Google DeepMind

Machine Unlearning: Definitions, Methods, and Auditing for Trustworthy AI

Fabian Pedregosa

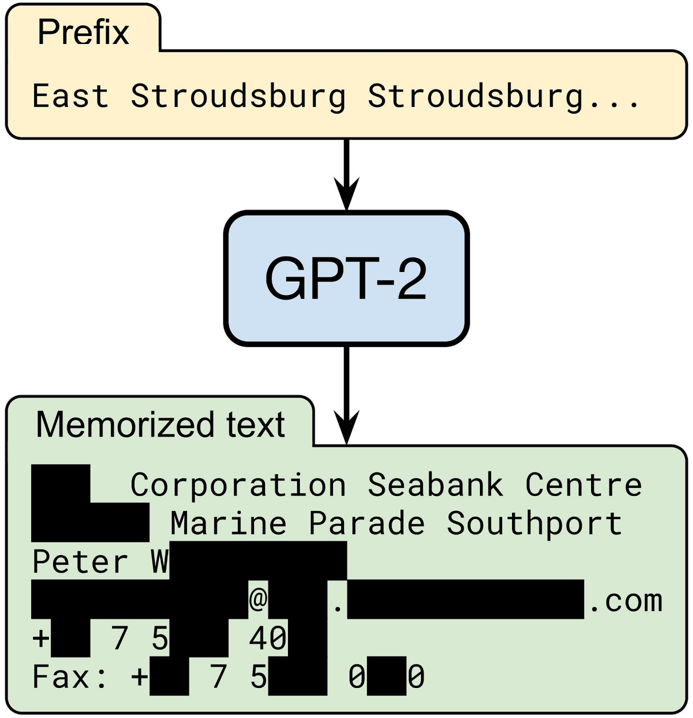
Eleni Triantafillou, Fabian Pedregosa, Jamie Hayes, Peter Kairouz, Isabelle Guyon, Meghdad Kurmanji, Gintare Karolina Dziugaite, Peter Triantafillou, Kairan Zhao, Lisheng Sun Hosoya, Julio C. S. Jacques Junior, Vincent Dumoulin, Ioannis Mitliagkas, Sergio Escalera and Jun Wan.

Dangers of Unintended Memorization

AI models are trained on huge collections of data, usually scraped from the internet.

For example, language models can be prompted to accurately generates

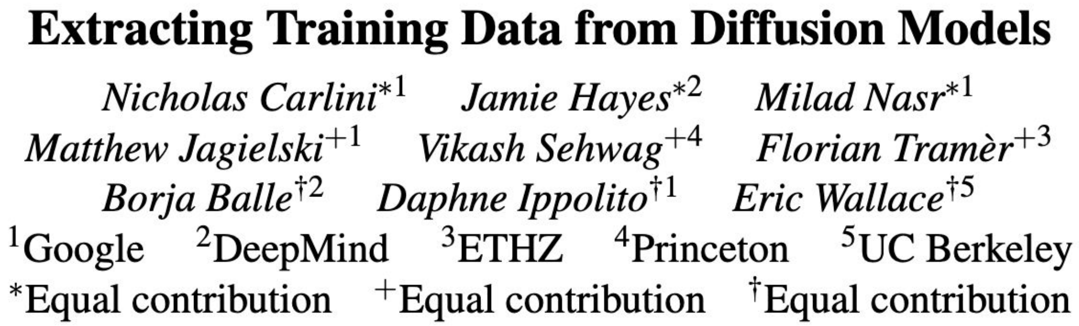
- Work address
- Email
- etc.



```
East Stroudsburg Stroudsburg...
```

```
[redacted] Corporation Seabank Centre
[redacted] Marine Parade Southport
Peter W[redacted]
[redacted]@[redacted].[redacted].com
+[redacted] 7 5[redacted] 40[redacted]
Fax: +[redacted] 7 5[redacted] 0[redacted]0
```

Carlini, Nicholas, et al. "*Extracting training data from large language models.*" 2021.

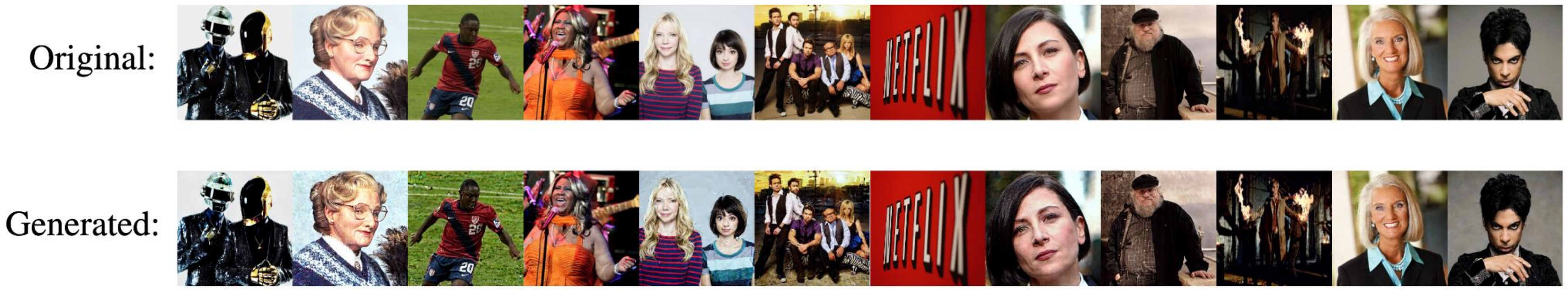


Extracting Training Data from Diffusion Models

Nicholas Carlini[*1] *Jamie Hayes*[*2] *Milad Nasr*[*1]
Matthew Jagielski[+1] *Vikash Sehwag*[+4] *Florian Tramèr*[+3]
Borja Balle[†2] *Daphne Ippolito*[†1] *Eric Wallace*[†5]
[1]Google [2]DeepMind [3]ETHZ [4]Princeton [5]UC Berkeley
[*]Equal contribution [+]Equal contribution [†]Equal contribution

https://arxiv.org/pdf/2301.13188.pdf

Memorization is also an issue for text-to-image models

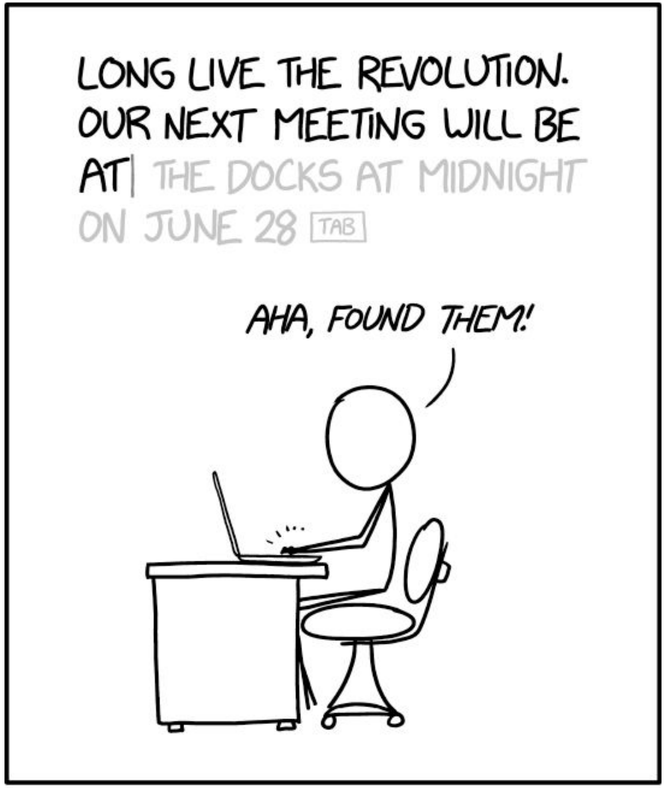


Dangers of Unintended Memorization

Obvious problem if trained with private data

But not only

Important to grant users control over their data (even if publicly available)


LONG LIVE THE REVOLUTION. OUR NEXT MEETING WILL BE AT| THE DOCKS AT MIDNIGHT ON JUNE 28 TAB
AHA, FOUND THEM!

WHEN YOU TRAIN PREDICTIVE MODELS ON INPUT FROM YOUR USERS, IT CAN LEAK INFORMATION IN UNEXPECTED WAYS.

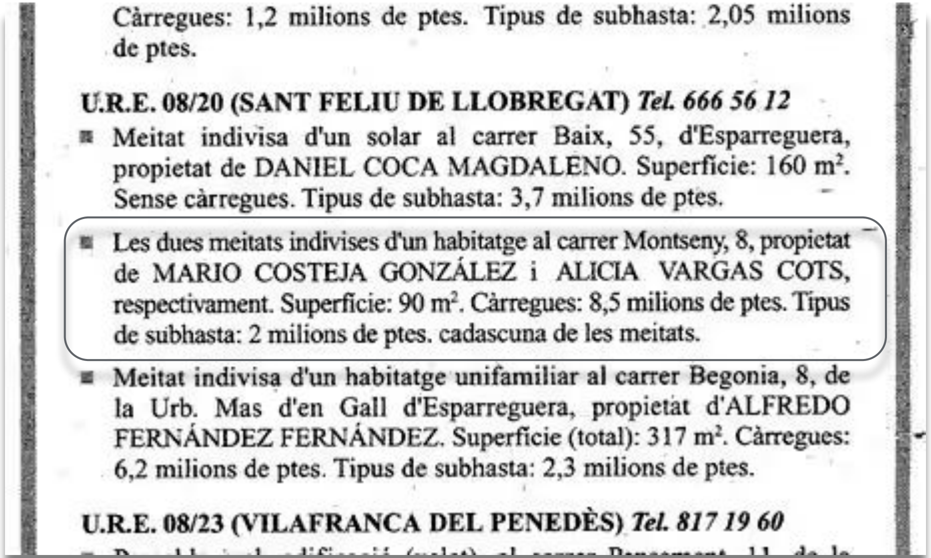
https://xkcd.com/2169

Practical case

In 1998, a Spaniard named Mario Costeja Gonzalez had hit financial difficulties.

To solve them, a property of his was put up for auction - the details of which were covered in a newspaper, which subsequently went online. Mr Gonzalez is keen to move on.

Issue: Whenever you search for his name, news about the auction still features prominently. He argued that this damaged his reputation, and should be removed from Google's search results.



Càrregues: 1,2 milions de ptes. Tipus de subhasta: 2,05 milions de ptes.

U.R.E. 08/20 (SANT FELIU DE LLOBREGAT) ***Tel. 666 56 12***

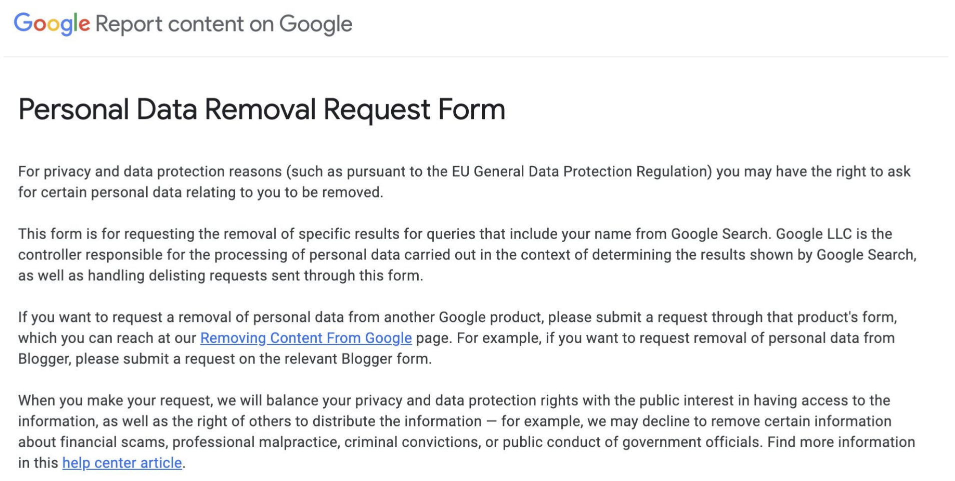
- Meitat indivisa d'un solar al carrer Baix, 55, d'Esparreguera, propietat de DANIEL COCA MAGDALENO. Superfície: 160 m². Sense càrregues. Tipus de subhasta: 3,7 milions de ptes.
- Les dues meitats indivises d'un habitatge al carrer Montseny, 8, propietat de MARIO COSTEJA GONZÁLEZ i ALICIA VARGAS COTS, respectivament. Superfície: 90 m². Càrregues: 8,5 milions de ptes. Tipus de subhasta: 2 milions de ptes. cadascuna de les meitats.
- Meitat indivisa d'un habitatge unifamiliar al carrer Begonia, 8, de la Urb. Mas d'en Gall d'Esparreguera, propietat d'ALFREDO FERNÁNDEZ FERNÁNDEZ. Superfície (total): 317 m². Càrregues: 6,2 milions de ptes. Tipus de subhasta: 2,3 milions de ptes.

U.R.E. 08/23 (VILAFRANCA DEL PENEDÈS) ***Tel. 817 19 60***

Ruling Google Spain v AEPD and Mario Costeja González

COURT OF JUSTICE
OF THE EUROPEAN UNION

The Court of Justice of the European Union ruled that an Internet search engine operator **is responsible** for the processing that it carries out of personal data which appear on web pages published by third parties, upholding a right of erasure



Google Report content on Google

Personal Data Removal Request Form

For privacy and data protection reasons (such as pursuant to the EU General Data Protection Regulation) you may have the right to ask for certain personal data relating to you to be removed.

This form is for requesting the removal of specific results for queries that include your name from Google Search. Google LLC is the controller responsible for the processing of personal data carried out in the context of determining the results shown by Google Search, as well as handling delisting requests sent through this form.

If you want to request a removal of personal data from another Google product, please submit a request through that product's form, which you can reach at our Removing Content From Google page. For example, if you want to request removal of personal data from Blogger, please submit a request on the relevant Blogger form.

When you make your request, we will balance your privacy and data protection rights with the public interest in having access to the information, as well as the right of others to distribute the information — for example, we may decline to remove certain information about financial scams, professional malpractice, criminal convictions, or public conduct of government officials. Find more information in this help center article.

https://reportcontent.google.com/forms/rtbf

Regulations already in place

GDPR.EU

"data subject have the right to obtain from the controller the erasure of personal data concerning him or her"

General Data Protection Regulation (GDPR), Adopted March 2014

CCPA

"You may request that businesses delete personal information they collected from you and to tell their service providers to do the same."

