Delving in the loss landscape to embed robust watermarks into neural networks

Enzo Tartaglione, Marco Grangetto,

Davide Cavagnino and Marco Botta

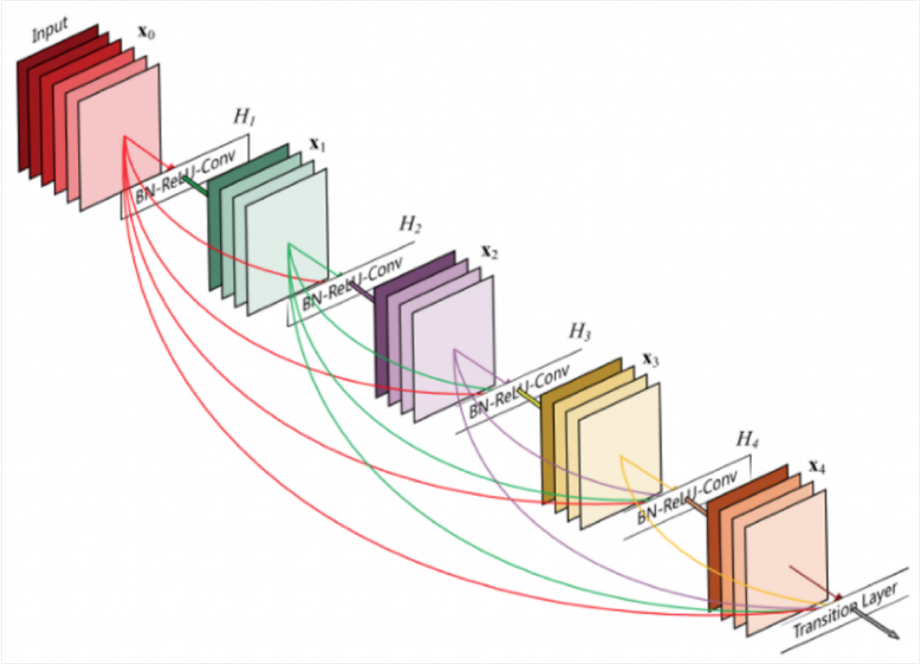
EIDOSlab

Image processing
Computer vision
Virtual reality

Università degli Studi di Torino

Deep learning

- Standard tool to tackle supervised learning problems.
- Many state-of-the-art applications
  - Image classification
  - Object detection
  - Image segmentation
  - Natural language processing
  - […]
- Little attention to security and model integrity.



