Machine Unlearning

Master's Degree in Information Engineering
Course of Multimedia Data Security

Nicola Cappellaro
Riccardo Zannoni

TABLE OF CONTENTS

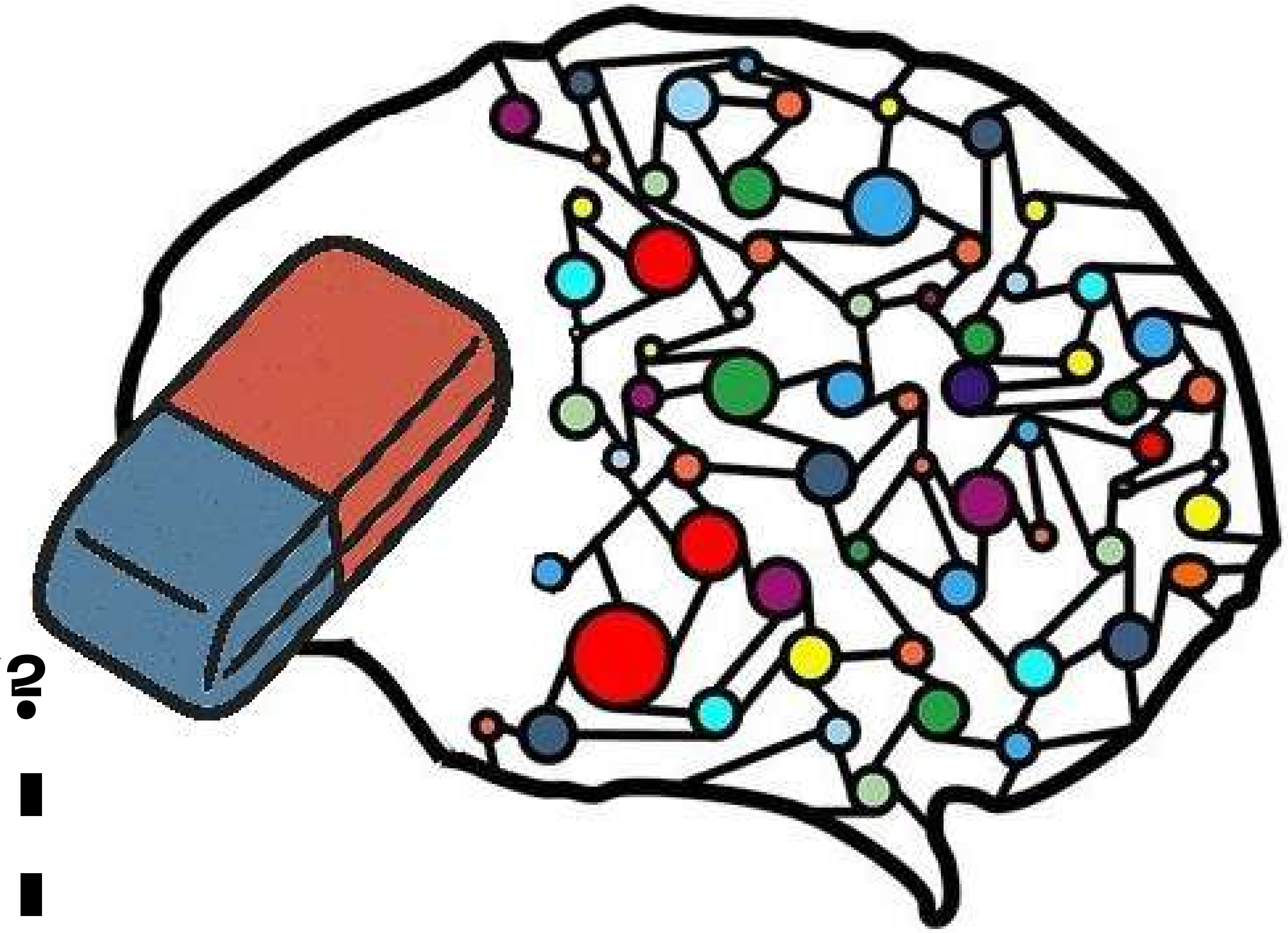
INTRODUCTION

WHAT?

