Is Retain Set All You Need in Machine Unlearning? Restoring Performance of Unlearned Models with Out-Of-Distribution Images Supplementary Material

Jacopo Bonato[1,2], Marco Cotogni[2*], and Luigi Sabetta[1,2]

[1] Leonardo Labs, Rome, Italy,
[2] These authors contributed equally
{jacopo.bonato, luigi.sabetta}.ext@leonardo.com, cotogni@proton.me

1 Additional Analyses

In this section, we present additional analyses that support the findings of the main paper. Specifically, we explore the impact of the distillation trick beyond the unlearning context. Additionally, we display the results for both CR and HR scenarios using the COCO dataset as $\mathcal{D}^{sur}$. We also examine the computational impact of our method. Lastly, we investigate the behavior of feature maps from various layers of the original and unlearned models in the feature space.

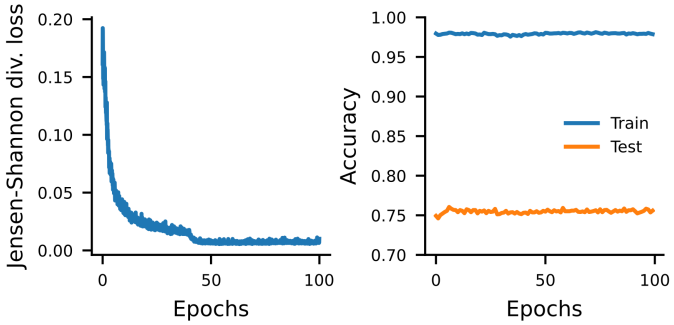

Fig. 1: Examples of train loss (left) and train set and test set accuracies (right) of a `resnet18` trained with the trick-distillation mechanism on the subset of Imagenet1K.

1.1 Distillation-Trick details

The distillation-trick mechanism has been verified outside the unlearning scope. We used as ϕ_θ a `resnet18` trained on CIFAR100 and as OOD surrogate dataset

* Work done while at Leonardo Labs

the subset of 10K images from Imagenet1K. ϕ_θ has been trained using the distillation-trick mechanism (eq. 1 and 7) directly on the surrogate dataset and tested on the test set of CIFAR100 (Fig. 1). For this training, it has been used Adam [9] as the optimizer, a weight decay of 5×10^{-4}, $lr = 5 \times 10^{-4}$ and StepLR scheduler with $\gamma = 0.1$ and $\text{epoch}_{\text{step}} = 40$. Overall, we observed that the accuracy of the test set after the first 5 epochs remains constant at $\approx 76\%$. This result demonstrates how this mechanism can be used as Model "Knowledge" regularization technique when training data are not available as in the CR and HR scenarios where the $\mathcal{D}_r$ is not available.

1.2 Results with Coco as $\mathcal{D}^{\text{sur}}$

We reported in Tab. 1 the results obtained for SCAR and SCAR self-forget in CR and SCAR in HR for both CIFAR100 and TinyImagenet using the subset of COCO as surrogate dataset. Overall, we observed compatible results in both scenarios which highlights once more how differences in terms of kind of information such as surrogate dataset classes and distribution of pixels do not affect SCAR.

		CIFAR100			TinyImagenet		
		$\mathcal{A}_r^t$	$\mathcal{A}_f^t$	AUS	$\mathcal{A}_r^t$	$\mathcal{A}_f^t$	AUS
CR	SCAR (Imagenet)	72.93(01.78)	02.00(02.00)	0.935(0.025)	62.99(01.39)	00.60(01.37)	0.940(0.019)
	SCAR S-F(Imagenet)	71.69(02.69)	00.70(00.95)	0.929(0.028)	60.79(02.15)	00.80(01.40)	0.917(0.025)
	SCAR (COCO)	71.61(01.92)	00.84(00.43)	0.933(0.023)	62.04(01.41)	01.00(01.41)	0.927(0.019)
	SCAR S-F(COCO)	71.27(02.95)	00.75(01.30)	0.930(0.032)	61.88(00.98)	01.20(01.69)	0.924(0.018)
		$\mathcal{A}^t$	$\mathcal{A}_f$	AUS	$\mathcal{A}^t$	$\mathcal{A}_f$	AUS
HR	SCAR (Imagenet)	73.23(00.74)	75.63(00.54)	0.934(0.011)	61.35(00.69)	66.51(00.44)	0.886(0.011)
	SCAR (COCO)	72.51(00.41)	75.64(00.57)	0.921(0.008)	61.56(00.73)	66.49(00.47)	0.890(0.011)