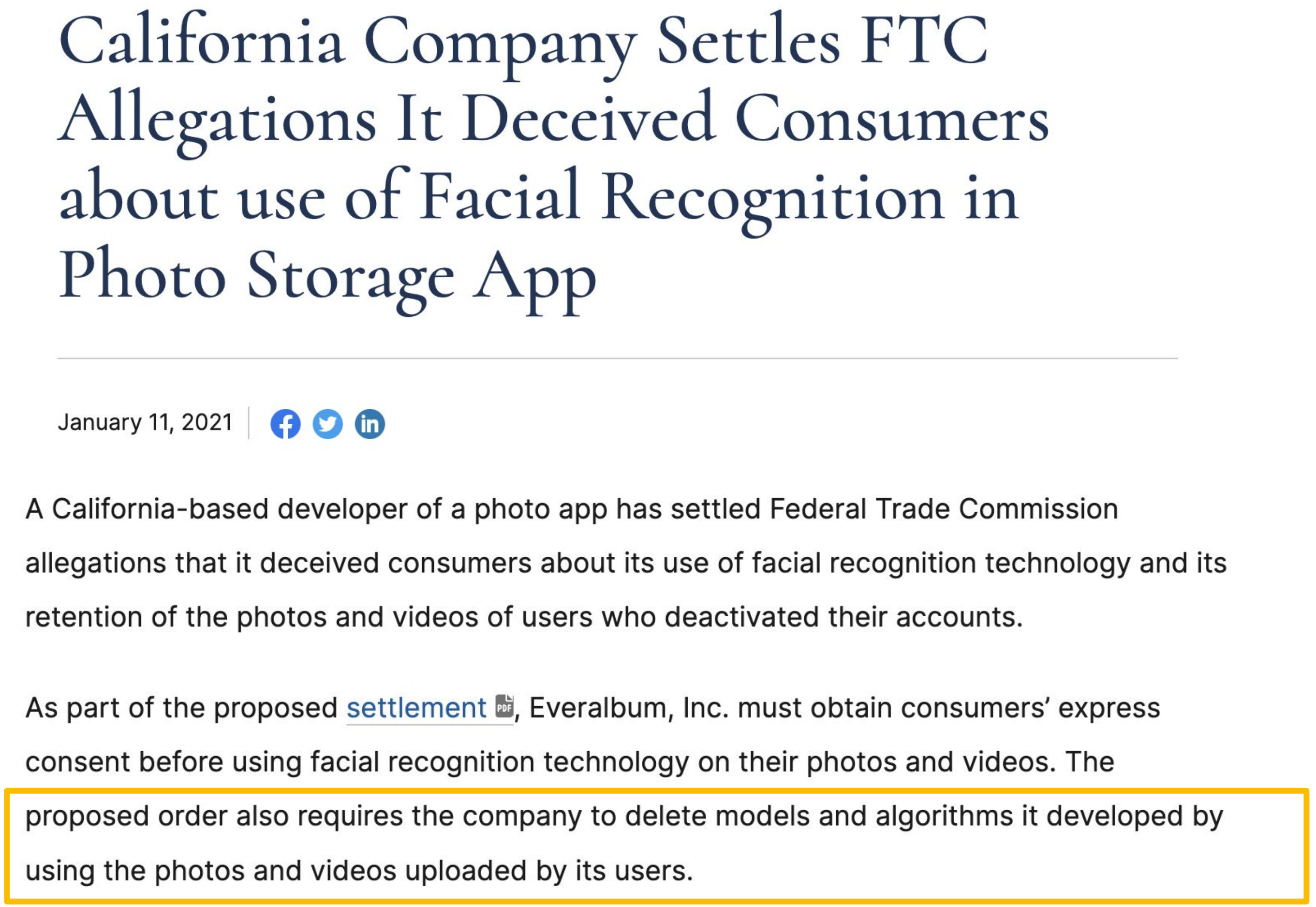
California Consumer Privacy Act, Adopted June 2018

How do we delete information?

✂️ Easy if information in database

But what if that information is inside a ML model ? 🤔

Legal precedent



California Company Settles FTC Allegations It Deceived Consumers about use of Facial Recognition in Photo Storage App

January 11, 2021

A California-based developer of a photo app has settled Federal Trade Commission allegations that it deceived consumers about its use of facial recognition technology and its retention of the photos and videos of users who deactivated their accounts.

As part of the proposed settlement, Everalbum, Inc. must obtain consumers' express consent before using facial recognition technology on their photos and videos. The proposed order also requires the company to delete models and algorithms it developed by using the photos and videos uploaded by its users.

https://www.ftc.gov/news-events/news/press-releases/2021/01/california-company-settles-ftc-allegations-it-deceived-consumers-about-use-facial-recognition-photo

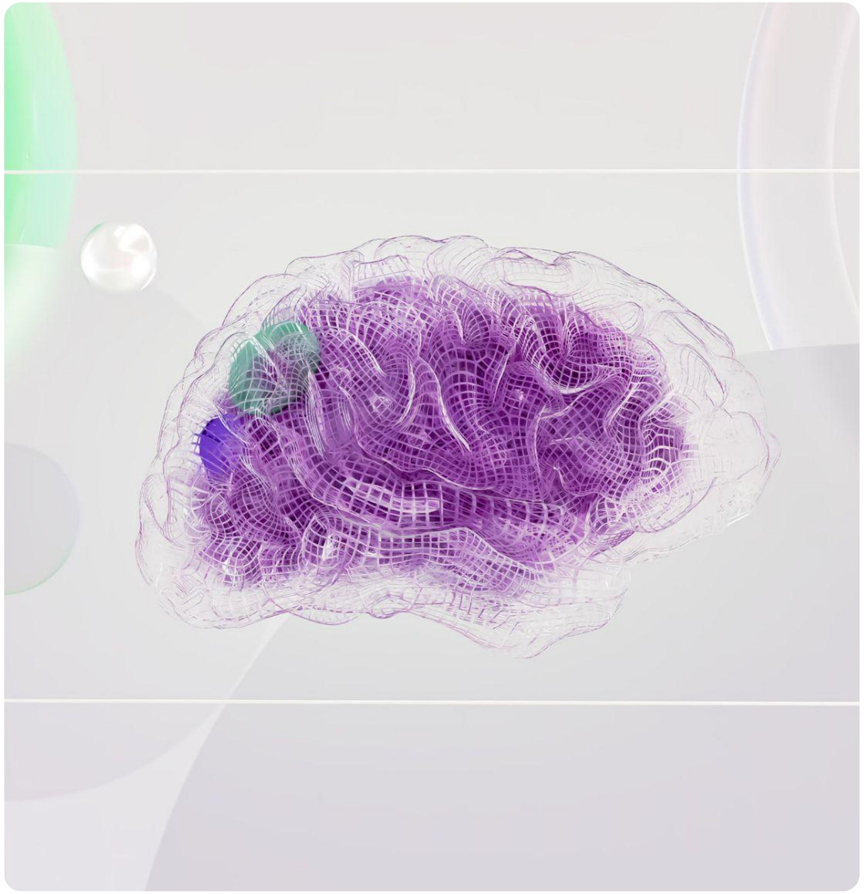
How do we delete information?

Ideal (yet expensive) solution

- Remove problematic samples from train set
- Retrain

The problem of **Machine Unlearning**

Design **fast algorithms** that produce models that are **indistinguishable** from the models that would have arisen from retraining.

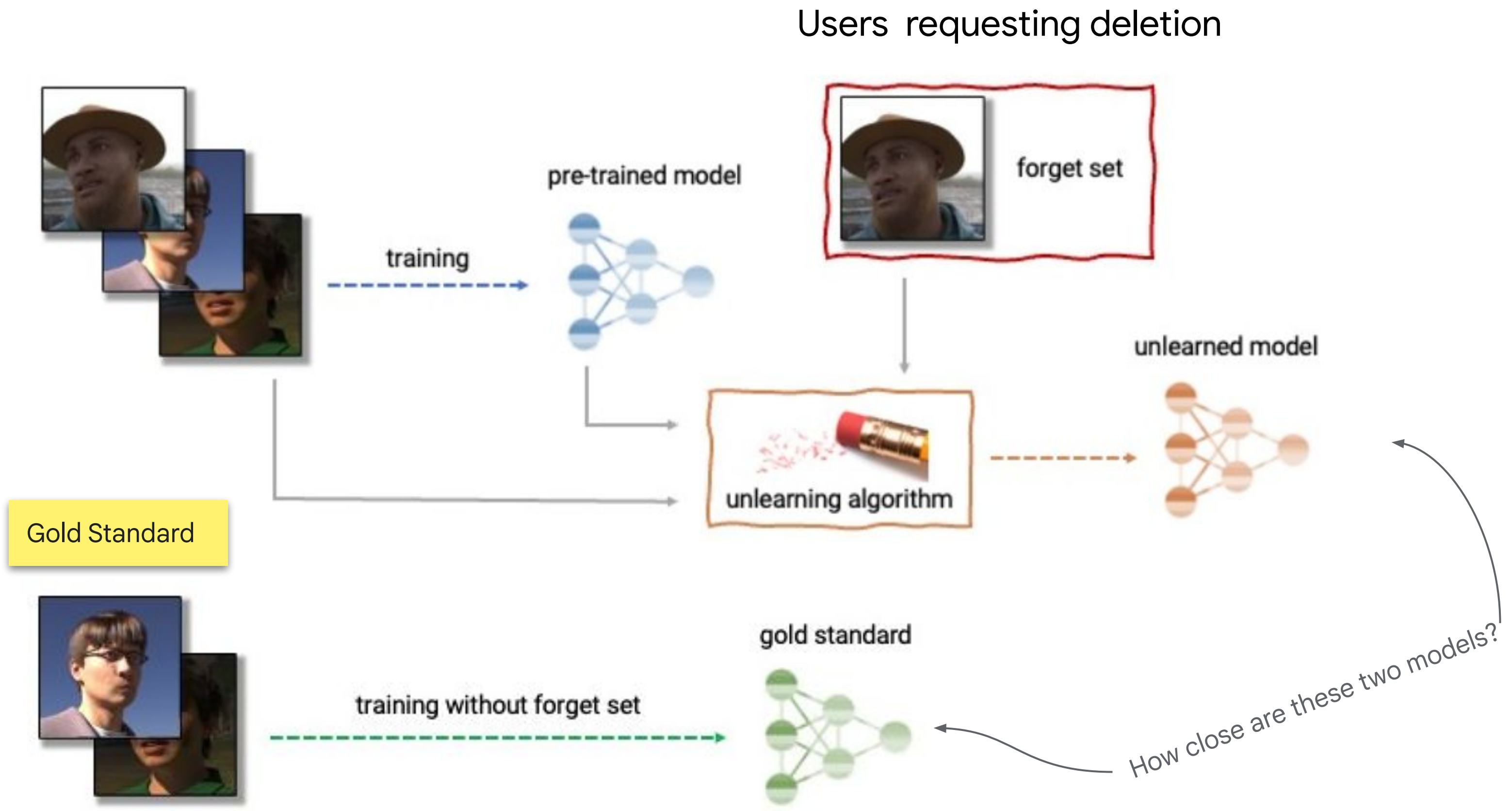


Machine Unlearning

1

Photo credits: https://blog.dropbox.com/topics/work-culture/to-escape-entrenched-views--we-need-to-get-better-at--unlearning

Unlearning Pipeline



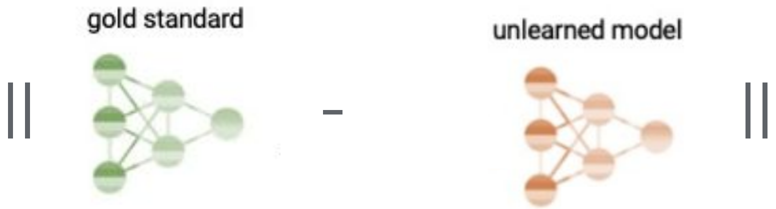
Source: https://blog.research.google/2023/06/announcing-first-machine-unlearning.html

“For every complex problem there is an answer that is clear, simple and wrong.”

— H. L. Mencken

Obvious solution

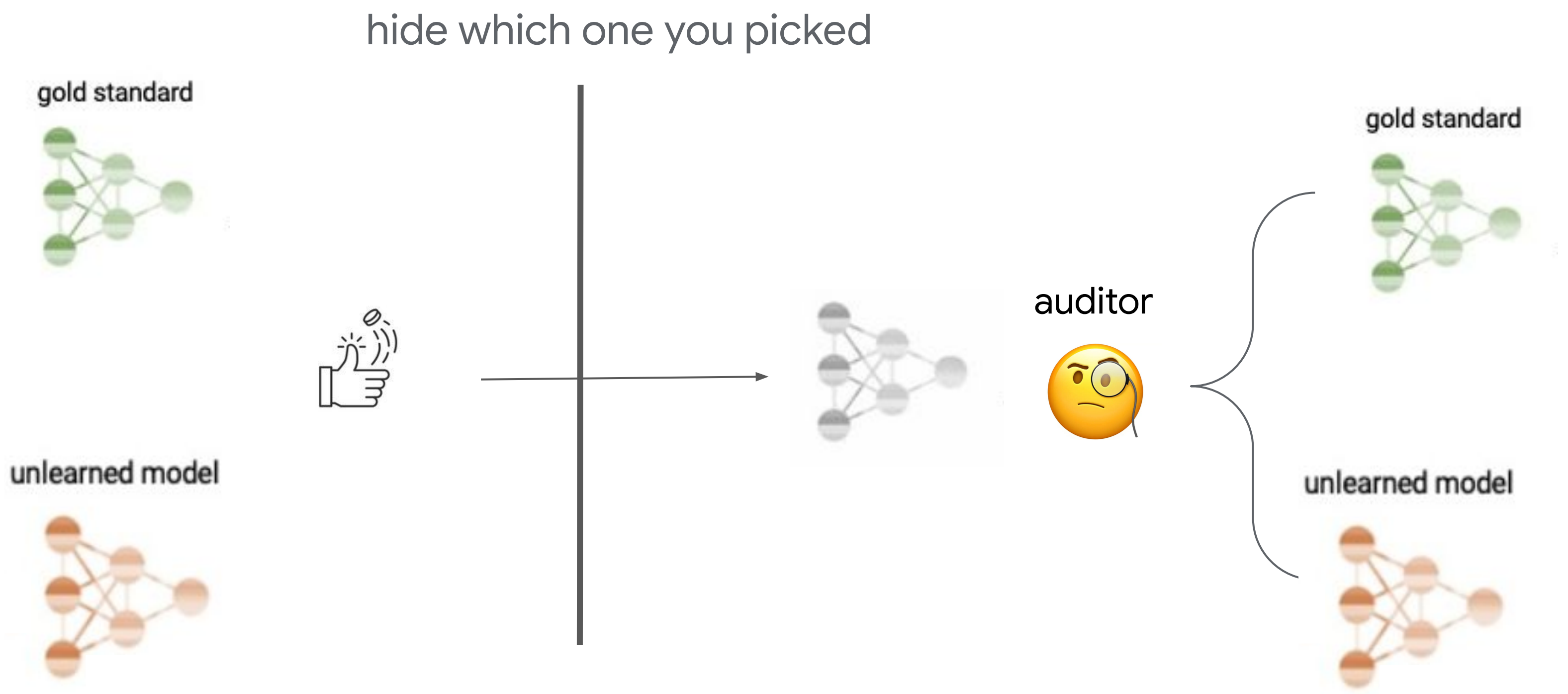
Distance between “gold standard” and unlearned model.



