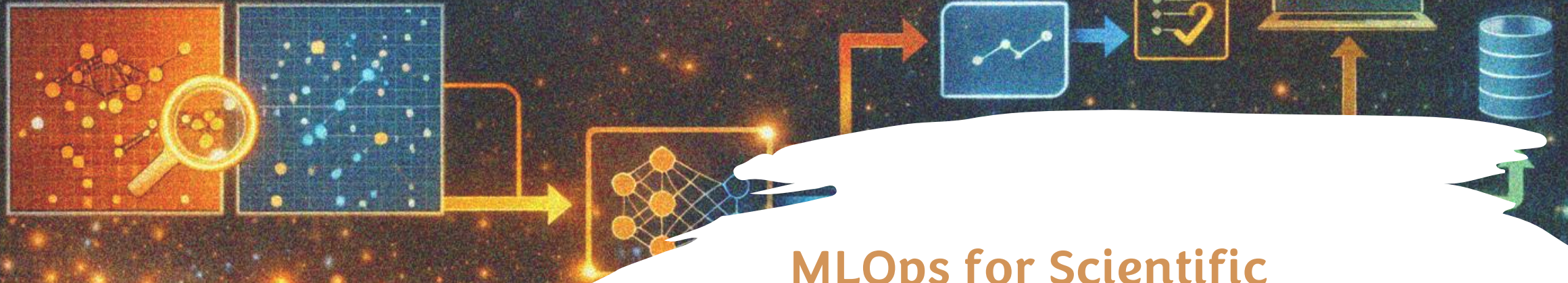


ISGC
2026

International Symposium on Grids & Clouds



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 Python 3.9.91
[4] df_data['age'].fillna(50, inplace=True)
    df_data['Income'] = df_data['Income'] / 1000
    learning_rate = 0.001
    epochs = 20
Python

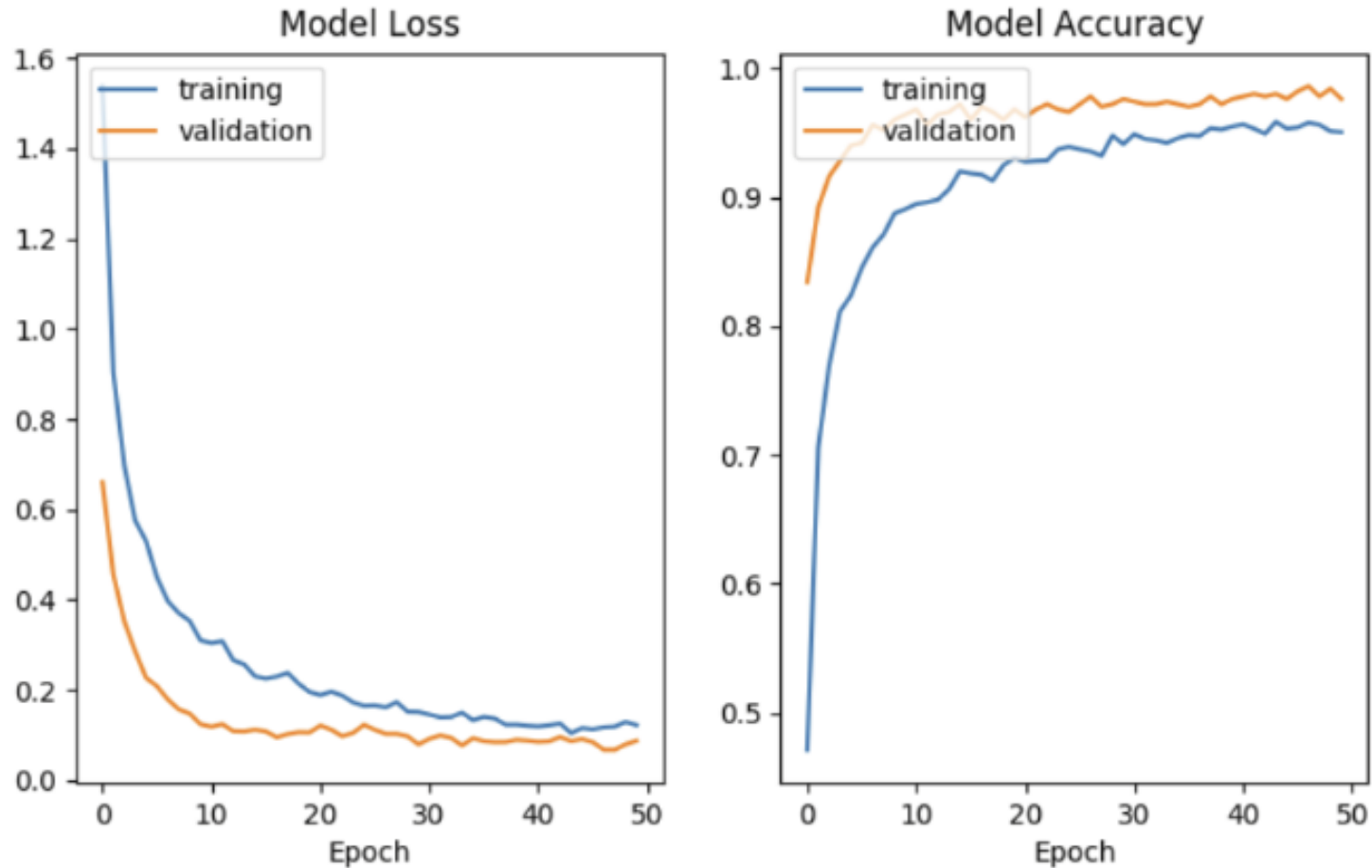
[2] df_data = pd.read_csv("./data.csv")
Python