Table 1: Performance of SCAR and Scar self-forget in CR and HR scenarios for CIFAR100 and TinyImagenet datasets using either the subset of Imagenet1K or the subset of COCO as surrogate datasets. The metrics are reported as mean ± std over ten runs. S-F stands for Self-forget

	TinyImagenet CR			TinyImagenet HR		
	$\mathcal{A}_r^t$	$\mathcal{A}_f^t$	AUS	$\mathcal{A}_r^t$	$\mathcal{A}_f^t$	AUS
Cosine Similarity	60.55(01.66)	01.00(01.41)	0.912(0.021)	61.27(00.81)	66.48(00.52)	0.883(0.012)
$\mathcal{L}_2$ Distance	57.74(01.66)	01.40(01.90)	0.881(0.023)	49.20(01.18)	48.69(01.01)	0.806(0.018)
Mahalanobis	62.99(01.39)	00.60(01.37)	**0.940(0.019)**	1.35(00.69)	66.51(00.44)	**0.886(0.011)**

Table 2: Results of the ablation study on CIFAR100 in CR and HR scenarios. The metrics for CR and HR are reported as mean ± std over ten runs.

1.3 Computational Analysis

We present in Table 3 a computational comparison between our method and other state-of-the-art approaches. We evaluated the methods based on the AUS metric, required unlearning time, and memory usage for the forget and retain data. Notably, our method only needs to store $\mathcal{D}^{\text{sur}}$ for the Self-Forget variant while $\mathcal{D}_f$ and $\mathcal{D}^{\text{sur}}$ for SCAR. Furthermore, our approach demonstrates a favorable balance between AUS, unlearning time, and storage efficiency, outperforming several methods that rely on both $\mathcal{D}_r$ and $\mathcal{D}_f$. Although slower than Retain-Free approaches and requiring marginally more memory, our methods exhibit significantly superior performance compared to these approaches and are comparable to more computationally intensive methods that depend on the Retain set.

Method	$\mathcal{D}_r$ free	$\mathcal{D}_f$ free	AUS	Time(s)	Retain-Data Stored(Mb)	Forget-Data Stored(Mb)	Total-Data Stored(Mb)
Fine Tuning	✗	✓	0.998(0.022)	366.99	145.02	0	145.02
DUCK	✗	✗	0.931(0.026)	12.56	145.02	1.46	146.48
Boundary S.	✗	✗	0.749(0.116)	23.19	145.02	1.46	146.48
Boundary E.	✗	✗	0.750(0.117)	66.50	145.02	1.46	146.48
SCRUB	✗	✗	0.977(0.051)	139.18	145.02	1.46	146.48
L1-Sparse	✗	✗	0.879(0.018)	35.29	145.02	1.46	146.48
ERM-KTP	✗	✗	0.814(-)	550.14	145.02	1.46	146.48
Bad Teacher	✗	✗	0.940(0.089)	56.07	145.02	1.46	146.48
Neg. Grad.	✓	✗	0.849(0.061)	09.02	0	1.46	1.46
Rand. Lab.	✓	✗	0.774(0.070)	17.03	0	1.46	1.46
SCAR	✓	✗	0.935(0.025)	109.11	29.30	1.46	30.76
SCAR Self-Forget	✓	✓	0.929(0.028)	101.61	29.30	0	29.30

Table 3: Comparison between our method, the self-forget variant, and methods from the state-of-the-art on the Cifar100 dataset in the CR scenario in terms of AUS, unlearning time, and memory requirements. For SCAR and SCAR Self-Forget, the amount of memory occupied for the retain-data refers to the $\mathcal{D}^{\text{sur}}$

1.4 t-SNE analysis on DNN layers

To gain deeper insights into the impact of the unlearning mechanism employed in our approach, we conducted an analysis of the embeddings derived from the neural network backbone (Φ_θ) both pre and post the unlearning process in the CR scenario. Utilizing t-SNE [10], a dimensionality reduction technique for high-dimensional data visualization, we analyzed the transformation of embeddings from various layers of the deep neural network (DNN). The resulting distributions, as illustrated in Fig. 2, depict each class within the dataset as a uniquely colored cluster for better visual distinction.