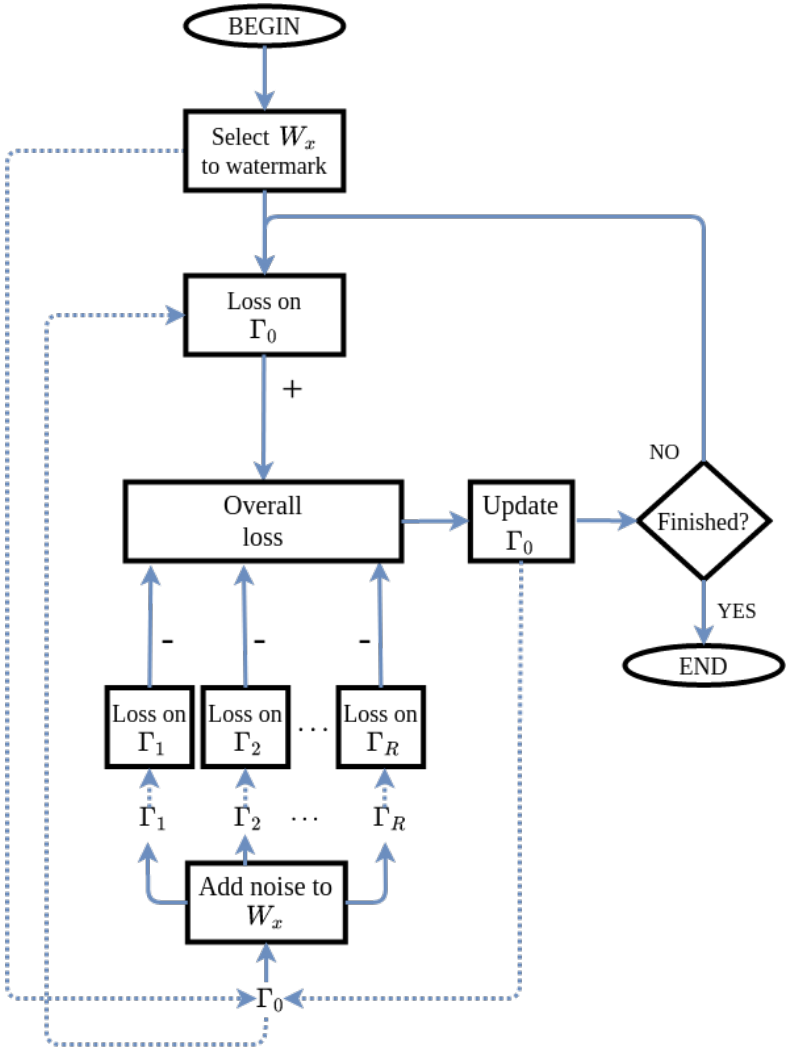
https://datascience.eu/it/apprendimento-automatico/una-panoramica-di-resnet-e-delle-sue-varianti/

Watermarking deep models

- Only few preliminary studies on digital watermarking applied to ANN.
- We start from a simple solution that embeds the watermark in the ANN model weights:
  - ANN models are usually highly redundant;
  - training can be done by fixing a subset of the model weights (the watermark);
  - no impact on the final ANN performance.
- We embed watermark in all the layers of the deep model.
- We train the model to make the model robust to fine-tuning attacks, where an adversary tries to change the model weights by running additional training epochs.
- We empirically study the effectiveness of the proposed approach by showing that it represents an efficient solution achieving a significant level of robustness to attacks without impacting on the ANN performance.