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

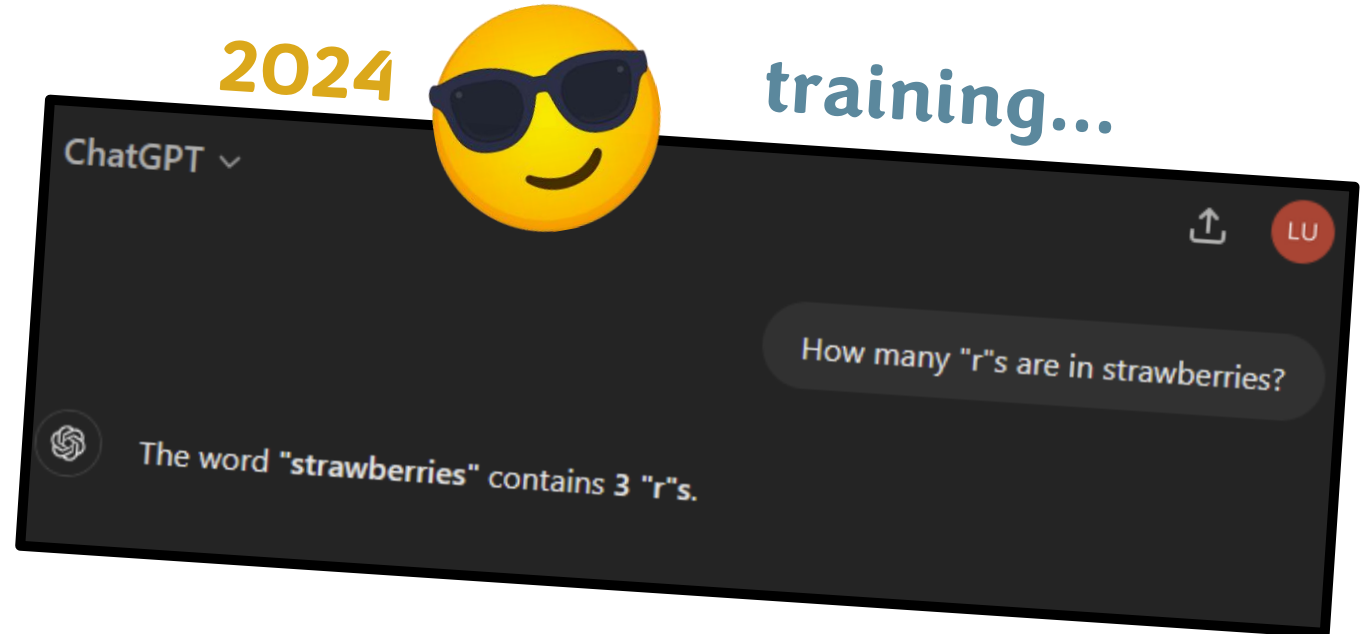
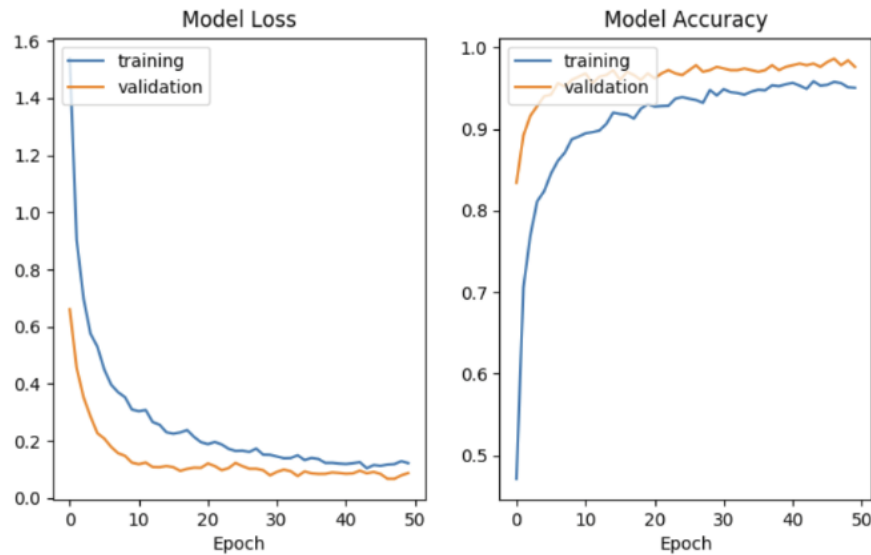
[5] data_df.head()