We generated two separate t-SNE visualizations: one representing the original model's embeddings (Fig. 2-A) and the other post-unlearning with SCAR (Fig. 2-B). Initially, in the original model, the embeddings of different classes, particularly from the last residual layer, formed distinct and cohesive clusters, signifying a high degree of separation within the test set. Contrastingly, after applying SCAR, while the embeddings of the samples meant to be retained preserved their clustered integrity, those designated for forgetting became interspersed among the clusters of other classes.

These findings suggest that the information content of the embeddings derived from Φ_θ for the forget-set no longer hold the necessary attributes for accurate sample classification, signaling a comprehensive dilution of relevant details not only in the final layer but also across preceding layers of the network. This observation underscores the profound efficacy of our unlearning process, which effectively destroys the pertinent data footprint throughout the neural architecture.

2 Reproducibility

In this section we report all the details about the experiments reported in the main paper and in this Supplementary Material.

2.1 Experimental Details and Hyperparameters

We trained the original model with the seed fixed to $S = 42$. Then for the two scenarios proposed in the paper, we adopted two different setups.

CR. In CR scenario, we set the seed to $S = 42$. Subsequently, we divided the training and testing datasets into forget sets (comprising all instances of a specific class q) and retain sets (comprising instances of the remaining classes). We applied the unlearning algorithms to these sets and recorded the metrics. This process of splitting and applying the unlearning procedure was repeated ten times, each time altering the class q designated for the forget set. Ultimately, we calculated the mean and standard deviation (std) for all metrics. For CIFAR10, one class at time constituted the forget sets ($q = [0, 1, 2...]$). For CIFAR100, the forget set contained at each time a class multiple of 10, starting from class 0 ($q = [0, 10, 20, ..., 90]$). Similarly, for TinyImagenet, starting from class 0, at each iteration, a class multiple of 20 constituted the forget set ($q = [0, 20, 40, 60, ..., 180]$)

HR. In the HR scenario, we select a seed S and split the training dataset into forget set (containing 10% of the training data independently from the class) and retain set (containing the remaining 90%). We applied the unlearning algorithms to these sets and recorded the metrics. This process of splitting and applying the unlearning procedure was repeated ten times, each time selecting a different seed in $S = [0, 1, 2, 3, 4, 5, 6, 7, 8, 42]$ designated for the forget set. Ultimately, we calculated the mean and standard deviation (std) for all metrics.

