Watermarking deep neural networks

MN906 – Multimedia content security

Enzo Tartaglione

LTCI, Télécom Paris

enzo.tartaglione@telecom-paris.fr

Outline

- Deep learning 101
- Watermarking and deep neural networks
  - Watermarking multimedia content
  - Watermarking the deep learning model
    - Black-box watermarking
    - White-box watermarking
  - Attacks
    - Gaussian noise addition
    - Fine-tuning attack
    - Quantization attack
    - Pruning attack

Deep learning 101

Supervised training

- Input sample ***x***=(x_1, x_2), compute *y* (*t* known)
- Compute gradient of *the loss L* w.r.t. to each w_n via *chain rule*, e.g.:

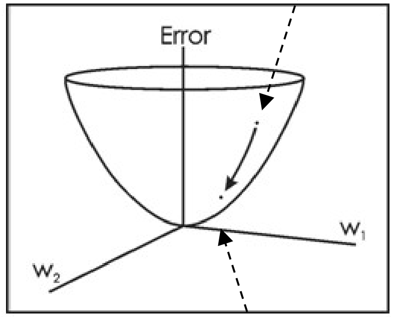
$$\frac{\partial E(w)}{\partial w_1} = \frac{\partial E(w)}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_1}$$

- Update parameters via *error gradient descent*

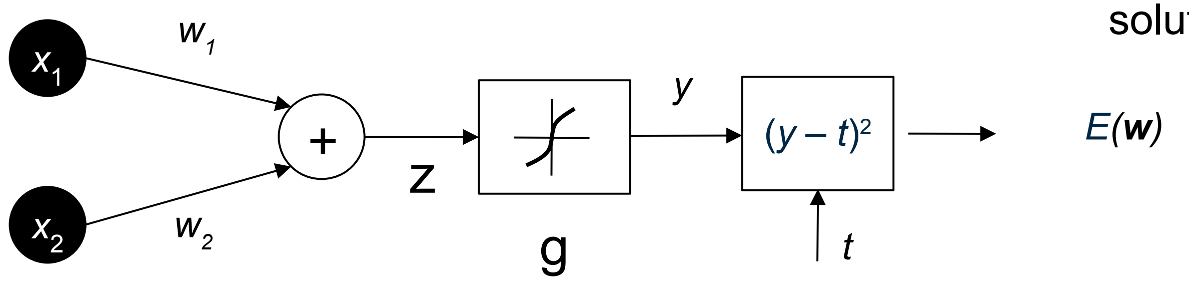
$$w^{t+1} = w^t - \eta \nabla_w E(w)$$

- Iterate on *k+1*-th sample

Current solution

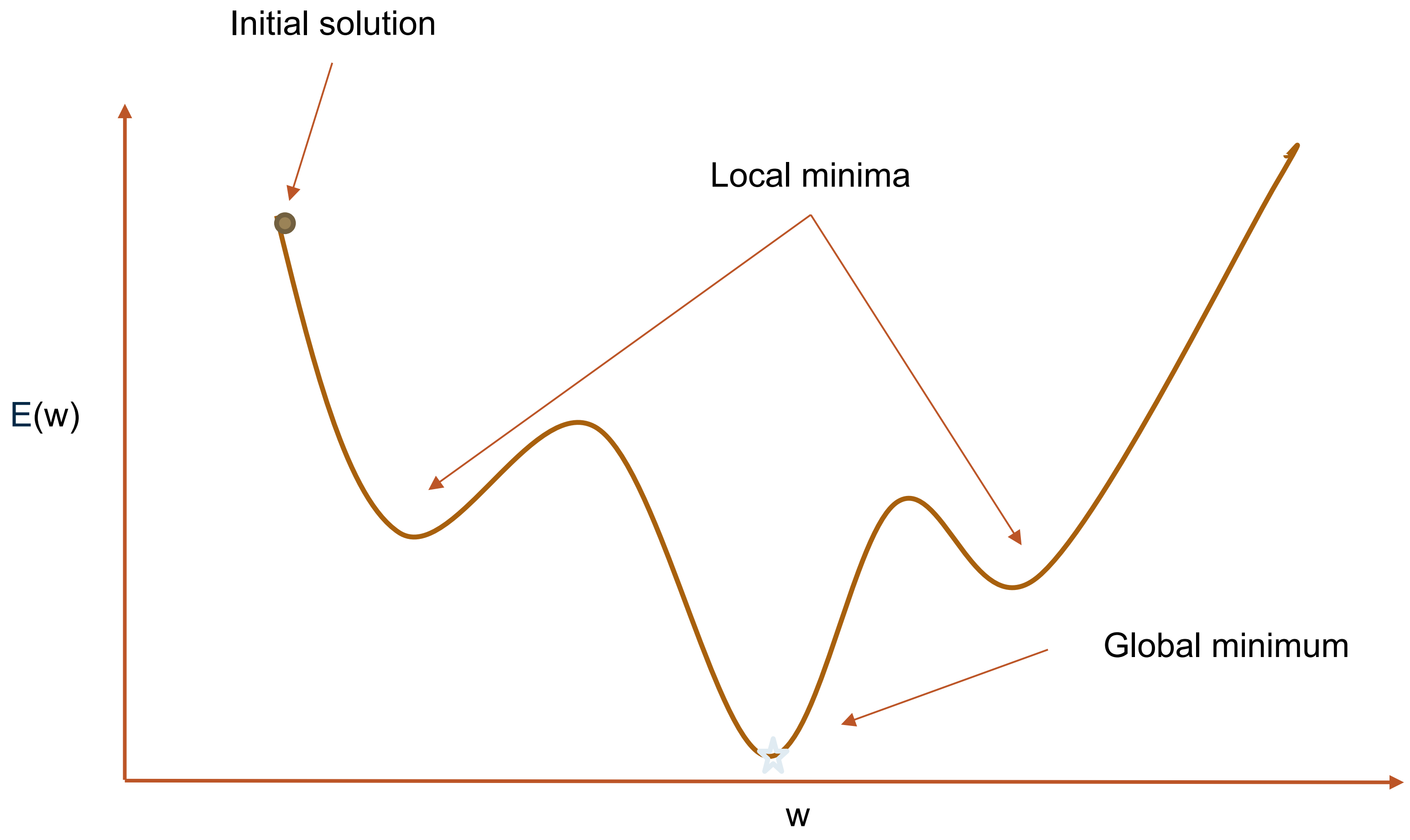


Updated solution



Learning rate choice

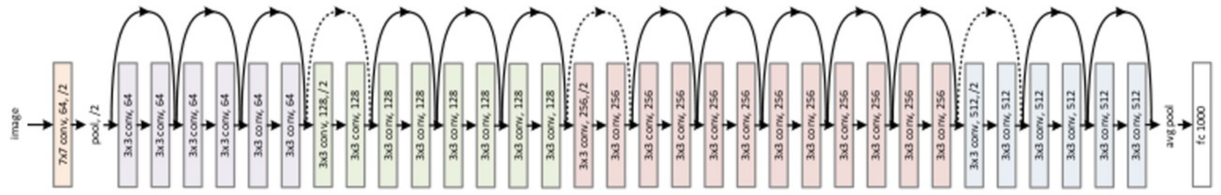
- How to choose *the* learning rate η ?



Transfer learning

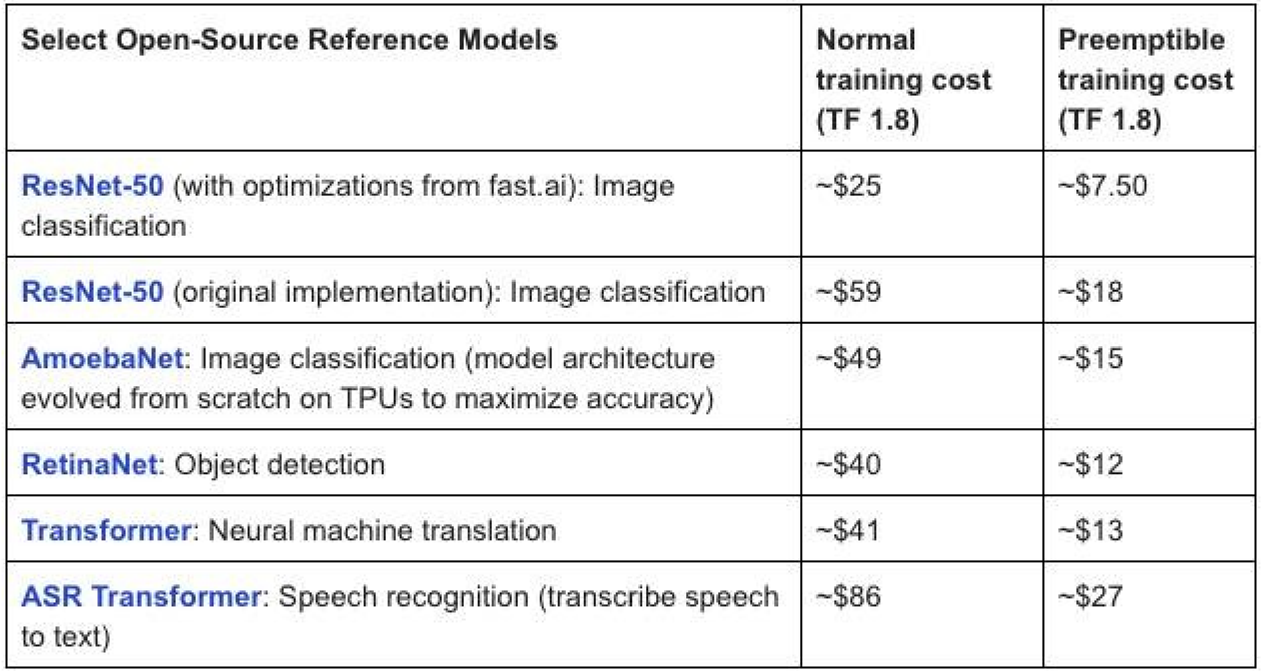
Training from scratch

- Train *ResNet* to recognize K custom objects classes
  - Long training time



Cost of training from scratch…

- AlexNet (2012) took 5~6 days over two GTX 580 GPUs
- Nowadays many startups *loan* training time

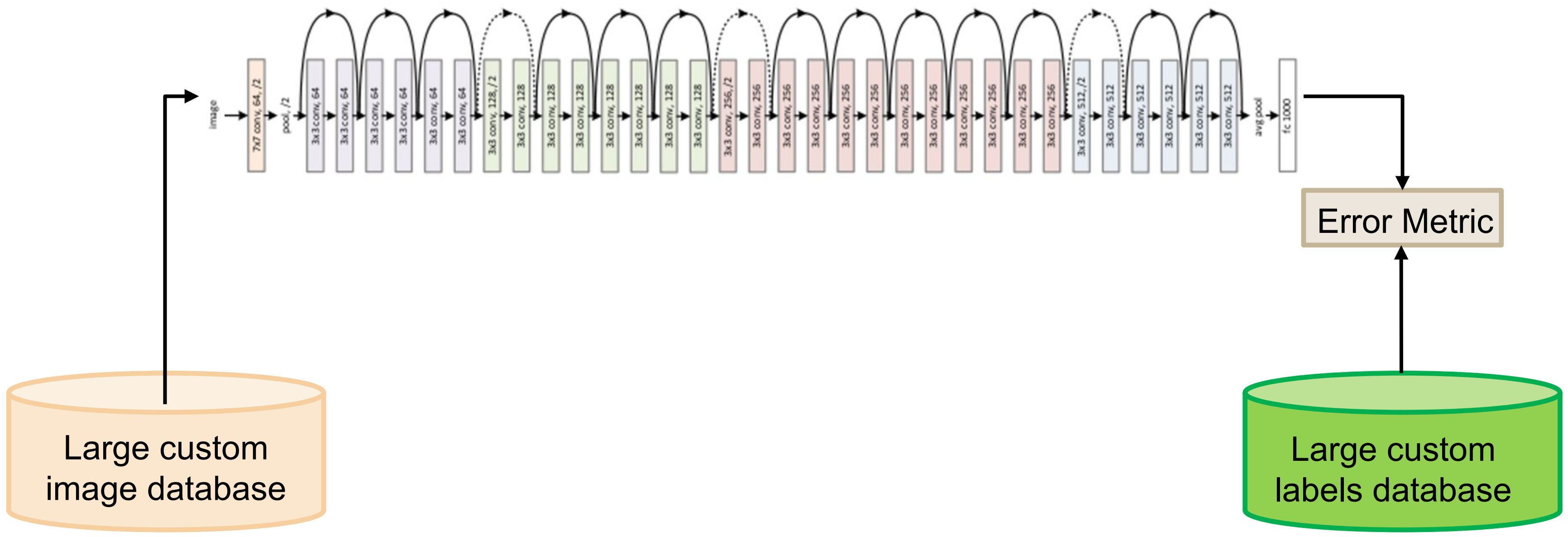


Select Open-Source Reference Models	Normal training cost (TF 1.8)	Preemptible training cost (TF 1.8)
ResNet-50 (with optimizations from fast.ai): Image classification	~$25	~$7.50
ResNet-50 (original implementation): Image classification	~$59	~$18
AmoebaNet: Image classification (model architecture evolved from scratch on TPUs to maximize accuracy)	~$49	~$15
RetinaNet: Object detection	~$40	~$12
Transformer: Neural machine translation	~$41	~$13
ASR Transformer: Speech recognition (transcribe speech to text)	~$86	~$27

Cost of training from scratch some deep convolutional networks over Google's TPU cloud (2017)
https://www.theregister.co.uk/2018/06/20/google_cloud_tpus/

Training from scratch

- Train *ResNet* to recognize K custom objects classes
  - Long training time
  - Must collect and label **many** train samples



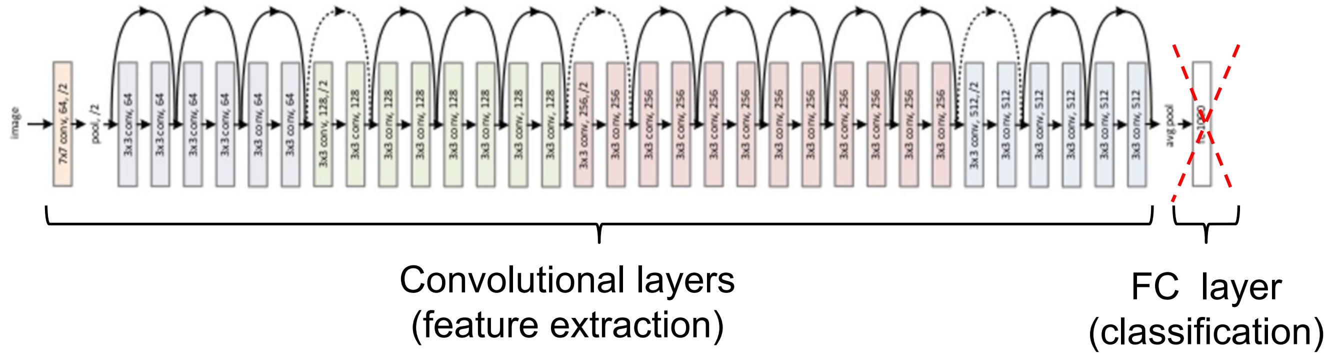
Transfer learning

- Transfer learning (TL) is **a research problem in machine learning (ML)** that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.

- In simpler words: you take a pre-trained model on a general large task and you use it as a base to train on your specific task!

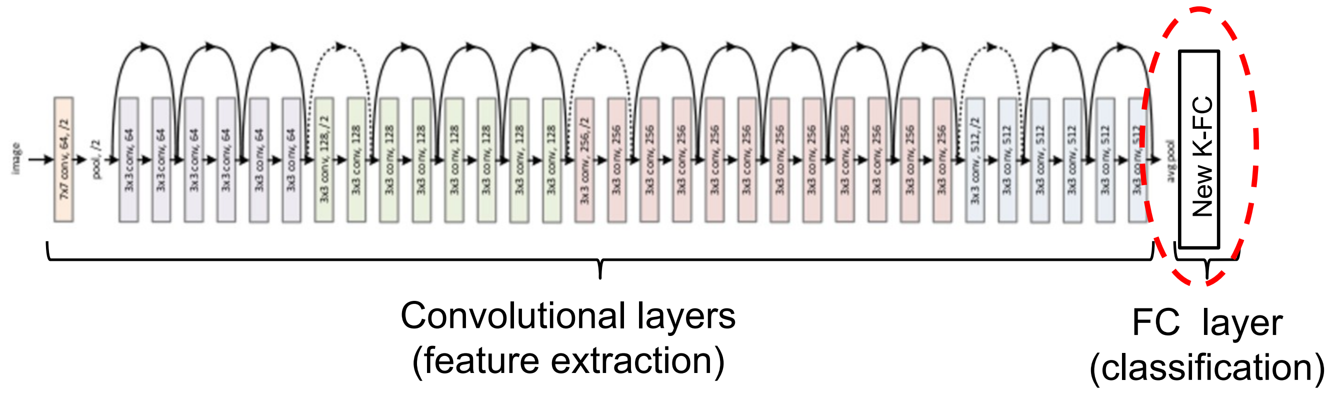
Transfer learning with ResNet models

- Take ResNet pretrained on *ImageNet*



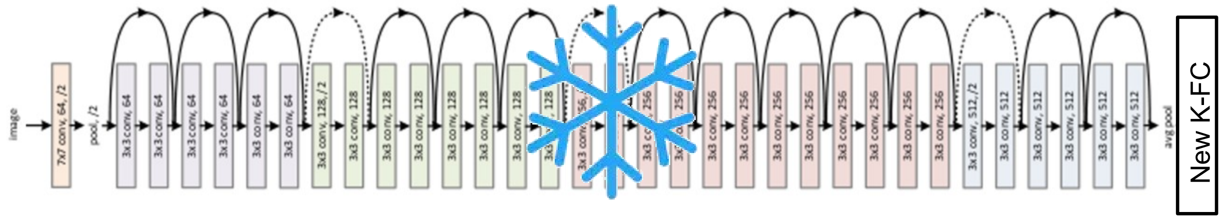
Transfer learning with ResNet models

- Take ResNet pretrained on *ImageNet*
- Replace FC layer(s) with ad-hoc K-units FC layer