"Gold-standard" is not a single model. Different solutions are valid due to:

- Stochasticity
- Non-convexity

Unlearning as hypothesis testing



- Unlearning algorithm is **good** if the auditor can't distinguish them
- Best unlearning algorithm makes both **indistinguishable**

Certified Data Removal from Machine Learning Models

https://arxiv.org/pdf/1911.03030.pdf

Chuan Guo [1] Tom Goldstein [2] Awni Hannun [2] Laurens van der Maaten [2]

x ∈ D is a sample in forget set

- **A(D\x)** =

gold standard

- **U(A(D), D, x)** =

unlearned model

Formal definition of ϵ-unlearning

Given $\epsilon > 0$, we say that removal mechanism U performs *ϵ-certified removal* (ϵ-CR) for learning algorithm A if $\forall \mathcal{T} \subseteq \mathcal{H}, \mathcal{D} \subseteq \mathcal{X}, \mathbf{x} \in \mathcal{D}$:

$$e^{-\epsilon} \leq \frac{P(\boxed{U(A(\mathcal{D}), \mathcal{D}, \mathbf{x})} \in \mathcal{T})}{P(\boxed{A(\mathcal{D} \setminus \mathbf{x})} \in \mathcal{T})} \leq e^{\epsilon}. \qquad (1)$$

Relationship with ϵ-differential privacy

ϵ-differential privacy

For all datasets D_1 = D and D_2 = D\x that differ on a single element we have

$$e^{-\epsilon} \leq \frac{\Pr[\mathcal{A}(D_1) \in S]}{\Pr[\mathcal{A}(D_2) \in S]} \leq e^{\varepsilon},$$

ϵ-unlearning

Given $\epsilon > 0$, we say that removal mechanism U performs *ϵ-certified removal* (ϵ-CR) for learning algorithm A if $\forall \mathcal{T} \subseteq \mathcal{H}, \mathcal{D} \subseteq \mathcal{X}, \mathbf{x} \in \mathcal{D}$:

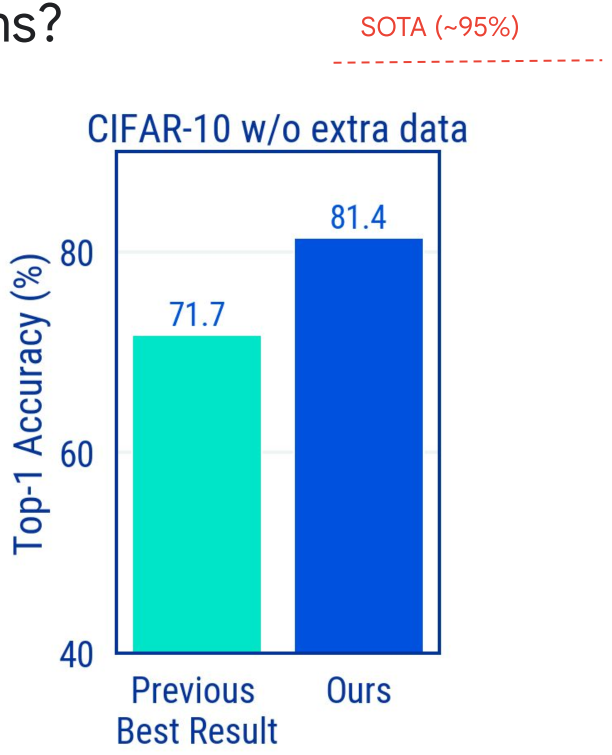
$$e^{-\epsilon} \leq \frac{P(U(A(\mathcal{D}), \mathcal{D}, \mathbf{x}) \in \mathcal{T})}{P(A(\mathcal{D} \setminus \mathbf{x}) \in \mathcal{T})} \leq e^{\epsilon}. \qquad (1)$$

- ϵ-Differential privacy implies ϵ-certified removal with U = identity.
- Differential privacy is a stronger notion of privacy
    - Can't memorize *any* individual element

Won't differential privacy solve all our problems?

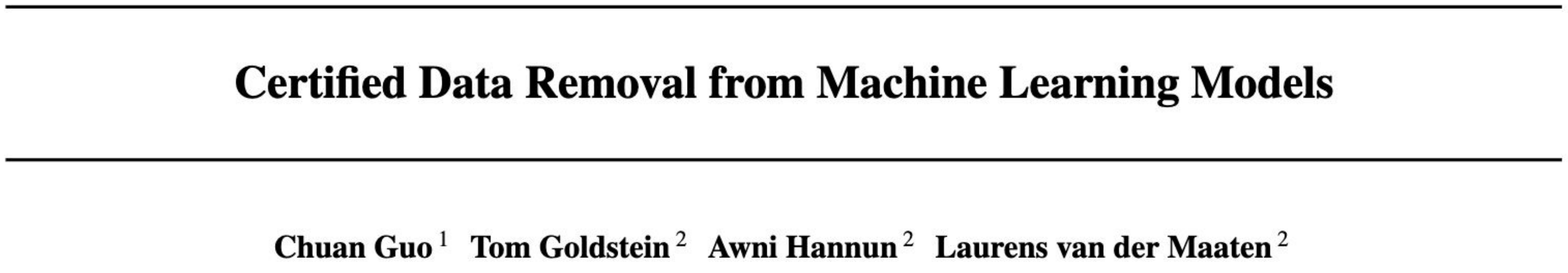
Yes but

- DP-SGD sacrifices utility
- We hope unlearning has a better trade-off
  - Provably the case (Sekhari et al. 2021)



De, Soham, et al. "*Unlocking high-accuracy differentially private image classification through scale.*" (2022).

Sekhari, Ayush, et al. "*Remember what you want to forget: Algorithms for machine unlearning.*" (2021)



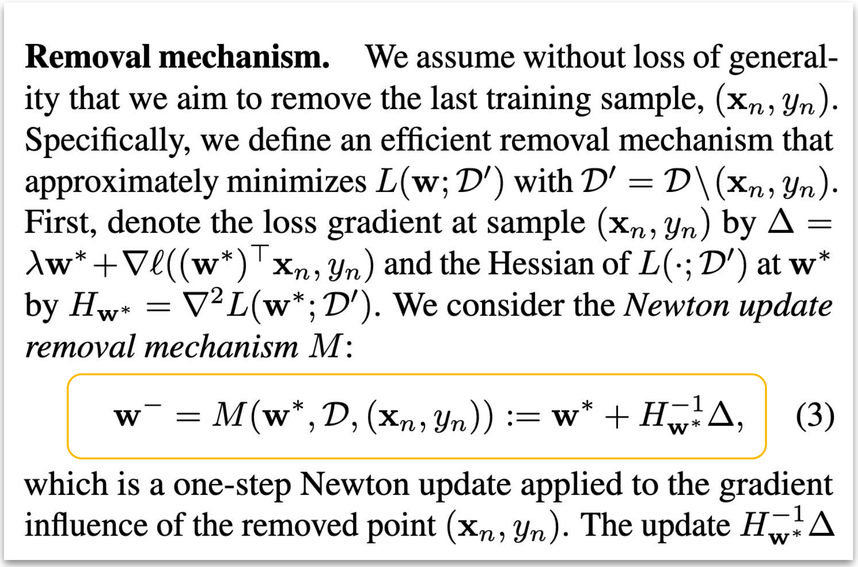
Certified Data Removal from Machine Learning Models

Chuan Guo [1] Tom Goldstein [2] Awni Hannun [2] Laurens van der Maaten [2]

✅ **Strong** unlearning guarantees

❌ **Applies** to strongly convex linear models
- Exact for least squares
- Iterative for logistic regression

❌ **Cost** solving a linear system



Removal mechanism. We assume without loss of generality that we aim to remove the last training sample, $(\mathbf{x}_n, y_n)$. Specifically, we define an efficient removal mechanism that approximately minimizes $L(\mathbf{w}; \mathcal{D}')$ with $\mathcal{D}' = \mathcal{D} \setminus (\mathbf{x}_n, y_n)$. First, denote the loss gradient at sample $(\mathbf{x}_n, y_n)$ by $\Delta = \lambda \mathbf{w}^* + \nabla \ell((\mathbf{w}^*)^\top \mathbf{x}_n, y_n)$ and the Hessian of $L(\cdot; \mathcal{D}')$ at $\mathbf{w}^*$ by $H_{\mathbf{w}^*} = \nabla^2 L(\mathbf{w}^*; \mathcal{D}')$. We consider the *Newton update removal mechanism* M:

$$\boxed{\mathbf{w}^- = M(\mathbf{w}^*, \mathcal{D}, (\mathbf{x}_n, y_n)) := \mathbf{w}^* + H_{\mathbf{w}^*}^{-1} \Delta,} \quad (3)$$

which is a one-step Newton update applied to the gradient influence of the removed point $(\mathbf{x}_n, y_n)$. The update $H_{\mathbf{w}^*}^{-1} \Delta$

Remember What You Want to Forget: Algorithms for Machine Unlearning

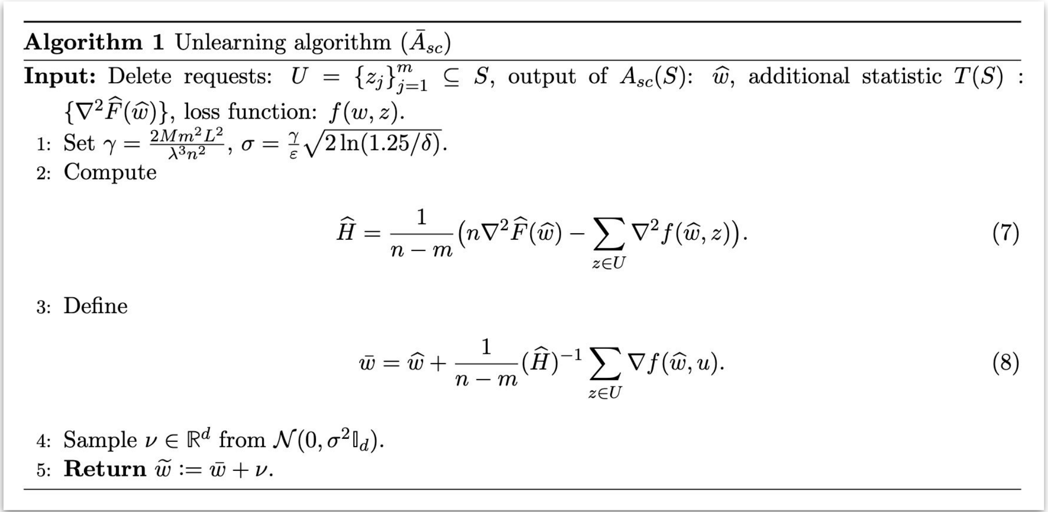
https://arxiv.org/abs/2103.03279

Ayush Sekhari[†] Jayadev Acharya[‡] Gautam Kamath[*] Ananda Theertha Suresh[§]

✅ Proves "strict separation between DP and machine unlearning"

❌ Applicable to **convex losses**

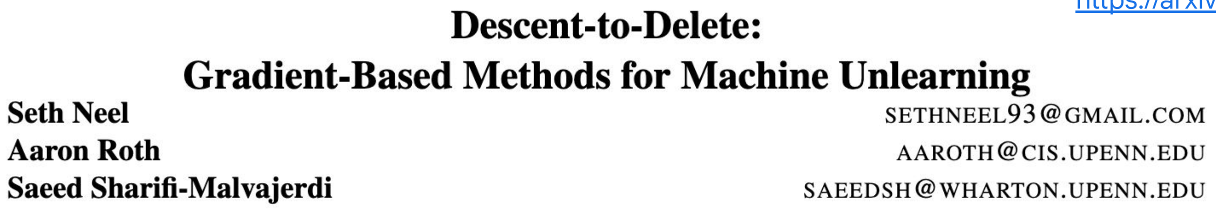
❌ Algorithm based on performing **Hessian inversion** + noise injection



Algorithm 1 Unlearning algorithm ($\bar{A}_{sc}$)

```
Input: Delete requests: $U = \{z_j\}_{j=1}^{m} \subseteq S$, output of $A_{sc}(S)$: $\widehat{w}$, additional statistic $T(S)$ : $\{\nabla^2 \widehat{F}(\widehat{w})\}$, loss function: $f(w, z)$.
1: Set $\gamma = \frac{2Mm^2L^2}{\lambda^3 n^2}$, $\sigma = \frac{\gamma}{\varepsilon}\sqrt{2\ln(1.25/\delta)}$.
2: Compute
     $$\widehat{H} = \frac{1}{n-m}\Big(n\nabla^2 \widehat{F}(\widehat{w}) - \sum_{z \in U} \nabla^2 f(\widehat{w}, z)\Big). \quad (7)$$
3: Define
     $$\bar{w} = \widehat{w} + \frac{1}{n-m}(\widehat{H})^{-1} \sum_{z \in U} \nabla f(\widehat{w}, u). \quad (8)$$
4: Sample $\nu \in \mathbb{R}^d$ from $\mathcal{N}(0, \sigma^2 \mathbb{I}_d)$.
5: Return $\widetilde{w} := \bar{w} + \nu$.
```

https://arxiv.org/abs/2007.02923

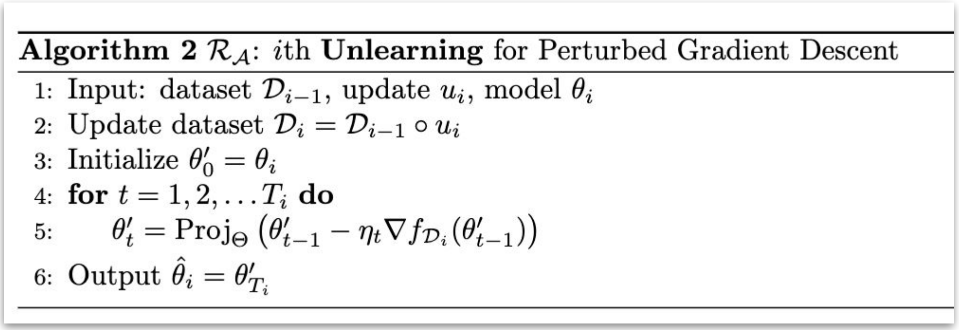


Descent-to-Delete: Gradient-Based Methods for Machine Unlearning

Seth Neel SETHNEEL93@GMAIL.COM
Aaron Roth AAROTH@CIS.UPENN.EDU
Saeed Sharifi-Malvajerdi SAEEDSH@WHARTON.UPENN.EDU

✅ **Scalable:** ~GD on retain set + noise

❌ Guarantees only applicable to convex objectives



Algorithm 2 $\mathcal{R}_{\mathcal{A}}$: ith **Unlearning** for Perturbed Gradient Descent

```
1: Input: dataset $\mathcal{D}_{i-1}$, update $u_i$, model $\theta_i$
2: Update dataset $\mathcal{D}_i = \mathcal{D}_{i-1} \circ u_i$
3: Initialize $\theta'_0 = \theta_i$
4: for $t = 1, 2, \ldots T_i$ do
5:     $\theta'_t = \text{Proj}_\Theta\left(\theta'_{t-1} - \eta_t \nabla f_{\mathcal{D}_i}(\theta'_{t-1})\right)$
6: Output $\hat{\theta}_i = \theta'_{T_i}$
```