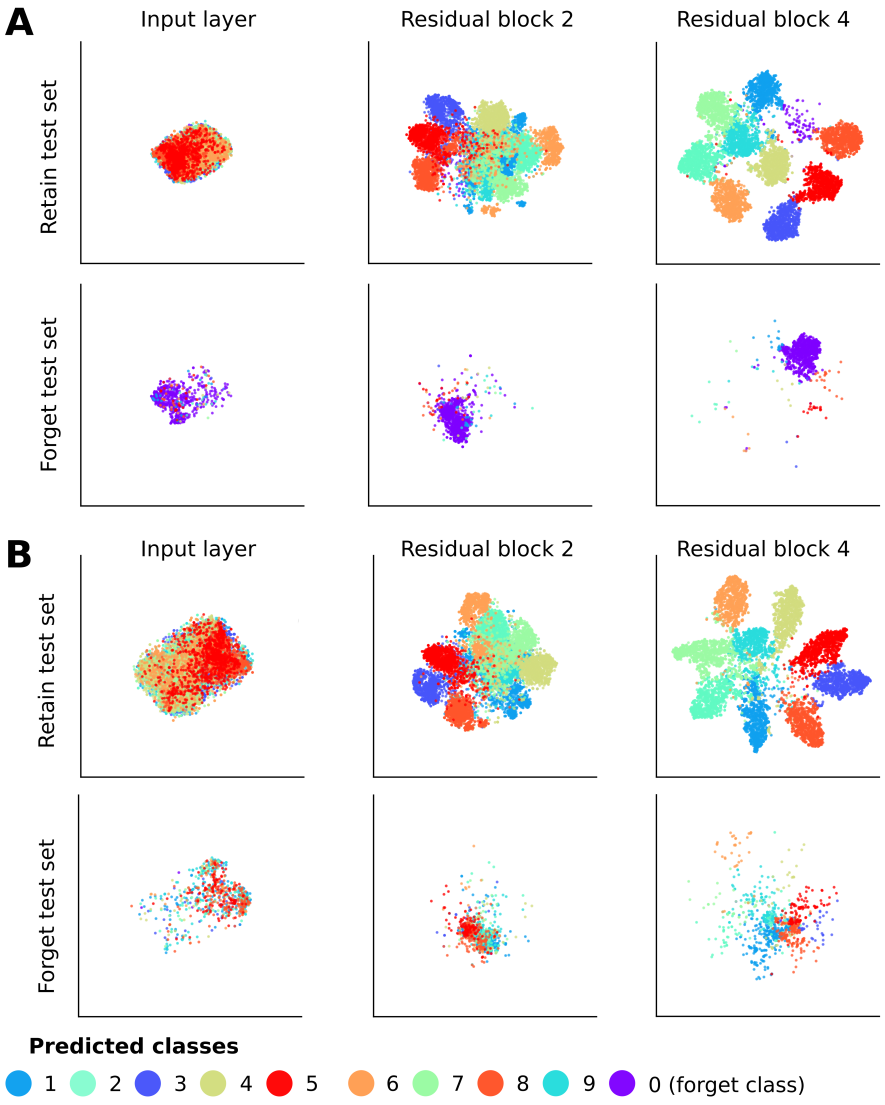

Fig. 2: T-SNE plot of feature vectors of the original model ϕ_θ (A) and unlearned model ϕ_θ^U (B) applied to $\mathcal{D}_r^t$ and $\mathcal{D}_f^t$ of CIFAR10. Feature vectors are extracted after the first convolutional layer, and after the first and last residual block. The original model or unlearned model predicted classes are reported in different colors.

All the hyperparameters of SCAR used to produce the results reported in Tab. 1 and 2 of the main text are detailed in Tab. 4, while the ones used for SCAR self-forget case are listed in Tab. 5. **For comprehensive reproducibility, all scripts necessary to replicate our findings are attached to this Supplementary Material** and will be released at https://github.com/jbonato1/SCAR.

	CR									HR								
	λ_1	λ_2	lr	BS	T	δ	γ_1	γ_2	$\mathrm{E}_{\max}$	λ_1	λ_2	lr	BS	T	δ	γ_1	γ_2	$\mathrm{E}_{\max}$
Cifar10	1	5	5×10^{-4}	1024	1	0.5	3	3	30	1	8	5×10^{-4}	1024	5	1	3	3	30
Cifar100	1	5	5×10^{-4}	1024	1	0.5	3	3	30	1	6	5×10^{-4}	1024	2.5	1	3	3	30
TinyImagenet	1	5	1×10^{-4}	1024	1	0.5	3	3	30	1	5	1×10^{-4}	1024	1.8	1	6	6	30

Table 4: Hyperparameters of SCAR used to produce the results reported in Tab. 1 and Tab. 2 of the main text for the two scenarios, CR on the left, HR on the right.

	λ_1	λ_2	lr	BS	T	δ	γ_1	γ_2	$\mathrm{Epoch}_{\max}$
Cifar10	1	5	7.5×10^{-4}	1024	2	0.5	3	3	25
Cifar100	1	4	1×10^{-3}	1024	2	0.5	3	3	25
TinyImagenet	1	5	1×10^{-4}	1024	2	0.5	3	3	25

Table 5: Hyperparameters of SCAR self-forget used to produce the results reported in Tab. 1 of the main text, for the CR scenario.