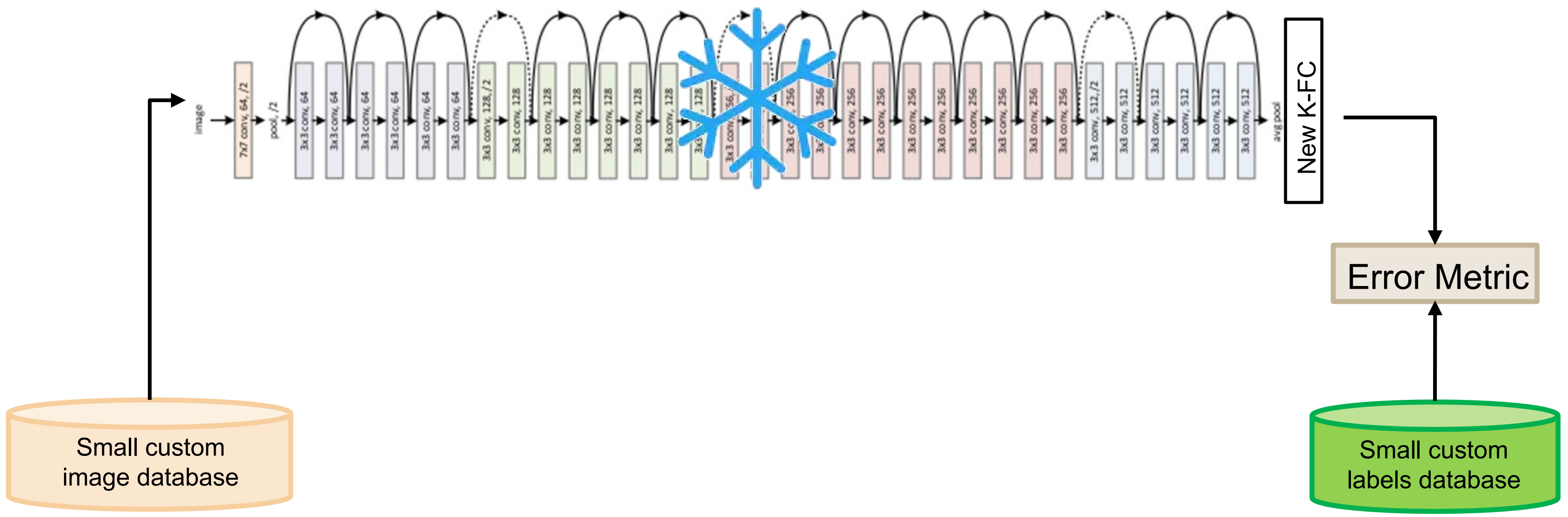
Transfer learning with ResNet models

- Take ResNet pretrained on *ImageNet*
- Replace FC layer(s) with ad-hoc K-units FC layer
- *Freeze* (early) convolutional layers (η=0 or close to)



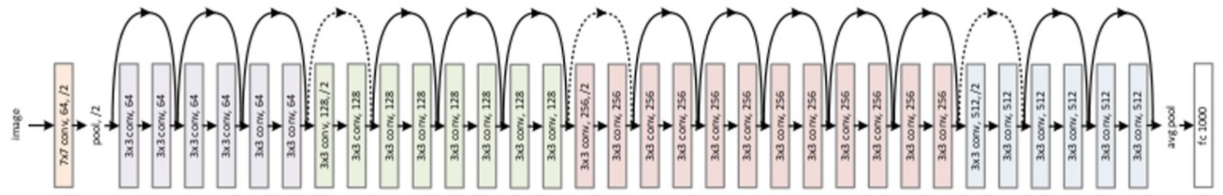
Transfer learning with ResNet models

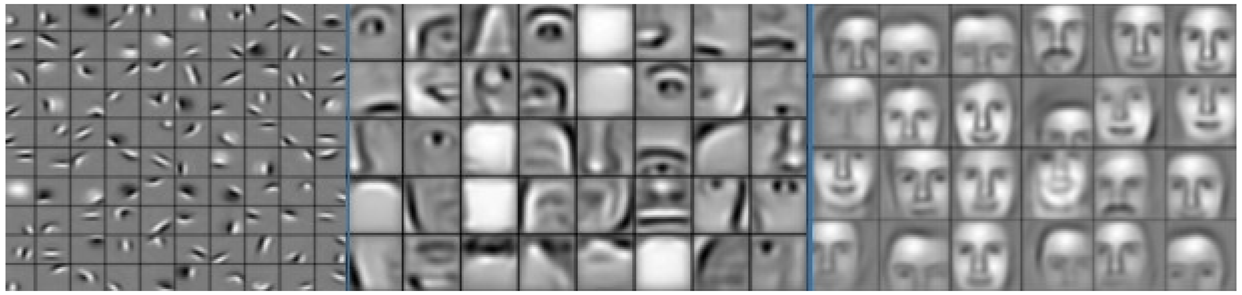
- Take ResNet pretrained on *ImageNet*
- Replace FC layer(s) with ad-hoc K-units FC layer
- *Freeze* (early) convolutional layers (η=0 or close to)
- *Refine* network over **small** custom dataset



Why does transfer learning work?

- Early conv. layers more difficult to trend (faint error gradients)
  - Very low level filters (edges, etc.)
    - «Reusing» pre-learned feature detectors





Watermarking and Deep Neural Networks

A parallel with watermarking for images

From Nagai et al., 2018

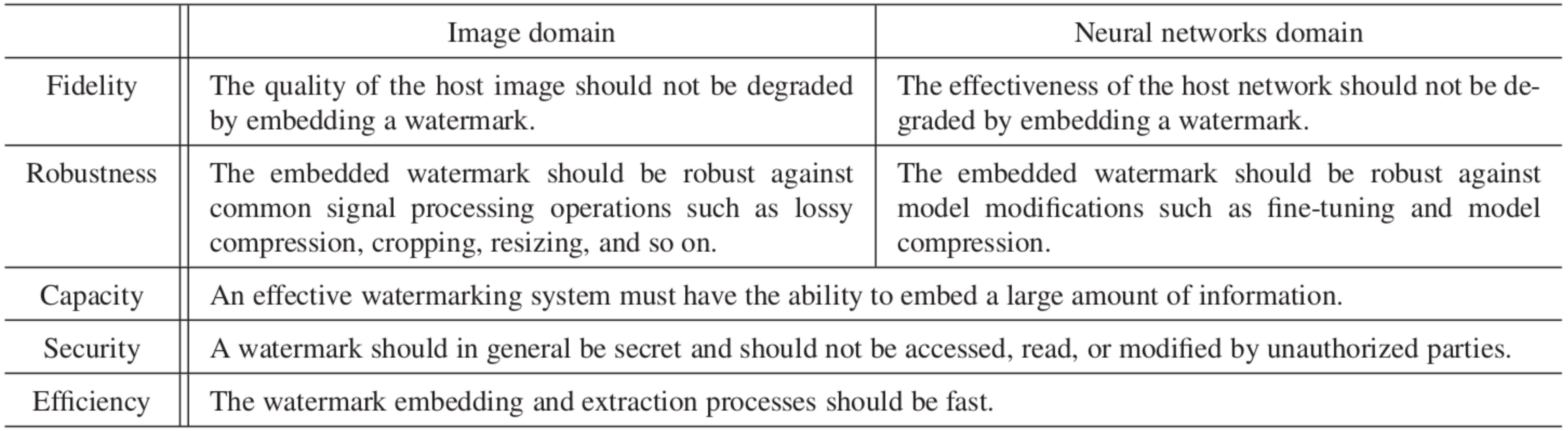


	Image domain	Neural networks domain
Fidelity	The quality of the host image should not be degraded by embedding a watermark.	The effectiveness of the host network should not be degraded by embedding a watermark.
Robustness	The embedded watermark should be robust against common signal processing operations such as lossy compression, cropping, resizing, and so on.	The embedded watermark should be robust against model modifications such as fine-tuning and model compression.
Capacity	An effective watermarking system must have the ability to embed a large amount of information.	
Security	A watermark should in general be secret and should not be accessed, read, or modified by unauthorized parties.	
Efficiency	The watermark embedding and extraction processes should be fast.	

Another taxonomy

- Watermarking tools guaranty the traceability and integrity of contents by finding the right balance between three principles:
  - imperceptibility,
  - Robustness,
  - data payload.

Imperceptibility

- Imperceptibility evaluate the impact on the content induced by the watermark, we want this impact to be minimal:
  "Prediction quality of the model on its original task should not be degraded significantly."

- Currently a common definition of Imperceptibility independent of the task and applicable on all field does not exist.

Robustness

- Robustness evaluates the resistance of the watermark against a set of attacks. In other word, if we can still detect the watermark after a modification of the content occurred. For neural network watermarking, since Uchida et al. [UCHI] the definition is

  “Watermark should be robust against removal attacks.”

- Another type of attacks borrowed from multimedia watermarking is the watermark overwriting and watermark forging, but they are not or partially explore yet.

Data payload

- Data payload is the quantity of inserted information under imperceptibility and robustness constraints.

- In neural network watermarking methods, it mostly considered as 0-bit watermarking, the watermark is detected or not, but papers and methods start deepen this field like Li et al. [LI21] who extend the work of Uchida to increase its capacity.

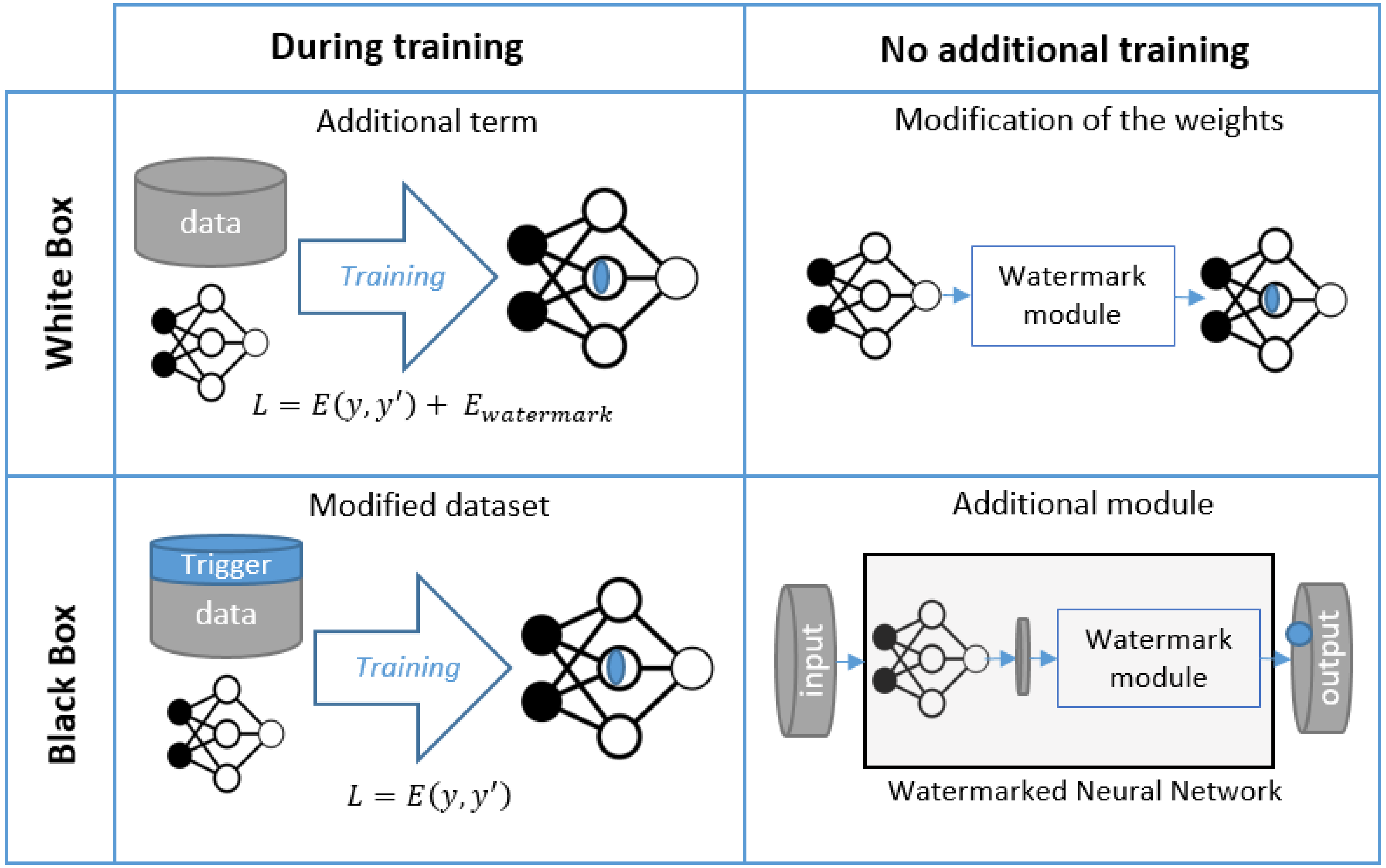
Watermarking VS Fingerprinting

- Fingerprint also deals with traceability with similar criteria of evaluation
- Imperceptibly is replaced by uniqueness: each content as is own fingerprint.
- For multimedia content, watermarking methods are considered “active”: we add something to the content, while fingerprinting is a “passive method”, which does not modify the content.
- In Neural Network this boundary is not easily define: most methods embed their watermark **during training**, thus we can see neural network watermarking techniques as methods that force the model to have a specific fingerprint.

Integrity

- A particular case of watermarking/fingerprinting appears when we have a very low robustness: the loss of integrity.
- We can use those methods to detect modification of a content (in our case, the parameters of the model, or the output of the model itself).
- One of the objectives here could be to detect any modification of the inference.

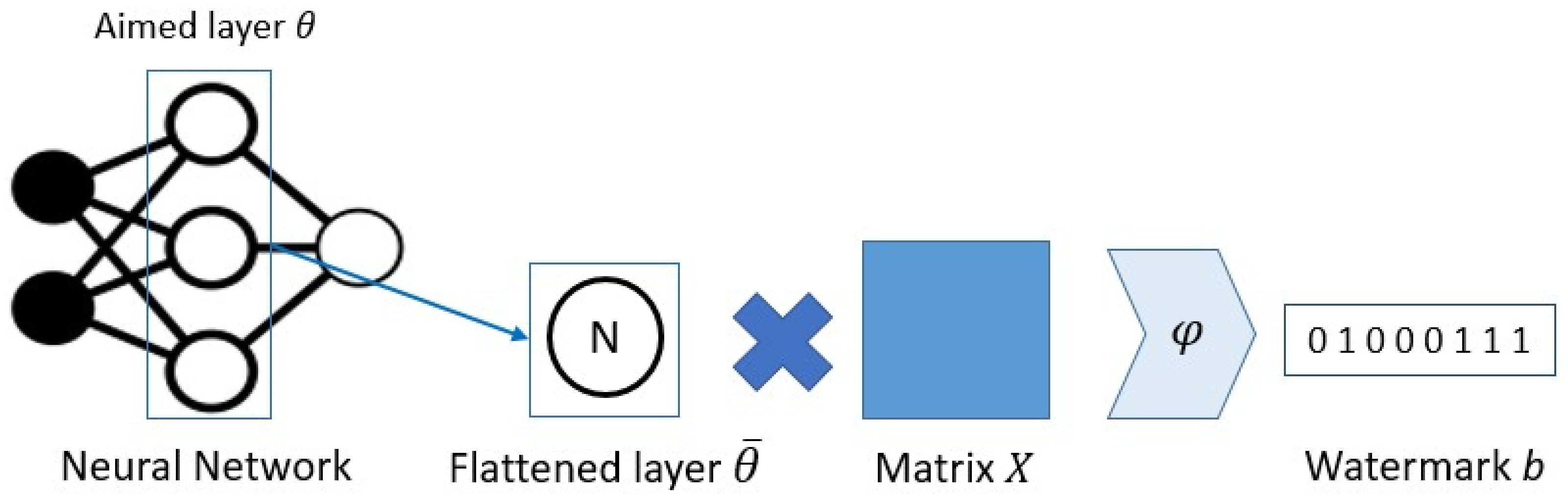
Black-box VS white-box methods



White-box watermarking

Learning an extraction matrix (Uchida et al.)

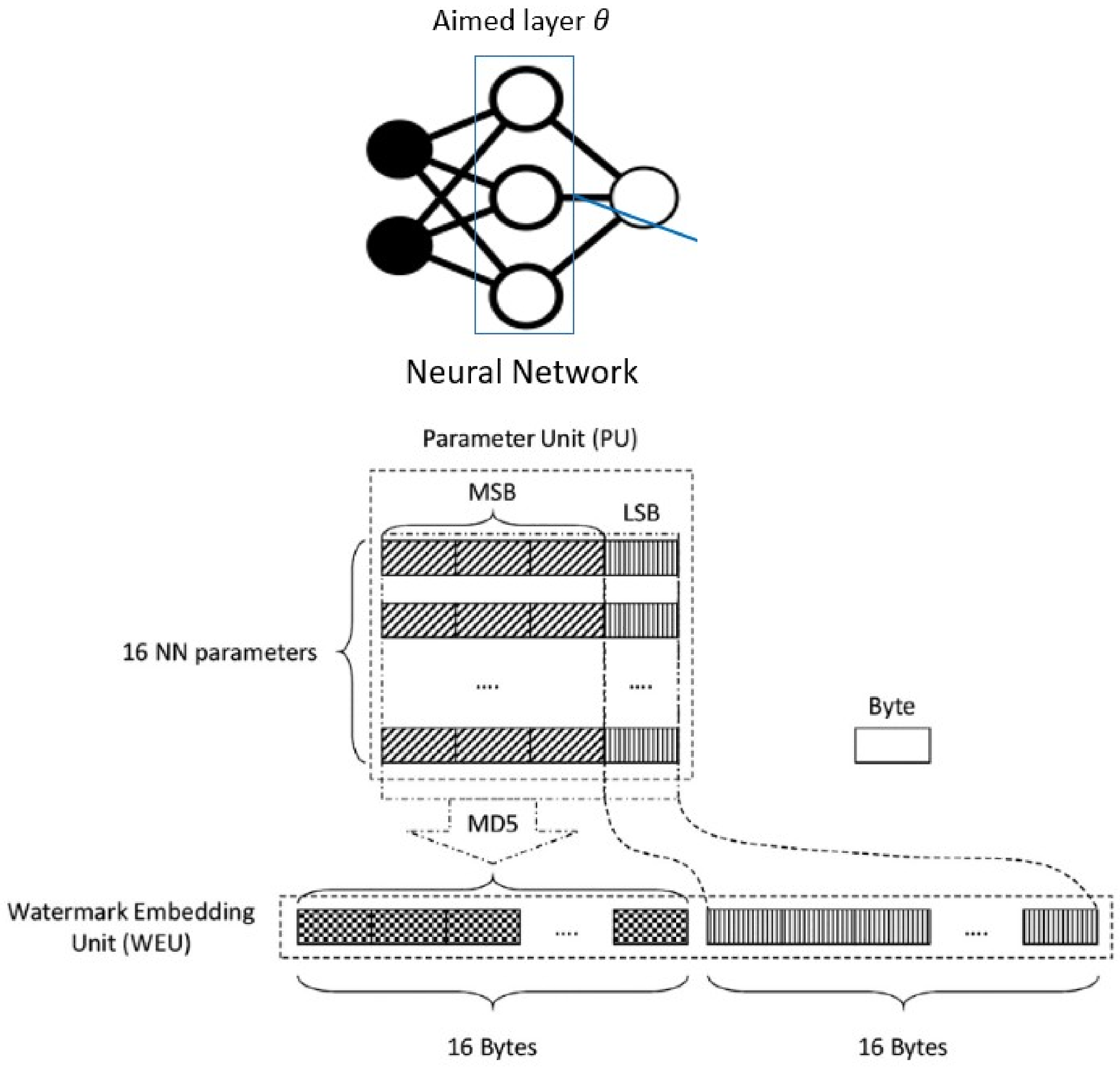
- Let us choose one layer in the deep model
- We learn a transformation matrix X such that the parameters are projected in a sub-space, which is our watermark



Check fragile watermarks (Botta et al.)