Existing approaches

❌ Apply to convex objectives

and/or

❌ Computationally costly

Need for unlearning algorithm

✅ Scalable

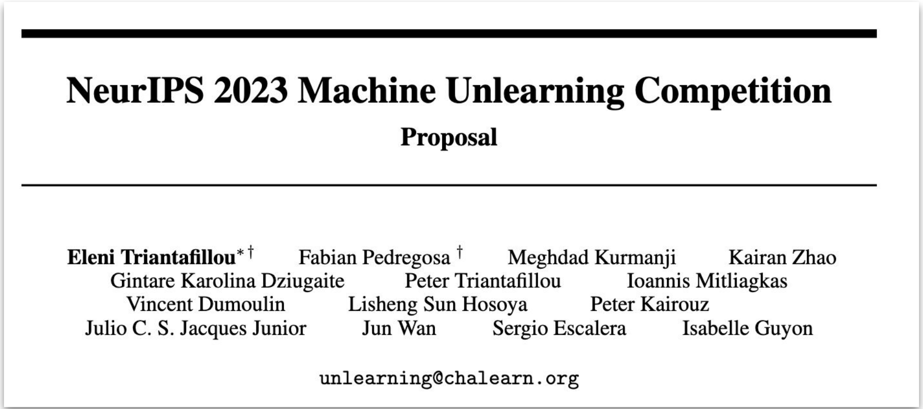
✅ Applicable to non-convex objectives

✅ Doesn't sacrifice (too much) utility

The 2023 NeurIPS Unlearning Challenge

NEURAL INFORMATION PROCESSING SYSTEMS

2



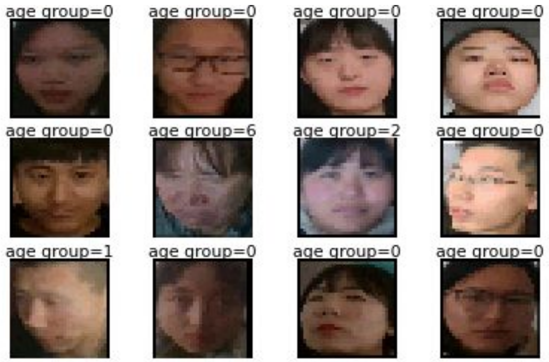
NeurIPS 2023 Machine Unlearning Competition

Proposal

Eleni Triantafillou[*†] Fabian Pedregosa[†] Meghdad Kurmanji Kairan Zhao Gintare Karolina Dziugaite Peter Triantafillou Ioannis Mitliagkas Vincent Dumoulin Lisheng Sun Hosoya Peter Kairouz Julio C. S. Jacques Junior Jun Wan Sergio Escalera Isabelle Guyon

unlearning@chalearn.org

- Accepted proposal to organize the first unlearning competition
- Decided on a dataset:
  - CASIA-SURF
  - Faces annotated with age groups



Zhang, Shifeng, et al. "Casia-surf: A large-scale multi-modal benchmark for face anti-spoofing." *IEEE Transactions on Biometrics, Behavior, and Identity Science* 2.2 (2020). https://arxiv.org/abs/1908.10654

Evaluating unlearning

Efficiency

Time (seconds)
FLOPs

Absolute or relative to retrain

↓

⚙️ **Threshold**: entries that take more than 10% of total training time are eliminated

Utility

Accuracy of the model

↓

⚙️ **Accuracy on retain/test set**

Forgetting quality

How well have we forgotten?

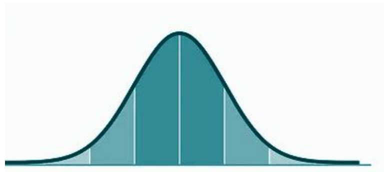
↓

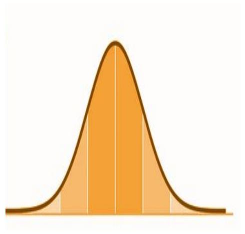
⚙️ 😱😱😱😱

Evaluating unlearning

Goal: evaluate ϵ in

$$e^{-\epsilon} \leq \frac{P(U(A(\mathcal{D}), \mathcal{D}, \mathbf{x}) \in \mathcal{T})}{P(A(\mathcal{D} \setminus \mathbf{x}) \in \mathcal{T})} \leq e^{\epsilon}.$$

- A(D\x) = 
training without forget set

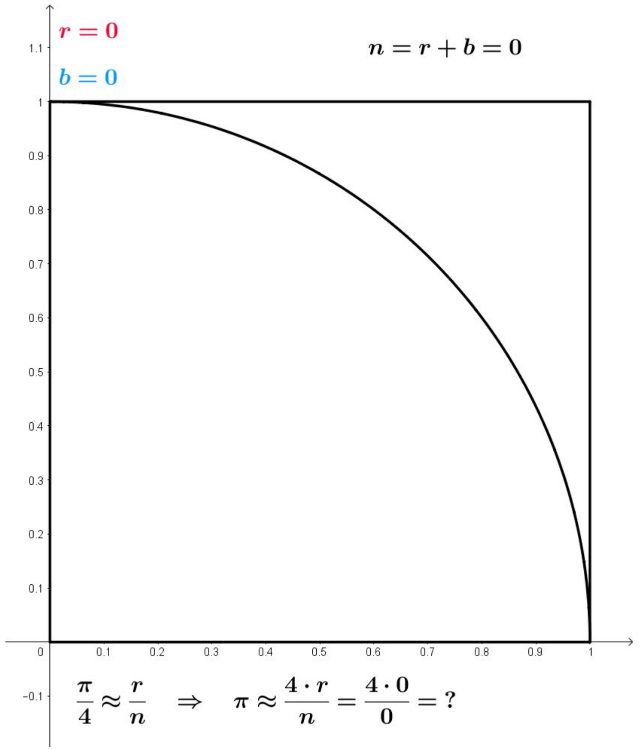
- U(A(D), D, x) = 
unlearning algorithm

Typically bounding ϵ is a theoretical contribution

- Can we ask participants to provide such a bound?
- How can we evaluate soundness of the this derivation?

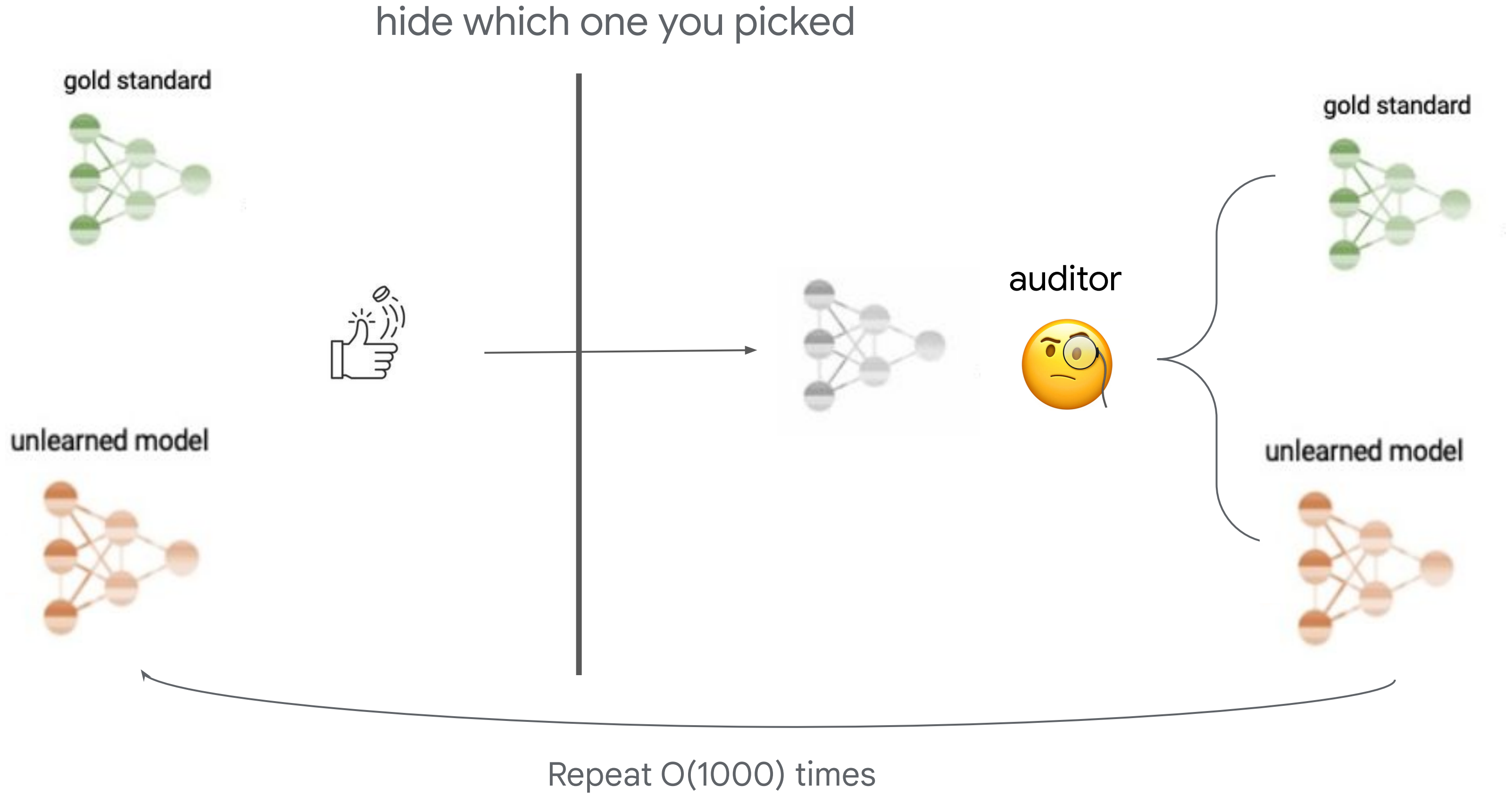
Monte carlo methods

Estimating complex probabilities can sometimes be achieved by running a large number of experiments

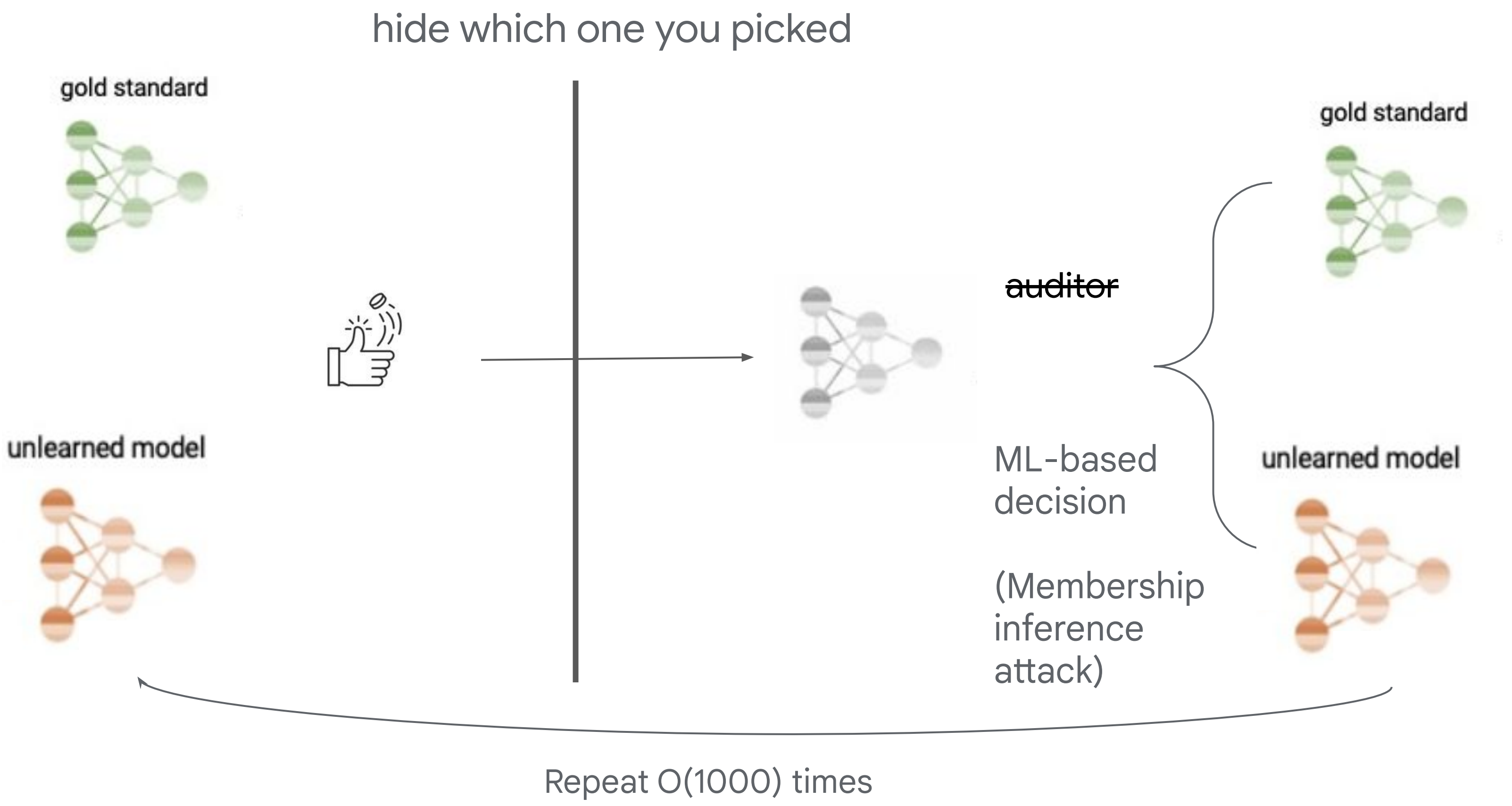

$r = 0$
$b = 0$
$n = r + b = 0$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
-0.1
$\frac{\pi}{4} \approx \frac{r}{n} \Rightarrow \pi \approx \frac{4 \cdot r}{n} = \frac{4 \cdot 0}{0} = ?$

https://en.wikipedia.org/wiki/Monte_Carlo_method

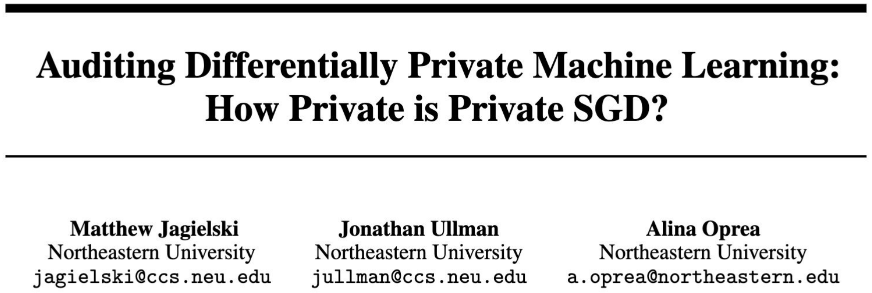
Unlearning as hypothesis testing



Unlearning as hypothesis testing



Inspiration from the differential privacy literature

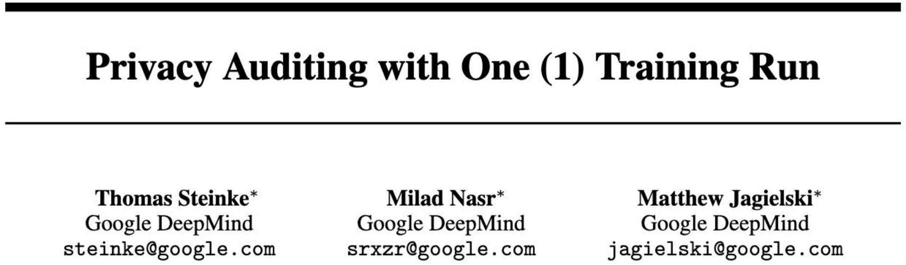


Auditing Differentially Private Machine Learning: How Private is Private SGD?

