UNIVERSITY OF BOLOGNA

Second Cycle Degree in Computer Engineering

Cybersecurity M Project

Dataset Anonymizer

AI-based system for anonymizing data, ensuring privacy while maintaining reliability for statistical analysis

Authors:

Francesco Simoni

Giacomo Vigarelli

Data:

16/12/2024

ACADEMIC YEAR 2024-2025

Project Goals

Project Goals

Developing an AI model that ensures the anonymization of a dataset with good accuracy

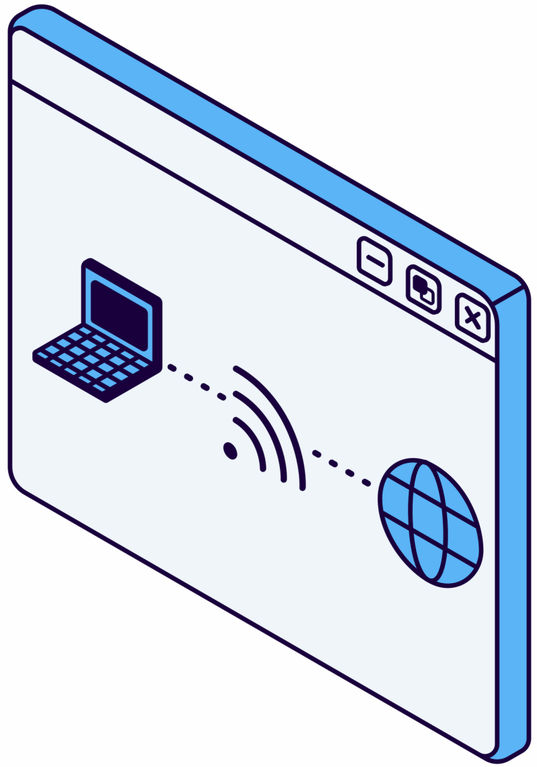
The following will be presented:

- Configuring, training, and evaluating ML models on the MNIST dataset
- Integrating Differential Privacy mechanisms
- Analyzing accuracy and loss metrics for models with different privacy levels