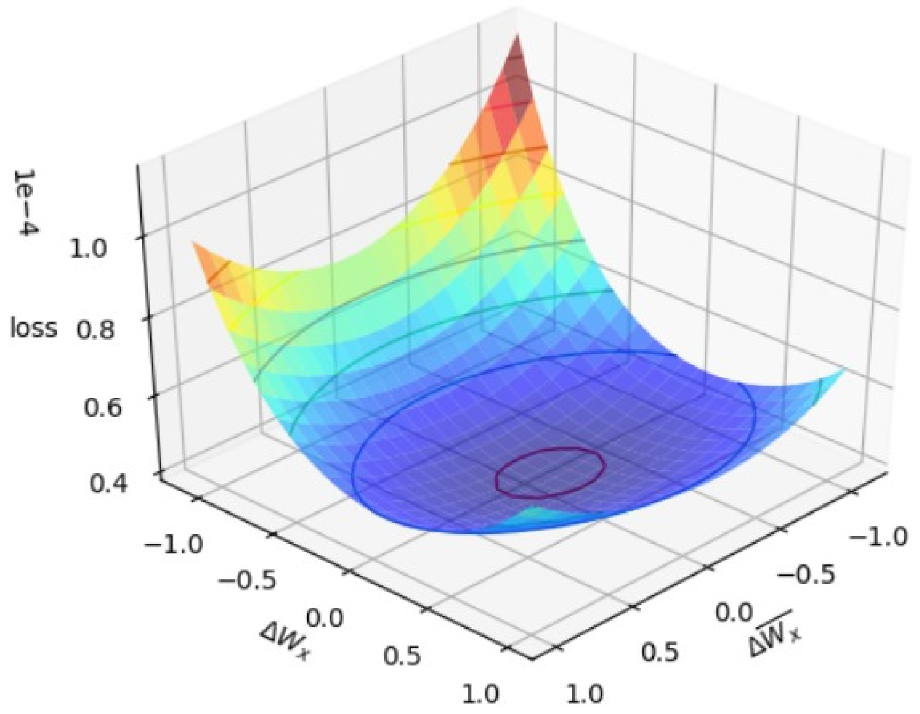
- IDEA:implicitly embed the watermark inside the parameters.
- LSB: is directly part of the watermark
- MSB: it is used MD5 hash generator to get a WEU.

- Advantage: very fast
- Disadvantage: just on fragile watermarks!

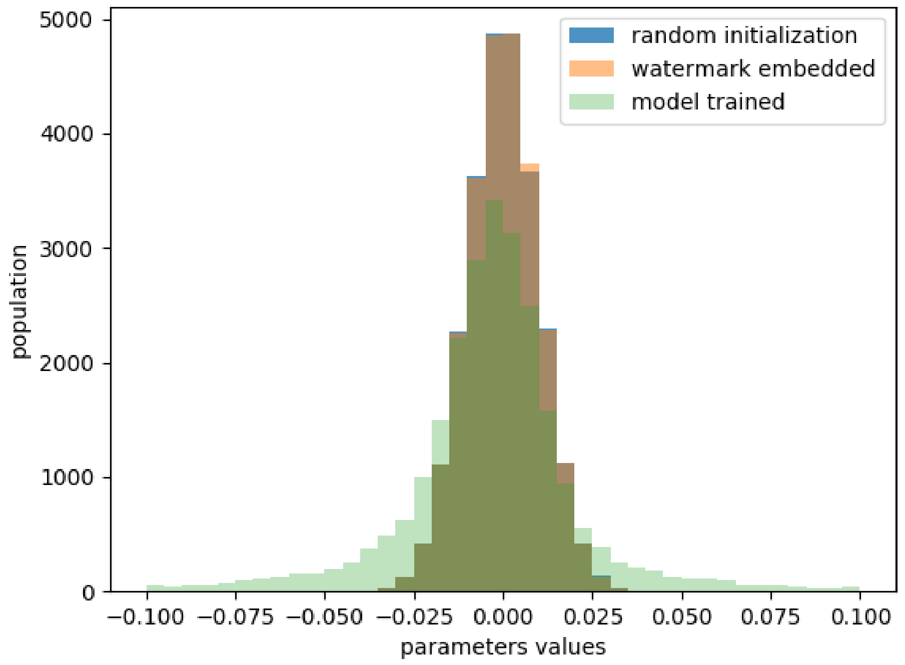


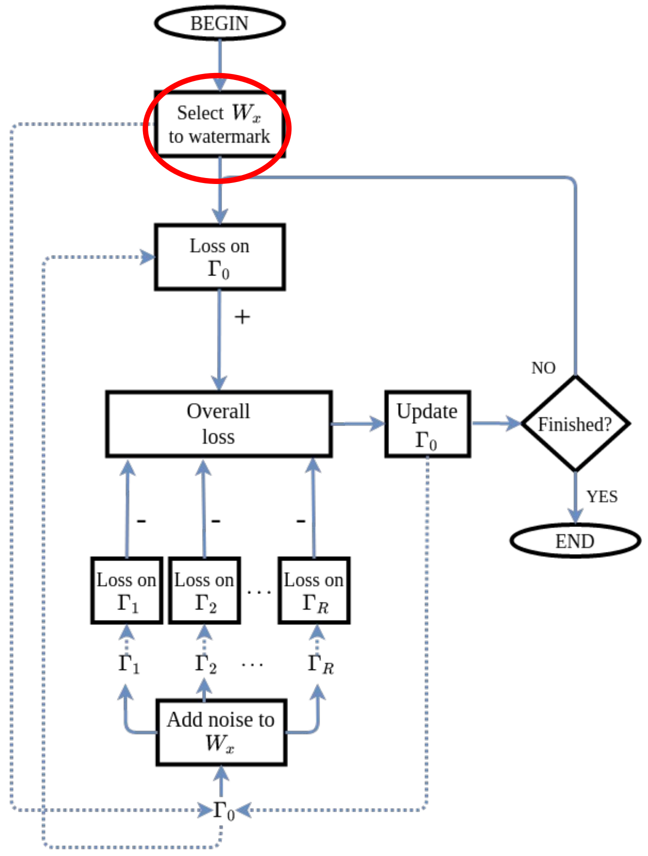
Find a special local minimum (Tartaglione et al.)

- Idea: make the watermark robust to any modification
  - In other words, when we modify the watermark, the error (loss) increases
- We select randomly parameters all along the model (any location)
- These parameters will constitute our watermark
- We want to find a solution such that, when modifying our watermark, the loss hoes high (narrow minimum) while, when modifying the non-watermarked parameters, the loss can remain low (wide minimum).

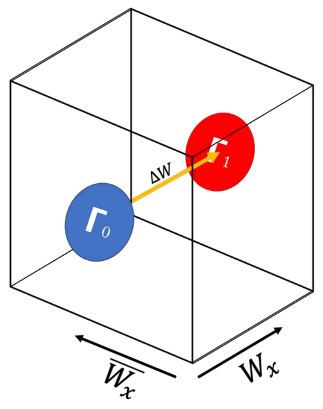


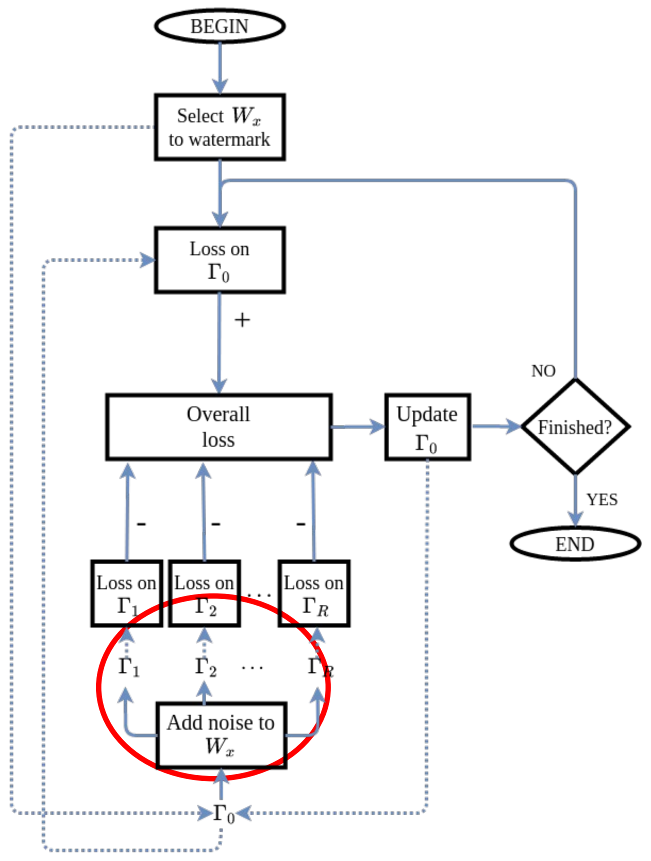
Find a special local minimum (Tartaglione et al.)



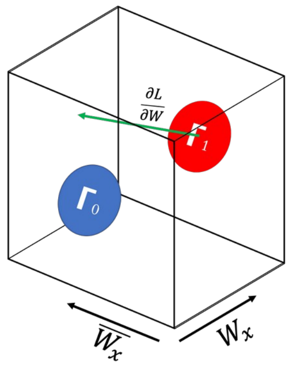


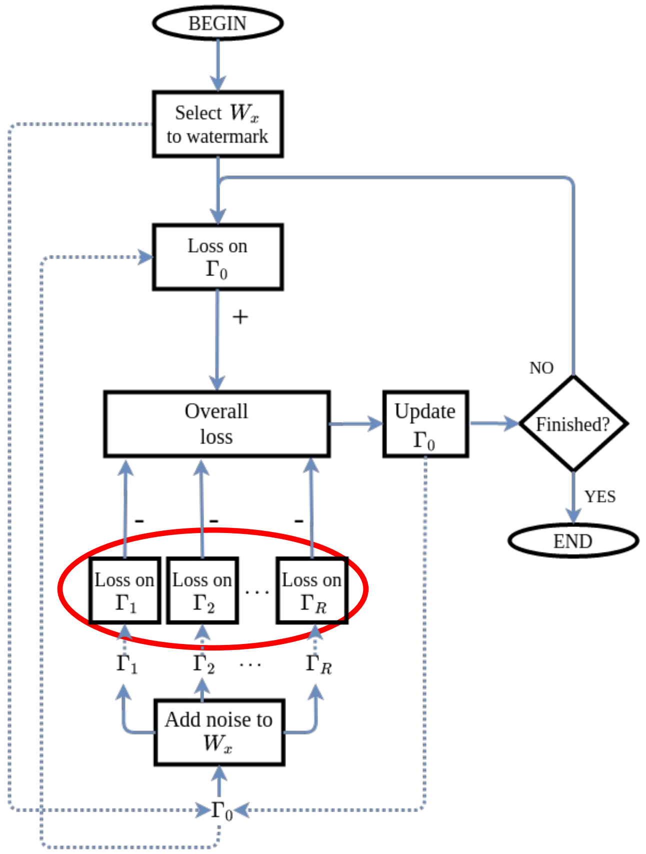
Find a special local minimum (Tartaglione et al.)



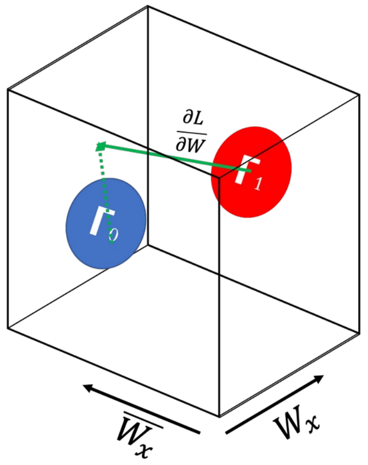


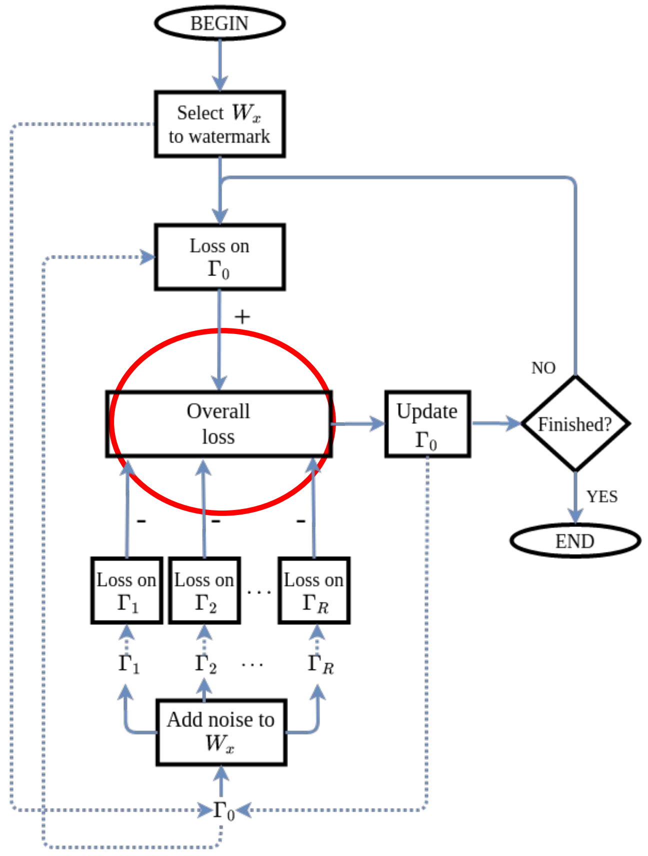
Find a special local minimum (Tartaglione et al.)



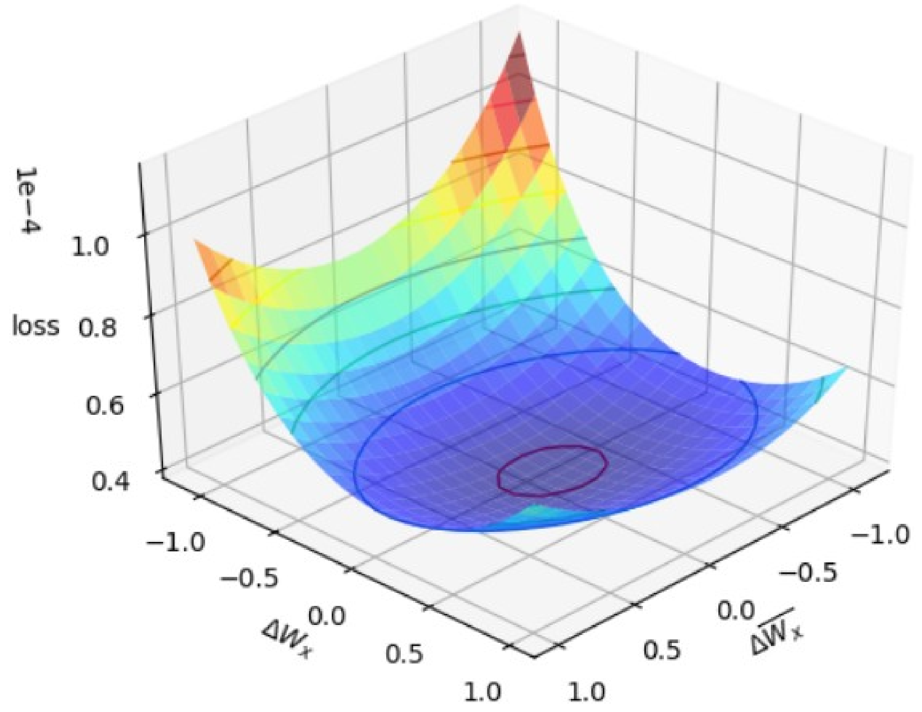


Find a special local minimum (Tartaglione et al.)



