Re-thinking Data Availability Attacks Against Deep Neural Networks

Supplementary Material

A. The Omitted Proofs

Lemma 1. *Let $f_\theta(x_i)[k]$ indicates the k-th value of the predicted vector. Let $\mathcal{R} = (\frac{1}{K}, \ldots, \frac{1}{K}) \in R^K$ denotes the random guess probability and $\Theta = (f_\theta(x_i)[0], \ldots, f_\theta(x_i)[K]) \in R^K$ denotes the predicted probability. Then, we have*

$$-\mathcal{R}\log\Theta = KL(\mathcal{R}|\Theta) + \log K. \tag{9}$$

Proof.

$$\begin{aligned}\mathrm{KL}(\mathcal{R}|\Theta) &= \sum_{j=1}^{K}\frac{1}{K}(\log\frac{1}{K} - \log(f_\theta(x)[j])) \\ &= \sum_{j=1}^{K}\frac{1}{K}\log\frac{1}{K} - \sum_{j=1}^{K}\frac{1}{K}\log(f_\theta(x)[j]) \\ &= -\log K + \mathcal{R}\log\Theta\end{aligned} \tag{10}$$

□

Theorem 2. *Let $f_\theta(x_i)[k]$ indicates the k-th value of the predicted vector. Let $\mathcal{R} = (\frac{1}{K}, \ldots, \frac{1}{K}) \in R^K$ denotes the random guess probability. Then, we have*

$$0 \le \frac{1}{K}\sum_{j=1}^{K}(f_\theta(x)[j] - \frac{1}{K})^2 < \frac{4}{K} \tag{11}$$

Proof. Since, $|f_\theta(x)[j] - \frac{1}{K}| \ge 0$, we have:

$$\begin{aligned}&\frac{1}{K}\sum_{j=1}^{K}(f_\theta(x)[j] - \frac{1}{K})^2 \\ \le&\frac{1}{K}\Bigg(\sum_{j=1}^{K}(f_\theta(x)[j] - \frac{1}{K})^2 \\ &+ \sum_{j=1}^{K}|f_\theta(x)[j] - \frac{1}{K}| \cdot \sum_{k=1,k\neq j}^{K}|f_\theta(x)[k] - \frac{1}{K}|\Bigg) \\ =&\frac{1}{K}(\sum_{j=1}^{K}|f_\theta(x)[j] - \frac{1}{K}|)^2\end{aligned} \tag{12}$$

Since, $f_\theta(x)[j] \ge 0$, then $|f_\theta(x)[k] - \frac{1}{K}| < |f_\theta(x)[k] + \frac{1}{K}|$. We have:

$$\begin{aligned}\frac{1}{K}(\sum_{j=1}^{K}|f_\theta(x)[j] - \frac{1}{K}|)^2 &< \frac{1}{K}(\sum_{j=1}^{K}|f_\theta(x)[j] + \frac{1}{K}|)^2 \\ &= \frac{1}{K}\Bigg(\sum_{j=1}^{K}(f_\theta(x)[j] + \frac{1}{K})\Bigg)^2\end{aligned} \tag{13}$$

Since, $\sum_{j=1}^{K} f_\theta(x)[j] = 1$, we have:

$$\frac{1}{K}\Bigg(\sum_{j=1}^{K}(f_\theta(x)[j] + \frac{1}{K})\Bigg)^2 = \frac{4}{K} \tag{14}$$

In summary, we have:

$$0 \le \frac{1}{K}\sum_{j=1}^{K}(f_\theta(x)[j] - \frac{1}{K})^2 < \frac{4}{K} \tag{15}$$

□

B. An Analysis of REM

B.1. The optimization object of REM's paper and code implementation are inconsistent.

The reason REM [8] believes that adversarial training can disrupt non-robust attacks is that the model can learn from the adversarial examples. These adversarial examples contribute to an increase in the model's classification loss, which may encode valuable knowledge that can be harnessed to update the model's parameters and improve the model's performance. As a result, REM [8] generates unlearnable noise exclusively for adversarial examples, rather than clean data, ensuring that unlearnable noise remains effective against adversarial training. Eq (16) represents the proposed optimization objective.