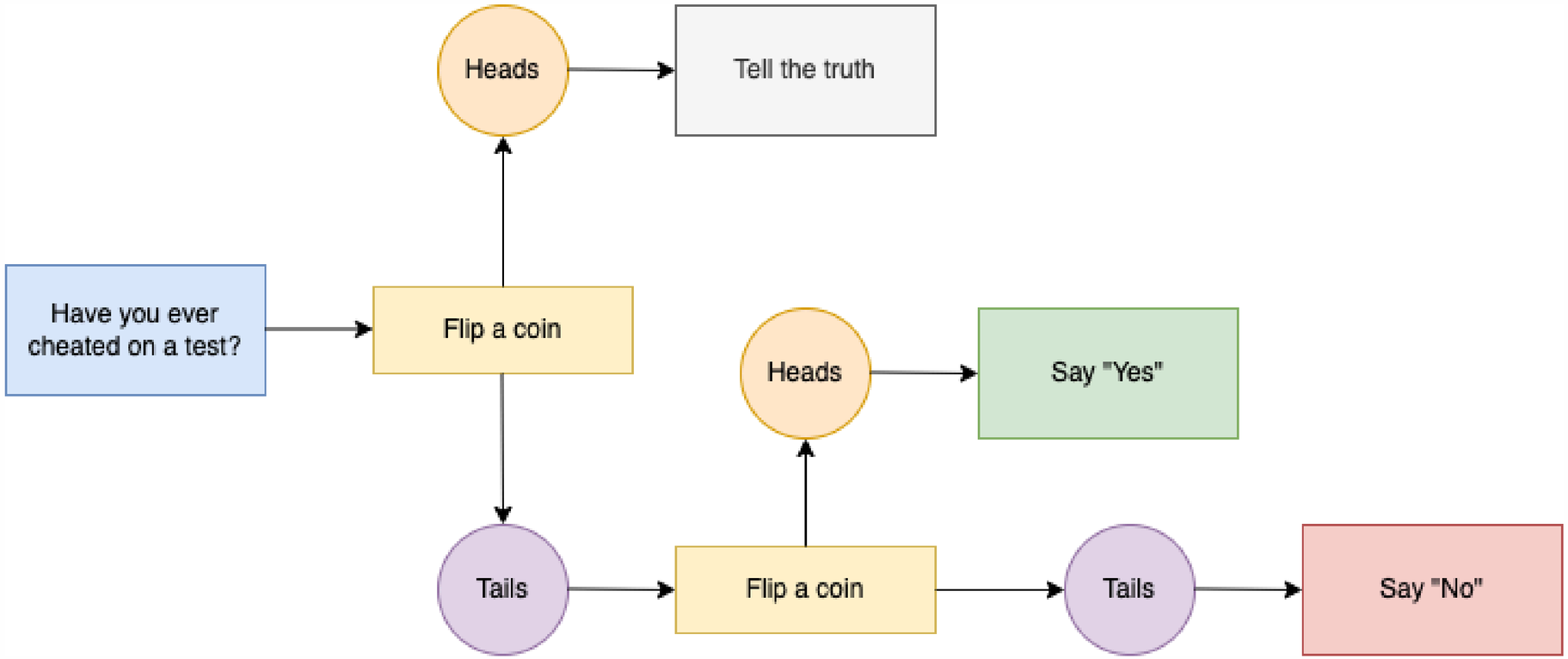
Differential Privacy

Key Concepts in Differential Privacy

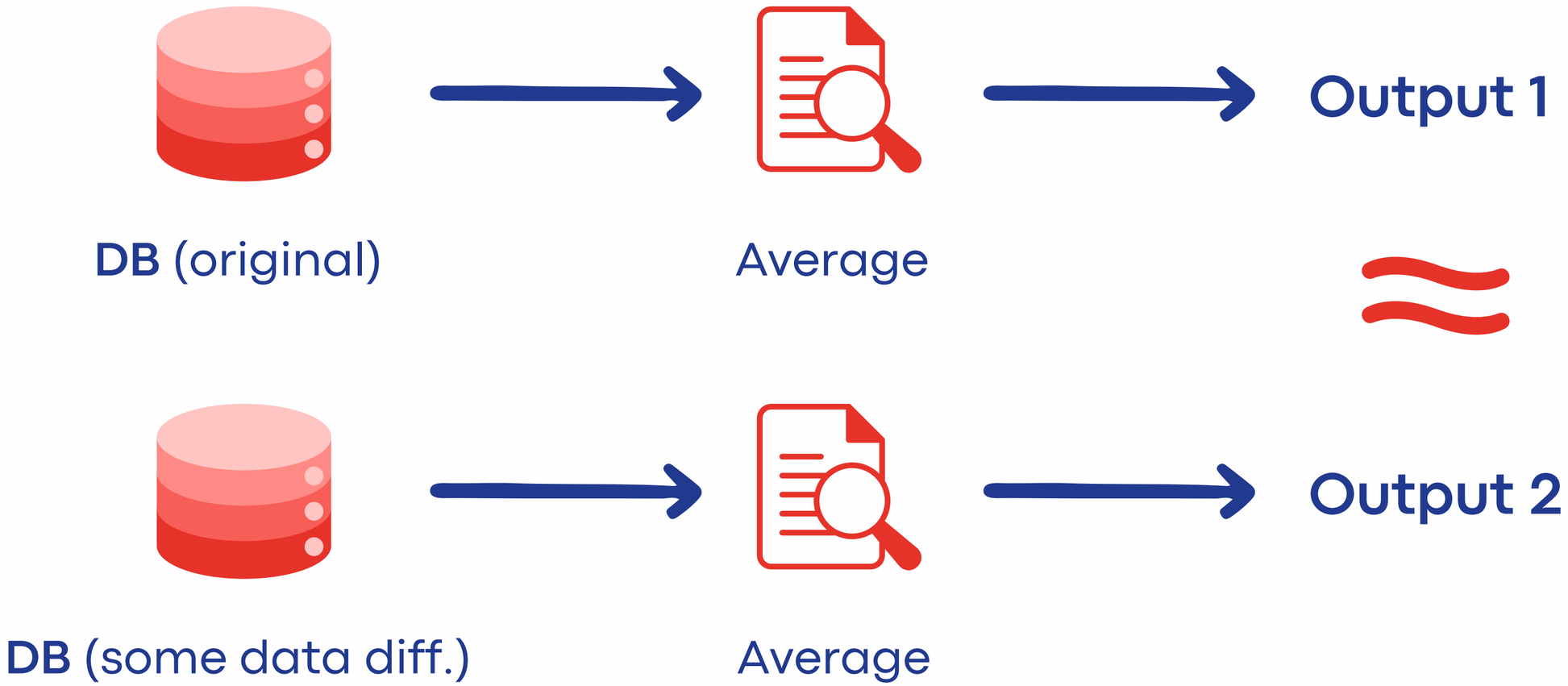
- Technique used to protect Individual Privacy in Dataset
- Ensure that presence/absence of specific data doesn’t change the output of a statistic analysis
- If an attack query is done, it is nearby impossible to determine the specific contribution