-----
ValueError: col1 rexerl lask call lioa.:>
2
n > 1_data_df.head()
ValueError: data_df not found
```

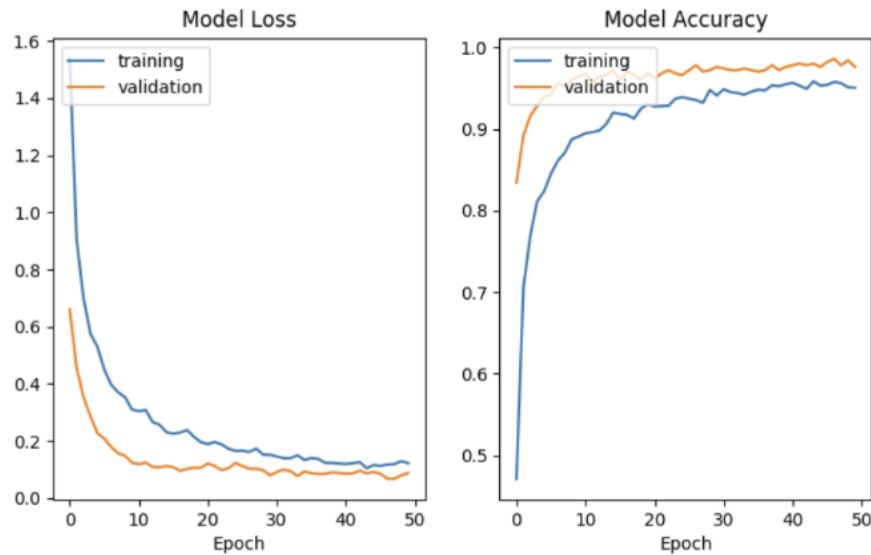
The problem of generalization



The problem of generalization



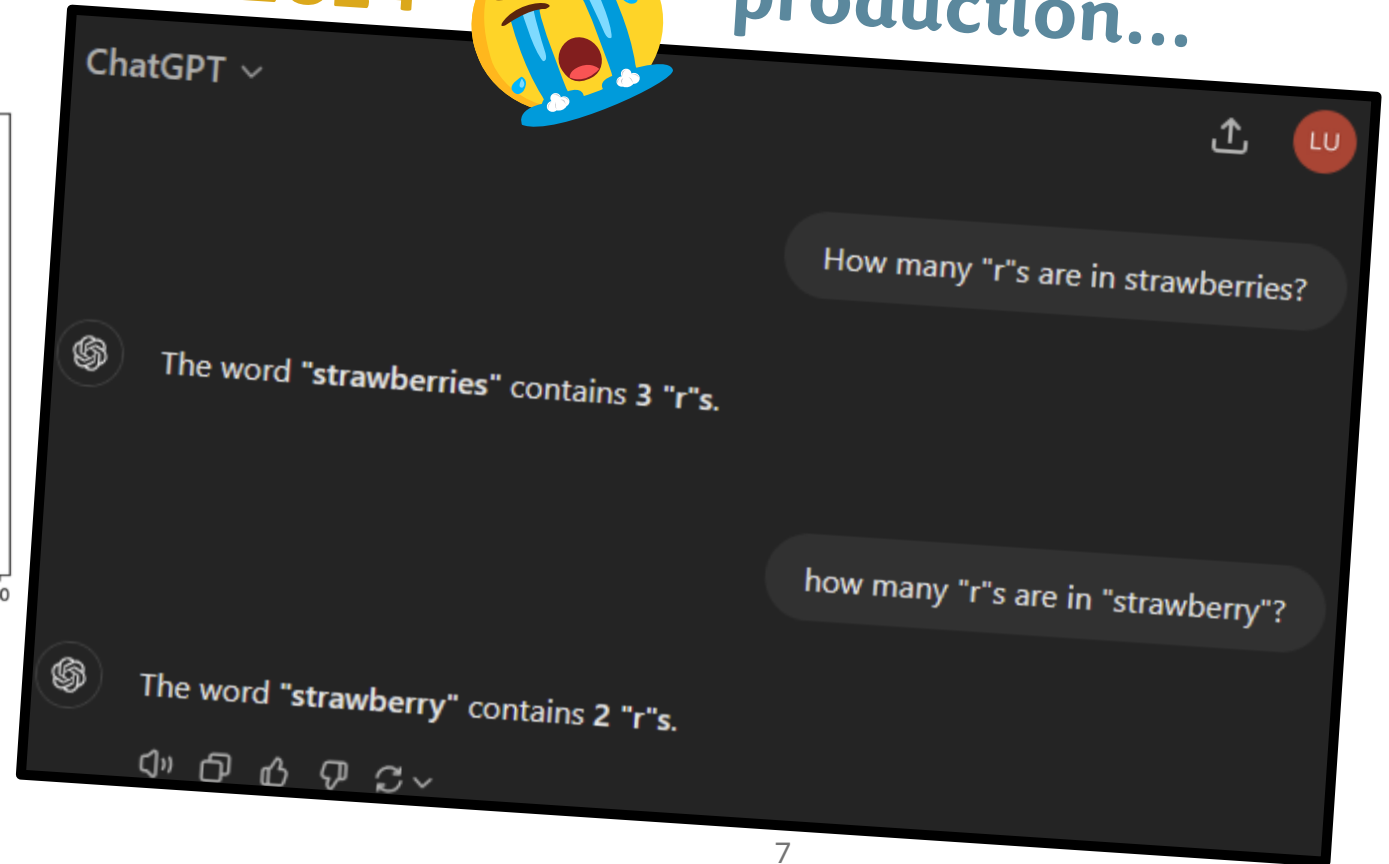
The problem of generalization



2024



production...

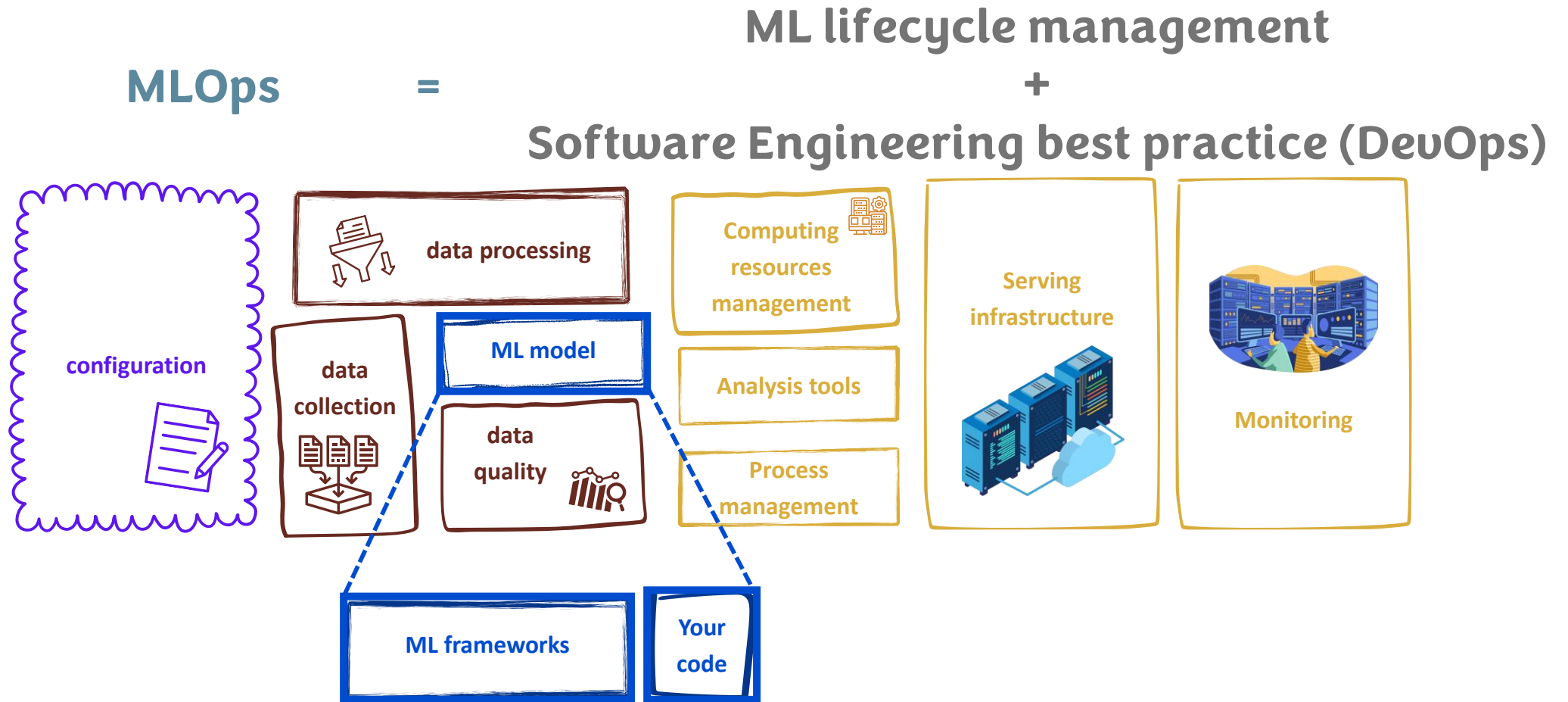


What can we do about it?

Machine Learning Operations (MLOps)



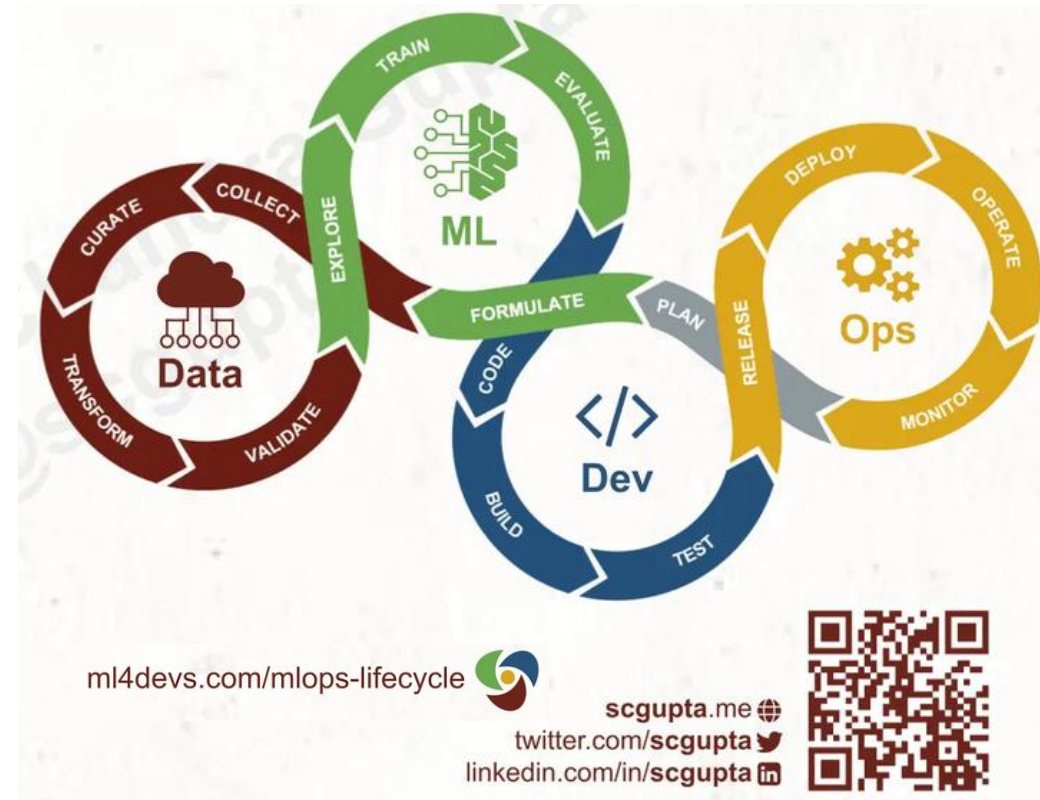
MLOps: one definition



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!

- 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!

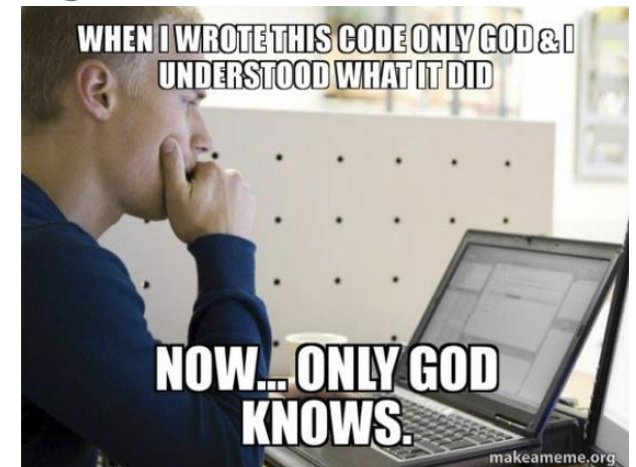
Consider a binary classification problem with a dataset composed of 200 entries. There are 160 negative examples (no failure) and 40 positive ones (failure).

Expected:

Training 75% (120 + 30)	Validation 15% (24+6)	Test 10% (16+4)
-------------------------------	-----------------------------	-----------------------

Random:

Training 75% (131 + 19)	Validation 15% (19+11)	Test 10% (10+10)
-------------------------------	------------------------------	------------------------



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

The logo for mlflow, featuring the text "mlflow" in a blue, lowercase, sans-serif font with a trademark symbol.The logo for W&B, consisting of a black rectangle with a grid of yellow dots on the left and the text "W&B" in white on the right.The logo for TensorBoard, featuring an orange stylized "T" icon above the text "TensorBoard" in a black, sans-serif font.The logo for neptune.ai, featuring a blue circle with a white waveform icon to the left of the text "neptune.ai" in a black, sans-serif font.

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](https://github.com/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!

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