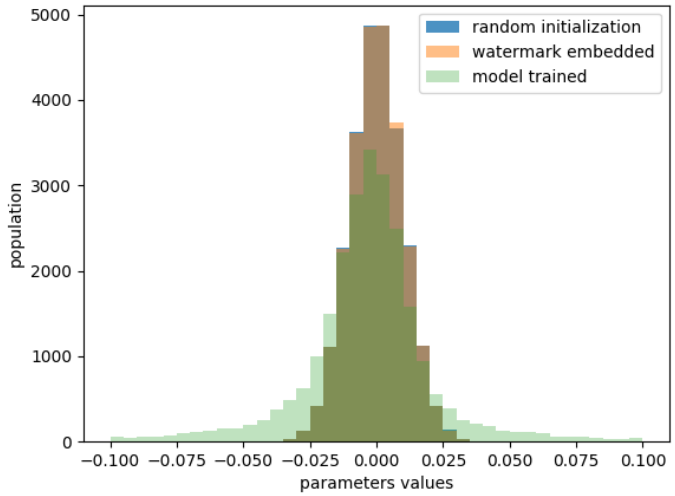
Overview on the technique

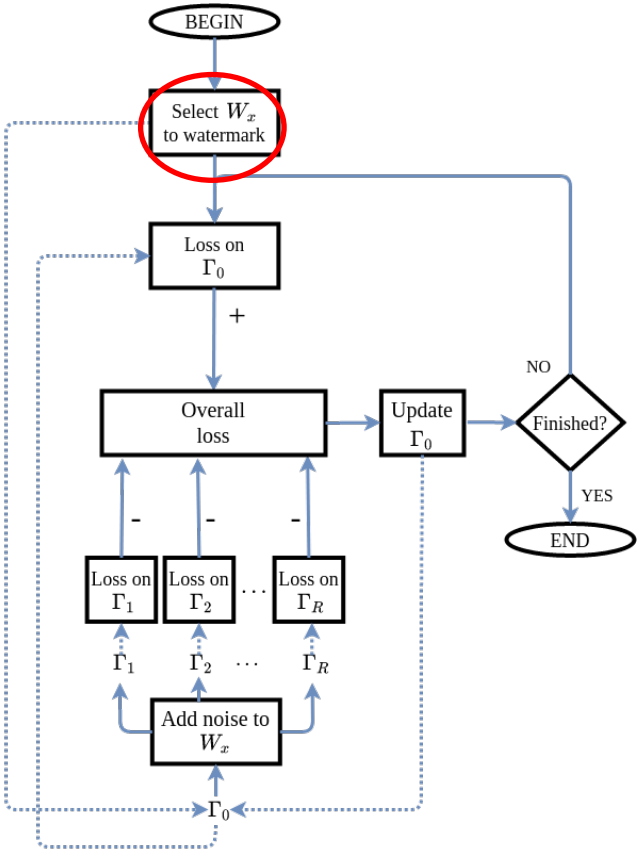
- Watermarking is applied after the model is trained.
- We enforce robustness of the watermark to the most-common attacks.



Step 1: watermark the parameters

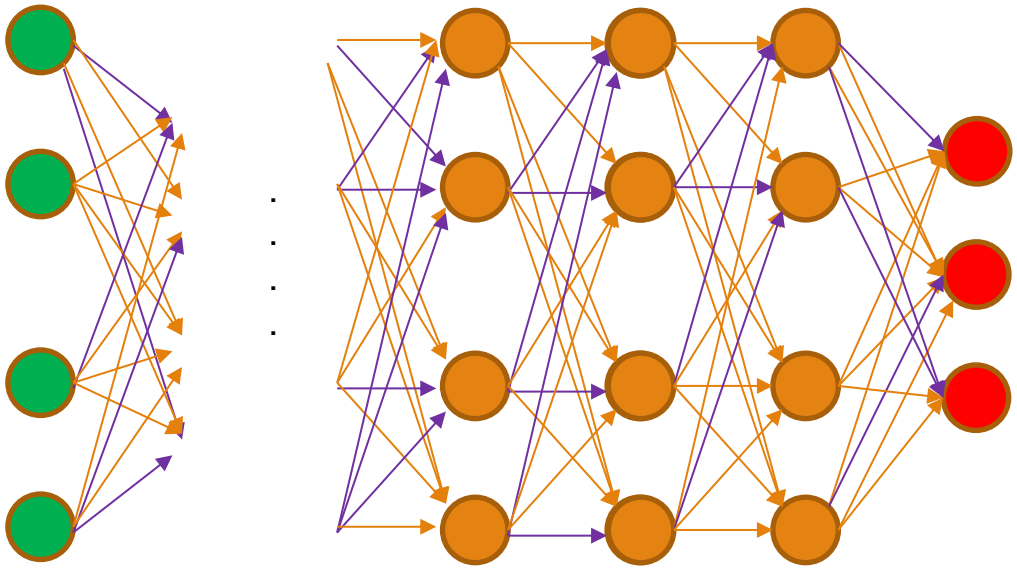
- We watermark the parameters.
- We already know the parameters' distribution.
- We are sure we will not choose outliers in the parameters' distribution.