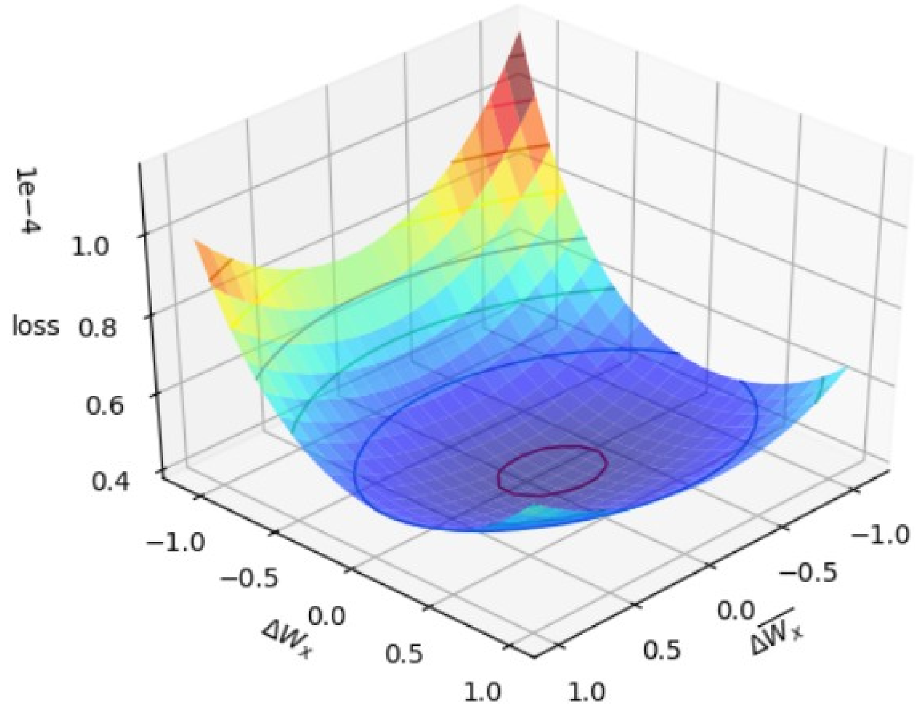


Drawbacks (Tartaglione et al.)



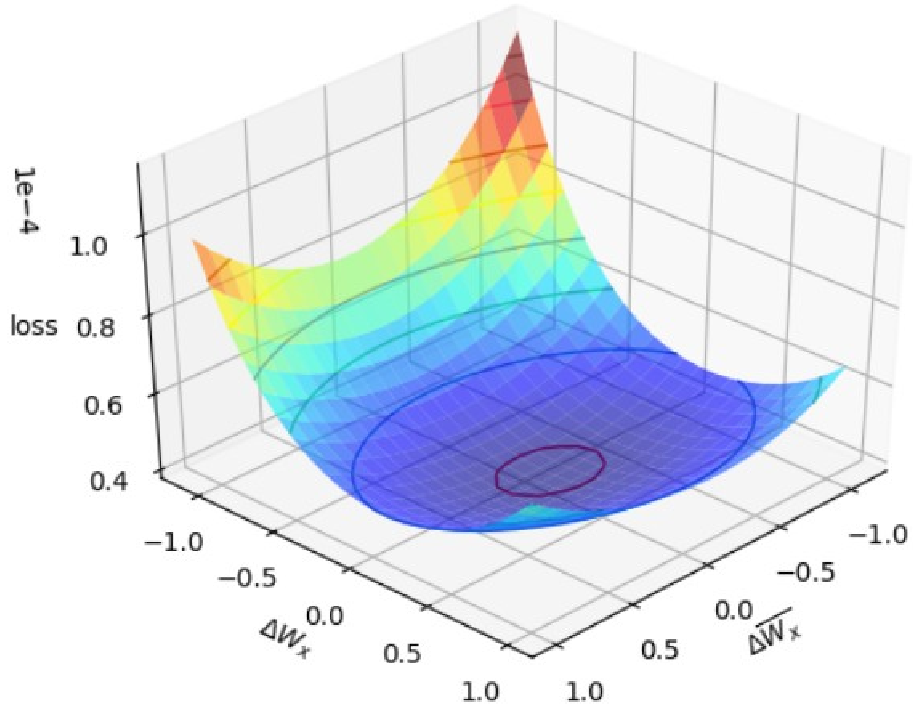
Drawbacks (Tartaglione et al.)

- Computational complexity
  - Every iteration, a new pool of R architectures must be instantiated, gradient calculated, and projected back.



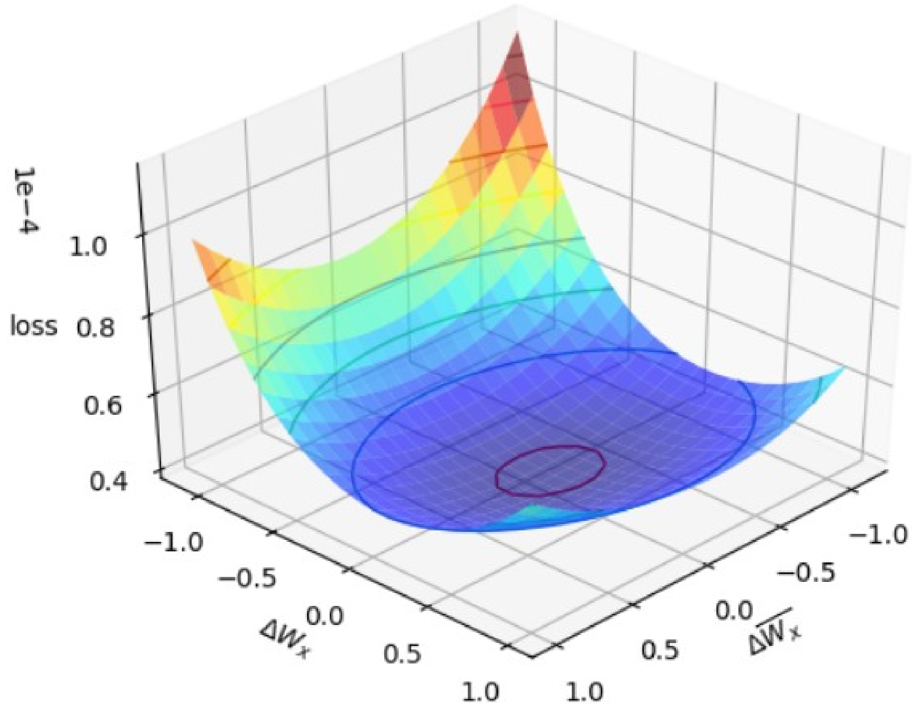
Drawbacks (Tartaglione et al.)

- Computational complexity
  - Every iteration, a new pool of R architectures must be instantiated, gradient calculated, and projected back.
- Memory complexity
  - If P is the number of parameters on your models, then the peak memory will be (R+1)*P → seems nothing, but imagine a model already occupying 10 GB, multiplied by R=8…



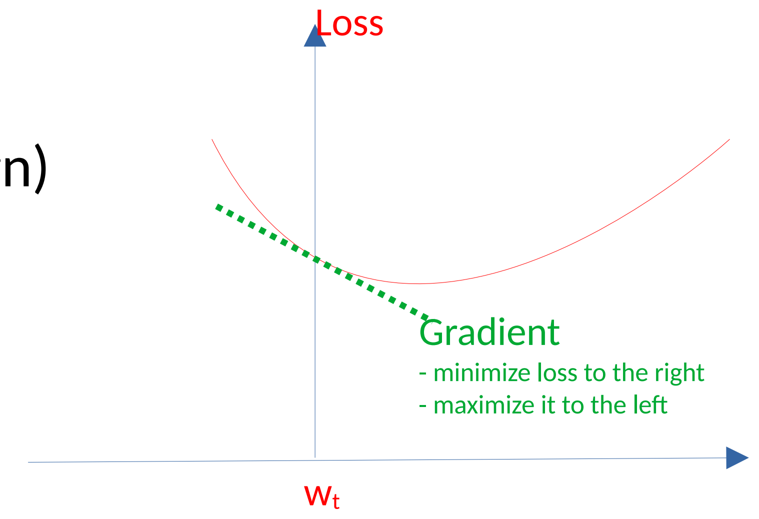
Drawbacks (Tartaglione et al.)

- Computational complexity
  - Every iteration, a new pool of R architectures must be instantiated, gradient calculated, and projected back.
- Memory complexity
  - If P is the number of parameters on your models, then the peak memory will be (R+1)*P → seems nothing, but imagine a model already occupying 10 GB, multiplied by R=8…
- Sampling
  - How large should R be…?



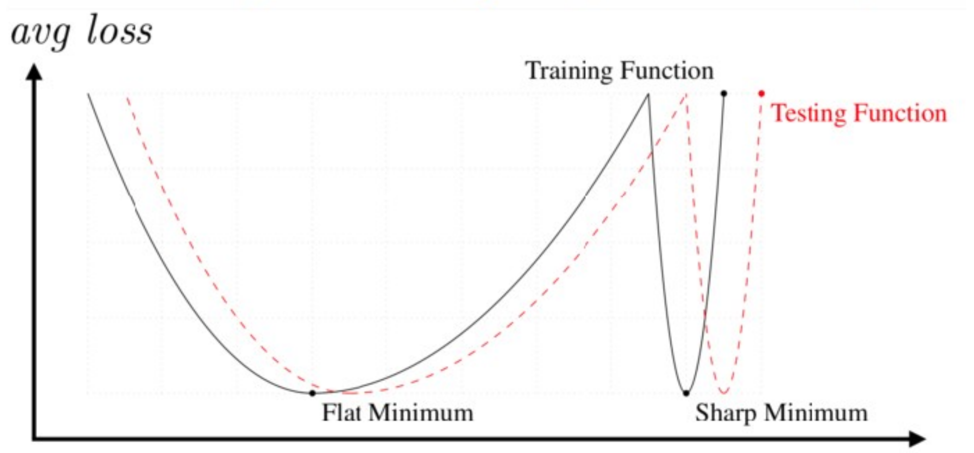
[Preliminary] Sharpness-aware Minimization (Foret et al.)

- **Assumption 1**: the loss function is locally linearizable
  - Once we compute the gradient, we also know the direction to maximize the loss (remember the "-" sign)



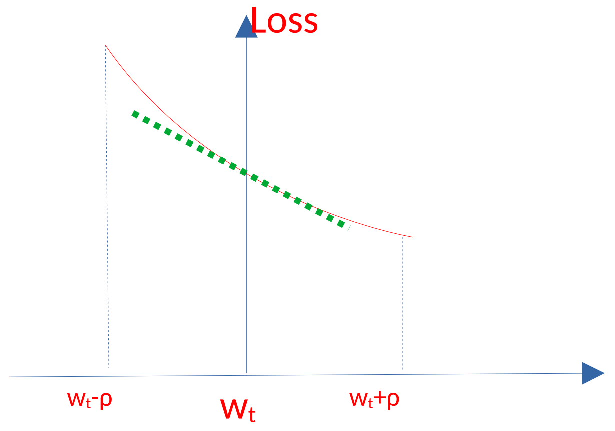
[Preliminary] Sharpness-aware Minimization (Foret et al.)

- **Assumption 1**: the loss function is locally linearizable
  - Once we compute the gradient, we also know the direction to maximize the loss (remember the “-” sign)
- **Assumption 2**: wide minima generalize well
  - Remember what seen in MN909…



[Preliminary] Sharpness-aware Minimization (Foret et al.)

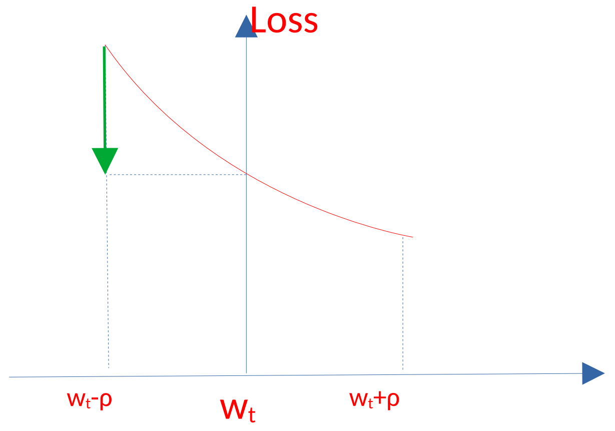
- **Assumption 1**: the loss function is locally linearizable
  - Once we compute the gradient, we also know the direction to maximize the loss (remember the “-” sign)
- **Assumption 2**: wide minima generalize well
  - Remember what seen in MN909…



- **Idea**: within an hypersphere of radius ρ, we minimize the maximum loss!
  - This means that within radius ρ the loss will be maximally flat, helping in the generalization.

[Preliminary] Sharpness-aware Minimization (Foret et al.)

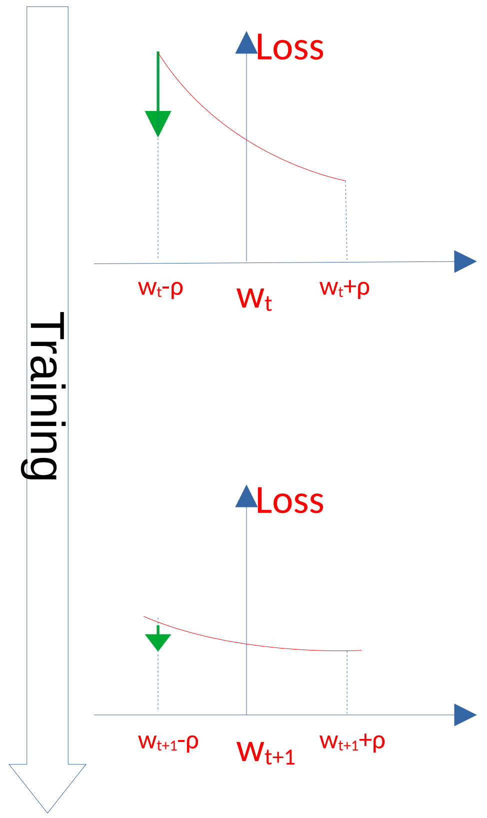
- **Assumption 1**: the loss function is locally linearizable
  - Once we compute the gradient, we also know the direction to maximize the loss (remember the "-" sign)
- **Assumption 2**: wide minima generalize well
  - Remember what seen in MN909…



- **Idea**: within an hypersphere of radius ρ, we minimize the maximum loss!
  - This means that within radius ρ the loss will be maximally flat, helping in the generalization.

[Preliminary] Sharpness-aware Minimization (Foret et al.)

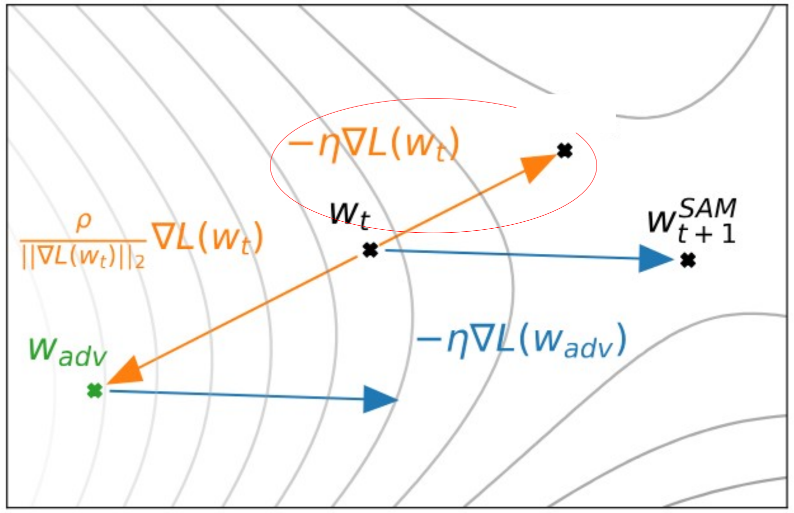
- **Assumption 1**: the loss function is locally linearizable
  - Once we compute the gradient, we also know the direction to maximize the loss (remember the “-” sign)
- **Assumption 2**: wide minima generalize well
  - Remember what seen in MN909…
- **Idea**: within an hypersphere of radius ρ, we minimize the maximum loss!
  - This means that within radius ρ the loss will be maximally flat, helping in the generalization.



[Preliminary] Sharpness-aware Minimization (Foret et al.)

HOW?

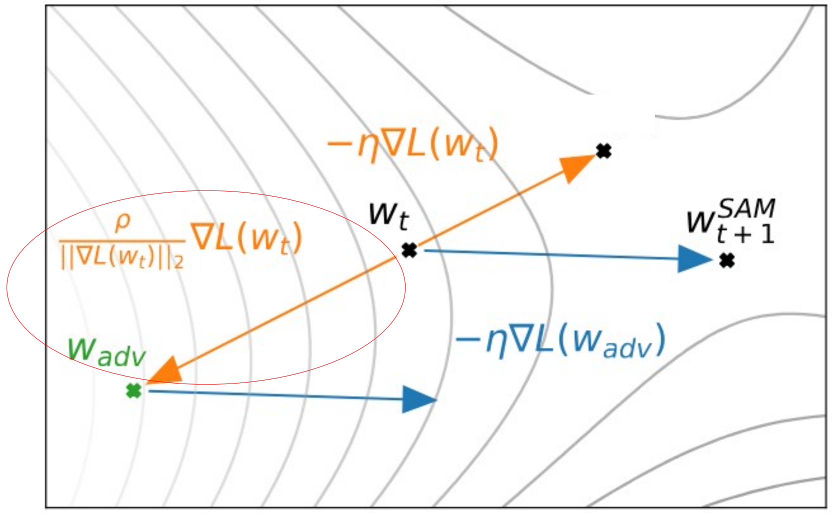
1) **Calculate gradient for w_t**



[Preliminary] Sharpness-aware Minimization (Foret et al.)

HOW?

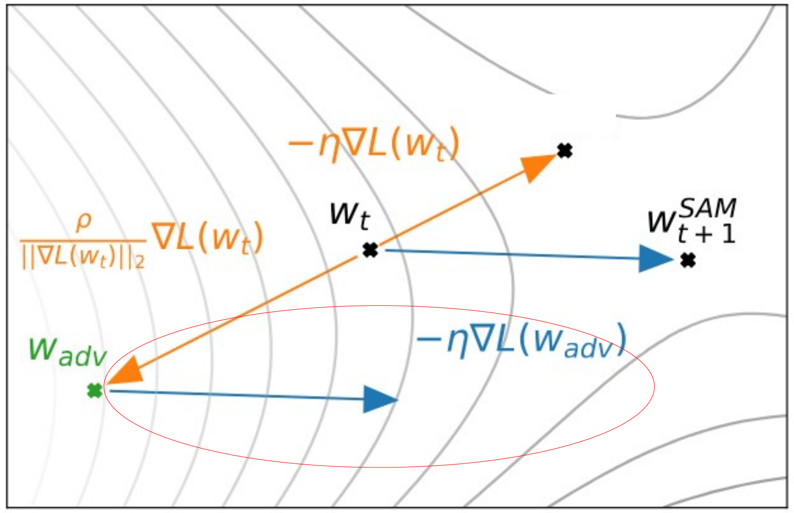
1) **Calculate gradient for w_t**
2) **Move to the maximizing direction to get w_{adv}**



[Preliminary] Sharpness-aware Minimization (Foret et al.)