```
#!/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
)
```

Experiment tracking

Logging metrics, parameters and outputs

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

```
# 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
```

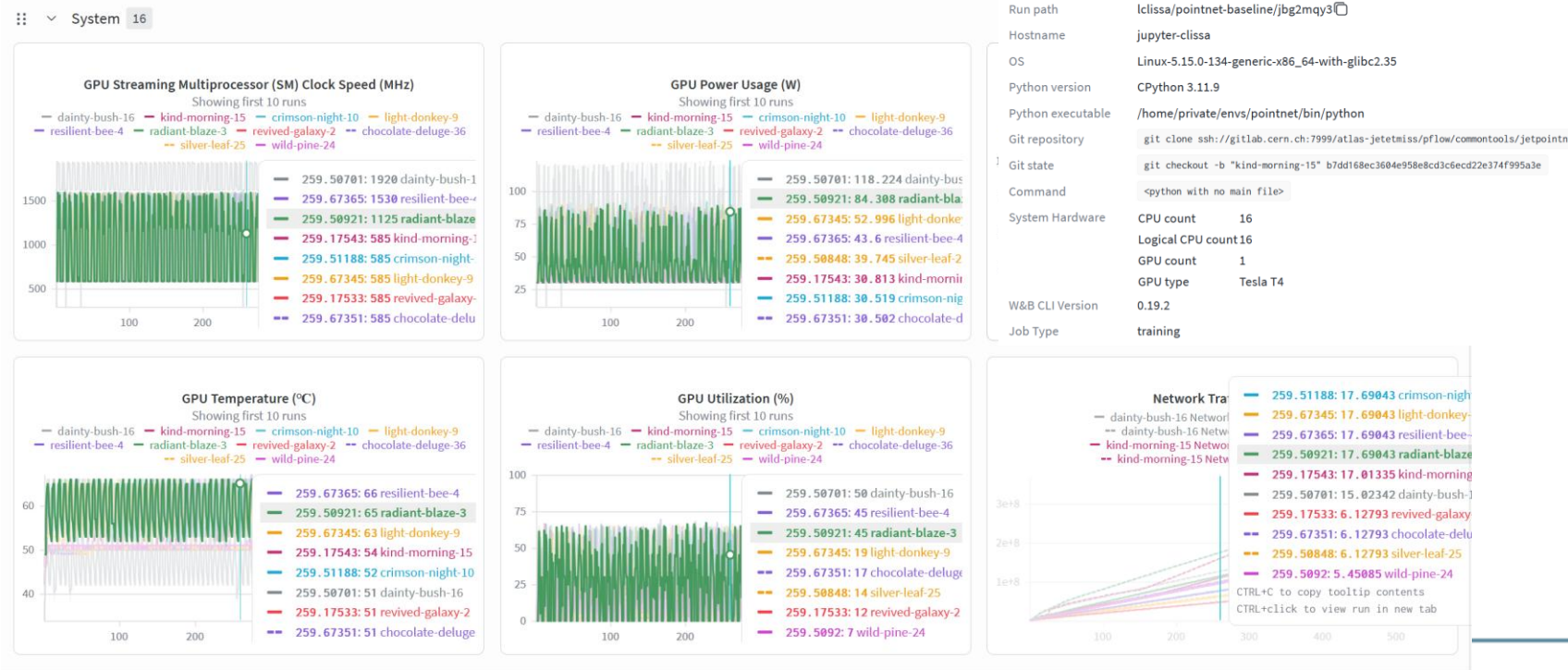
```
wandb.log({
    "roc_curve": wandb.plot.roc_curve(y_true, y_pred)
})
```