II Example


Heads
Tell the truth
Have you ever cheated on a test?
Flip a coin
Heads
Say "Yes"
Tails
Flip a coin
Tails
Say "No"

II Example

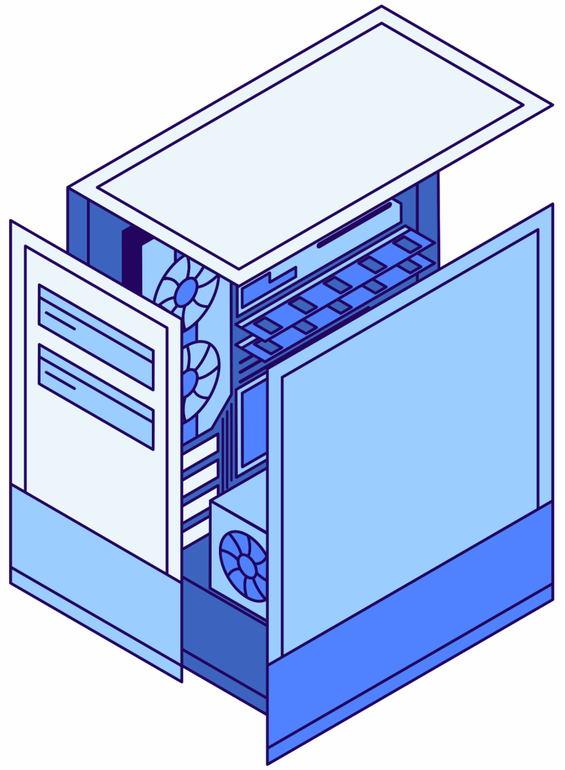

DB (original)
Average
Output 1
≈
DB (some data diff.)
Average
Output 2

II Epsilon Parameter

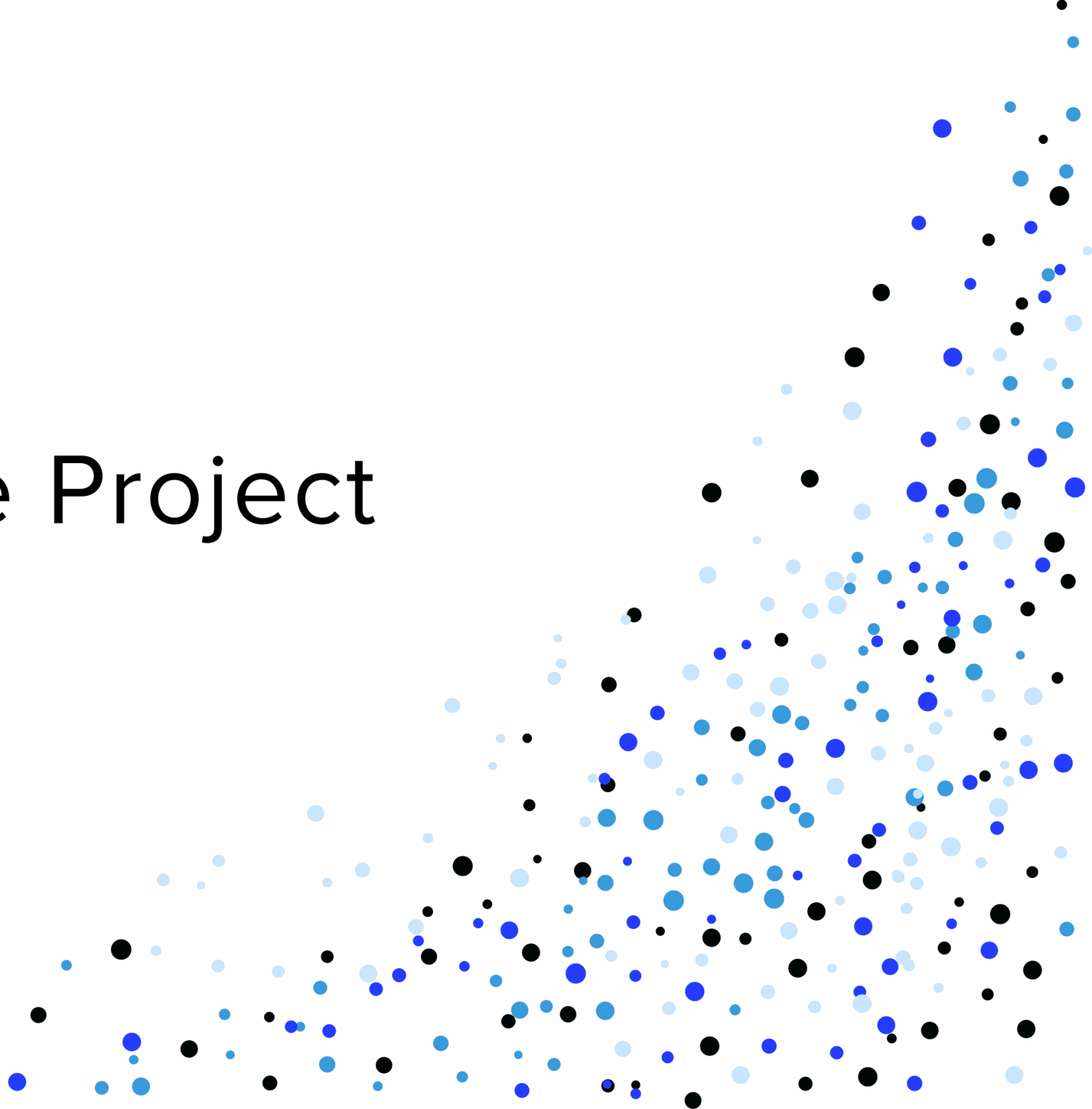
Properties of ε, derived from the DP formula:

- Measures privacy level protection
- Low **ε**: Higher privacy (Lower accuracy)
- High **ε**: Higher accuracy (Lower privacy)
- Different **ε** tests to find optimal balance

The parameter will be fundamental for the analysis and final results.