Machine unlearning refers to techniques designed to **selectively remove** the influence of specific data from an already-trained model. The goal is to make the model "forget" specific information **without requiring a full and expensive retraining** from scratch.

WHY?

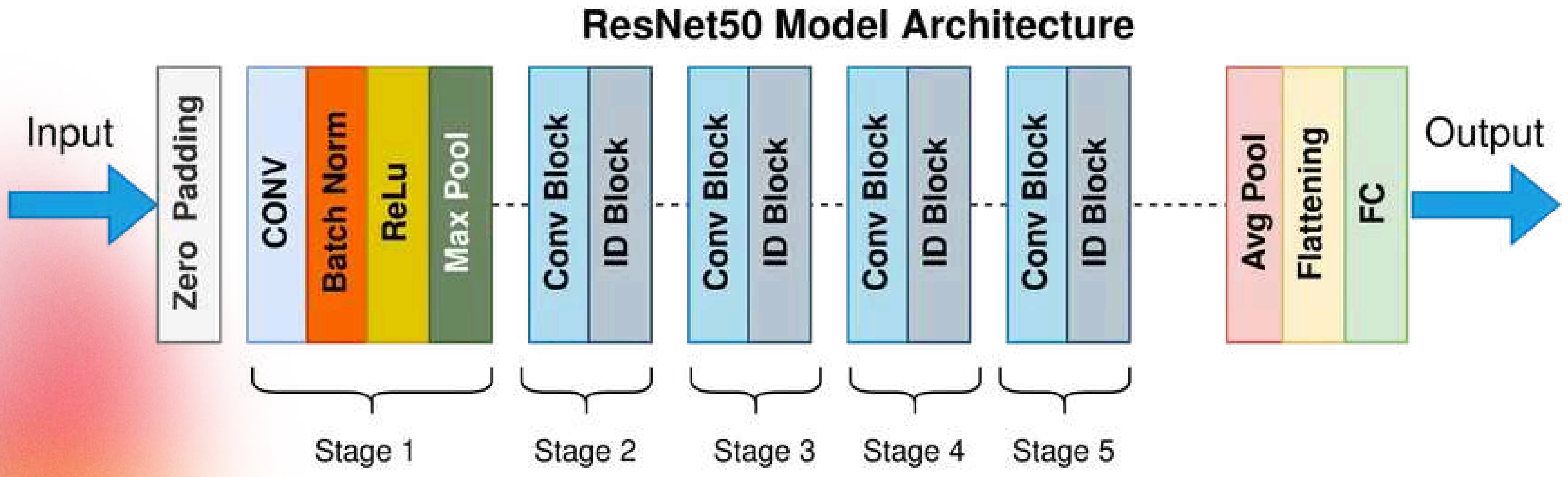
- Privacy
- GDPR
- Data poisoning
- Right to be forgotten



NN MODEL

The model is a modified ResNet50 where:

- The first convolutional layer is modified to prevents the early spatial downsampling.
- all pretrained weights are kept
- The fully connected layer is replaced to produce a single output