```
# 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



kind-morning-15

Charts Overview Logs Files Artifacts

Notes pointnet_segmentation using normalized_x, normalized_y, normalized_z, category, n

Tags +

Author Iclissa

State Finished

Start time June 5th, 2025 9:50:00 PM

Runtime 9h 16m 9s

Tracked hours 9h 16m 7s

Run path Iclissa/pointnet-baseline/jbg2mqy3

Hostname jupyter-clissa

OS Linux-5.15.0-134-generic-x86_64-with-glibc2.35

Python version CPython 3.11.9

Python executable /home/private/envs/pointnet/bin/python

Git repository git clone ssh://gitlab.cern.ch:7999/atlas-jetmet/flow/common/tools/jetpointnet.git

Git state git checkout -b "kind-morning-15" b7dd168ec3604e958e8cd3c6ecd22e374f995a3e

Command <python with no main file>

System Hardware CPU count 16
Logical CPU count 16
GPU count 1
GPU type Tesla T4

W&B CLI Version 0.19.2

Job Type training

Config [View raw data](#)

Config parameters are your model's inputs. [Learn more](#)

Search keys with regex

Config parameters: 24 keys

Artifact Outputs

This run produced these artifacts as outputs. Total: 6. [Learn more](#)

Search

Type	Name
dataset	training_data_jbg2mqy3:v0
model	model_best_jbg2mqy3:v0
model	model_final_jbg2mqy3:v0
dataset	training_history_jbg2mqy3:v0
plots	training_plots_jbg2mqy3:v0
wandb-history	run-jbg2mqy3-history:v0

batch_size: 128
buffer_size: 1,024
epochs: 100
es_patience: 15
es_warmup: 50
feature_names: 6 items
init_filters: 32
initial_lr: 0.002
loss_alpha: 0.25
loss_fn: "focal"
loss_gamma: 2
loss_name: "weighted_masked_focal_loss"
lr_patience: 7
lr_reduction_factor: 0.5
lr_scheduler_name: "cosine_annealing"
lr_warmup: 10
min_delta: 0.01
min_lr: 0.000001
n_points: 800
n_steps: 1,000