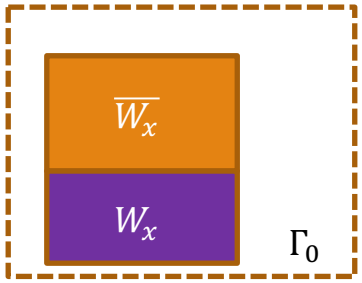




Step 1: watermark the parameters (II)

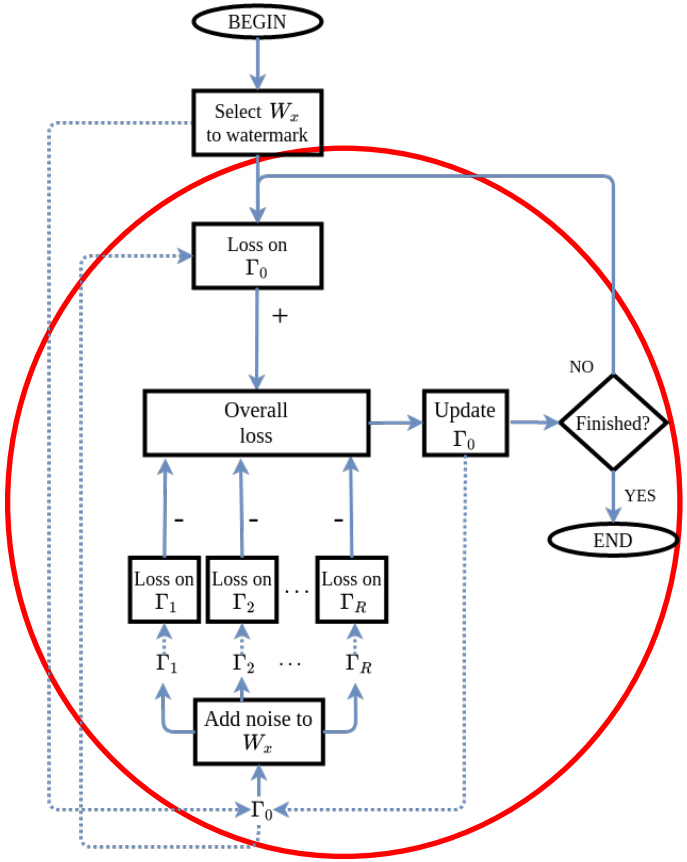
- Parameters are chosen randomly.
- All the learned layers are involved, including the output layer!
- The watermarked parameters are indicated as W_x, while the non-watermarked are $\overline{W_x}$.
- Now on, W_x will not be modified within the model Γ_0.





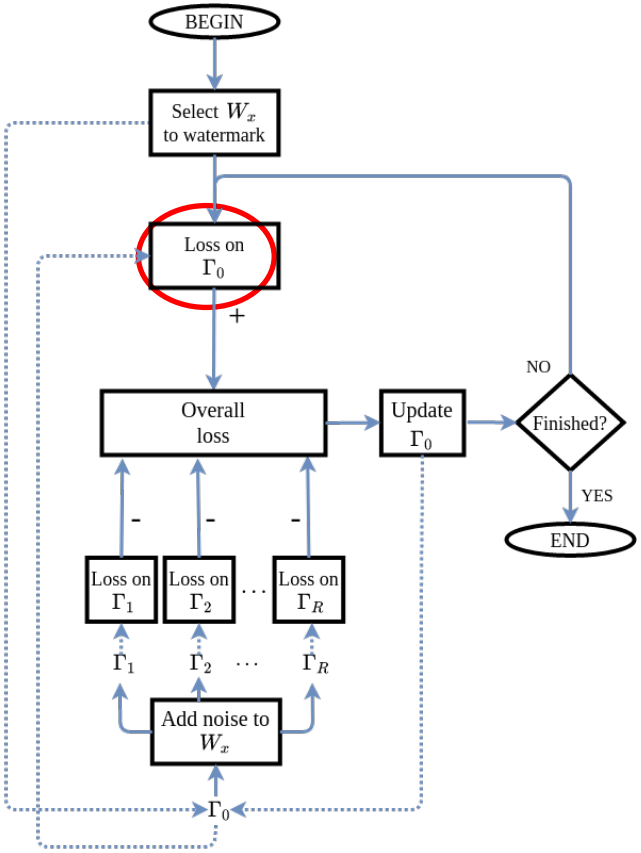
Step 2: Delving in the loss

- Iteratively, we will tune the non-watermarked parameters $\overline{W_x}$ such that the watermarked parameters W_x will be robust to attacks:
  - If W_x parameters are modified, the performance of the model drops;
  - Fine-tuning attacks (keeping the loss low) will naturally change values for $\overline{W_x}$, but not W_x.
- To achieve this, we delve in the loss, driving Γ_0 in a steep valley for the subspace W_x.