HOW?

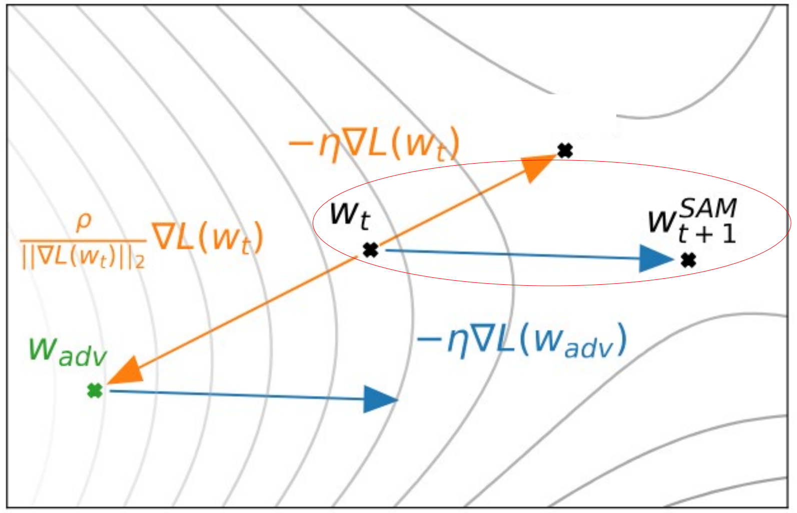
1) **Calculate gradient for w_t**
2) **Move to the maximizing direction to get w_{adv}**
3) **Calculate the gradient in w_{adv}**



[Preliminary] Sharpness-aware Minimization (Foret et al.)

HOW?

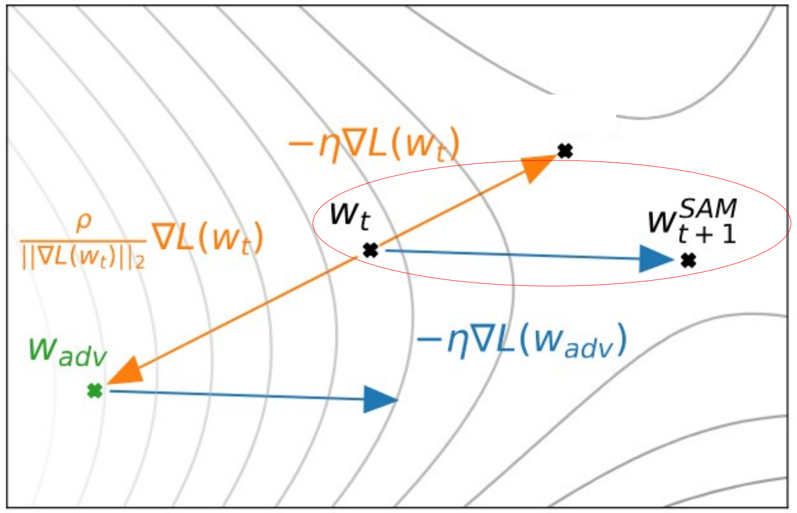
1) **Calculate gradient for w_t**
2) **Move to the maximizing direction to get w_{adv}**
3) **Calculate the gradient in w_{adv}**
4) **Apply the gradient calculated in w_{adv} to w_t and repeat.**



[Preliminary] Sharpness-aware Minimization (Foret et al.)

Questions:

1) We are applying the gradient of w_{adv} to w_{t} – arewe still minimizing the loss in w_t?



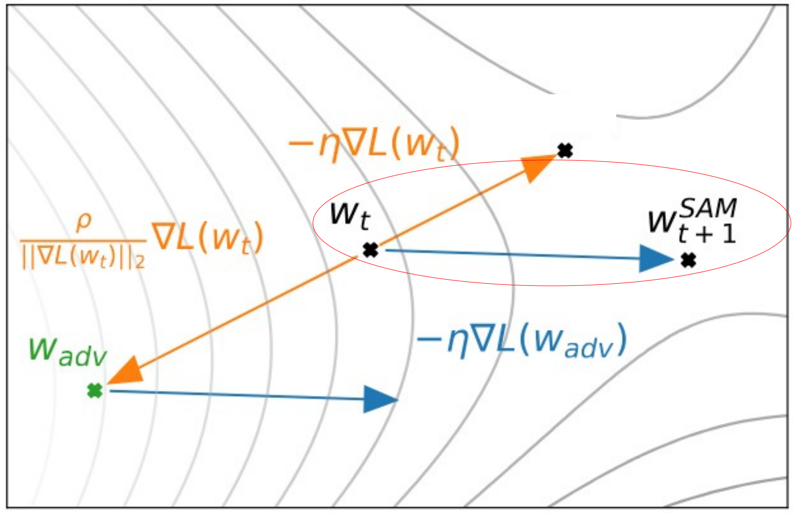
[Preliminary] Sharpness-aware Minimization (Foret et al.)

Questions:

1) We are applying the gradient of w_{adv} to w_t – arewe still minimizing the loss in w_t?

   Yes, for low values of ρ.

2) What is the computational complexity of SAM?



[Preliminary] Sharpness-aware Minimization (Foret et al.)

Questions:

1) We are applying the gradient of w_{adv} to w_t – arewe still minimizing the loss in w_t?

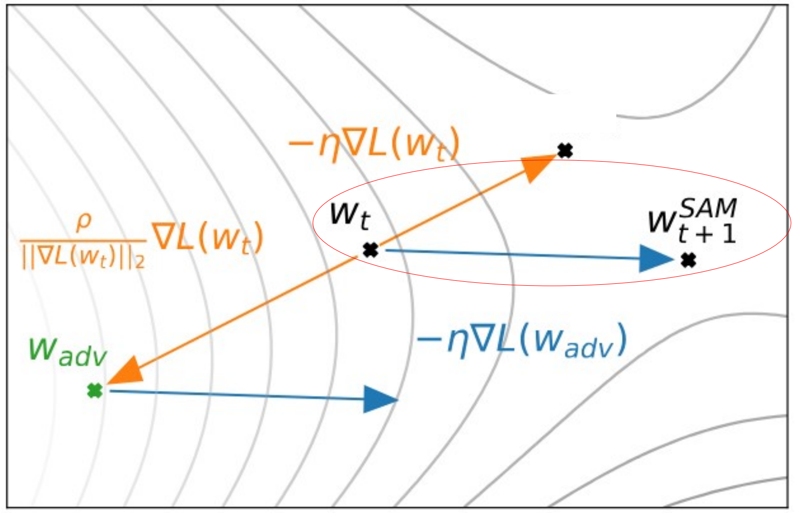
   Yes, for low values of ρ.

2) What is the computational complexity of SAM?

   Roughly twice a regular training (we need two backpropagations).

3) What is the natural application for SAM?

   Federated or continual learning, where you want to change the parameters while ensuring robustness… what about watermarking?



WaterMAS (De Sousa Trias et al.)

- We can use the same idea as SAM, but applied differently.
- As in the previous approach (Tartaglione et al.), we have some watermarked parameters where we want sharpness (opposite objective as SAM). How can we do that?

WaterMAS (De Sousa Trias et al.)

- We can use the same idea as SAM, but applied differently.
- As in the previous approach (Tartaglione et al.), we have some watermarked parameters where we want sharpness (opposite objective as SAM). How can we do that?
  - We can go to the loss MINIMUM (within the watermarked parameter’s space) and MAXIMIZE the loss at distance ρ.
- What about the unwatermarked parameters?

WaterMAS (De Sousa Trias et al.)

- We can use the same idea as SAM, but applied differently.
- As in the previous approach (Tartaglione et al.), we have some watermarked parameters where we want sharpness (opposite objective as SAM). How can we do that?
  - We can go to the loss MINIMUM (within the watermarked parameter's space) and MAXIMIZE the loss at distance ρ.
- What about the unwatermarked parameters?
  - They will follow regular gradient descent.

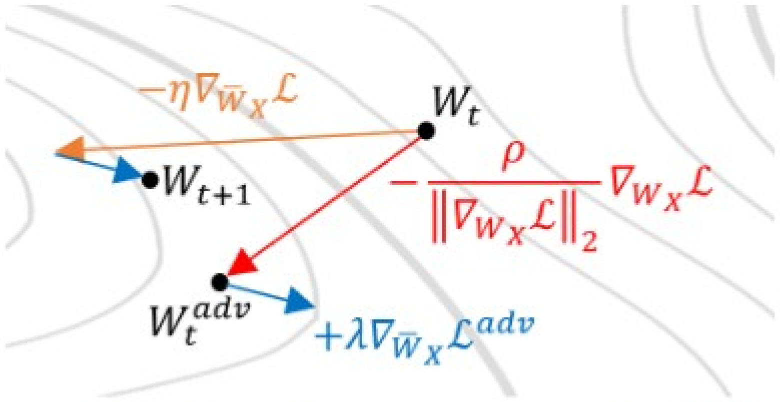
WaterMAS (De Sousa Trias et al.)

- We can use the same idea as SAM, but applied differently.
- As in the previous approach (Tartaglione et al.), we have some watermarked parameters where we want sharpness (opposite objective as SAM). How can we do that?
  - We can go to the loss MINIMUM (within the watermarked parameter's space W_X) and MAXIMIZE the loss at distance ρ.
- What about the unwatermarked parameters?
  - They will follow regular gradient descent.

WaterMAS (De Sousa Trias et al.)

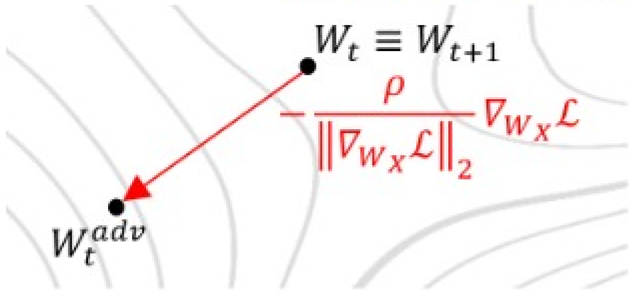
HOW?

1) **Calculate gradient for w_t in W_X and find the minimum at distance** ρ named W^{adv}.



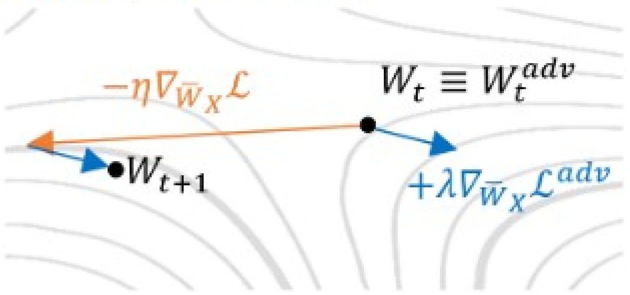
W_t
$-\eta \nabla_{\overline{W}_X} \mathcal{L}$
W_{t+1}
$-\frac{\rho}{\|\nabla_{W_X}\mathcal{L}\|_2} \nabla_{W_X}\mathcal{L}$
W_t^{adv}
$+\lambda \nabla_{\overline{W}_X} \mathcal{L}^{adv}$

(a) Schematic of the parameter update in MAS.



$W_t \equiv W_{t+1}$
$-\frac{\rho}{\|\nabla_{W_X}\mathcal{L}\|_2} \nabla_{W_X}\mathcal{L}$
W_t^{adv}

(b) Projection in the space W_X.



$-\eta \nabla_{\overline{W}_X} \mathcal{L}$
$W_t \equiv W_t^{adv}$
W_{t+1}
$+\lambda \nabla_{\overline{W}_X} \mathcal{L}^{adv}$

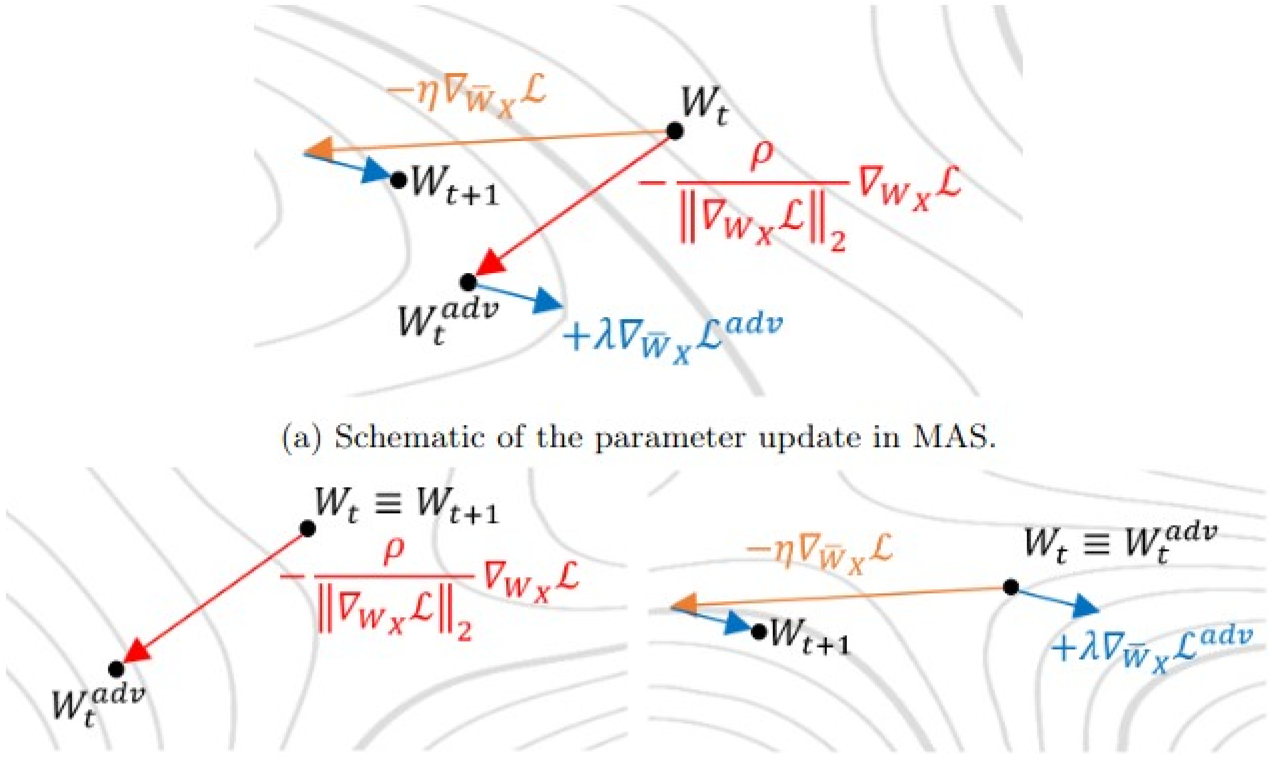
(c) Projection in the space $\overline{W}_X$.

WaterMAS (De Sousa Trias et al.)

HOW?

1) **Calculate gradient for w_t in W_X and find the minimum at distance** ρ named W^{adv}.

2) **Calculate the gradient in** W^{adv} and maximize the loss [make loss landscape sharp for W_X]


$-\eta\nabla_{\overline{W}_X}\mathcal{L}$
W_t
W_{t+1}
$-\frac{\rho}{\|\nabla_{W_X}\mathcal{L}\|_2}\nabla_{W_X}\mathcal{L}$
W_t^{adv}
$+\lambda\nabla_{\overline{W}_X}\mathcal{L}^{adv}$
$W_t \equiv W_{t+1}$
$W_t \equiv W_t^{adv}$

(a) Schematic of the parameter update in MAS.

(b) Projection in the space W_X.

(c) Projection in the space $\overline{W}_X$.

WaterMAS (De Sousa Trias et al.)

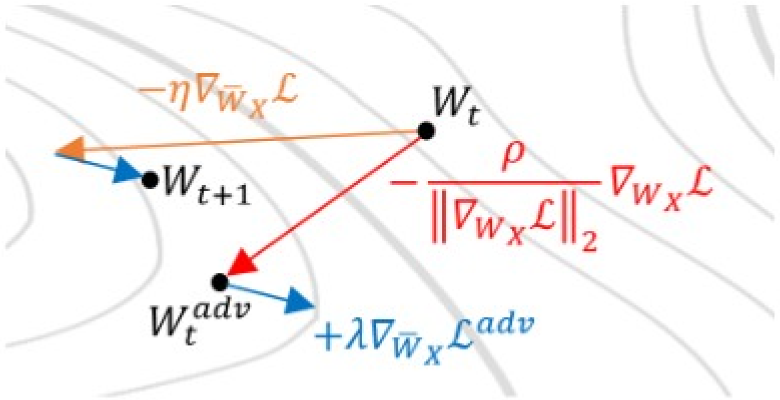
HOW?

1) **Calculate gradient for w_t in W_X and find the minimum at distance** ρ named W^{adv}.

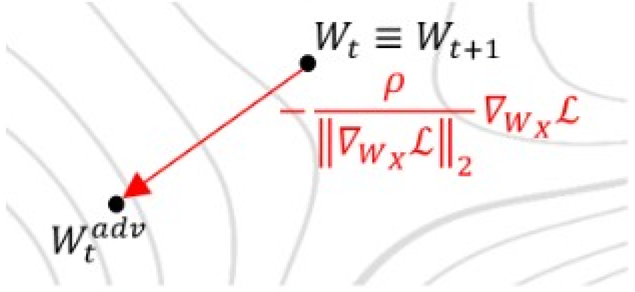
2) **Calculate the gradient in** W^{adv} and maximize the loss [make loss landscape sharp for W_X]

3) Calculate the loss to minimize for the unwatermarked parameters

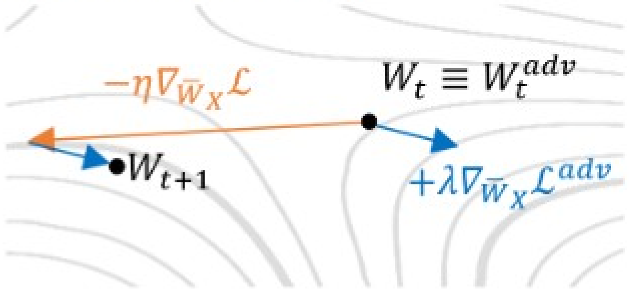
4) Update using both watermarked parameters sharpness enhancing and learning one.