Matthew Jagielski Northeastern University `jagielski@ccs.neu.edu`

Jonathan Ullman Northeastern University `jullman@ccs.neu.edu`

Alina Oprea Northeastern University `a.oprea@northeastern.edu`

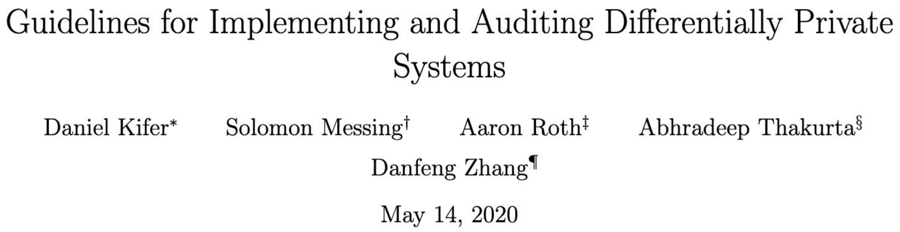


Privacy Auditing with One (1) Training Run

Thomas Steinke[*] Google DeepMind `steinke@google.com`

Milad Nasr[*] Google DeepMind `srxzr@google.com`

Matthew Jagielski[*] Google DeepMind `jagielski@google.com`



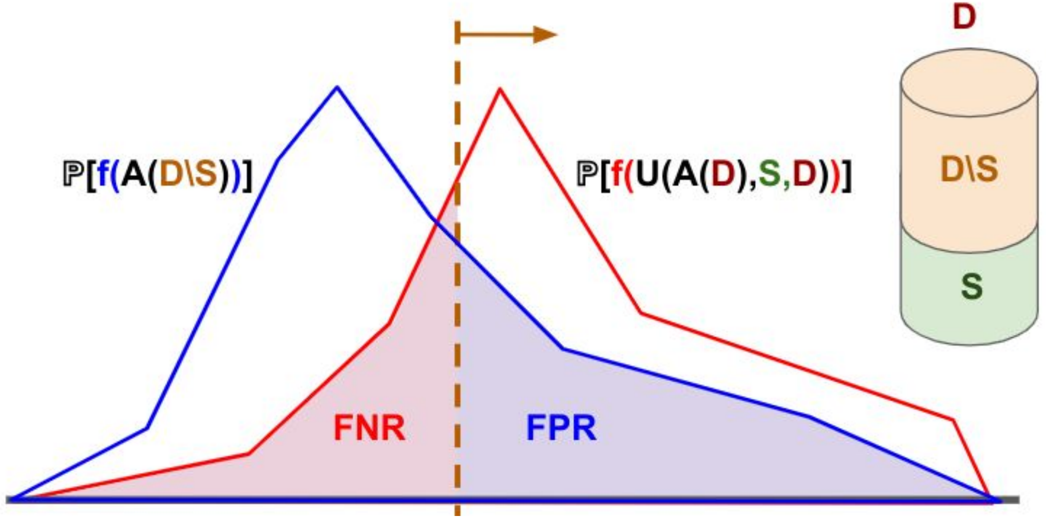
Guidelines for Implementing and Auditing Differentially Private Systems

Daniel Kifer[*] Solomon Messing[†] Aaron Roth[‡] Abhradeep Thakurta[§]
Danfeng Zhang[¶]

May 14, 2020

Simplifications for efficiency

- We estimate the distributions of retrained / unlearned outputs **for each example**
- Instead of considering weight space, we consider distributions of (scalar) outputs when receiving 'forget examples' as input
- We run many 'attacks', compute accuracy of and keep the worse (largest) ϵ



After having computed each example's ϵ, we aggregate them via a bucketing procedure

Evaluating unlearning

Efficiency

Time (seconds)
FLOPs

Absolute or relative to retrain

Threshold: entries that take more than 10% of total training time are eliminated

X

Utility

Accuracy of the model

Accuracy on retain/test set

X

Forgetting quality

How well have we forgotten?

Monte-Carlo approach

Unusual competition

k Standard Kaggle competition

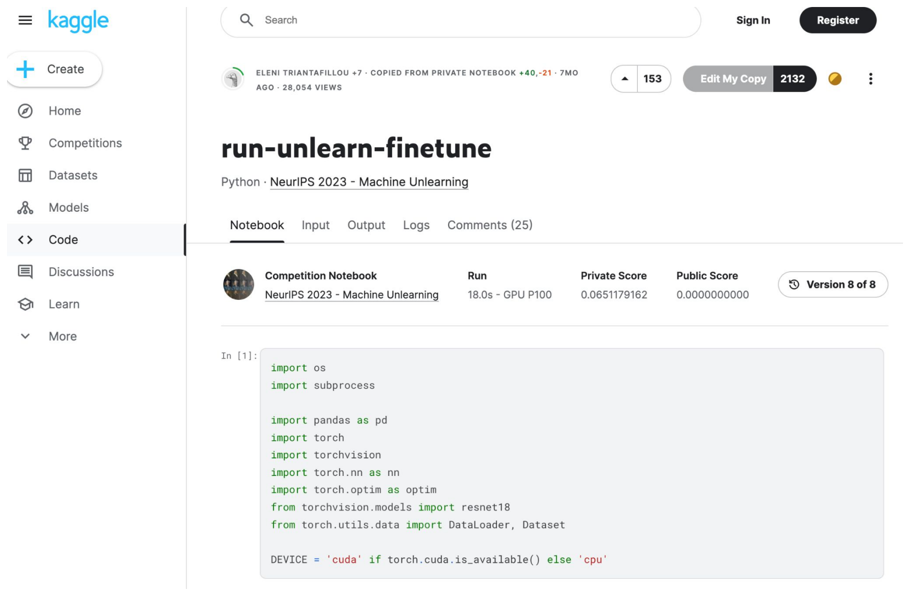
- Download dataset
- Submit labels

This one was **very different**. Participants had

- No access to dataset
  - Can't run locally

- No implementation to evaluation (only rough description)
  - Avoid overfitting to metric

kaggle



ELENI TRIANTAFILLOU +7 · COPIED FROM PRIVATE NOTEBOOK +40,-21 · 7MO AGO · 28,054 VIEWS

run-unlearn-finetune

Python · NeurIPS 2023 - Machine Unlearning

Notebook Input Output Logs Comments (25)

Competition Notebook	Run	Private Score	Public Score	
NeurIPS 2023 - Machine Unlearning	18.0s - GPU P100	0.0651179162	0.0000000000	Version 8 of 8

In [1]:
```
import os
import subprocess

import pandas as pd
import torch
import torchvision
import torch.nn as nn
import torch.optim as optim
from torchvision.models import resnet18
from torch.utils.data import DataLoader, Dataset

DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'
```

We launched on September 2023

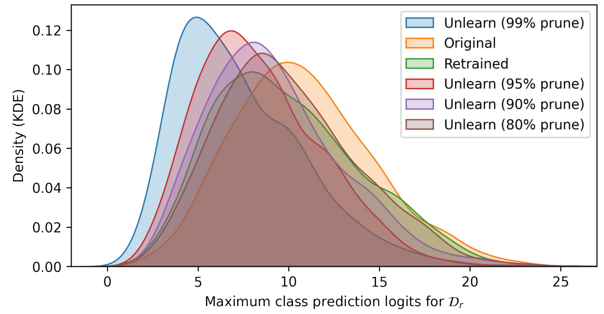
3-months after schedule

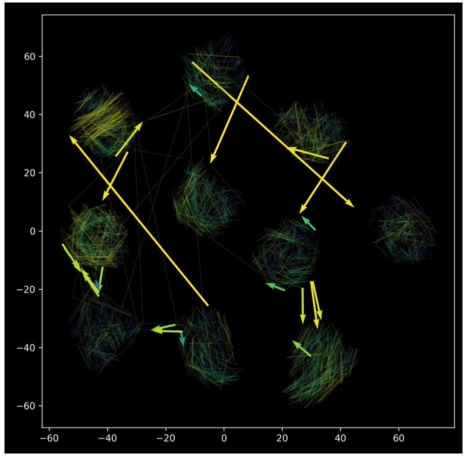
- Baseline of "fine-tuning" (training for few epochs on "retain set")
-

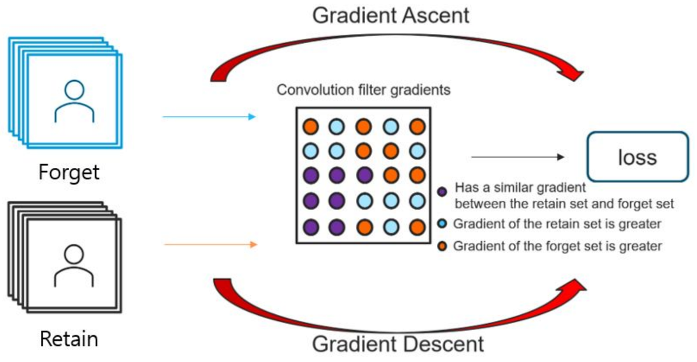
Nightmare scenarios

- Problem is too hard: nobody does better than baseline (happened to others)
- Participants find a "backdoor" in the evaluation, manage to win without really unlearning

A look at the top submissions







Gradient Ascent
Convolution filter gradients
Forget
Retain
loss
Has a similar gradient between the retain set and forget set
Gradient of the retain set is greater
Gradient of the forget set is greater
Gradient Descent

3

In numbers

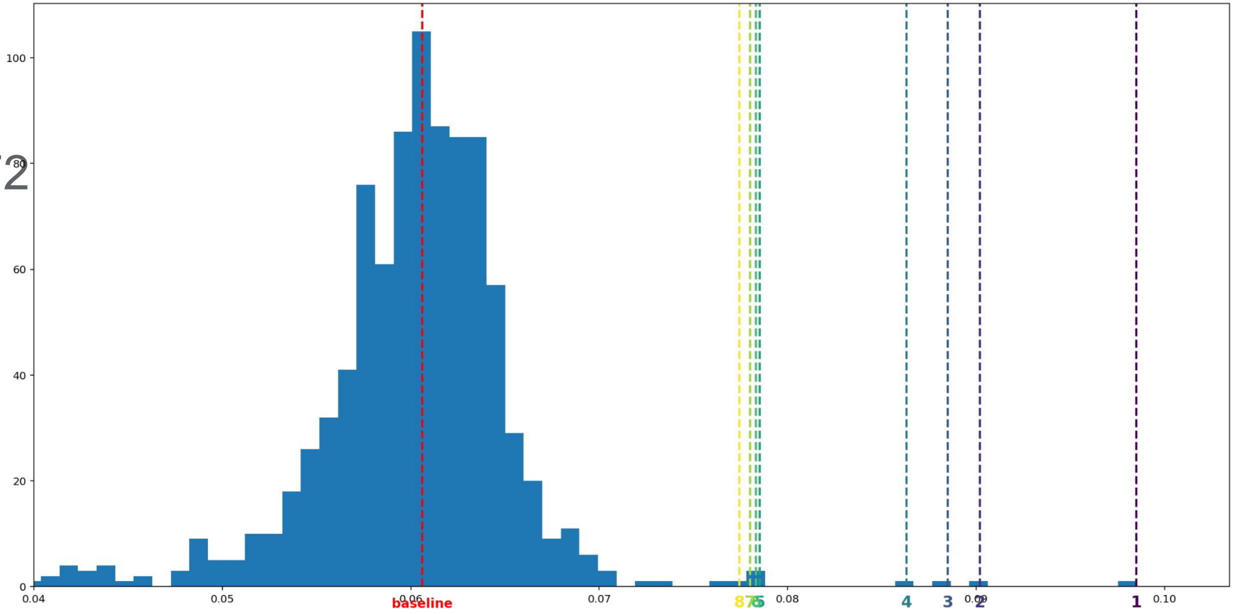
5,161 registrations

1,338 participants from 72 countries.
For 500 (including 44 in the top 100!), this was their first competition

1,121 teams

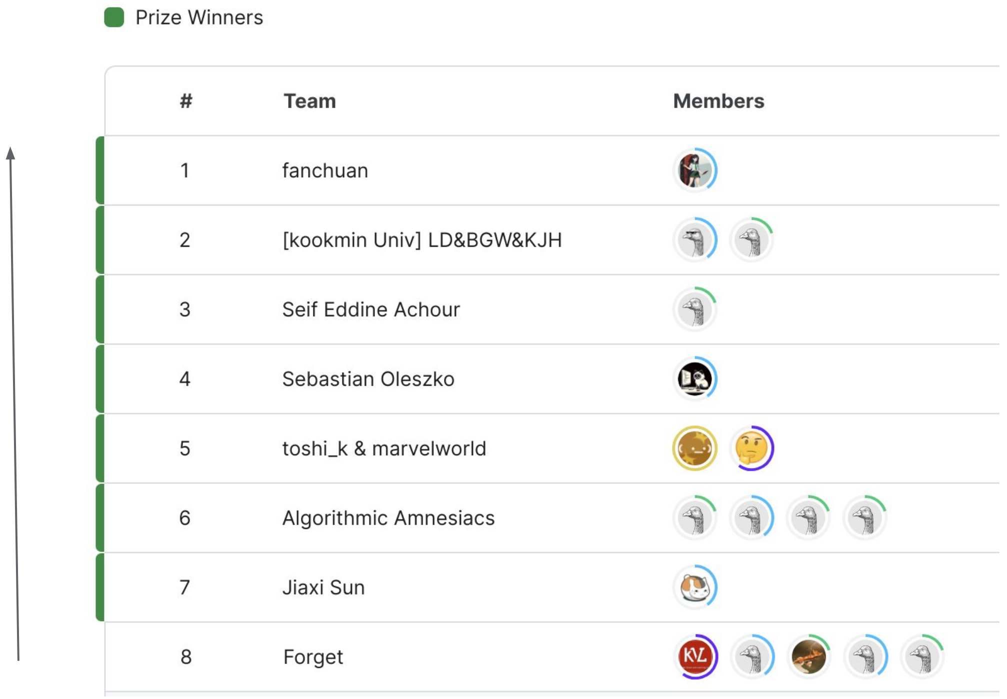
1,923 submissions

🤯



Leaderboard: 40% scored above baseline

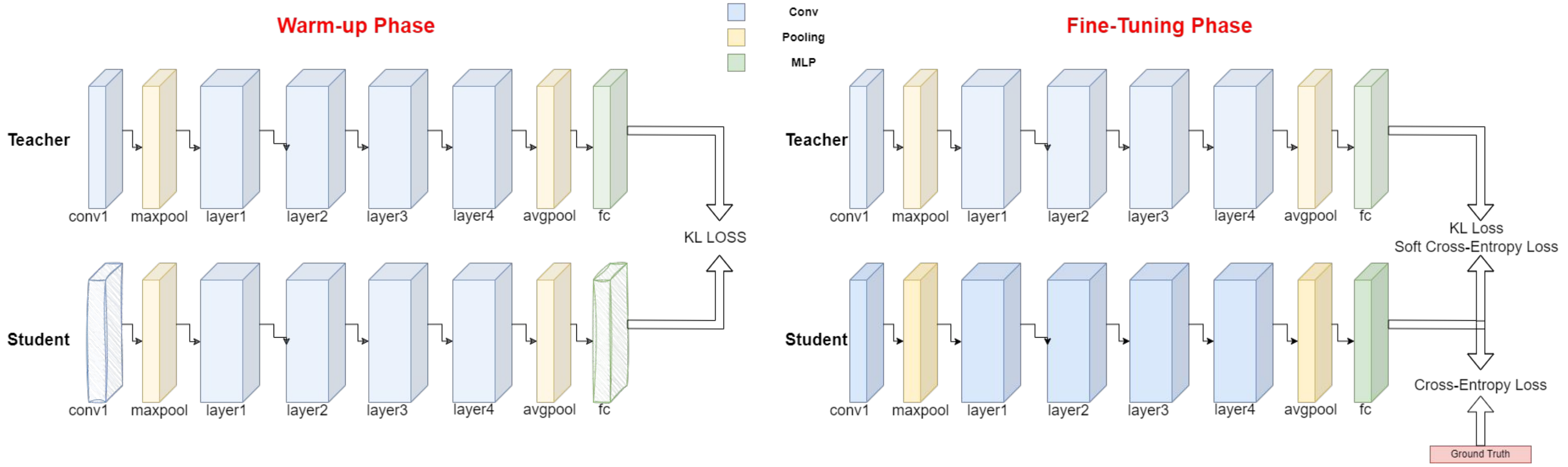
Top submissions



🟩 Prize Winners

#	Team	Members
1	fanchuan	
2	[kookmin Univ] LD&BGW&KJH	
3	Seif Eddine Achour	
4	Sebastian Oleszko	
5	toshi_k & marvelworld	
6	Algorithmic Amnesiacs	
7	Jiaxi Sun	
8	Forget	