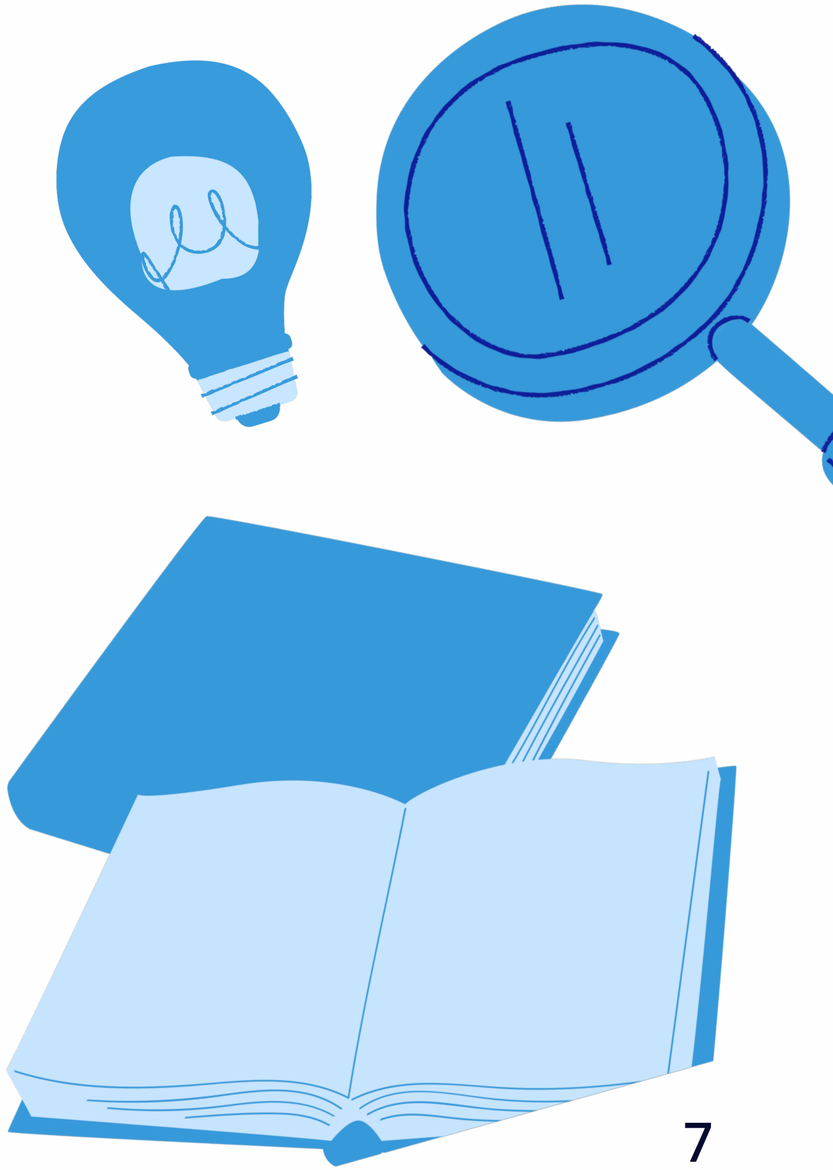


The Project

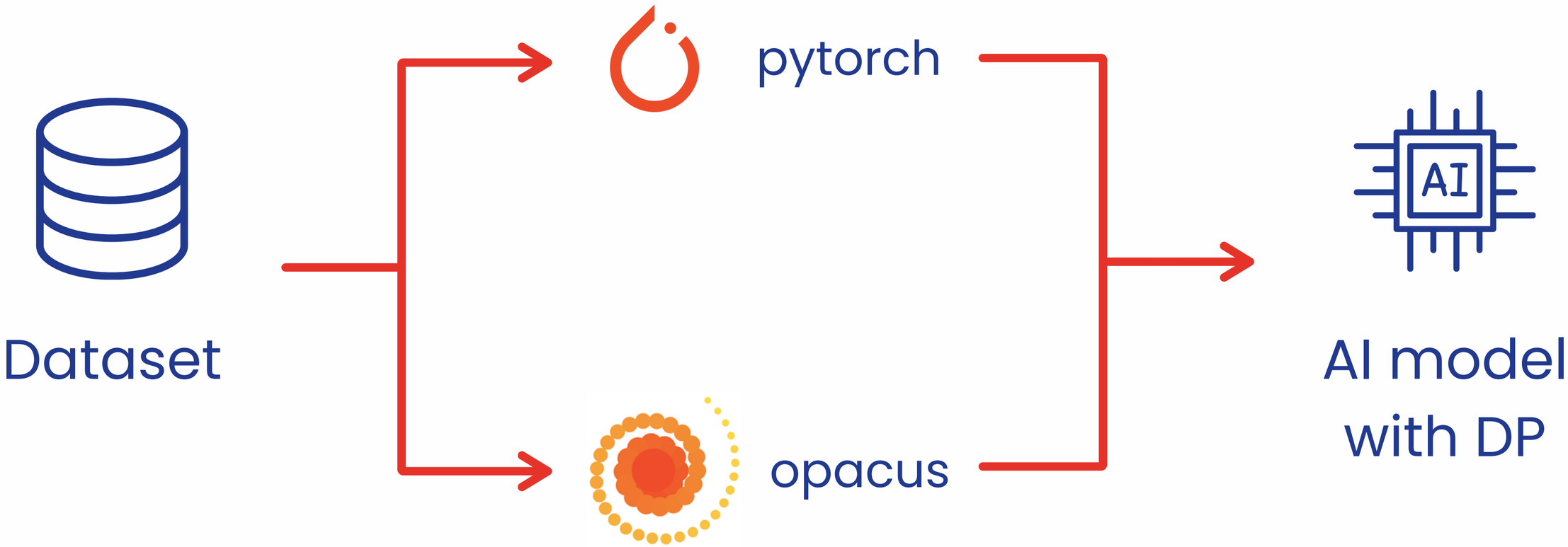


Technologies Used

- **Python 3.11**: the most common programming language for ML models
- **PyTorch**: a training framework for Python along with *TorchVision*, a specific extension for the MNIST dataset
- **Opacus**: a library to implement DP in PyTorch-based models
- **MatPlotLib**: a library for graph visualization



III Implementation



Which dataset should we use?

We evaluated two datasets:

- **MNIST**: An image dataset containing 60,000 samples of handwritten digits (0 to 9)
- **ADULTS** (UCI) : A textual dataset with 48,000 records, each containing an individual's name, surname, annual gross income and other sensible data

We chose MNIST due to technical incompatibilities with textual datasets and the tools used.

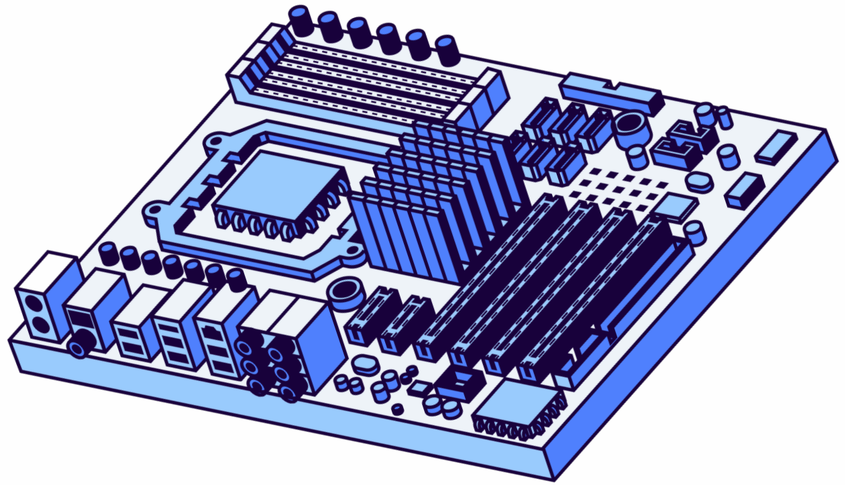
III Project Modular Structure

We designed a *modular* structure in Python to ensure that each *.py* file has a clear and specific responsibility. This approach improves **code maintainability** and **scalability**. The ***src*** folder contains the following files:

```
src
 |
 |-- plots                  # The resulting graphs are saved
 |-- model.py               # Define and validate the ResNet model
 |-- data.py                # Load and prepare the MNIST dataset
 |-- plotter.py             # Generate metric visualizations
 |-- trainer.py             # Train and evaluate the ML model
 |-- main.py                # Orchestrate all process
```

III Model

ResNet18: A convolutional neural network (CNN) designed for image classification. Its ***efficiency*** and architecture make it suitable for tasks with limited computational resources



Modifications:

- Adapted the first convolutional layer to handle grayscale images (single-channel input)
- Configured for 10 output classes to align with the MNIST dataset