Let $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ represent a dataset consisting of n samples, where $x_i \in \mathcal{X}$ is the i-th sample and $y_i \in \mathcal{Y} = \{1, \ldots, K\}$ is the corresponding label. Let a parameterized machine learning model be denoted by $f_\theta : \mathcal{X} \to \mathcal{Y}$, where $\theta \in \Omega$ is the model parameter. Let ℓ denote a loss function. Then, the training objective for the robust noise generator is as follows:

$$\min_\theta \frac{1}{n}\sum_{i=1}^{n} \underbrace{\min_{\|\delta_i^u\|\le\rho_u} \max_{\|\delta_i^a\|\le\rho_a} \ell(f'_\theta(x_i + \delta_i^u + \delta_i^a), y_i)}_{\text{unlearnable noise for adversarial examples}} \underbrace{\phantom{\min_{\|\delta_i^u\|\le\rho_u}}}_{\text{standard training}} \tag{16}$$

where ρ_u is the defensive perturbation radius that compels the generated robust unlearnable noise to be imperceptible.

However, the code implementation of REM [8][2] does not exactly follow the optimization goals in the paper, which are

[2] https://github.com/fshp971/robust-unlearnable-examples/blob/main/generate_robust_em.py

presented below:

$$\underbrace{\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta_i^{a\prime}}}_{\textbf{adversarial training}} \quad \underbrace{\min_{\delta_i^u} \max_{\delta_i^a} \ell(f'_\theta(x_i + \delta_i^u + \delta_i^a + \delta_i^{a\prime}), y_i)}_{\textbf{unlearnable noise for adversarial examples}} \tag{17}$$

We have intercepted some fragments of the code. In the code pie segment below, line 8 is the extra optimization process that REM's authors added to the code. Obviously, REM's authors performed an additional process of maximizing before training the surrogate model.

```
1    if args.cpu:
2        def_x = train_trans(x + torch.
             tensor(def_noise[ii]))
3    else:
4        def_x = train_trans(x + torch.
             tensor(def_noise[ii]).cuda
             ())
5    def_x.clamp_(-0.5, 0.5)
6
7    # the additional max step
8    adv_x = attacker.perturb(model,
         criterion, def_x, y)
9
10   model.train()
11   _y = model(adv_x)
12   def_acc = (_y.argmax(dim=1) == y).
         sum().item() / len(x)
13   def_loss = criterion(_y, y)
14   optim.zero_grad()
15   def_loss.backward()
16   optim.step()
```

B.2. Both Paper and Code Are not The Proper Optimization Object

The optimization object in REM's paper is equal to EM. As displayed in Eq (16), we divided the optimization objective of REM in the paper into two parts: the inner part, which generates unlearnable noise, and the outer part, which trains the model normally. The training process of this surrogate model is essentially the same as EM [13], with the only difference being the target of the generated unlearnable noise. REM's unlearnable noise targets adversarial examples, while EM's unlearnable noise targets clean examples. As shown in Table 7, the unlearnable noise generated for either example is not resistant to adversarial training. This observation aligns with our theoretical perspective, leading us to posit that the emergence of robust unlearnable noise is attributable to the inherent robustness of the surrogate model. The consistency between our findings and this hypothesis further substantiates the connection between model robustness and unlearnable noise generation.

The optimization object in REM's code is complex and incorrect. As illustrated in Eq. (17), we similarly divide the optimization objective employed by the REM-code into two distinct components: an internal component responsible for generating unlearnable noise against the samples, and an external component dedicated to training robust alternative models. By comparing the REM-paper and REM-code as presented in Table 7, it becomes evident that the additional optimization process of maximization, introduced in the REM code, is a critical factor in enabling unlearnable noise to withstand adversarial training. This observation underscores our conviction that the generation of robust unlearnable noise hinges upon the presence of a robust alternative model. However, two issues arise with the current optimization process:

1. The unlearnable noise generated by REM is specific to the adversarial example, which is not a valid assumption. Consequently, we propose optimizing the internal component of the REM code by retaining only the minimization optimization process.
2. Although the REM-code employs adversarial training to enhance model robustness, it fails to consider the definition of unlearnable examples. As a solution, we refine the training process of the alternative model by introducing ASR while accounting for the model's randomness in predicting probabilities for clean samples.

Taking these factors into account, we put forth our refined optimization objective, which aims to address the aforementioned concerns and foster a more accurate and comprehensive understanding of robust unlearnable noise generation.

Table 7 clearly demonstrates that generating unlearnable noise for adversarial examples remains susceptible to corruption by adversarial training. Comparing the REM-paper [8] and EM [13] approaches under different adversarial training perturbation radii, we observe that both techniques yield similar protection effects. This result supports our hypothesis that REM-paper [8] is fundamentally equivalent to EM [13]. Additionally, the evidence presented confirms that non-robust surrogate models are incapable of generating unlearnable examples that can withstand adversarial training.