6th place solution - Algorithmic Amnesiacs



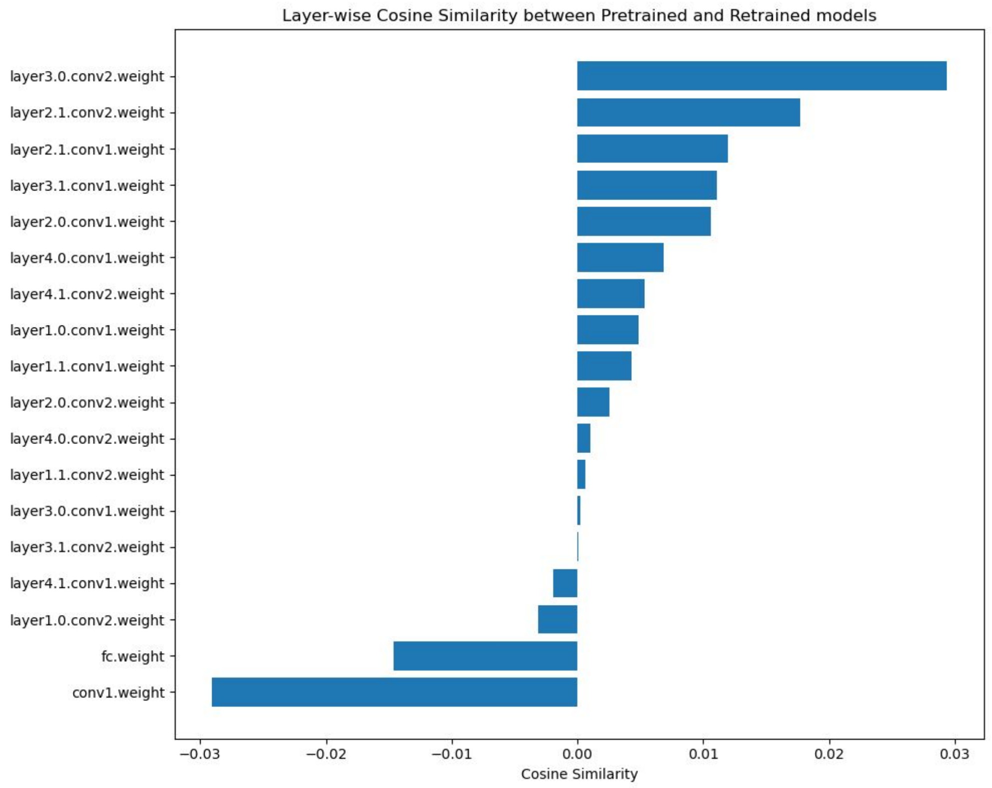
1. Reset first and last layer of the original model.
2. Warm-up phase employing knowledge distillation
3. Fine-tuning phase.

6th place solution - Algorithmic Amnesiacs

Reset first and last layers:

- First layer significantly influences the rest of the model layers and the last layer determines the model's final output distribution.

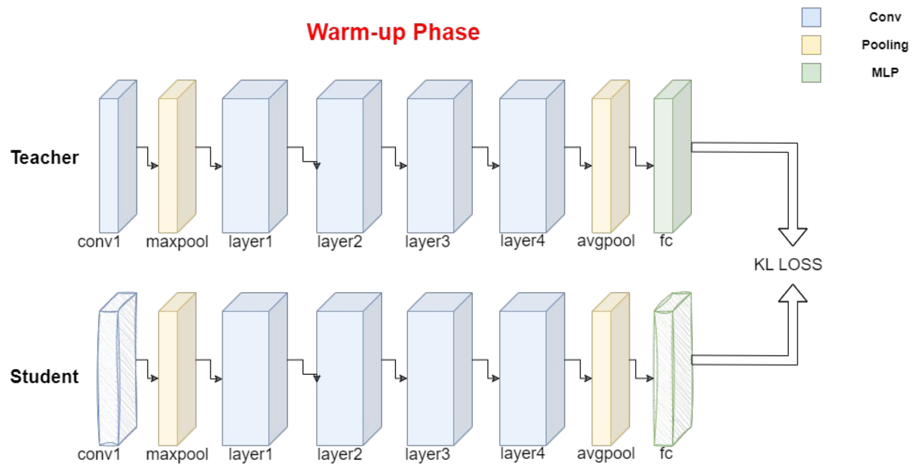
- On CIFAR-10: these two layers exhibited the most negative cosine similarity between model weights trained on the full training set and models trained from scratch on a smaller subset (i.e., the retain set).



6th place solution - Algorithmic Amnesiacs

Warm-up phase

Minimize KL divergence between the outputs of the original pre-trained model (teacher) and the reinitialized model (student) **on the validation set**.



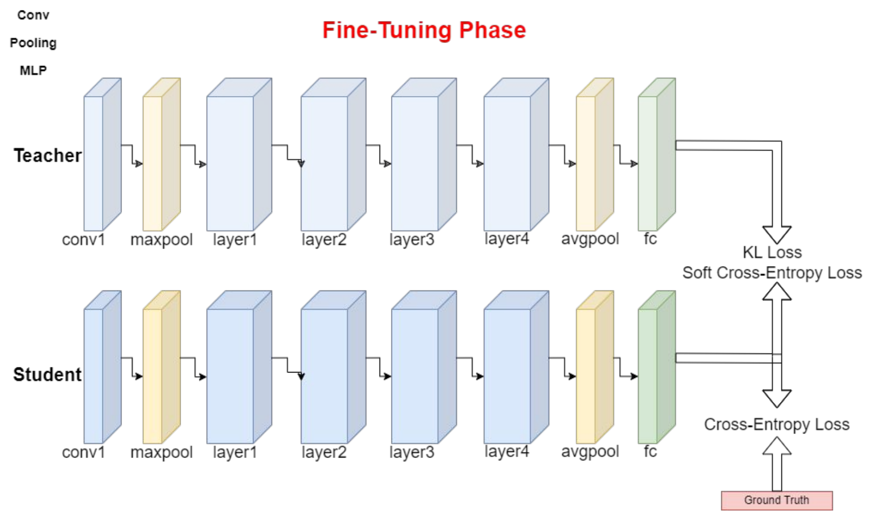
6th place solution - Algorithmic Amnesiacs

Fine-tuning using 3 losses

Cross-entropy for model's accuracy using hard labels on retain set.

Soft cross-entropy for predictions of the student model with soft labels from the teacher model.

KL divergence combined with the soft cross-entropy facilitates rapid knowledge transfer and broader information capture.



Common trends

All submissions followed a strategy of two stages: **Forget** and **fine-tuning**

Forgetting

Random reinitialization / untargeted

- 8th: model parameters are stochastically selected and re-initialized.
- 7th: reset last layer + add noise to N=9 (randomly chosen) layers
- 6th: reset first and last layer
- 5th: permute weights

Optimization-based forgetting (lack of a better name ...)

- 4th: prune weights based on L1 norm
- 2nd: difference of gradients
- 1st: minimize KL-divergence between output logits and a uniform pseudo label on forget set. Also, there's a "forget round: Maximize dissimilarity between logits of forget and retain set"

Fine-tuning

7th: standard fine-tuning

6th: knowledge distillation + fine-tuning + uses 3 losses (the sum)

5th: pseudo-labels

4th: regularize with entropy

2nd: standard fine-tuning but with a very small learning rate (1/10th of original)

1st: standard fine-tuning

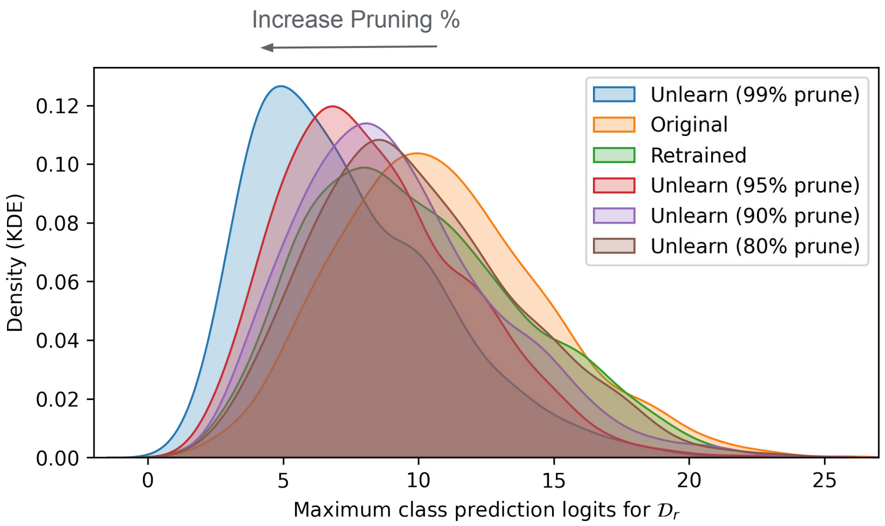
4th place solution - Sebastian Oleszko

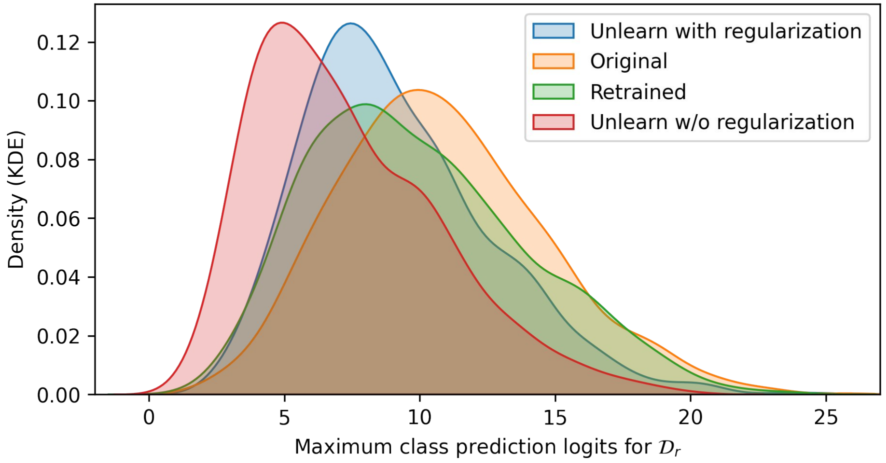
1. Re-initializing/pruning 99% of parameters based on L1-norm (Unstructured)
   - Weights: Pytorch default initialization
   - Biases: Set to zero (prune)
2. Fine-tune on retain dataset
   - Regularize using entropy
   - Cross entropy class weights as $N_c^{-0.1}$

$$\min_{w} \underbrace{\sum_{(x_r, y_r)\in\mathcal{D}_r} H(y_r, f(x_r; w))}_{\text{Cross-entropy}} + \underbrace{\sum_{x_r\in\mathcal{D}_r} \left(H(f(x_r; w)) - H(f(x_r; \overset{\substack{\text{Initial weights}\\ \downarrow}}{w^o}))\right)^2}_{\text{MSE of entropy}}$$

4th place solution - Sebastian Oleszko

- CIFAR-10 experiment
- Impact of most important hyperparameters: Learning rate/epochs and pruning percentage
- Effect of including entropy regularization
- Tuning on public submission scores





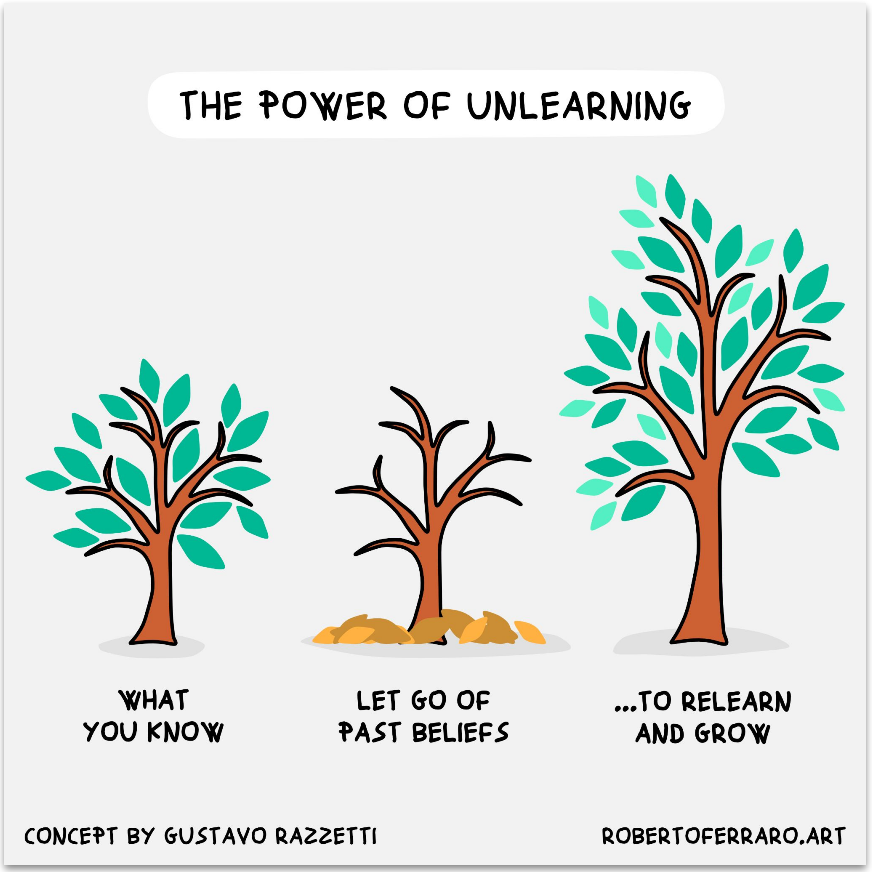
Conclusion

🔐Machine Unlearning is **essential** to safely deploy AI systems at scale

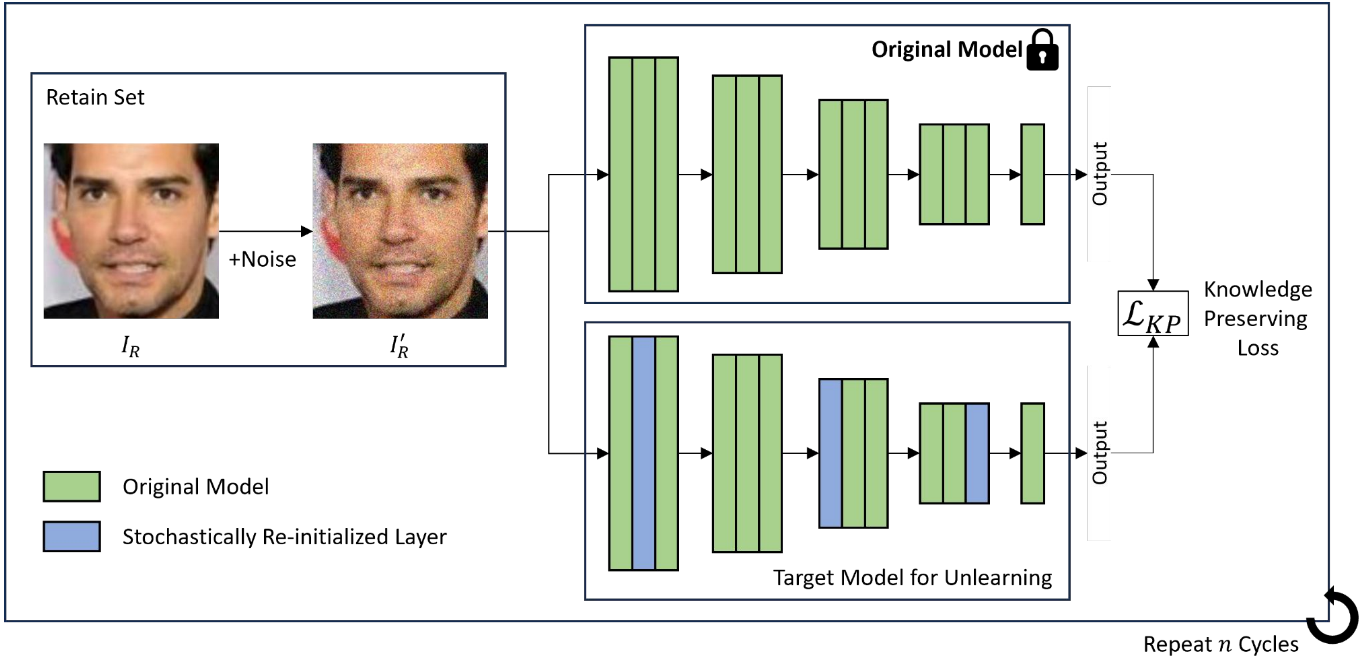
📚 There are rigorous definitions of machine unlearning

💻 Theoretical notions, but which can be approximated with computational methods

🏆 Organizing a machine learning challenge is hard – but also fun!