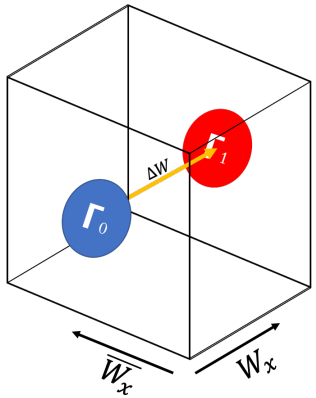
Step 2.1: loss on Γ_0

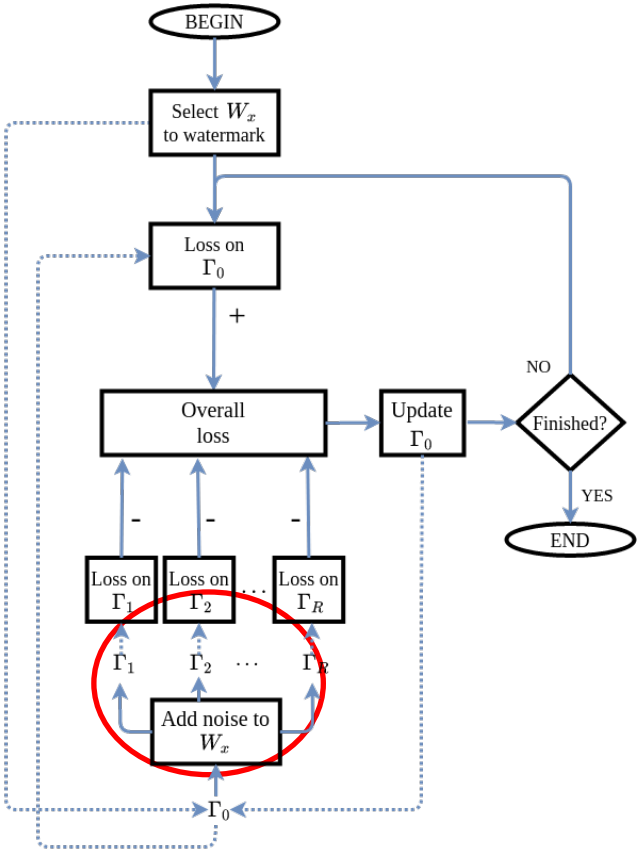
- We compute the loss over Γ_0
- We aim at minimizing this loss (over $\overline{W_x}$) to keep the error low.



Step 2.2: Generate R replicas

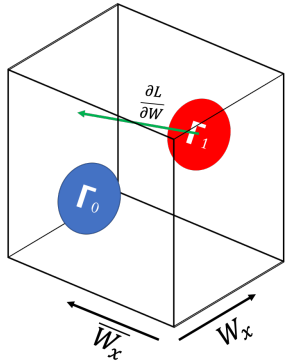
- We generate R replicas of Γ_0
- To these, we add some noise ΔW just to the watermarked parameters W_x
- In this way, we have R samplings at a small distance from Γ_0, in W_x