2.2 AUS

The AUS metric [3] has been formulated to address the need for a measure that considers both the performance on the retain and forget-set of an untrained model. It serves the pivotal function of enabling the ranking of different methods based on their ability to balance memory retention and forgetting. This metric is particularly designed to provide a singular score reflecting each method's efficiency in maintaining high test set accuracy while successfully engaging in the unlearning process. The metric is outlined as follows:

$$\mathrm{AUS} = \frac{1 - (\mathcal{A}_t^{Or} - \mathcal{A}_t)}{1 + \Delta}, \quad \Delta = \begin{cases} |0 - \mathcal{A}_f| & \text{if CR} \\ |\mathcal{A}_t - \mathcal{A}_f| & \text{if HR} \end{cases}, \tag{1}$$

In this formulation, A_t represents the accuracy on the test set and A_f represents the accuracy on the forget-set of the untrained model, whereas A_t^{Or} denotes the accuracy on the test samples of the original model. To simplify notation,

we sobstituted $\mathcal{A}_f$ with $\mathcal{A}_f^t$, $\mathcal{A}_t^{Or}$ with $\mathcal{A}_r^{t,Or}$, and $\mathcal{A}_t$ with $\mathcal{A}_r^t$. By offering a unified score, the AUS metric simplifies the comparative assessment of different models, underscoring their performance in both retaining essential information and effectuating targeted unlearning.

2.3 MIA

In the HR scenario, through the usage of a membership inference attack (MIA) [3], the goal is to investigate whether forget set samples were used to train an input model Φ_θ. To achieve this, the MIA tries to assess whether, from the model perspective, the forget data can be distinguished from the test set data (the actual set of data never seen by the model Φ_θ during training). Failure of the MIA indicates the inability to differentiate forget data from test data and consequently the impossibility to classify the former samples as training data.

Summarizing the step of the MIA proposed in [3, 11, 12]: given the datasets $\mathcal{D}_f$ and $\mathcal{D}^t$, we combine $\mathcal{D}_f$ and an equal number of samples from $\mathcal{D}^t$ into a single dataset, which we then split into a training set $\mathcal{D}_{\text{mia}}$ (containing 80% of the samples) and a test set $\mathcal{D}_{\text{mia}}^t$ (containing 20% of the samples).
Subsequently, an SVM with a Gaussian kernel classifier is trained. This SVM takes as input the probability distribution across classes obtained from the $\texttt{softmax}(\Phi_\theta^U(x_i))$, where x_i belongs to either $\mathcal{D}_{\text{mia}}$ or $\mathcal{D}_{\text{mia}}^t$. The SVM is trained on $\mathcal{D}_{\text{mia}}$ to distinguish between training (i.e. forget) and test instances.
SVM hyperparameters are optimized through a 3-fold cross-validation grid search and finally, the SVM is evaluated on $\mathcal{D}_{\text{mia}}^t$. Failure of the Membership Inference Attack (MIA) indicates that information about forget-samples has been successfully removed from the untrained model. The MIA performance is assessed based on the mean F1-score over 10 iterations of the SVM, each utilizing different train and test splits of $\mathcal{D}_f$ and $\mathcal{D}^t$.

In the CR scenario, the forget set consists solely of samples from a single class, differently from the HR scenario where the test set is composed of instances from multiple classes. In this scenario, the application of MIA as defined before, would lead the SVM to detect biases in the logits related to the identity of the classes, thereby skewing results away from reflecting true sample membership. To mitigate the introduction of these biases, stemming from the differing class distributions between the two datasets, we employed the forget subset of the test data to ensure a fair and unbiased comparison. Rather than utilizing the entire $\mathcal{D}^t$ for the MIA test, $\mathcal{D}_f$ is combined with $\mathcal{D}_f^t$ to ensure uniform class identification across both subsets, thus mitigating the introduction of class-related biases in $\mathcal{D}^t$. The SVM is trained to discriminate forget train instances vs. forget test instances. The number of forget samples ($N = 5000$ for CIFAR10 and $N = 500$ for CIFAR100) and forget test samples ($N = 1000$ for CIFAR10 and $N = 100$ for CIFAR100) are unevenly distributed. To address this imbalance, we resample $\mathcal{D}_f$ to create a less imbalanced $\mathcal{D}_{\text{mia}}$ with a ratio of 1:3 forget test