8th place solution - Team Forget



(1) Forgetting phase: model parameters are stochastically selected and re-initialized.
- FC, Projection-shortcut layers are excluded from the selection pool.

(2) Remembering phase: knowledge preserving loss is calculated between the original model and the target unlearning model.
- Knowledge Preserving Loss:

$$\mathbb{E}\{|f_O(\mathbf{I}'_R) - f_U(\mathbf{I}'_R)|^2\}$$

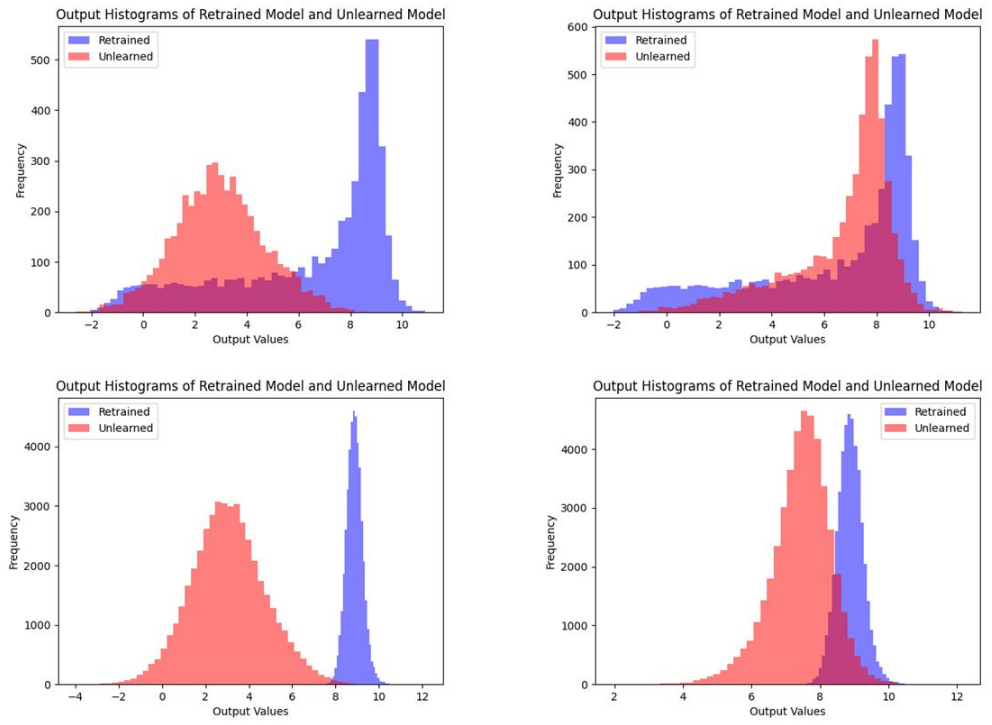
- Gaussian noise is added to the image as data augmentation.
- It reminds the target model about the retain set.

(3) Forgetting phase and Remembering phase are repeated for n cycles to enhance unlearning performance.

8th place solution - Team Forget

Forget Set

Retain Set



(a) CE

(b) MSE

Figure. Comparison of logit distributions between CE loss and MSE loss

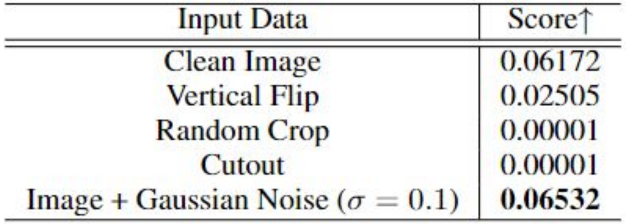
- Histograms of logits from retrained model and unlearned models are visualized.

- This observation is acquired from local experiments on CIFAR-10.

- MSE loss makes closer distributions than CE loss for both forget set and retain set.

8th place solution - Team Forget

Table 1. Comparison between different data augmentation techniques



Input Data	Score↑
Clean Image	0.06172
Vertical Flip	0.02505
Random Crop	0.00001
Cutout	0.00001
Image + Gaussian Noise ($\sigma = 0.1$)	**0.06532**

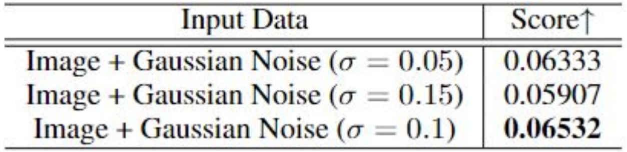
- Gaussian Noise (σ=0.1) is the best data augmentation compared with other data augmentation techniques.

- Compared with CE loss and L1 loss, MSE loss demonstrates the best score.

- Additionally, increasing the cycles highly improves the performance.

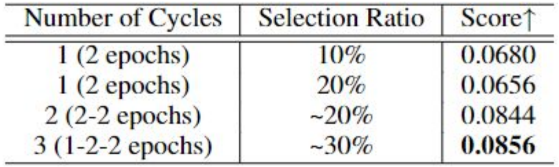
- From these observations, we build the final submission.

Table 2. Comparison between different sigma of gaussian noise



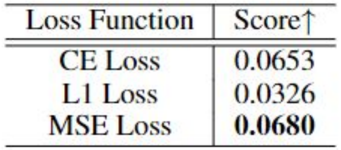
Input Data	Score↑
Image + Gaussian Noise ($\sigma = 0.05$)	0.06333
Image + Gaussian Noise ($\sigma = 0.15$)	0.05907
Image + Gaussian Noise ($\sigma = 0.1$)	**0.06532**

Table 4. Effect of cycles



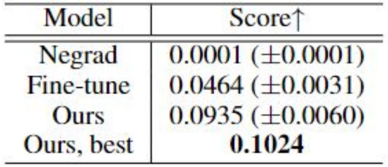
Number of Cycles	Selection Ratio	Score↑
1 (2 epochs)	10%	0.0680
1 (2 epochs)	20%	0.0656
2 (2-2 epochs)	~20%	0.0844
3 (1-2-2 epochs)	~30%	**0.0856**

Table 3. Comparison between different loss functions



Loss Function	Score↑
CE Loss	0.0653
L1 Loss	0.0326
MSE Loss	**0.0680**

Table 5. Final submission score compared with other unlearning methods



Model	Score↑
Negrad	0.0001 (±0.0001)
Fine-tune	0.0464 (±0.0031)
Ours	0.0935 (±0.0060)
Ours, best	**0.1024**

7th place solution - Jiaxi Sun

Solution that only makes use of retain set

1. Reset parameters of last layer

2. Randomly selecting N=9 layers from the network and add noise
   a. Adding noise helps the network 'forget' the information it has learned, and the randomness of the layer selection contributes to enhancing the model's diversity.

3. Fine-tune all network layers

5th place solution - toshi_k & marvelworld

Summary

Our solution is the ensemble of two approaches:
(1) Retraining from transposed weights
(2) Fine-tuning with pseudo-labels.

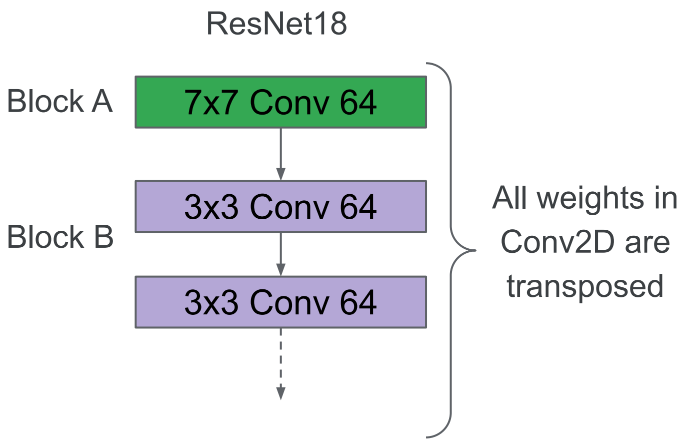
Our solution is built upon two distinctive approaches, contributing to the stability of our solution in the private LB.

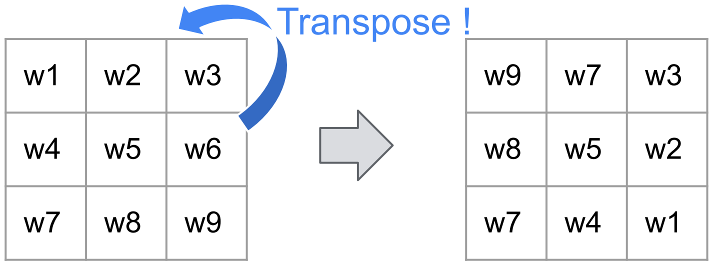
(1) Retraining from transposed weights	(2) Fine-tune with pseudo-labels	Public LB	Private LB
512 models	0 models	0.0720386947	-
0 models	512 models	0.0707241647	-
246 models	266 models	-	**0.0785184178**
266 models	246 models	-	0.0756313425

5th place solution - toshi_k & marvelworld

(1) Retraining from transposed weights

- This part retrains the model using a modified version of the original model.
- In this modification, all weights in Conv2D are transposed. This process helps in forgetting samples in the forget-set, enabling the reuse of valuable features from the original model.





```
for module in local_model.modules():
    if isinstance(module, torch.nn.modules.conv.Conv2d):
        module.weight = torch.nn.Parameter(module.weight.swapaxes(2, 3))
```

The modification is carried out simply like this.

5th place solution - toshi_k & marvelworld

(2) Fine-tuning with pseudo-labels

- This part reproduces behavior (errors) on the forget data with pseudo-labels from two functions.

```
Algorithm 1 Unlearning with pseudo-labels
Input: forget data : $(x_f, y_f) \in D_F$
       retrain data : $(x_r, y_r) \in D_R$
       pretrained model : $f_\theta^p$
       simple finetuning model : $f_\theta^l = f_\theta^p(x_r) \rightarrow \hat{y}^{pr} \rightarrow Loss(\hat{y}^{pr}, y^r)$
       simple scratch model : $f_\theta^s = f_\theta(x_r) \rightarrow \hat{y}^r \rightarrow Loss(\hat{y}^r, y^r)$
       confidence threshold : $T$

function INCORRECT DIRECTION PSEUDO-LABELS
    for $x_f \in D_F$ do
        if $f_\theta^p(x_f) = y^f$ and $f_\theta^p(x_f) \neq f_\theta^l(x_f)$ then
            $D_P \ni \{x_f, \hat{y}^{l^f}\}$
        end if
    end for
    return $D_P$
end function

function HIGH CONFIDENCE INCORRECT PSEUDO-LABELS
    for $x_f \in D_F$ do
        if $Entropy(f_\theta^s(x_f)) < T$ and $f_\theta^s(x_f) \neq y^f$ then
            $D_P \ni \{x_f, \hat{y}^{s^f}\}$
        end if
    end for
    return $D_P$
end function
```

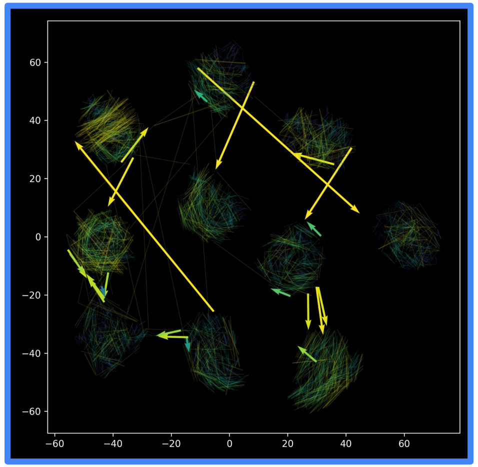


Figure 1: Inference shifts between the pretrained model and the fine-tuned model, showcasing significant shifts in the incorrect direction.

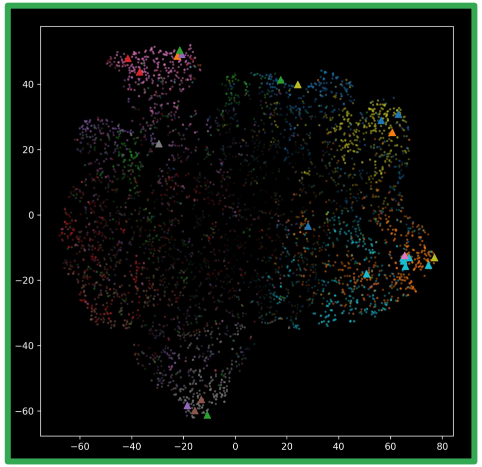


Figure 2: Inference results of the scratch model. Triangle-up marker indicates high confidence but they are making incorrect inferences.

4th place solution - Sebastian Oleszko

Entropy-based regularization

Helps to achieve a more similar prediction distribution/confidence.

Unlearning through pruning/re-initialization

Effective as unlearning technique. Most of the performance is retainable even with high pruning percentage.