DATASET

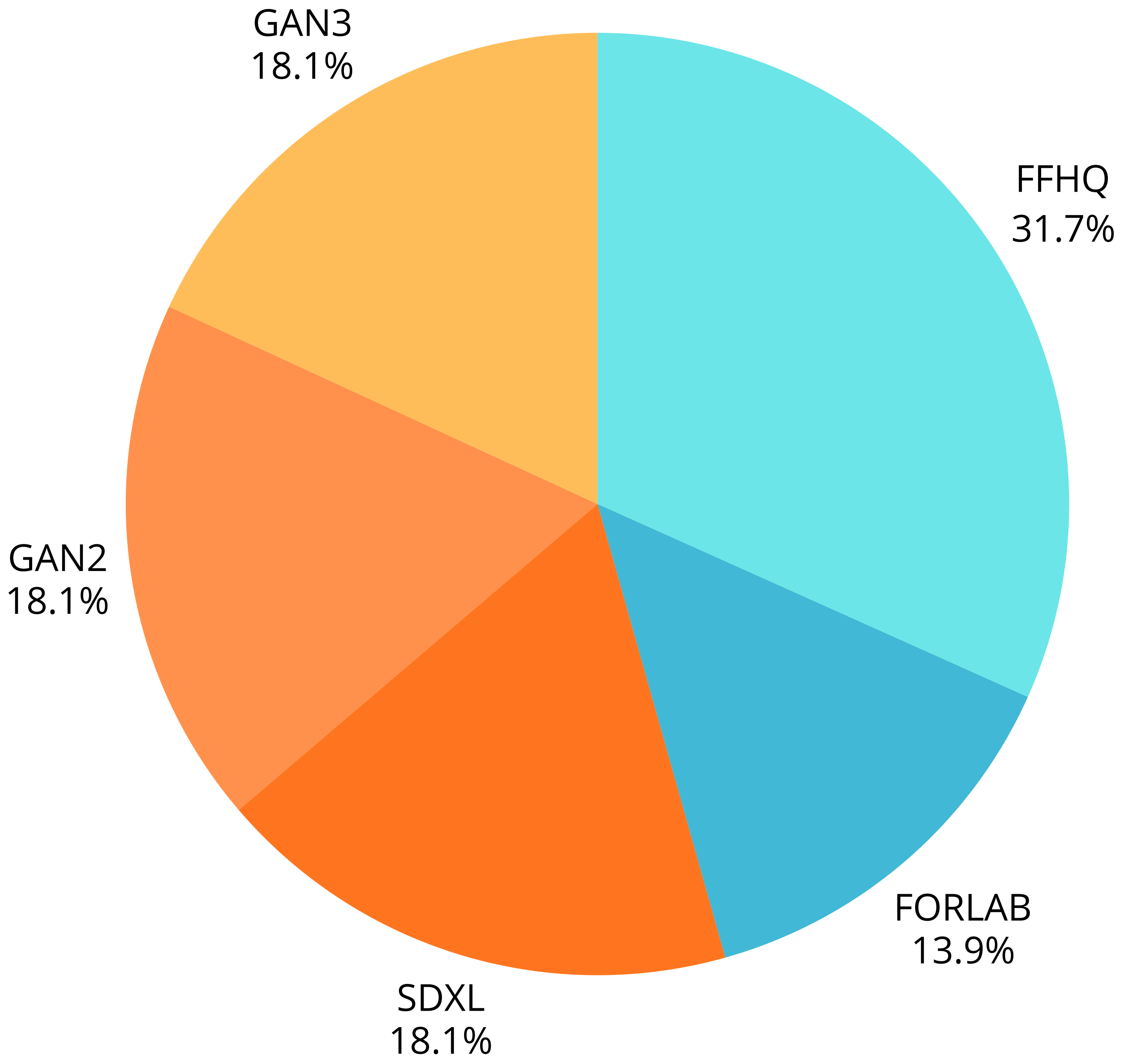
Path: /media/NAS/TrueFake
Only **PreSocial** folder:

- **Real**

FFHQ: 70000 images
FORLAB: 30719 images

- **Fake**

SDXL: 40000 images
GAN2: 40000 images
GAN3: 40000 images



PROJECT STRUTURE

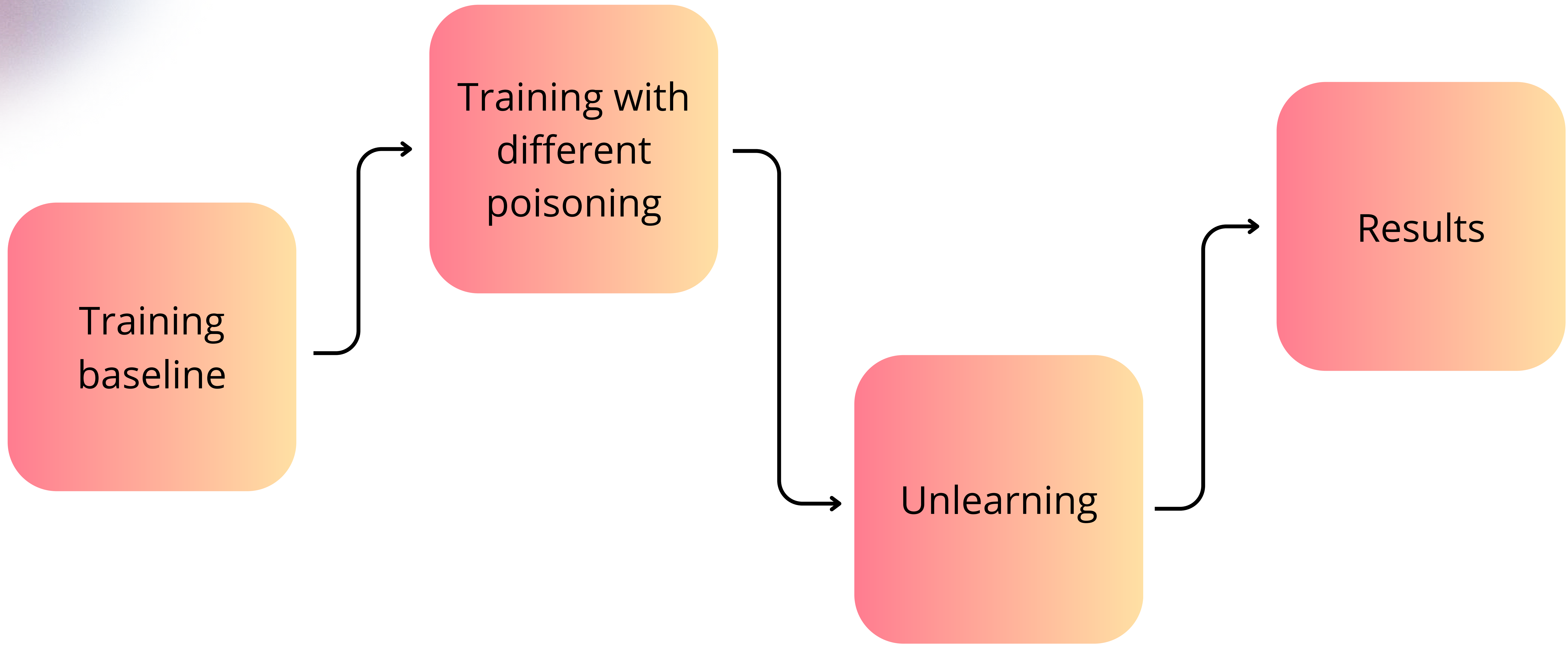
UNIVERSITÀ DI TRENTO

- **dataset.py**: dataset loading, preprocessing, and controlled data poisoning.
- **networks.py:** model architecture (modified ResNet50) with optional layer freezing.
- **train.py** : standard training pipeline for the ResNet-based classifier.
- **unlearn_trueface.py**: implementation of the projected-gradient unlearning procedure.
- **unlearn_utils.py**: SVD computation, subspace projection utilities, and evaluation helpers.

```
├── runs
│   └── 0_poison
│   └── 20_poison
│   └── 50_poison
│
├── dataset.py
├── launcher.py
├── networks.py
├── parser.py
├── train.py
│
├── unlearn_trueface.py
├── unlearn_utils.py
├── run_unlearning.sh
│
├── environment.yml
└── README.md
```

IMPLEMENTATION

OVERVIEW:



TRAINING

UNIVERSITÀ DI TRENTO

TRAINING PARAMETERS:

- Model nodown
- Freeze ResNet50 weight
- Learning rate 0.0001
- lr_decay_epochs 3
- num_epochs 10
- batch_size 16
- optimizer: Adam
- loss: BCEWithLogitsLoss

DATA AUGMENTATION

- Image normalization,
- resize
- crops
- blur
- JPEG compression

→ improve robustness and mimic real world distortions.