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

```
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)
```

```
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:

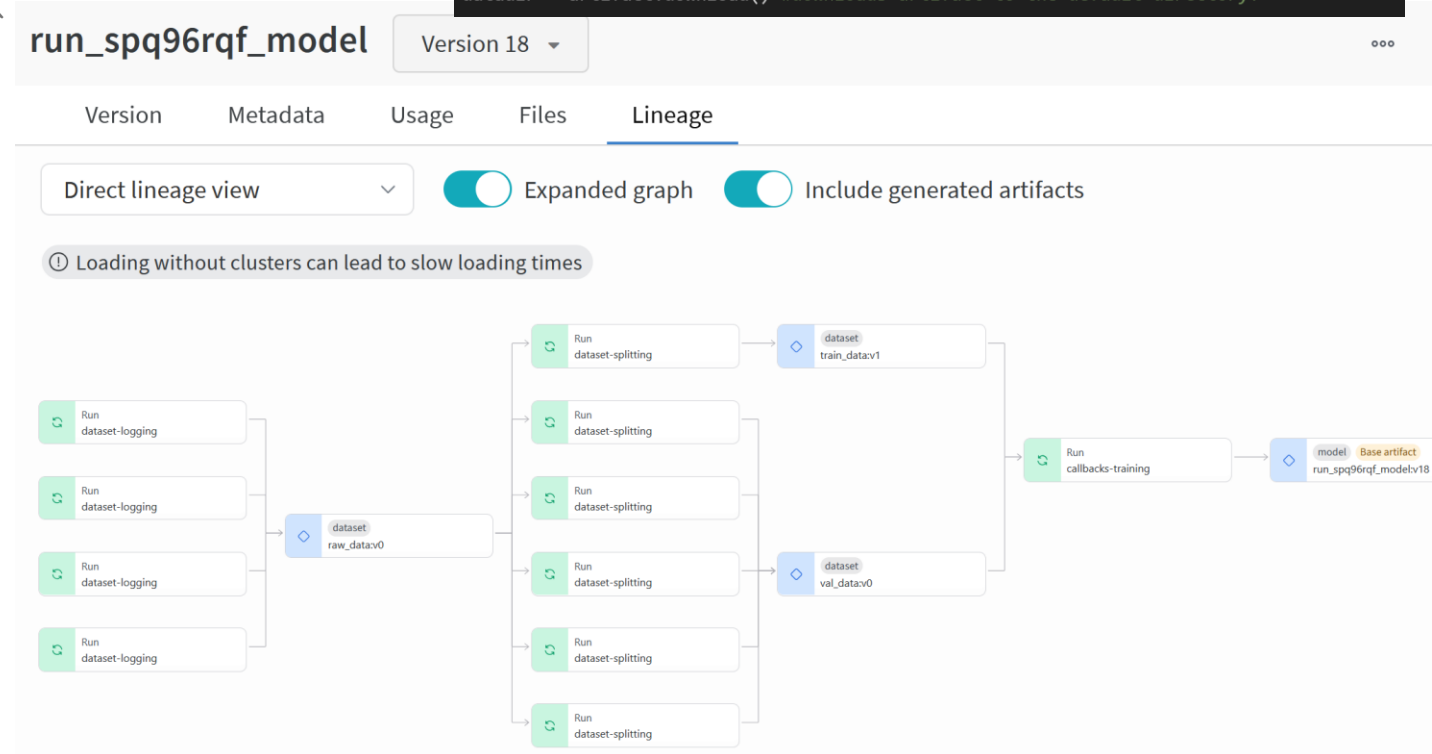
```
`run.use_artifact("my-dataset:latest")`
```

- 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

```
# 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]},
    },
}
```



```
<> 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]
```



```
# 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

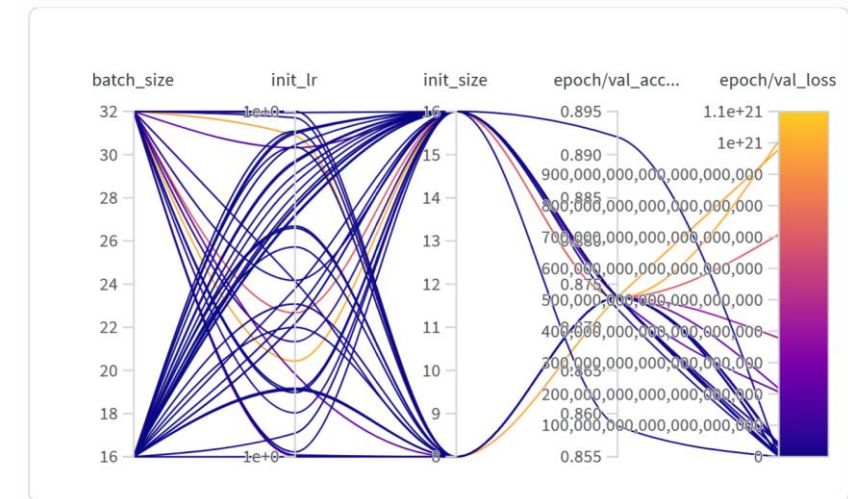
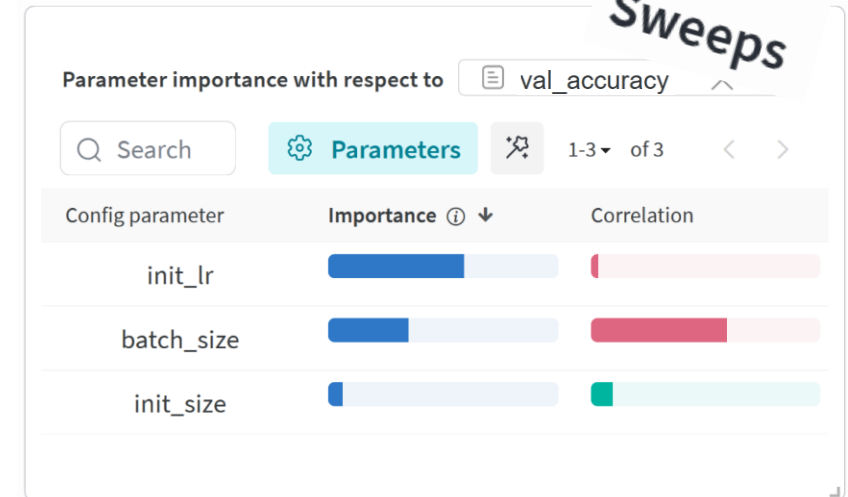


```
SWEEP_ID=$(wandb sweep sweep.yaml | tee /dev/stderr | grep "Create sweep with ID" | cut -d' ' -f2)
wandb agent atlas-ml/mlops-ISGC2026/$SWEEP_ID
```

