon. https://agenda.infn.it/event/43129/contributions/247489/

Backup