Therefore, we believe that a robust surrogate model is key to generating unlearnable examples that can resist adversarial training. Furthermore, the supplementary ASR correction term introduced in our methodology places particular emphasis on the performance of the alternative model when dealing with clean samples. To summarize, our proposed approach eliminates unnecessary optimization steps while providing a more comprehensive evaluation of relevant factors. Consequently, it demonstrates superior performance in comparison to other methodologies under investigation.

Dataset	Adv. Train. ρ_a	Clean	EM	TAP	NTGA	REM-paper $\rho_a = 4/255$	REM-code $\rho_a = 4/255$	EntF $\rho_a = 4/255$	Ours $\rho_a = 4/255$
CIFAR-10	0	94.66	13.20	22.51	16.27	12.81	22.93	94.65	**12.71**
	1/255	93.74	22.08	92.16	41.53	37.42	30.00	93.56	**14.71**
	2/255	92.37	71.43	90.53	85.13	80.46	30.04	92.00	**15.38**
	3/255	90.90	87.71	89.55	89.41	88.92	31.75	91.04	**15.51**
	4/255	89.51	88.62	88.02	88.96	88.70	48.16	89.52	**23.12**

Table 7. Test accuracy (%) of models trained on data protected by different availability attacks via standard training and adversarial training.

C. Detailed Experimental Settings

C.1. Data Augmentation

In our experiments, we employ distinct data augmentation techniques tailored to different datasets. For CIFAR-10 and CIFAR-100 [15], we apply data augmentation comprising random flipping, padding of 4 pixels on each side, random cropping to a size of 32×32, and rescaling per pixel in the range $[-0.5, 0.5]$ for individual images. In the case of the ImageNet subset, we implement data augmentation through random cropping, resizing to dimensions of 224×224, random flipping, and rescaling per pixel in the range $[-0.5, 0.5]$ for each image.

C.2. Adversarial Training

Adversarial training [19] is a commonly used method to improve model robustness. Standard adversarial training aims to solve the following min-max optimization problem:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{||\delta_i|| \leq \rho_a} \ell(f_\theta(x_i + \delta_i), y_i) \tag{18}$$

C.3. Projected Gradient Descent

PGD [19] is a standard approach for solving inner maximization and minimization problems. It performs iterative projection updates to search for the optimal perturbation as follows:

$$\delta^{(k)} = \prod_{\|\delta\| \leq \rho} \left[\delta^{(k-1)} + c \cdot \alpha \cdot \text{sign} \left(\frac{\partial}{\partial \delta} \ell(f_\theta(x + \delta^{(k-1)}), y) \right) \right] \tag{19}$$

where k is the current iteration step (K steps at all), $\delta^{(k)}$ is the perturbation found in the k-th iteration, $c \in \{-1, 1\}$ is a factor for controlling the gradient direction, α is the step size, and $\prod_{\|\delta\| \leq \rho}$ means the projection is calculated in the ball sphere $\{\delta : \|\delta\| \leq \rho\}$. The final output perturbation is $\delta^{(K)}$. Throughout this paper, the coefficient c is set as 1 when solving maximization problems and -1 when solving minimization problems.

C.4. Detailed Settings for Our Method

In our approach, we build upon REM [8] and utilize ResNet-18 [11] as the surrogate model f'_θ. In our experiments, we employ L_∞-bounded noise constraints, denoted as $\|\delta_u\|_\infty \leq \rho_u$ and $\|\delta_a\|_\infty \leq \rho_a$, where ρ_u represents the unlearnable noise perturbation radius and ρ_a denotes the adversarial perturbation radius. Both quantities assume various values. The PGD settings are detailed in Table 8.

Regarding the CIFAR-10 and CIFAR-100 datasets, each surrogate model undergoes training via SGD over $5,000$ iterations, utilizing a batch size of 128, a momentum factor of 0.9, a weight decay factor of 0.0005, an initial learning rate of 0.1, and a learning rate scheduler that reduces the learning rate by a factor of 0.1 every $2,000$ iteration.

Similarly, for the ImageNet subset, we train each surrogate model using SGD for $3,000$ iterations with a batch size of 128, a momentum factor of 0.9, a weight decay factor of 0.0005, an initial learning rate of 0.1, and a learning rate scheduler that decays the learning rate by a factor of 0.1 every $1,200$ iteration.