III Trainer

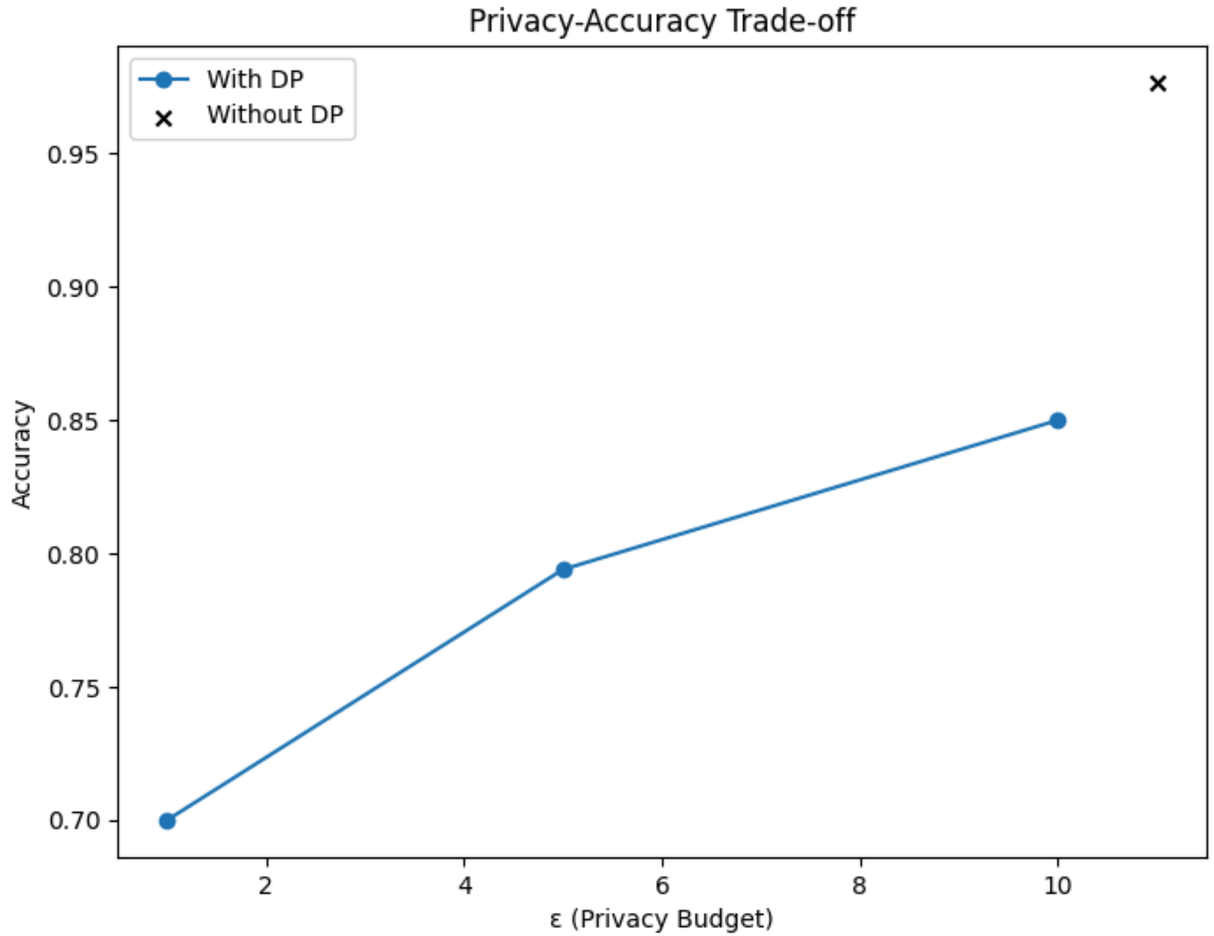
The training pipeline implements the integration of DP using the **Opacus** framework. The PrivacyEngine ensures privacy by adding noise to gradients and clipping them, balancing privacy and accuracy. *Custom parameters* like epsilon and epochs can be insert easily

```python
# Enables DP for the specific model
def _enable_dp(self):
    self.privacy_engine = PrivacyEngine()
    self.model, self.optimizer, self.train_loader = self.privacy_engine.make_private_with_epsilon(
        module=self.model,
        optimizer=self.optimizer,
        data_loader=self.train_loader,
        epochs=self.epochs,
        target_epsilon=self.epsilon,
        target_delta=self.delta,
        max_grad_norm=self.clipping_threshold,
    )
    print(f"[DEBUG] Using sigma={self.optimizer.noise_multiplier} and C={self.clipping_threshold}")
```