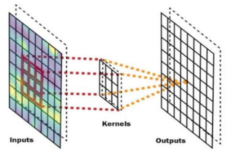
Concluding thoughts

- Hyperparameter tuning is **very** important to achieve high scores
- Final submission was only fine-tuned for 3.2 epochs - maybe not optimal

3rd place solution - Seif Eddine Achour

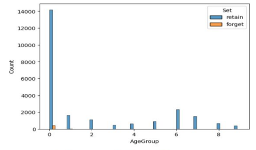
Competition approach: vision confusion - reconstitution

Vision Confusion



Details forgotten but the general idea was retained

Class Imbalance

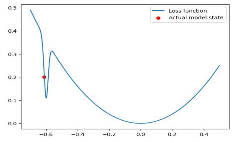


(b) Retain and forget histograms

Weighted cross entropy

Vision reconstitution on the Retain Set through 3 epochs

Hard Differentiability



Slight confusion & a final stabilisating epoch

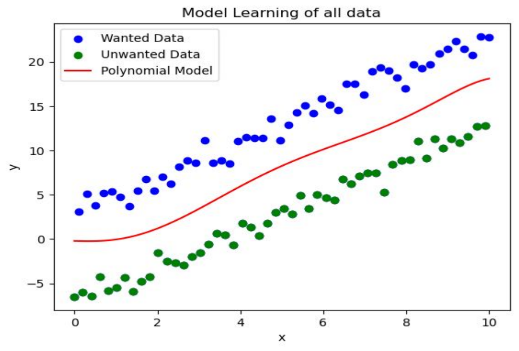
Model Unlearned!

The confusion process is instant. The whole computation is dedicated to the vision reconstitution (Time Efficient).

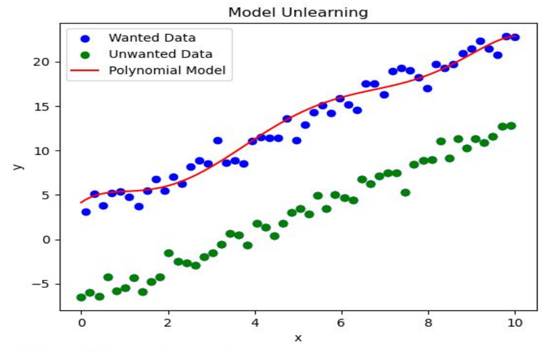
3rd place solution - Seif Eddine Achour

Paper approach: Loss landscape adjuster

Regression



```
R-squared wanted: 0.28077359745782393
```



```
R-squared wanted: 0.9653162393871904
```

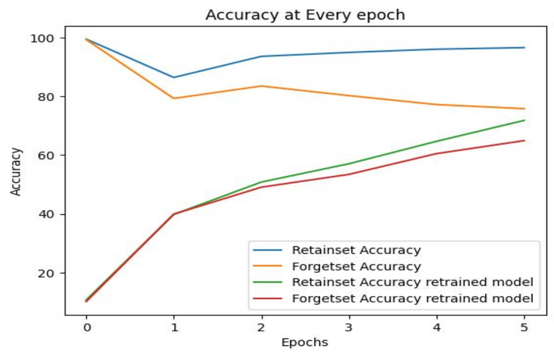
The model forgot totally about the unwanted data despite its big size

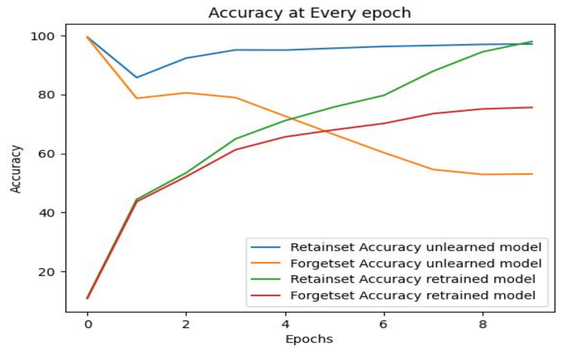
Original r-squared = 0.28 vs Unlearned r-squared = 0.97!

3rd place solution - Seif Eddine Achour

Paper approach: Loss landscape adjuster

Classification





- **Good results without considering the Forget Set (1st approach)**

➡ **The proper use of the Forget Set will certainly improve results**

- **The retrained model is not always the ideal one**

➡ **-1% accuracy on Retain Set lead to -24% of accuracy on Forget Set**

➡ **The metric which check the similitude between the unlearned and retrained model is not that representative for the unlearning performance**

2nd place solution - [kookmin Univ] LD&BGW&KJH

Gradient-based re-initialization method
We assumed that if the gradients of the weights in the model, specifically in the retain set and forget set, are similar, it becomes challenging to forget information from the forget set during the retraining of the retain set.

proposed gradient-based re-initialization method for unlearning consists of three main steps:

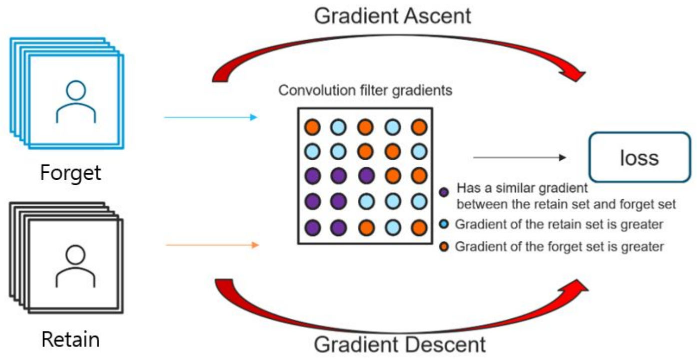
1. **Gradient Collection**:
   Gradient information is collected from the forget set and the retain set.

2. **Weight initialization**:
   Based on the gradient information collected in the first step, a percentage of the convolution filter weights are re-initialized.

3. **Retraining**:
   The model is retrained with the retain set. The learning rate for the Uninitialized weights uses 1/10 of the base learning rates.

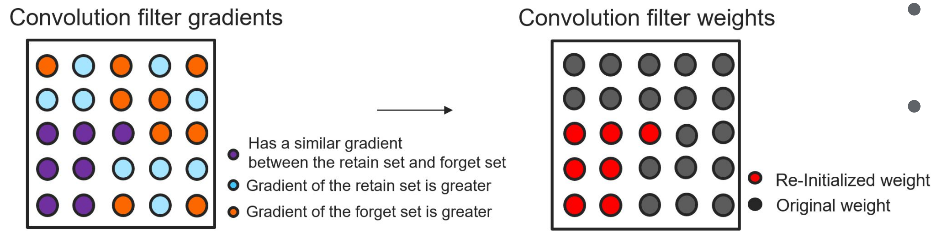
2nd place solution - [kookmin Univ] LD&BGW&KJH

1. Gradient Collection


Gradient Ascent
Convolution filter gradients
Forget
Retain
loss
Has a similar gradient between the retain set and forget set
Gradient of the retain set is greater
Gradient of the forget set is greater
Gradient Descent

- Collect gradients of forget set using gradient ascent
- Collect gradients of retain set using gradient descent
- Random sampling was used from the retain set to match the number of samples in the forget set
- In short, this is simply subtracting forget set's gradient from the retain set's gradient

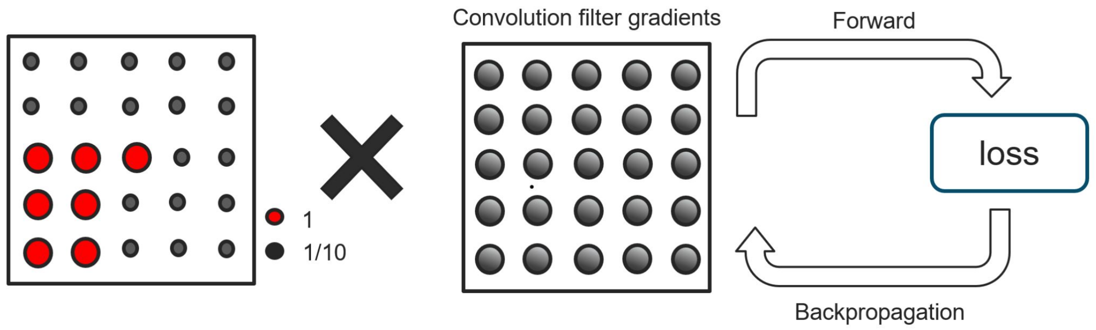
2. Weight initialization


Convolution filter gradients
Has a similar gradient between the retain set and forget set
Gradient of the retain set is greater
Gradient of the forget set is greater
Convolution filter weights
Re-Initialized weight
Original weight

- Based on the gradient information a percentage of the convolution filter weights are re-initialized
- Our best method re-initialized 30% of convolution filter weights

2nd place solution - [kookmin Univ] LD&BGW&KJH

3. Retraining



- Re-initialized Model is trained using the retain set
- Learning rate for the uninitialized weights uses 1/10 of the base learning rates (accomplished by scaling the gradient of uninitialized weight)
- Used a linear decay learning rate scheduler with a few warmup epoch
    - Consistently produces better results than other learning rate schedulers
    - Used warmup epoch of 3
      (0.00033 to 0.001 in the first 3 epochs, and then linearly decreases from 0.001 to 0.00033 in the last 2 epochs)

1st place solution - fanchuan

1. **Forget phase**: minimize KL-divergence between output logits and a uniform pseudo label on forget set.
2. **Adversarial fine-tuning phase**. Alternate between "forget" and "retain" rounds:

Forget round: Maximize dissimilarity between logits of forget and retain set

$$l_i = -\frac{1}{bactshsize2} \sum_{t=0}^{batch_2} log(\frac{e^{sim(x_i,y_t)/\tau}}{\sum_{j=0}^{batch_2} e^{sim(x_i,y_j)/\tau}})$$

$$L_{forget} = \frac{1}{batchsize_1} \sum_{i=0}^{batch_1} l_i$$

Retain: original loss (cross entropy) on retain set.

Trick: Increase batch size from 64 to 258 to be able to perform more epochs (6 -> 8)

Conclusion

4

Thank you.

Google DeepMind

Thank you.

Firstname Lastname
Title

Firstname Lastname
Title

Firstname Lastname
Title

Firstname Lastname
Title

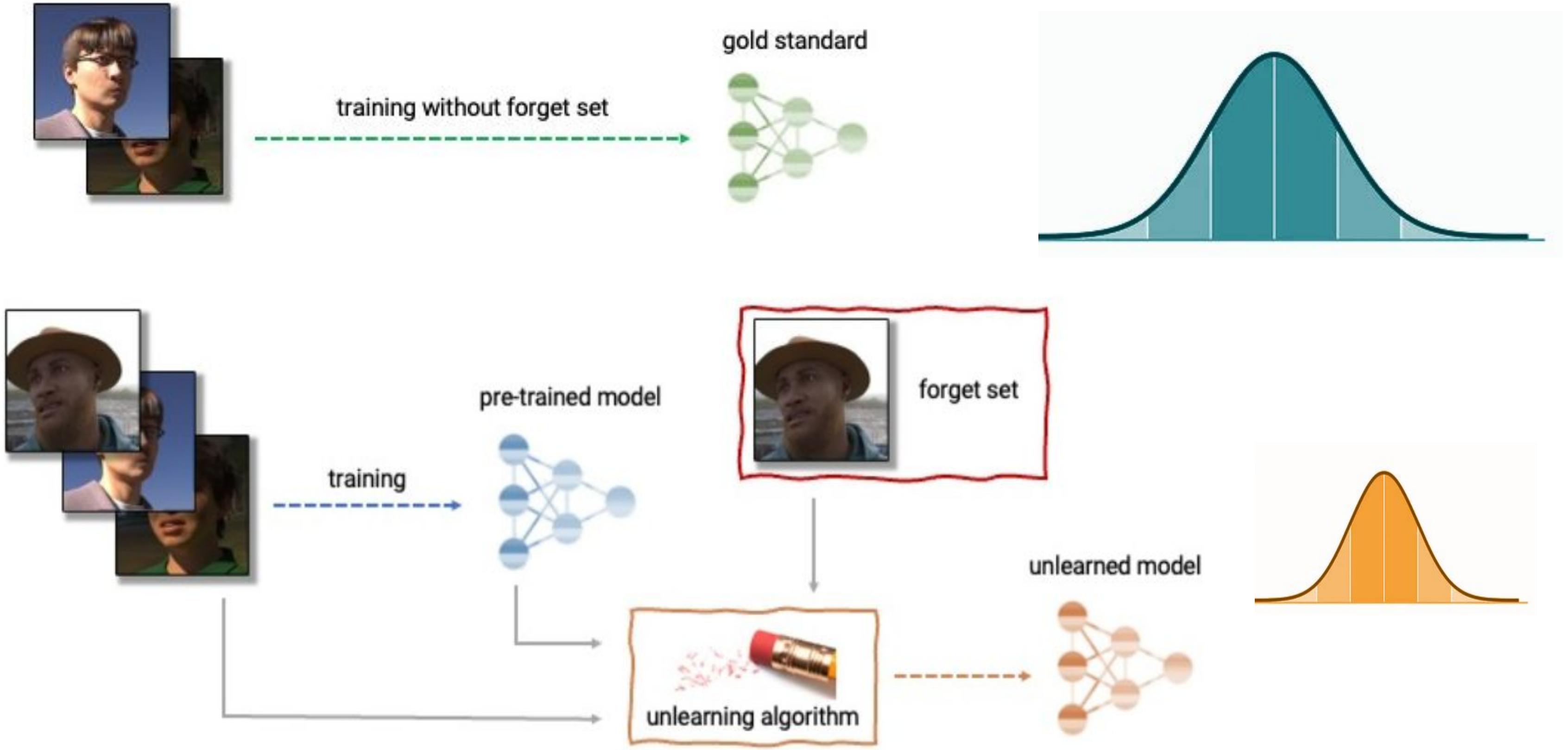
Firstname Lastname
Title

Google DeepMind

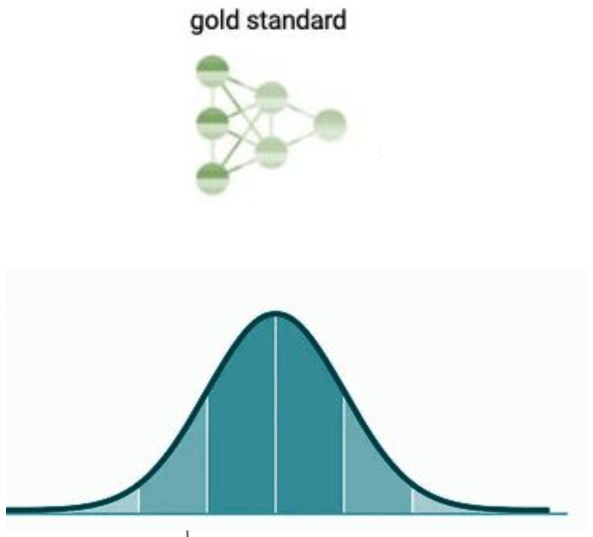
Colors

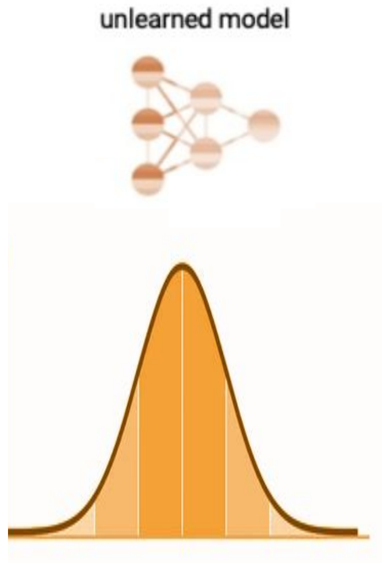
900	900	900	900	900
800	800	800	800	800
700	700	700	700	700
600	600	600	600	600
500	500	500	500	500
400	400	400	400	400
300	300	300	300	300
200	200	200	200	200
100	100	100	100	100
50	50	50	50	50

The gold standard of unlearning



How good is our approximation?





How close are these two distributions?