$-\eta\nabla_{\overline{W}_X}\mathcal{L}$
W_t
W_{t+1}
$-\frac{\rho}{\|\nabla_{W_X}\mathcal{L}\|_2}\nabla_{W_X}\mathcal{L}$
W_t^{adv}
$+\lambda\nabla_{\overline{W}_X}\mathcal{L}^{adv}$

(a) Schematic of the parameter update in MAS.


$W_t \equiv W_{t+1}$
$-\frac{\rho}{\|\nabla_{W_X}\mathcal{L}\|_2}\nabla_{W_X}\mathcal{L}$
W_t^{adv}

(b) Projection in the space W_X.


$-\eta\nabla_{\overline{W}_X}\mathcal{L}$
$W_t \equiv W_t^{adv}$
W_{t+1}
$+\lambda\nabla_{\overline{W}_X}\mathcal{L}^{adv}$

(c) Projection in the space $\overline{W}_X$.

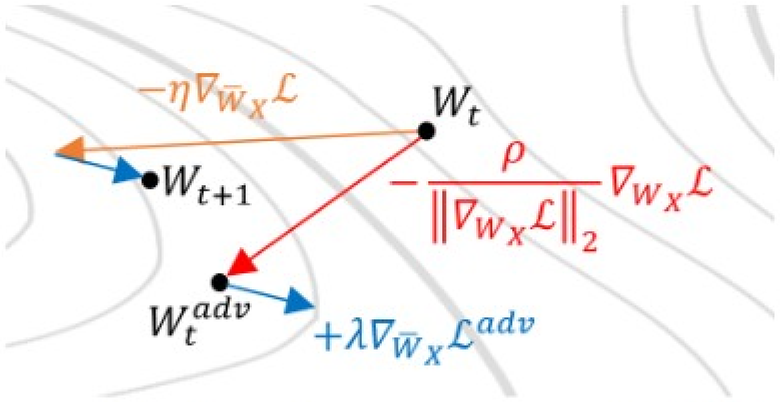
WaterMAS (De Sousa Trias et al.)

1) Why do we have only the rho projection in W_X?

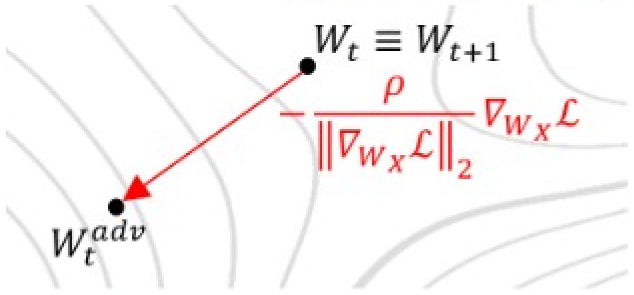
   We are not allowed to change watermarked parameters. All the gradients that we calculate are applied on unwatermarked parameters (that are allowed to change).

2) What is the computational complexity of WaterMAS?

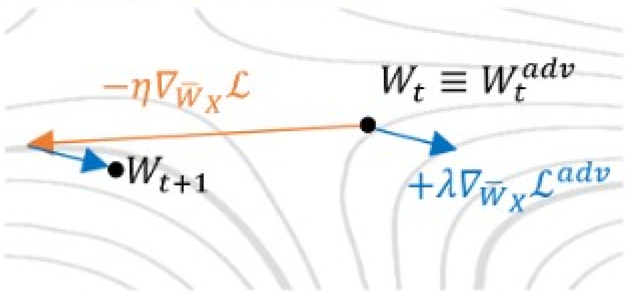
   Similar to SAM (2x backprops), making it more efficient than Tartaglione et al, and solving the sampling space issue.


$-\eta \nabla_{\overline{W}_X} \mathcal{L}$
W_t
W_{t+1}
$-\frac{\rho}{\|\nabla_{W_X}\mathcal{L}\|_2} \nabla_{W_X}\mathcal{L}$
W_t^{adv}
$+\lambda \nabla_{\overline{W}_X} \mathcal{L}^{adv}$

(a) Schematic of the parameter update in MAS.


$W_t \equiv W_{t+1}$
$-\frac{\rho}{\|\nabla_{W_X}\mathcal{L}\|_2} \nabla_{W_X}\mathcal{L}$
W_t^{adv}

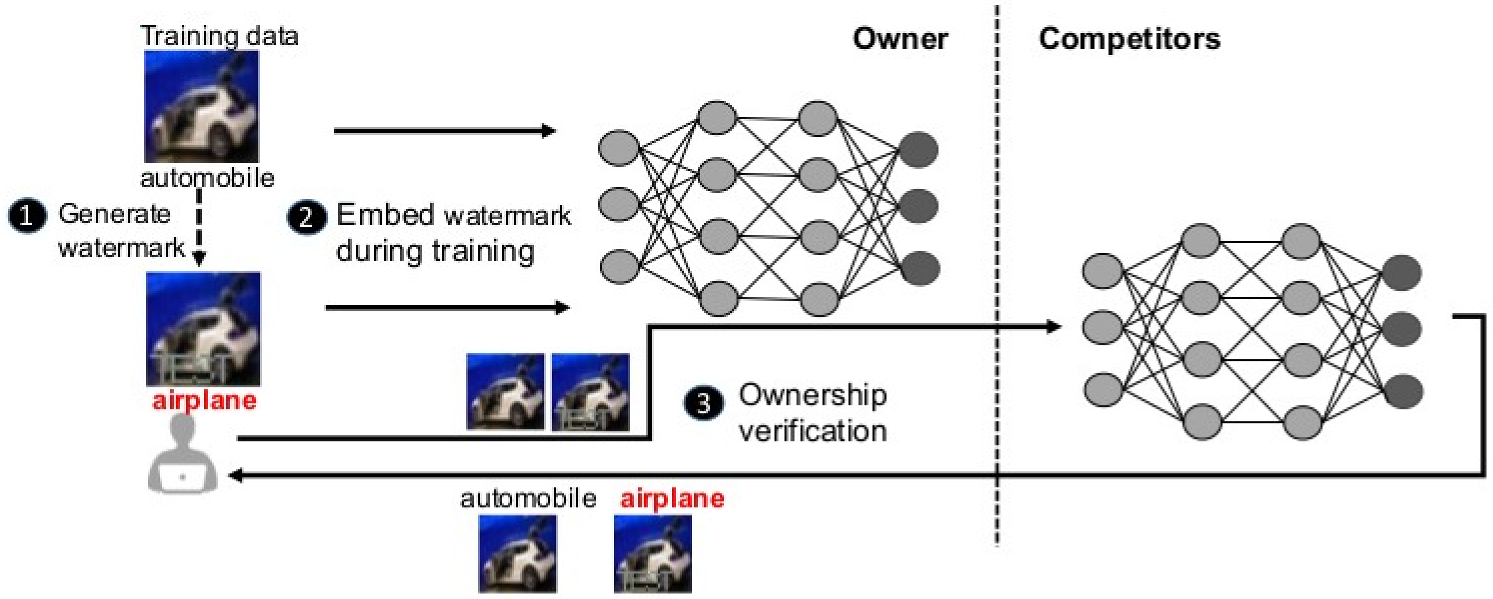
(b) Projection in the space W_X.


$-\eta \nabla_{\overline{W}_X} \mathcal{L}$
$W_t \equiv W_t^{adv}$
W_{t+1}
$+\lambda \nabla_{\overline{W}_X} \mathcal{L}^{adv}$

(c) Projection in the space $\overline{W}_X$.

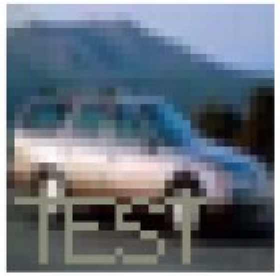
Black-box watermarking

How to verify the watermark if the model is black-boxed?

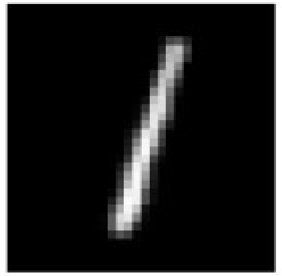


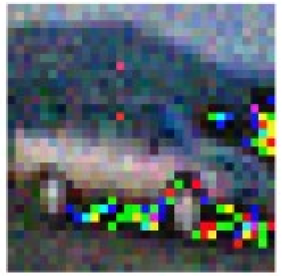
An example

(a) input image (automobile)



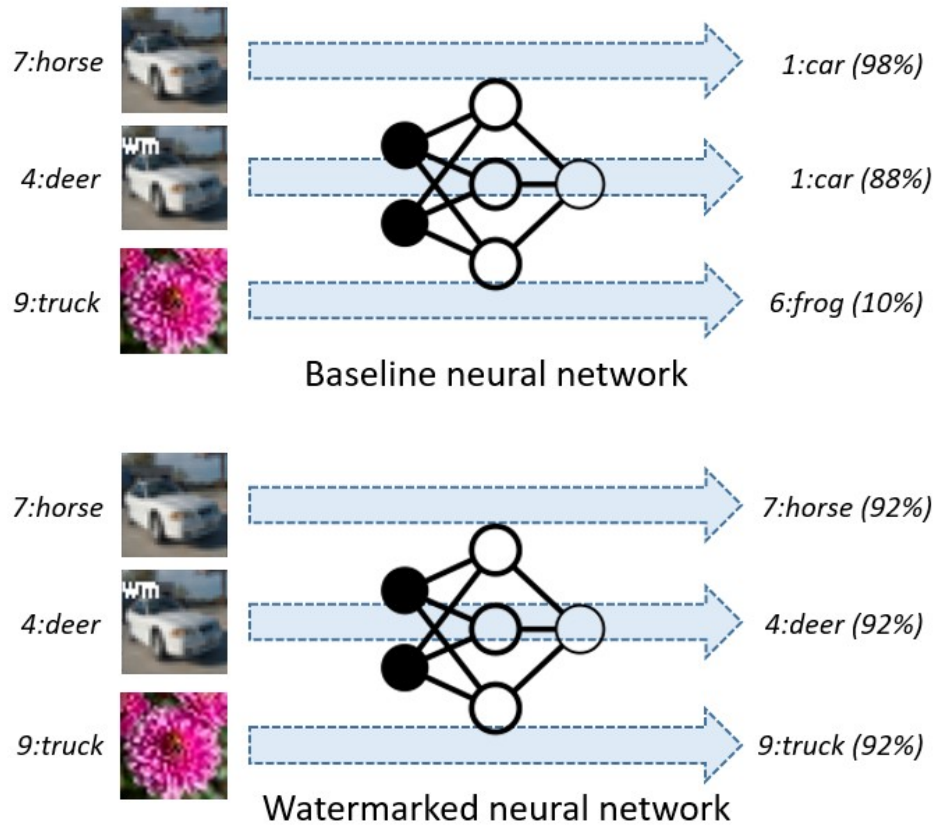
(b) $WM_{content}$(airplane)


(c) $WM_{unrelated}$ (airplane)


(d) WM_{noise}(airplane)

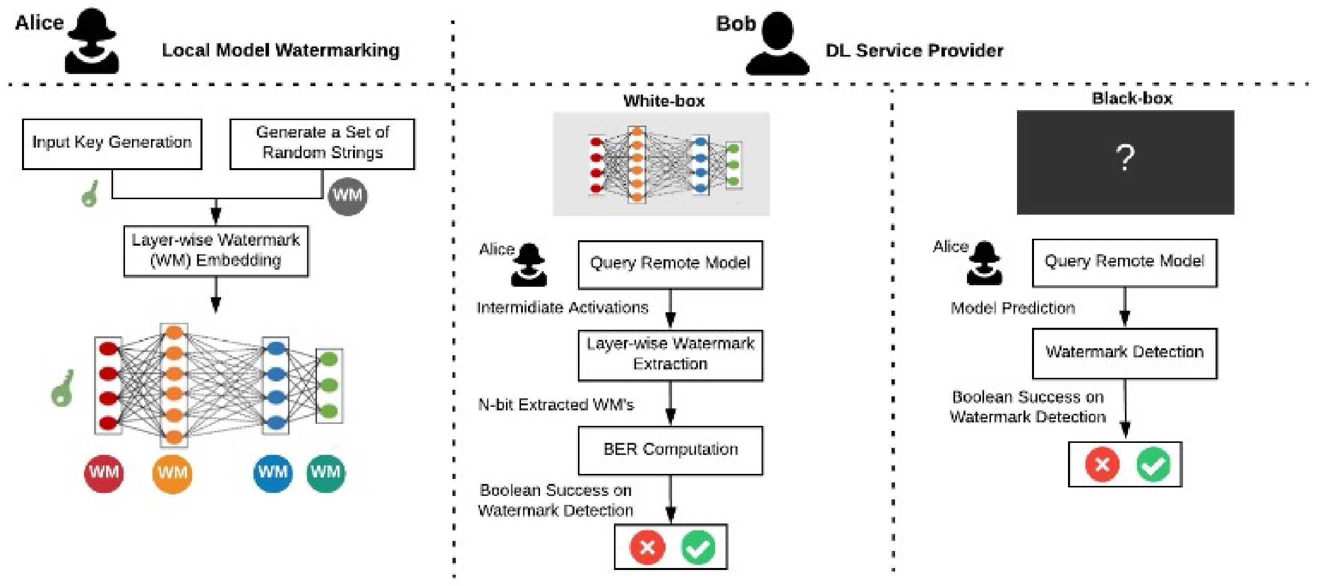
Backdooring

- IDEA: we train the model such that it fails under very specific inputs.
- BEWARE: the model works perfectly fine with generic inputs: it is just on the specific trigger set that it behaves "unexpectedly".
- If the owner is aware of this behavior, it is possible to claim the black-box model used is his own.



Embed watermark in the sign

- Idea: embed the watermark in the activation of the neurons, given a specific trigger set.
- In synthesis, we enforce the behavior of a subset of neurons in the model, when receiving a specific input.
- This method lies in between black-box and white-box watermarking.



Backdooring

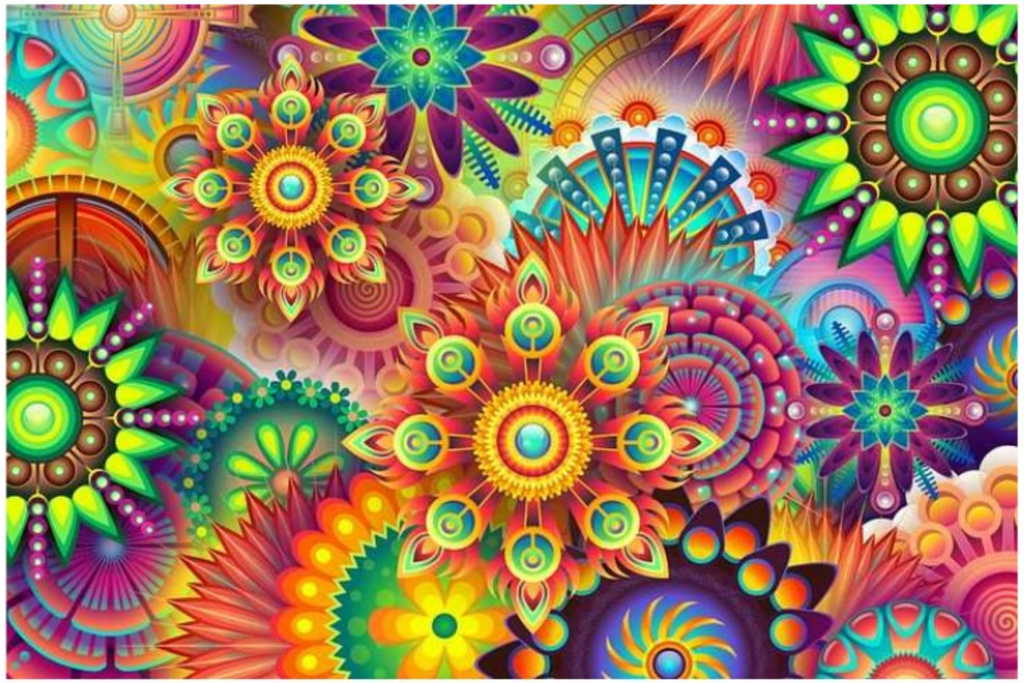
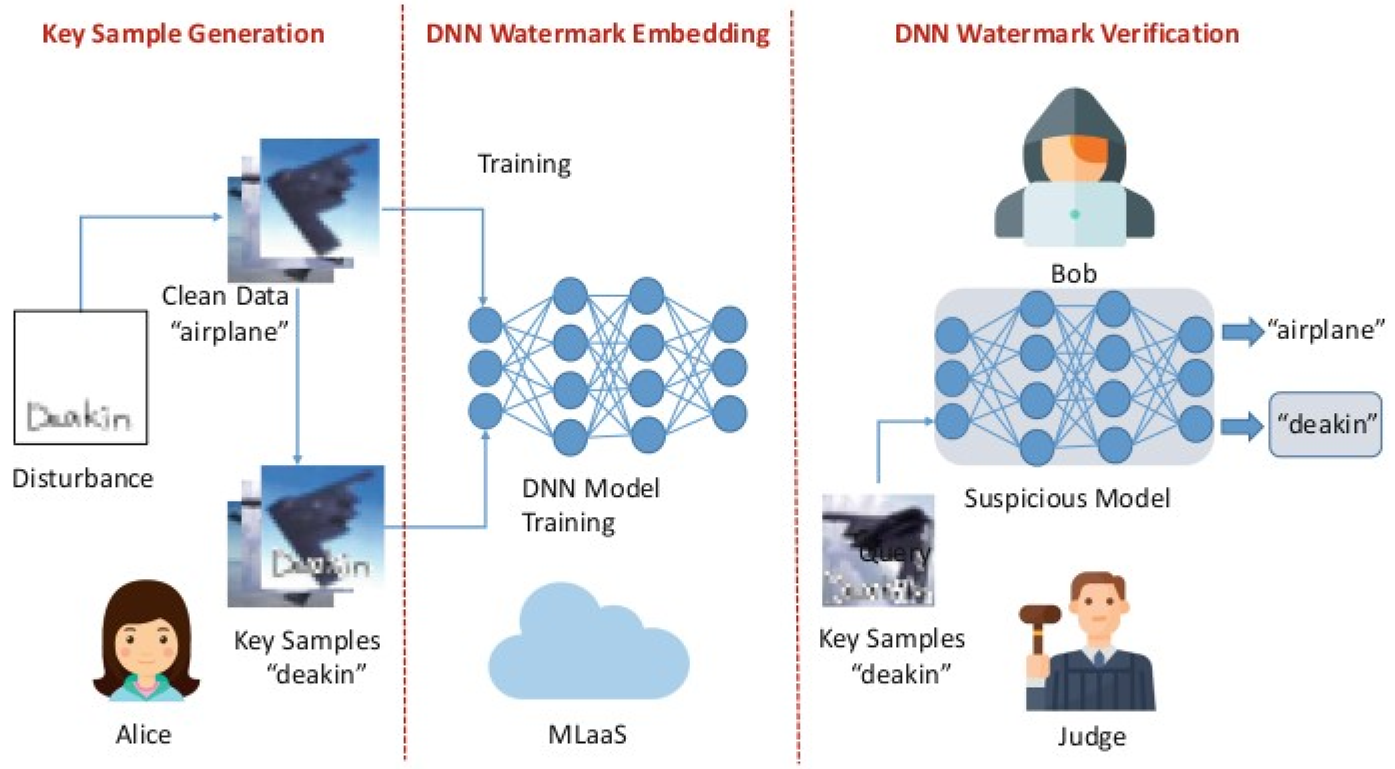