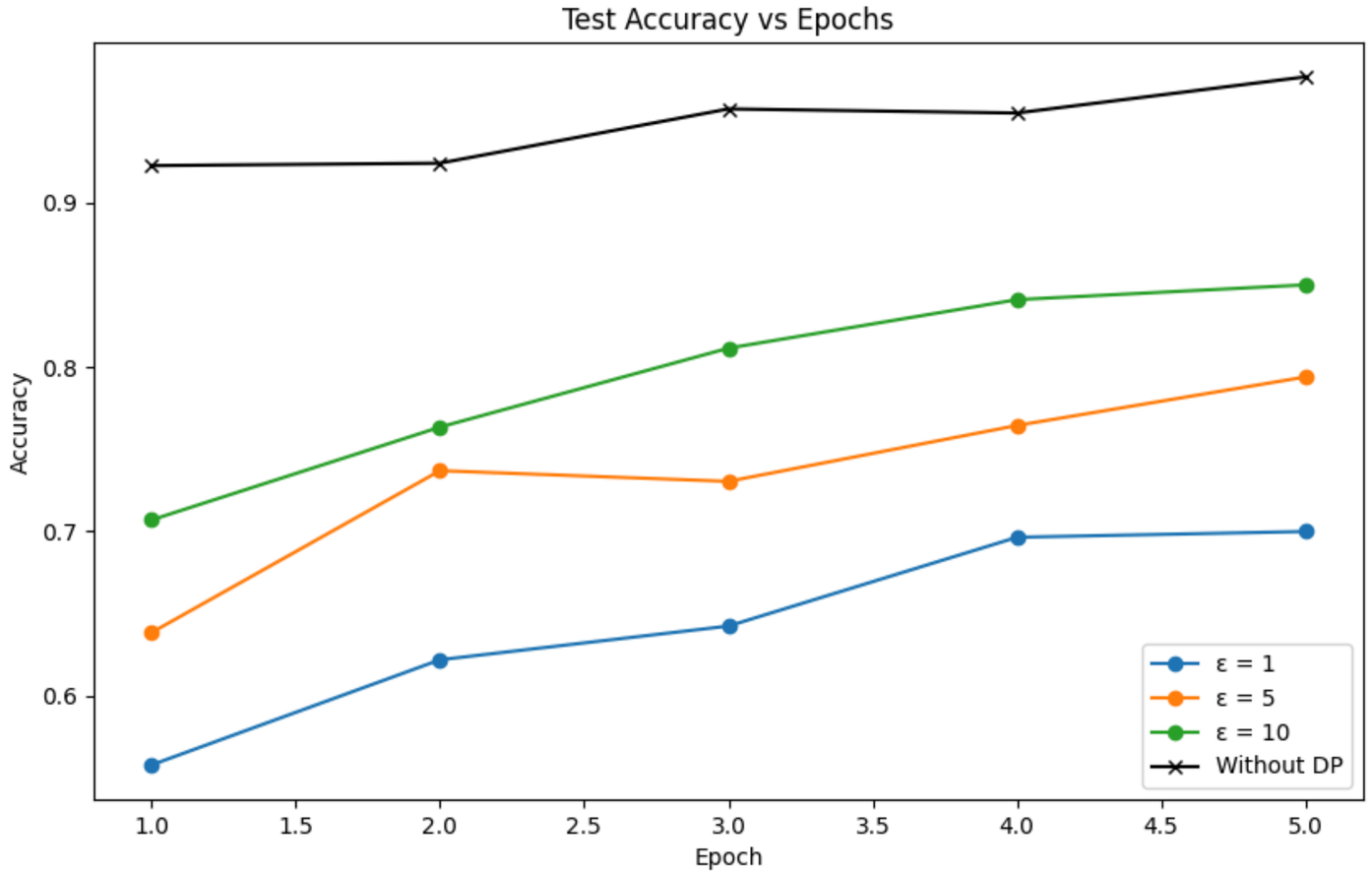
How Opacus integrates with PyTorch.

Results & Conclusions

IV Result: *Trade-OFF*



IV Result: *Accuracy vs Epochs*



V Conclusions

- Successfully implemented a ML prototype for data anonymization using differential privacy
- Training configuration included 5 epochs, [1, 5, 10] epsilon values and reduced datasets MNIST (10,000 training images and 2,000 test images) to optimize computational resources
- There is a clear trade-off between privacy and performance: as privacy weakened (higher ϵ), the model's accuracy improved
- Showcased the importance of customizing existing models to meet specific requirements

16/12/2024

MSc Computer Engineering, Unibo

Thanks for the attention

Authors:

Francesco Simoni

Giacomo Vigarelli