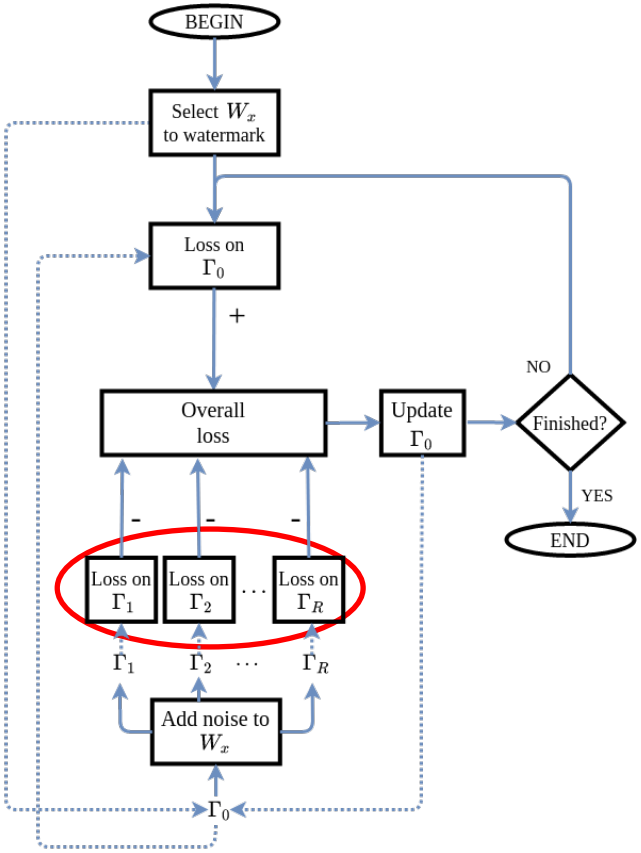




Step 2.3: loss on $\Gamma_1, \Gamma_2, \cdots \Gamma_R$

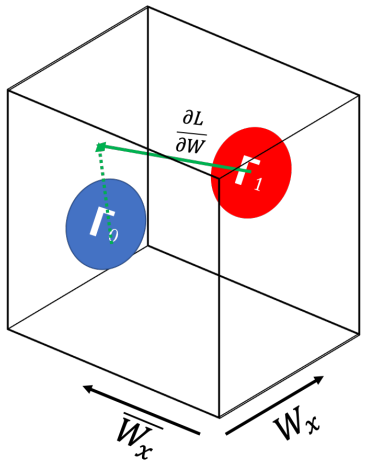
- For each of the R replicas we compute the loss value.
- We aim at maximizing it.
- In this way we guarantee that, at a small distance ΔW from the watermark, the loss value is high.
- And this is how we delve in the loss function, looking for narrow minima in the subspace W_x