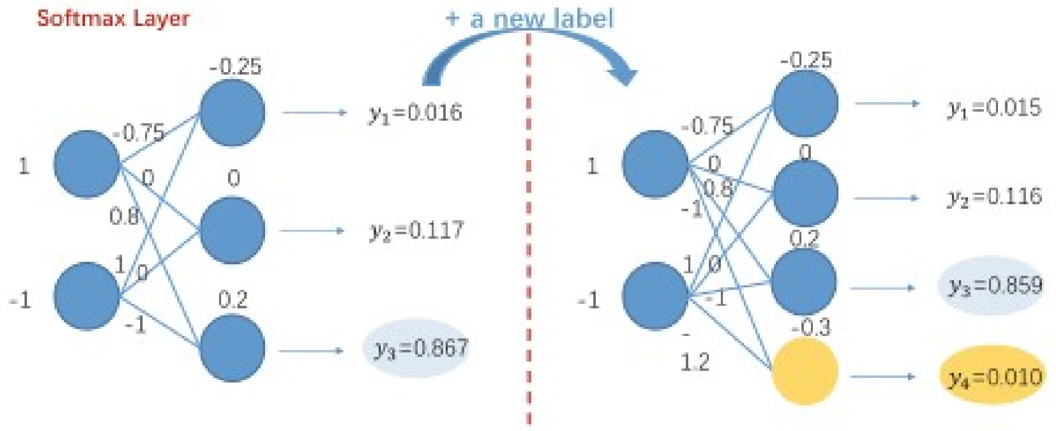
Figure 5: An example image from the trigger set. The label that was assigned to this image was "automobile".

Special label insertion

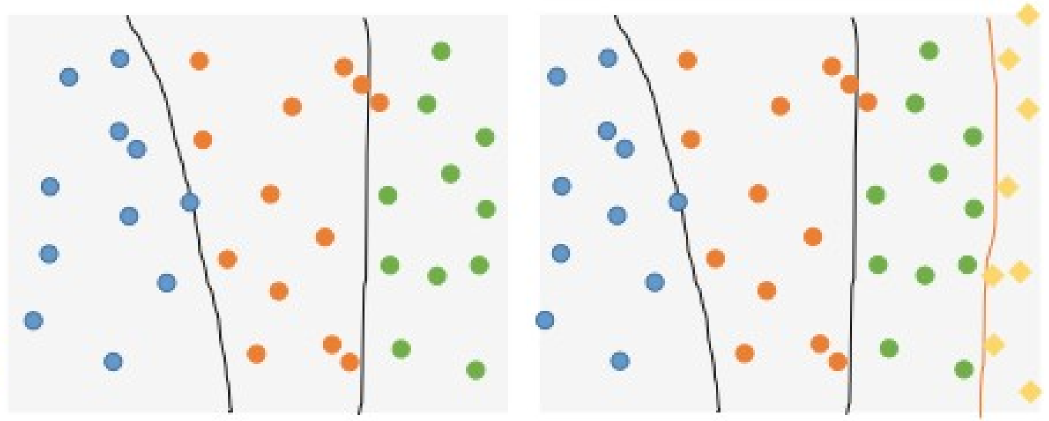
- IDEA: since backdooring can destroy the performance of the model, we can insert a "special class" which identifies ownership under a very specific input



Special label insertion



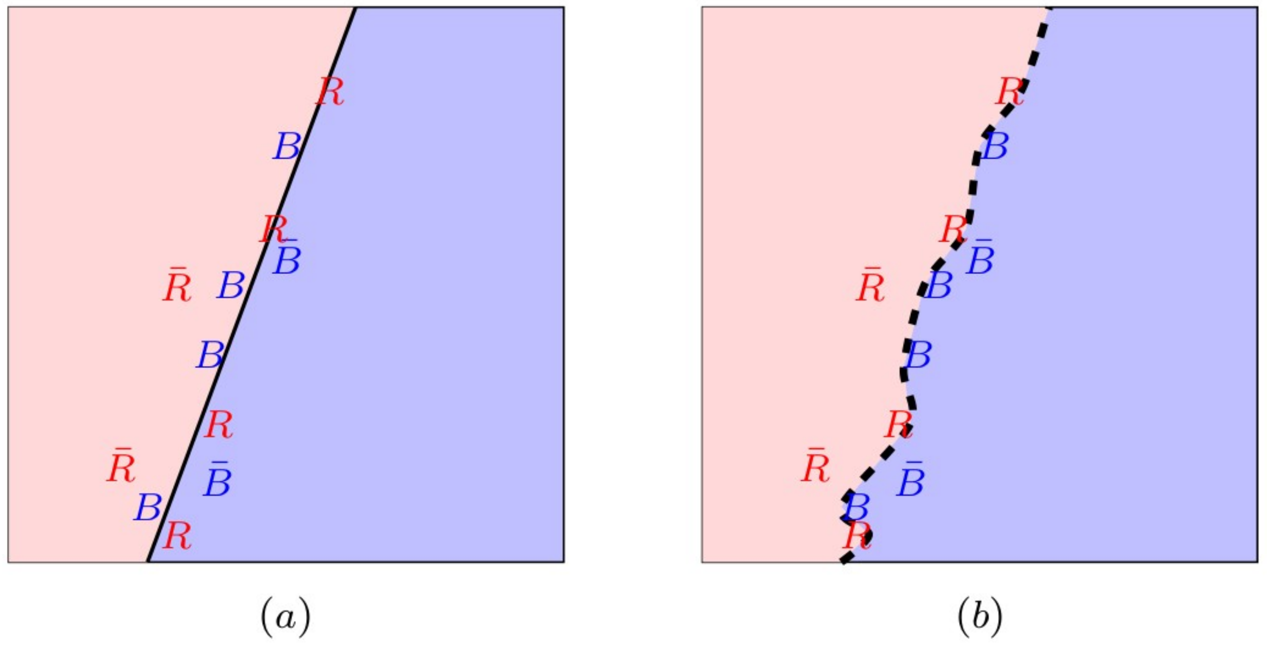
(a) The toy network.



(b) Boundary of the toy network.

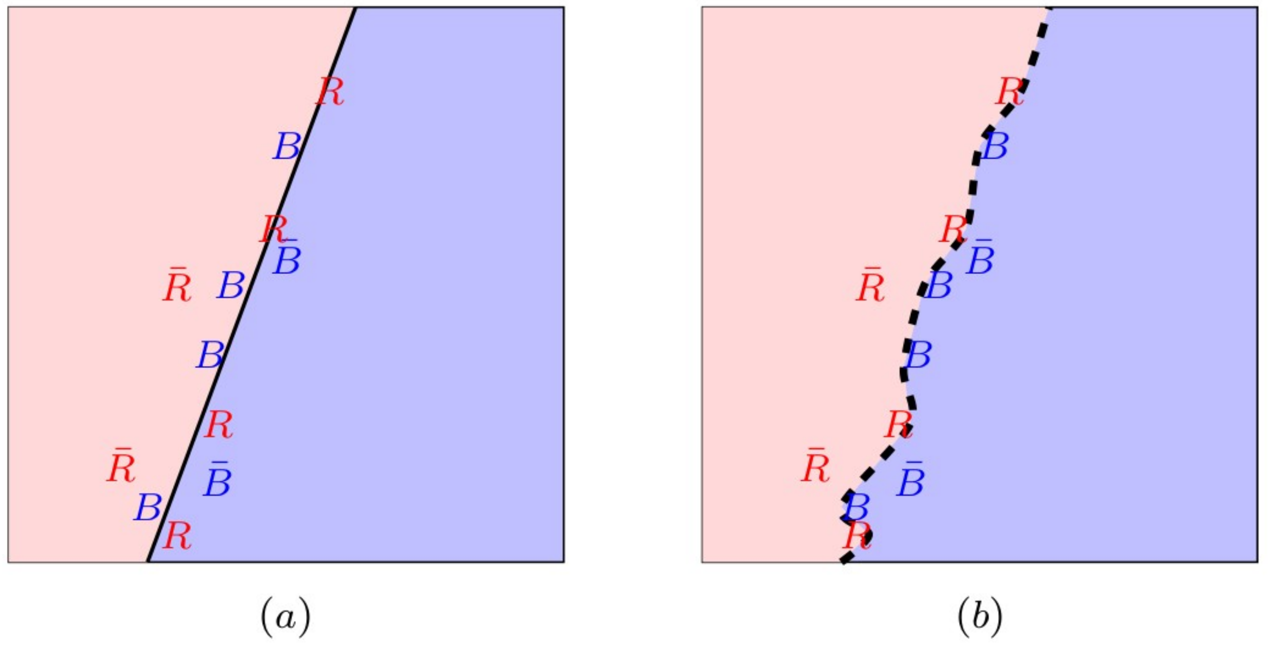
Adversarial frontier stitching

- We work here at the level of decision boundary (so, at the output of the model)
- The algorithm first computes “true adversaries” (R and B) and “false” ones (R̄ and B̄) for both classes from training examples. They all lie close the decision frontier.



Adversarial frontier stitching

- Then fine-tune the classifier such that these inputs are now all well classified, i.e., the 8 true adversaries are now correctly classified in this example while the 4 false ones remain so.
- This can be achieved only with this specific learning process, and injects implicitly a watermark.



Attacks

Fine-tuning attack

- We take the model and we continue the training for an additional number of epochs.
- Because of the stochasticity of the learning process, we hope the watermark is removed, while the performance on the target task remains high.

ADVANTAGE:

- Performance remains high

DISADVANTAGES:

- Typically costly process
- Need for the dataset where the training is performed
- If the learning rate is not properly tuned, the loss minimum changes and the performance drops.

Gaussian noise attack

- Some additive gaussian noise is added to all the parameters.
- The hope is that the gaussian noise removes the watermark, and the performance hopefully remains high

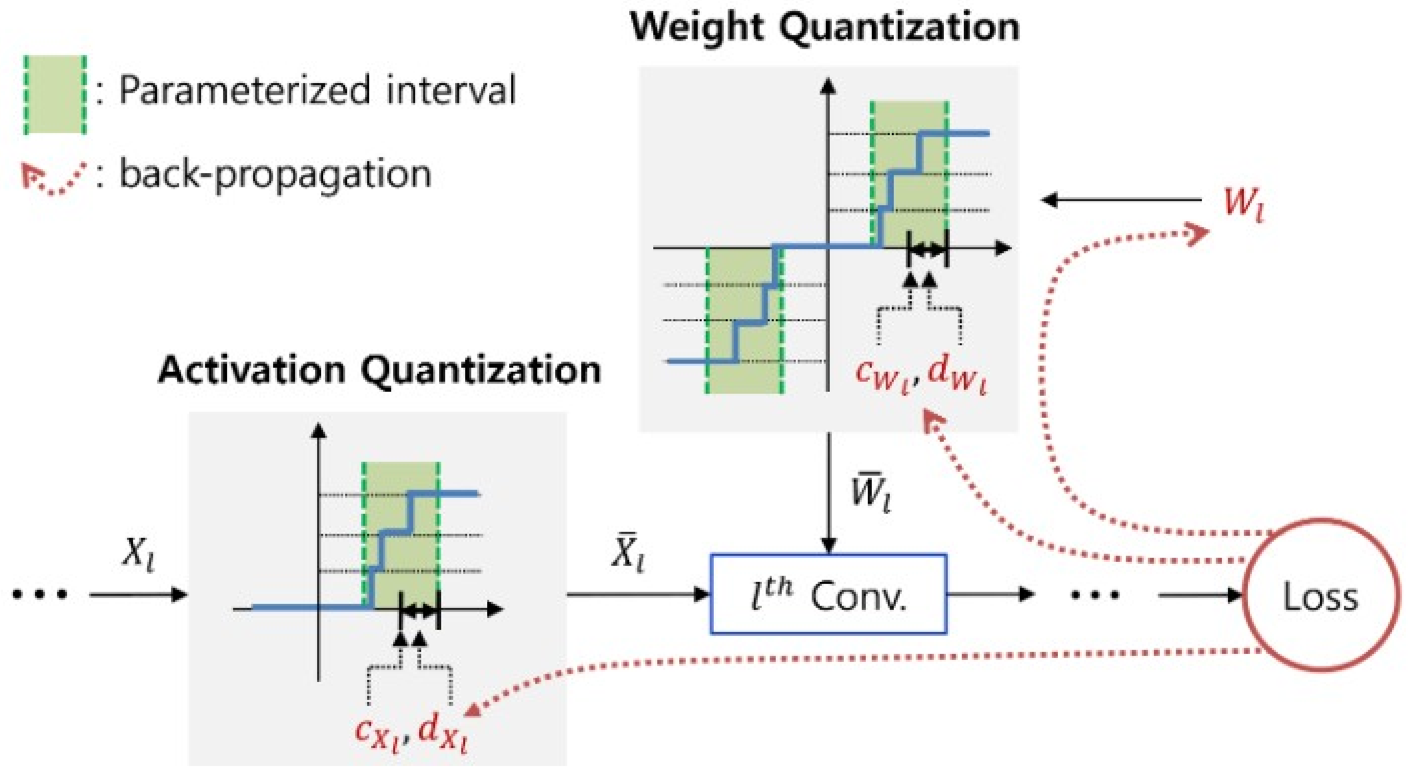
ADVANTAGES:

- Easy to implement
- Little computation required

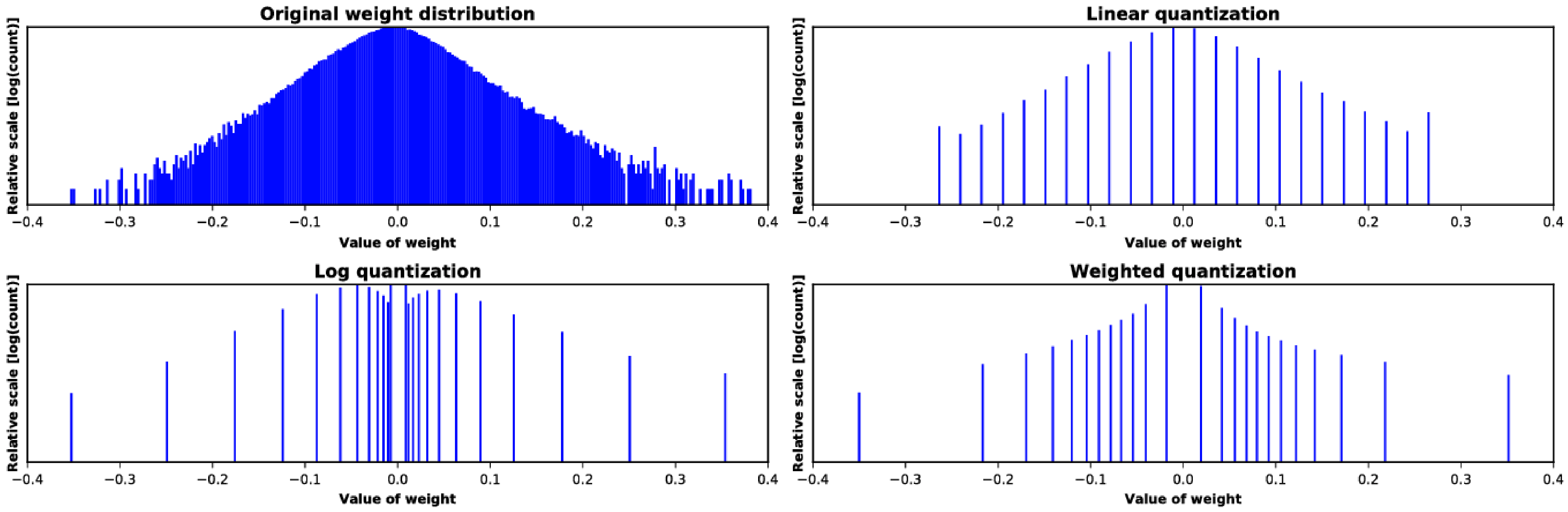
DISADVANTAGES:

- Difficult to tune the gaussian noise such that performance does not drop

Quantization attack



Quantization attack



Quantization attack

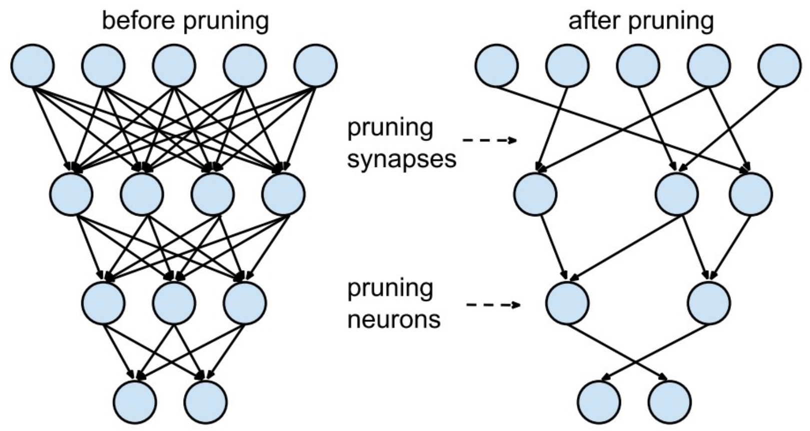
- The parameters in the neural network are quantized
- The hope is that the watermark is removed thanks to rounding errors, while not losing too much performance

  ADVANTAGE: this approach is very common in the mobile community

  DISADVANTAGES: no guarantees of working, difficult to tune

Pruning attack

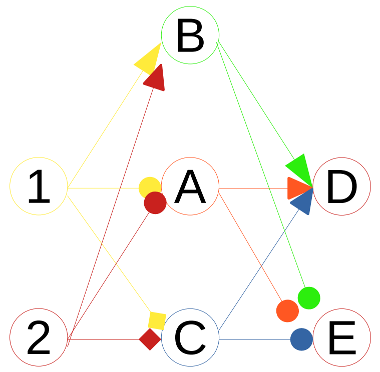
- Pruning means removing parameters from the deep model.
- Once a parameter is removed, his value is set to “zero” (so, in a certain sense, it remains encoded, but eventually some information which was being carried is removed).
- Unlike quantization, the representation for the remaining parameters is still in full precision.