Datasets	Noise Type	α_u	K_u	α_a	K_a
CIFAR-10 CIFAR-100	EM	$\rho_u/5$	10	-	-
	TAP	$\rho_u/125$	250	-	-
	NTGA	$\rho_u/10 \times 1.1$	10	-	-
	REM	$\rho_u/5$	10	$\rho_a/5$	10
	EntF	$\rho_u/5$	10	$\rho_a/5$	10
	Ours	$\rho_u/5$	10	$\rho_a/5$	10
ImageNet Subset	EM	$\rho_u/5$	7	-	-
	TAP	$\rho_u/50$	100	-	-
	NTGA	$\rho_u/8 \times 1.1$	8	-	-
	REM	$\rho_u/4$	7	$\rho_a/5$	10
	EntF	$\rho_u/4$	7	$\rho_a/5$	10
	Ours	$\rho_u/4$	7	$\rho_a/5$	10

Table 8. The settings of PGD [19] for the noise generations of error-minimizing noise (EM) [13], targeted adversarial poisoning noise (TAP) [7], neural tangent generalization attack noise (NTGA) [37], robust error-minimizing noise (REM) [8], and our method in different experiments. ρ_u denotes the defensive perturbation radius of different types of noise, while ρ_a denotes the adversarial perturbation radius of the robust error-minimizing noise.

C.5. Model Training Details

In accordance with Eq. (18), we carry out adversarial training [19] as described in appendix C.2. Analogous to the training of the noise generator, we focus on the L_∞-bounded noise, denoted as $\|\rho_a\|_\infty \leq \rho_a$, during the adversarial train-

Dataset	Random Seed	Method	VGG16	ResNet18	ResNet50	DenseNet121	WRN-34-10	Average
CIFAR-10	Random-1	EM	86.17	86.25	85.18	82.65	72.43	82.54
		REM	49.45	42.31	39.12	64.65	42.27	47.56
		Ours	**35.94**	**27.71**	**22.26**	**60.82**	**29.86**	**35.32**
	Random-20	EM	87.70	89.21	90.18	82.36	87.72	87.43
		REM	53.60	47.42	43.37	68.01	48.71	52.22
		Ours	**34.35**	**22.15**	**18.61**	**57.21**	**22.05**	**30.87**
	Random-42	EM	87.24	88.62	89.66	81.77	79.87	85.43
		REM	65.23	48.16	40.65	82.38	48.39	56.96
		Ours	**37.78**	**23.12**	**19.30**	**72.42**	**18.67**	**34.26**
CIFAR-100	Random-1	EM	56.35	63.77	66.44	53.56	68.13	61.65
		REM	50.51	27.54	26.85	54.62	26.43	37.19
		Ours	**40.64**	**12.29**	**16.42**	**44.12**	**16.00**	**25.89**
	Random-20	EM	56.37	63.32	66.43	53.74	68.35	61.64
		REM	39.03	27.80	27.81	54.84	27.27	35.35
		Ours	**31.41**	**11.35**	**12.55**	**46.05**	**15.33**	**23.34**
	Random-42	EM	56.94	63.43	66.43	53.52	68.27	61.72
		REM	58.07	27.35	26.03	56.63	27.71	39.16
		Ours	**55.05**	**23.00**	**21.47**	**52.25**	**20.14**	**34.38**

Table 9. Test accuracy (%) of different models adversarially trained on CIFAR-10 and CIFAR-100 with different random seeds.

ing process.

Throughout our experiment, the model undergoes training using SGD for $40,000$ iterations with a batch size of 128, a momentum factor of 0.9, a weight decay factor of 0.0005, an initial learning rate of 0.1, and a learning rate scheduler that reduces the learning rate by a factor of 0.1 every $16,000$ iteration. For the CIFAR-10 and CIFAR-100 datasets, the step number K_a and the step size α_a in PGD are set to 10 and $\rho_a/5$, respectively. For the ImageNet subset, we set the step number K_a and step size α_a to 8 and $\rho_a/4$, respectively. This ensures a consistent and well-structured experimental setup across various datasets.

D. Time Consumption

Our method requires less time to train the robust noise generator than REM. Table 10 shows the time cost of three different methods. It should be noted that the time cost for CIFAR-10 and CIFAR-100 was tested using a V100, while the time cost for ImageNet-Subset was tested using 4 V100s.

	EM	TAP	NTGA	REM	Ours
CIFAR-10	0.4	0.5	5.2	22.6	5.6
CIFAR-100	0.4	0.5	5.2	22.6	5.6
ImageNet-Subset	3.9	5.2	14.6	51.2	11.3

Table 10. Time consumption (h) of different methods on different datasets.

E. Random Seeds

The random seed for all the experiments in the paper is set to 42. We also test different random seeds and the results of the experiments are shown in the the following table 9. ResNet-18 is used as the noise generator.