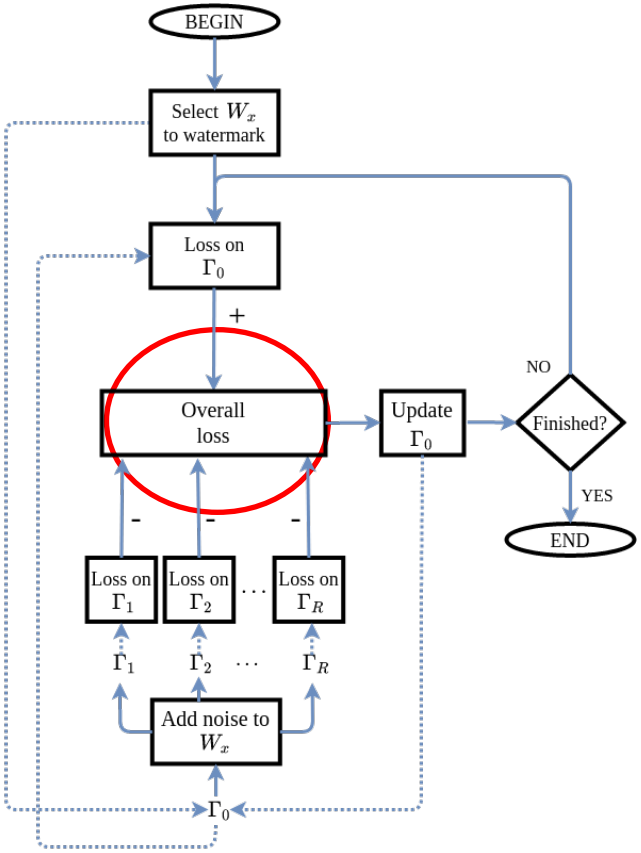




Step 2.4: Overall loss on Γ_0

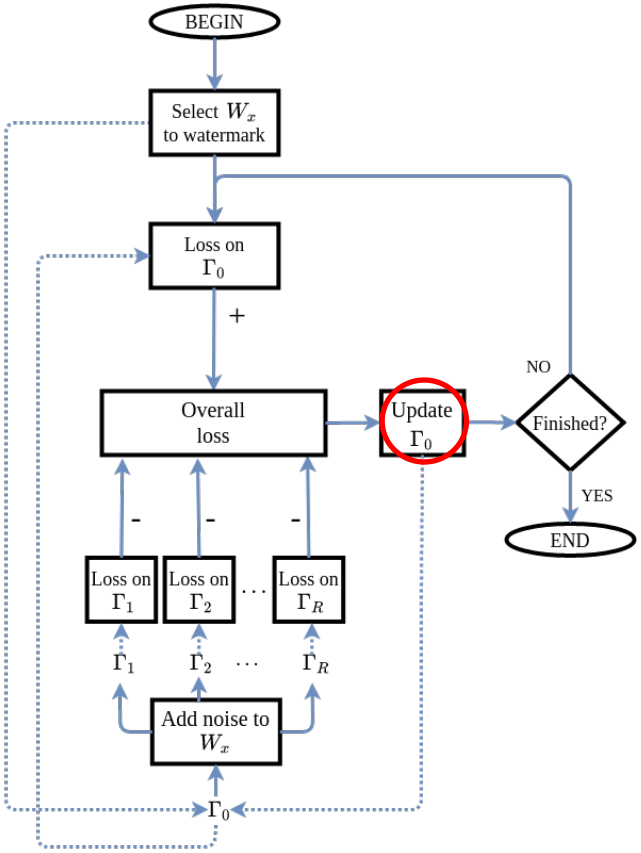
- We get a loss value from Γ_0 we minimize (positive contribution).
- Besides that, we get R other loss contributions we aim at maximizing (negative contribution)