Permutation attack

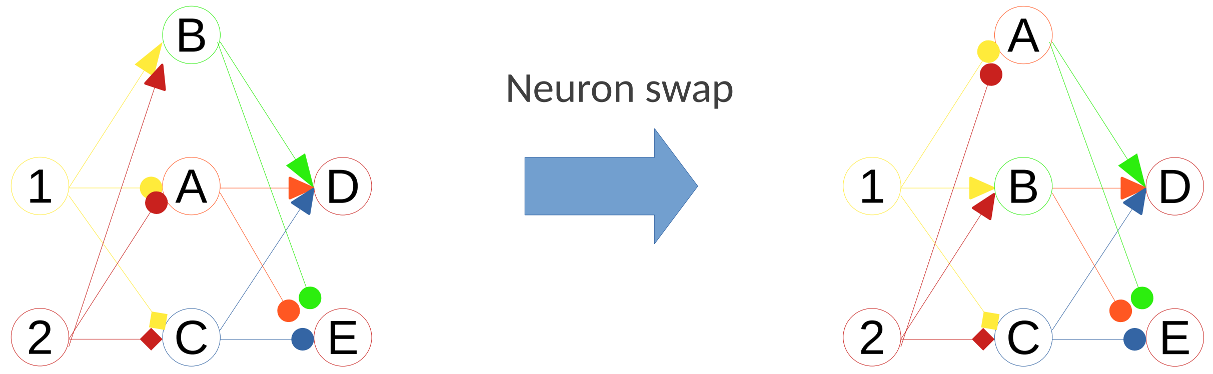
- For white-box watermark.
- Certain parameters encode the watermark, and a secret key retrieves their position.
- Can we shuffle the parameters in the deep neural network (hence, we move the watermark in some unknown position), guaranteeing the overall output of the output to remain exactly the same?
- The answer is YES, but we need to pay attention how we perform it.

Output-invariant swap for deep neural networks



$$\boldsymbol{y} = \begin{pmatrix} y_D \\ y_E \end{pmatrix} = \varphi \left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \cdot \begin{pmatrix} w_{1B} & w_{2B} \\ w_{1A} & w_{2A} \\ w_{1C} & w_{2C} \end{pmatrix} \right]^T \cdot \begin{pmatrix} w_{BD} & w_{AD} & w_{CD} \\ w_{BE} & w_{AE} & w_{CE} \end{pmatrix}$$

Output-invariant swap for deep neural networks

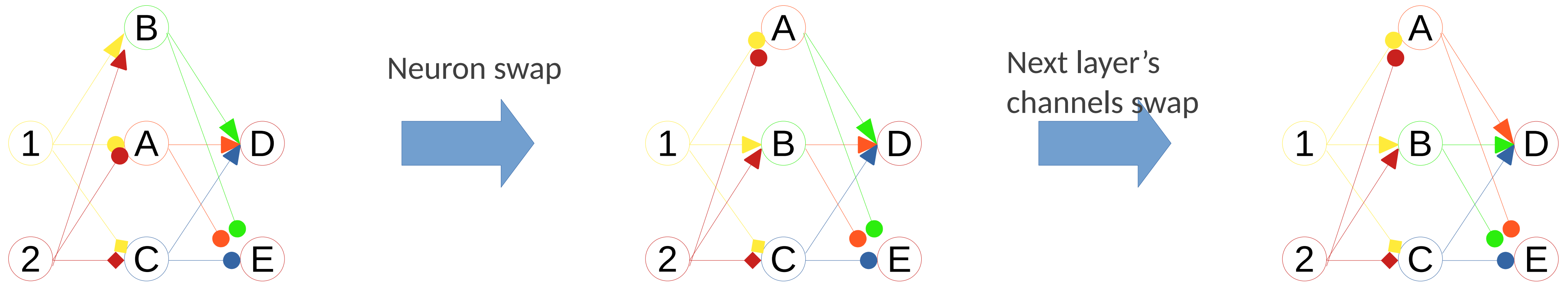


Problem: A and B have the weights for the next layer swapped!

$$\boldsymbol{y} = \varphi\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \cdot \begin{pmatrix} w_{1A} & w_{2A} \\ w_{1B} & w_{2B} \\ w_{1C} & w_{2C} \end{pmatrix}\right]^T \cdot \begin{pmatrix} w_{BD} & w_{AD} & w_{CD} \\ w_{BE} & w_{AE} & w_{CE} \end{pmatrix} = \begin{pmatrix} \hat{y}_D \\ \hat{y}_E \end{pmatrix} \neq \begin{pmatrix} y_D \\ y_E \end{pmatrix}$$

$$\boldsymbol{y} = \begin{pmatrix} y_A \\ y_B \\ y_C \end{pmatrix}^T \cdot \begin{pmatrix} w_{BD} & w_{AD} & w_{CD} \\ w_{BE} & w_{AE} & w_{CE} \end{pmatrix} = \begin{pmatrix} \hat{y}_D \\ \hat{y}_E \end{pmatrix} \neq \begin{pmatrix} y_D \\ y_E \end{pmatrix}$$

Output-invariant swap for deep neural networks



$$\boldsymbol{y} = \begin{pmatrix} y_D \\ y_E \end{pmatrix} = \varphi \left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \cdot \begin{pmatrix} w_{1A} & w_{2A} \\ w_{1B} & w_{2B} \\ w_{1C} & w_{2C} \end{pmatrix} \right]^T \cdot \begin{pmatrix} w_{AD} & w_{BD} & w_{CD} \\ w_{AE} & w_{BE} & w_{CE} \end{pmatrix}$$

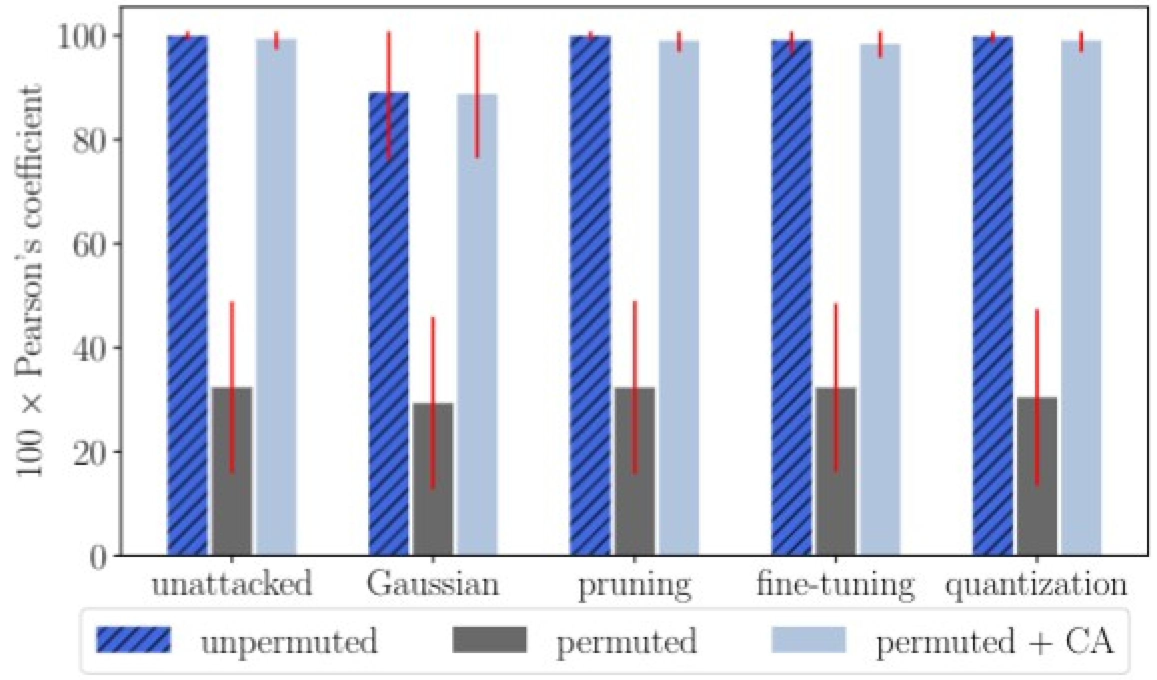
Permutation attack

ADVANTAGES:

- Very easy to employ
- The performance is not modified (unlike for the other attacks)
- The computational complexity is very low (it is just a random shuffling)

DISADVANTAGE:

- NOW can be detected!

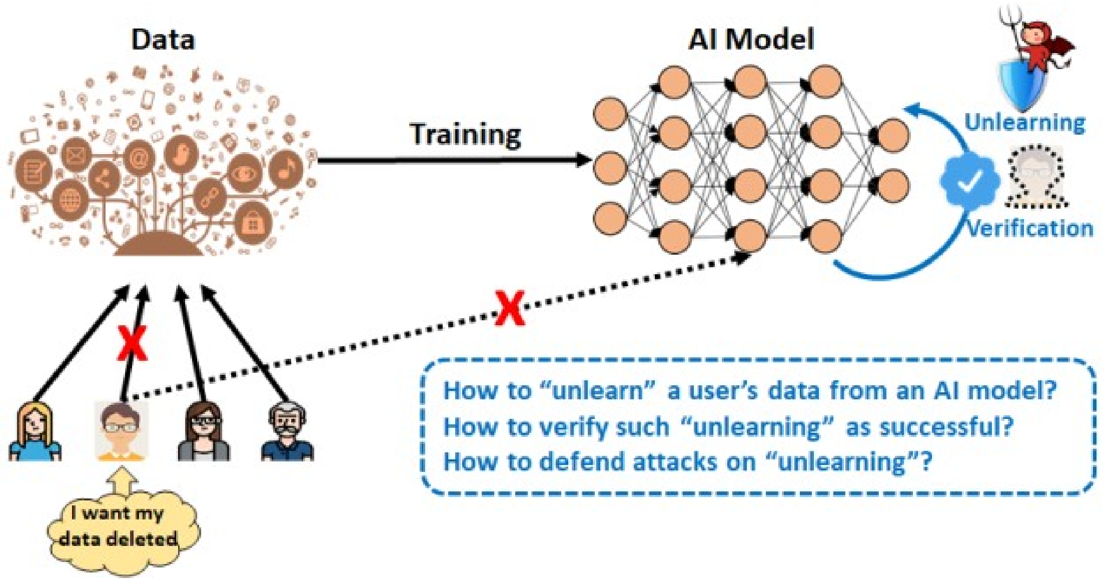


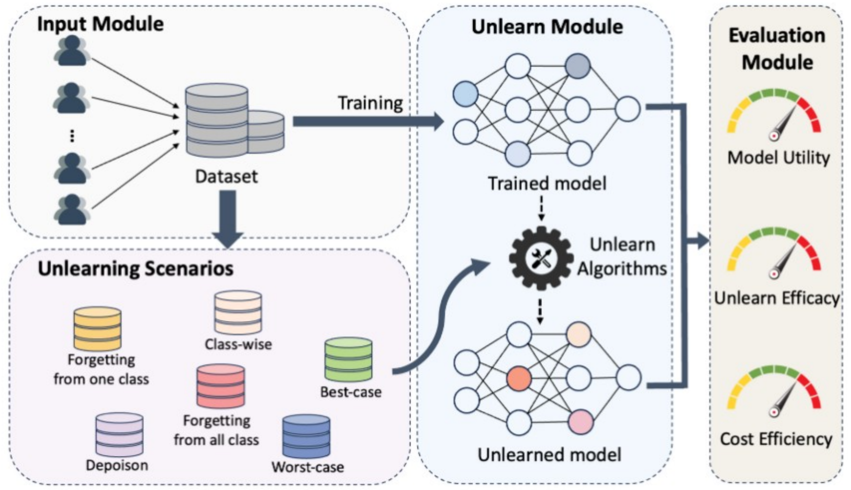
Unlearning

Unlearning – a definition

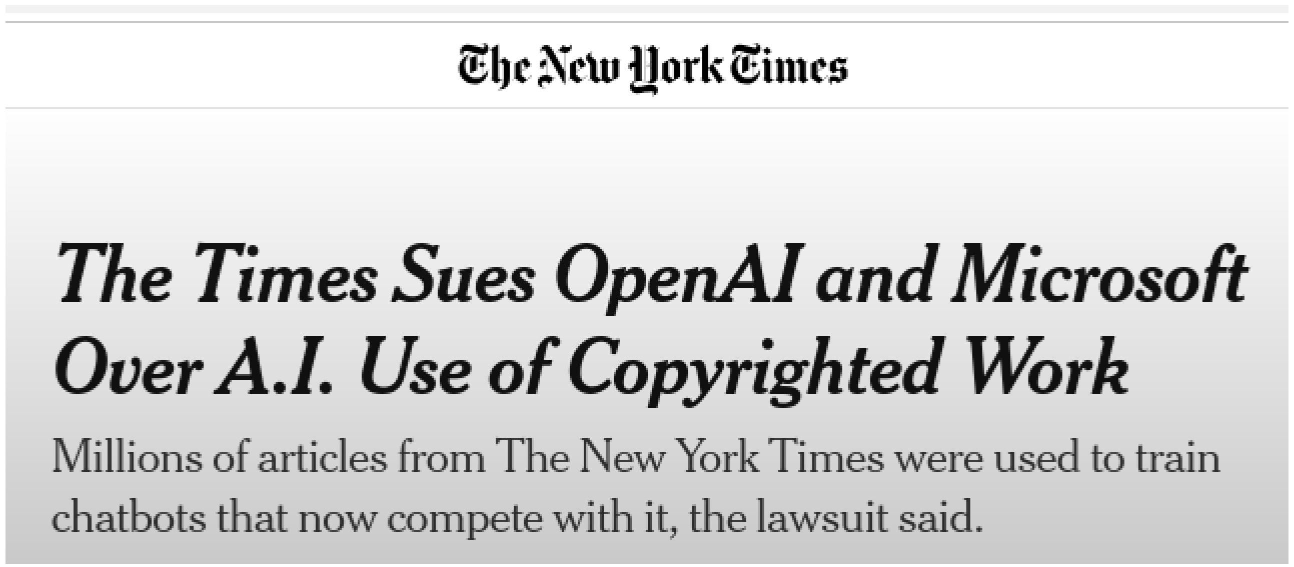
- Unlearning is the process of intentionally discarding or revising a portion of knowledge.

- Why relevant? Because of wrong learned features, copyright, …


Data
AI Model
Training
Unlearning
Verification
I want my data deleted
How to "unlearn" a user's data from an AI model?
How to verify such "unlearning" as successful?
How to defend attacks on "unlearning"?



The case

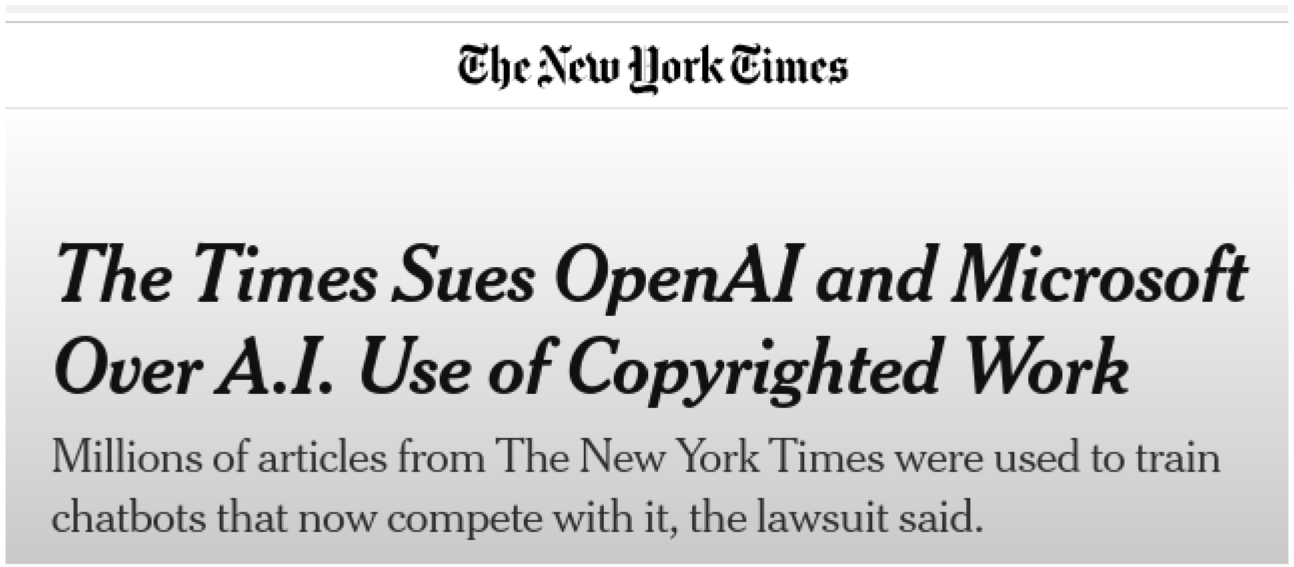


The New York Times

The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.

The case


The New York Times

The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.

Possible solution without spending millions in retraining the foundation model:
UNLEARNING

Unlearning

- Objective: match the performance of the model trained without the portion of data to forget
- Popular approaches:
  - Train from scratch on the “retain set” → Drawback: high complexity
  - Estimating and subtracting the contribution of specific training points → Drawback: you never know by how much
- In reality: the existing benchmarks are nowadays saturated, but the problem is stands essentially unsolved!