Sweeps visualizations

W&B sweeps provide great default vizzes 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

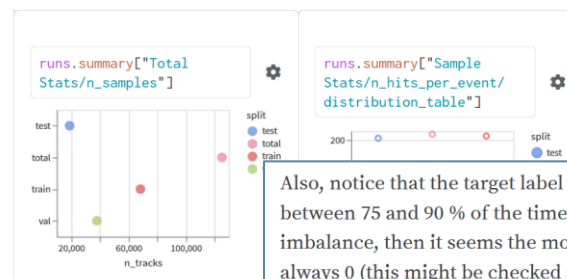
Experimental setting

Data

The analysed dataset is `ttbar (mltree_large.root)`:

- 9 input features: `X`, `Y`, `Z`, `min_distance_to_track`, `Energy`, `point type`
- `X`, `Y`, `Z`, `min_distance_to_track` are chunk normalized
- `Energy` (GeV for hits, MeV for cells)
- `point type` is "1-hot" encoded. Note that associated hits have increasing index, though (instead of 1-hot vector)

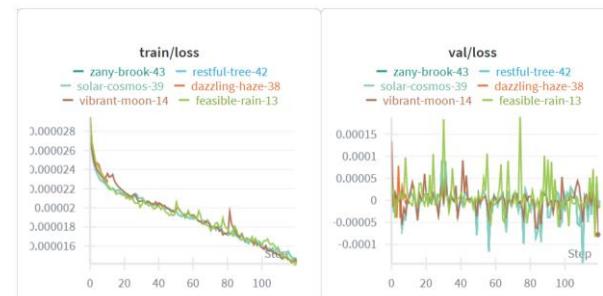
Notice that in total we have **~19M input points**, of which **~2.5M are actual hits** (the remaining are padding points). However, **only ~2M actually contribute to the loss** since we are just masking out non-cell hits.



Also, notice that the target label is obtained as `Fraction_Label > 0.5`, which happens between 75 and 90 % of the times. Hence, if we approximate to roughly 80/20 class imbalance, then it seems the model simply converged to a dummy classifier predicting always 0 (this might be checked actually!!)

Results

The training does not seem to go well. One indication of suspect behavior come when comparing training and validation losses. In fact, validation loss simply juggles randomly around 0, while training loss keeps decreasing.



[demo report](#)

Conclusions



- Reproducibility and stability are key for trustworthy scientific research
- Development and production have different needs
- MLOps can help:
 - mitigate development challenges
 - speed up experimentation

→ it seems overdoing at first

	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



Conclusions



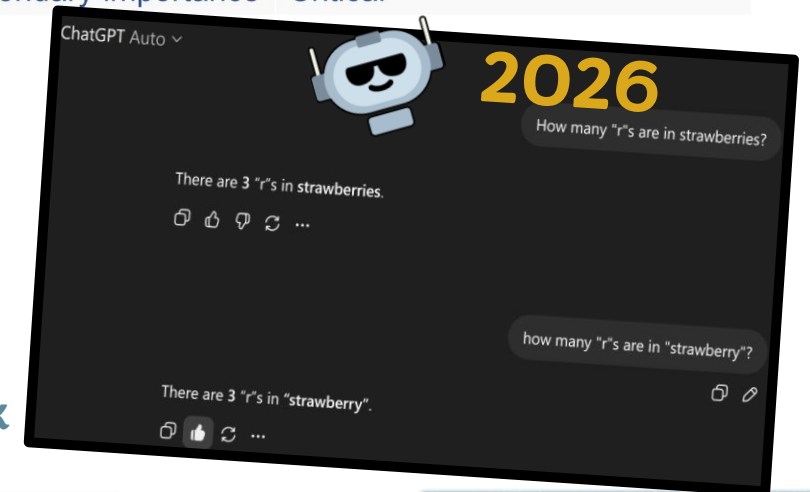
- Reproducibility and stability are key for trustworthy scientific research
- Development and production have different needs
- MLOps can help:
 - mitigate development challenges
 - speed up experimentation

	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

→ 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

**THANK
YOU**



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_INFNO Advanced Hackathon. <https://agenda.infn.it/event/43129/contributions/247489/>



Backup