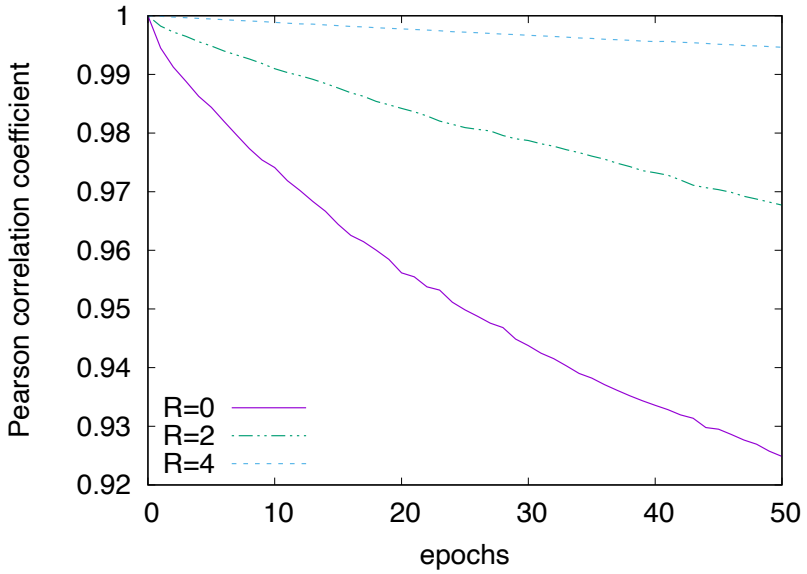


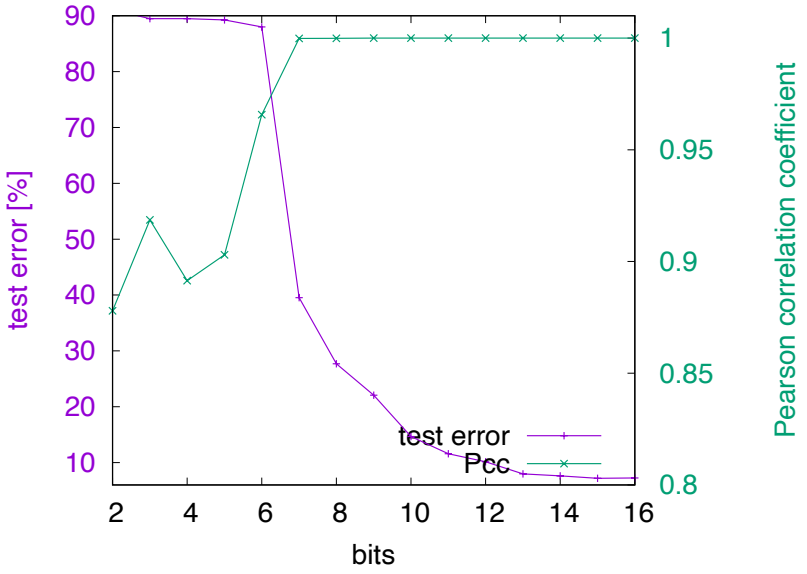
Step 2.5: update

- Finally, we update Γ_0.
- Only the non-watermarked parameters $\overline{W_x}$ are updated: the watermarked ones remain freezed!
- Then, the process is iterated: the more the iterations as well as the replica R used in the process, the best the final robustness of the watermark.

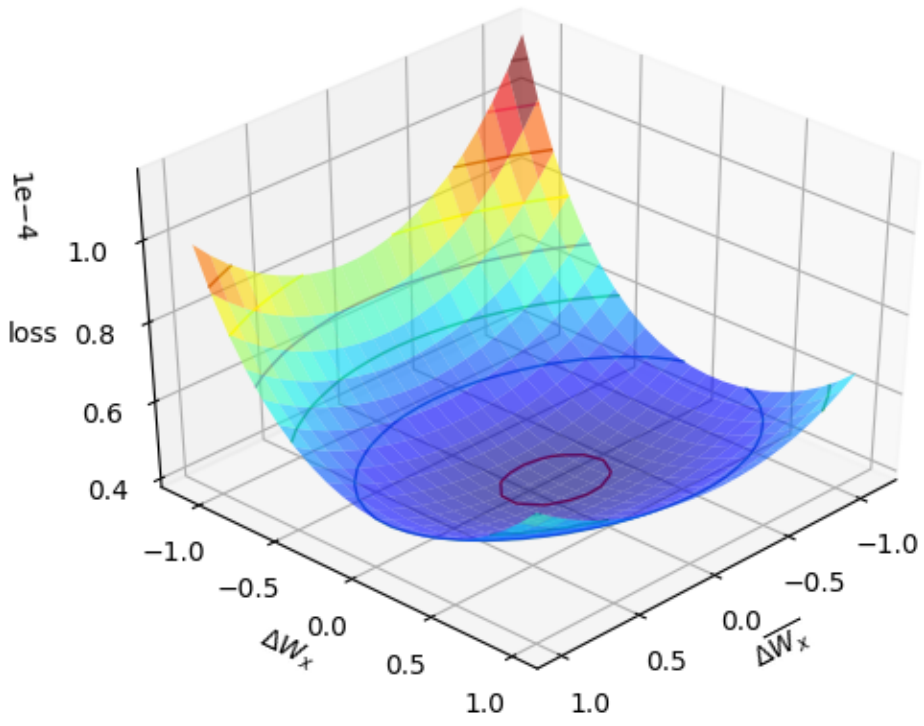


Results – ResNet32 (CIFAR-10)





R	Error on the test set (%)
0	7.14
2	7.08
4	7.29



Thank you!