TRAINING

TRAINING LOOP

Epoch loop

- Load batch → get images and labels
- Forward pass → scores = model(data)
- Compute Loss → loss = criterion(scores, label)
- Zero grad → optimizer.zero_grad()
- Backward → loss.backward()
- Update weights → optimizer.step()

POISON

UNIVERSITÀ DI TRENTO

We apply poison by flipping the dataset labels with a given **poison rate** (e.g., 10%, 20%)

Selection: random.sample() on dataset indices

Labels are **flipped** as follows:

Real (0) → Fake (1)

Fake (1) → Real (0)

Tracking: poisoned indices saved to runs/poison_info/poison_xx.pkl for unlearning phase

UNLEARNING

DETAILED OVERVIEW:

Based on **Projected Gradient Unlearning** (PGU)

Goal:

- Reduce the model's confidence on the data that must be "forgotten"
- Preserve performance on the remaining clean data

Three **key components:**

- Unlearning loss
- SVD and CGS Construction
- Projected gradient update

LOSS FUNCTION

UNIVERSITÀ DI TRENTO

Loss 1: reduce the model's confidence on the poisoned label:

$$loss1 = -log(1 - \mid p - (1 - label) \mid - offset)$$

Loss 2: increase the entropy of the predictions:

$$loss2 = -H(p), where$$
$$H(p) = -p\log(p) - (1-p)\log(1-p)$$

Total loss: weighted sum of the two terms:

$$L = w1 \cdot loss1 + w2 \cdot loss2$$

SVD + CGS

UNIVERSITÀ DI TRENTO

We want to **remove** the influence of poisoned data without damaging the model

To understand **which directions** represent useful knowledge, we compute the **SVD** of clean-data activations.

We keep only the **most important directions** (e.g., 95% variance): this forms the clean-data subspace U_k

We then build the projection matrix using **CGS**:

$$P = U_k U_k^T$$

This matrix is used to **project gradients** onto the clean-data subspace.

$$g_{proj} = Pg$$

GRADIENT UPDATE

After computing the unlearning loss and projecting the gradient, we update **only the safe part** of the model. The projected gradientis g_{proj} is applied to the parameters:

$$\theta \leftarrow \theta - \eta\, g_{\text{proj}}$$

This ensures the update:

1. **removes** the influence of poisoned data
2. **preserves** the clean-data subspace learned by the model, avoiding catastrophic forgetting.

Only the final fully connected layer is updated when freeze=True.

➜ The model is updated using only the gradient components that do not interfere with clean-data knowledge.

RESULTS

UNIVERSITÀ DI TRENTO

Performance 0% poison
Accuracy: 97.79%
Precision: 98.03%
Recall: 97.91%
F1 Score: 97.97%

Performance 20% poison
Accuracy: 96.39%
Precision: 94.17%
Recall: 99.54%
F1 Score: 96.78%

Performance 20% unlearn
Initial Clean Train Acc: 96.48%
Final Clean Train Acc: 97.52%
Clean Acc: +1.04%

Initial Poison Train Acc: 99.48%
Final Poison Train Acc: 98.73%
Poison Acc: -0.75%

Initial Test Acc: 96.67%
Final Test Acc: 97.64%
Test Acc Change: +0.97%

Performance 50% poison
Accuracy: 42.62%
Precision: 46.19%
Recall: 31.74%
F1 Score: 37.62%

Performance 50% unlearn
Initial Clean Train Acc: 42.20%
Final Clean Train Acc: 48.74%
Clean Acc: +6.54%

Initial Poison Train Acc: 55.04%
Final Poison Train Acc: 0.00%
Poison Acc: -55.04%

Initial Test Acc: 42.62%
Final Test Acc: 45.47%
Test Acc Change: +2.86%

CONCLUSION

UNLEARNING 20% POISON

1. The model with 20% poison was already excellent, almost identical to the clean model.
2. The amount of harmful information is small → unlearning only slightly changes the behavior.

Unlearning removes the "poisoned" component, but since the model was already very good, this results in only a slight improvement (~1%).

UNLEARNING 50% POISON

1. The model is severely corrupted: accuracy collapses to ~42%, so unlearning has much more to remove.
2. The unlearning process successfully eliminates this poisoned influence (poison accuracy drops to 0%), and the model partially recovers: clean accuracy improves (+6.5%)

Although performance cannot return to clean-model levels, this is expected: the goal of PGU is to forget the poisoned data without harming the clean data.

THANK YOU FOR YOUR ATTENTION

Nicola Cappellaro – nicola.cappellaro@studenti.unitn.it
Riccardo Zannoni – riccardo.zannoni@studenti.unitn.it