	CIFAR10				CIFAR100			
	$\mathcal{A}^t$	$\mathcal{A}_f$	AUS	$\mathcal{F}_1$	$\mathcal{A}^t$	$\mathcal{A}_f$	AUS	$\mathcal{F}_1$
Original	88.64(00.63)	88.34(00.62)	0.531(0.005)	77.63(1.33)	77.55(00.11)	77.50(02.80)	0.563(0.009)	83.15(4.12)
Retrained	88.05(01.28)	00.00(00.00)	0.994(0.014)	74.91(0.10)	77.97(00.42)	00.00(00.00)	1.004(0.022)	75.47(0.75)
SCAR	87.71(01.62)	00.97(00.24)	0.981(0.017)	74.91(0.19)	72.93(01.78)	02.00(02.00)	0.935(0.025)	75.60(0.44)

Table 6: Results obtained applying the MIA defined in [3, 11, 12] adapted for the CR scenario.

samples per forget samples, resulting in a chance level of .75. Results for CR scenario in CIFAR10 and CIFAR100 are reported in Tab. 6

In the HR scenario, both $\mathcal{D}^t$ and $\mathcal{D}_f$ encompass all classes, with an equal number of samples per dataset. Consequently, the chance level for the MIA test in this case is .5.

3 Baselines

In this section, we report additional details about the baselines and competitors we included in the comparisons.

Original: This is the original model trained on the entire dataset. This model is used as a starting point for SCAR, the competitors, and the baselines considered. The goal of every unlearning algorithm is to remove the knowledge about the forget set from this model. The model, as Resnet18, has been trained for 300 epochs with cosine annealing as lr scheduler.

Retrained: The "Retrained" model, is a model initialized from random weights and then trained only on the retain set $\mathcal{D}_r$. This is considered an upper bound since it does not have any knowledge of the forget data $\mathcal{D}_f$. The model has been trained for 200 epochs with cosine annealing as lr scheduler.

Finetuning [4]: This method fine-tunes the original model using the retained data $\mathcal{D}_r$. The model has been fine-tuned for 30 epochs and with X of learning rate. Following [1] the fine-tuning method results are effective but time-consuming. Importantly, this method requires access to the entire retained set.

Negative Gradient (NG) [4]: In this method, the original model is tuned on the forget data minimizing the inverse of gradient. Results are reported from [3].

Random Label (RL) [5]: In this method, the original model is tuned, using the cross-entropy loss, matching the exemplars from the forget set with random labels among the ones of the retain data. Results are reported from [3].

Boundary Expanding (BE) [1]: This method assigns an additional shadow class to each sample in the forget-set, thereby shifting the decision boundary and leveraging the decision space.

Boundary Shrink (BS) [1]: In this method, the decision boundary is adjusted matching for each sample in the forget-set the closest wrong class label and then finetuning the model with the forget set with these new labels and the retain set using cross-entropy loss. Results are reported from [3]

ERM-KTP (ERM) [8]: This approach switches between the Entanglement-Reduced Mask and the Knowledge Transfer and Prohibition phases to erase the

data related to the forget-set while enhancing the accuracy on the retain-set. Results are reported from [3]

SCRUB [7]: This method employs a distillation mechanism to forget the $\mathcal{D}_f$ pushing away the student network predictions (unlearned model) from the teacher network (original model). This procedure is called max-steps. Unfortunately, this mechanism is extremely disruptive, and SCRUB alternates to the max-steps a specific procedure to regain performance on the retain set called min-steps which combines distillation and cross-entropy losses. The distillation loss is weighted .001 whereas the cross-entropy is .999. Additionally, the authors conducted a limited number of min-steps to recover the lost knowledge pertaining to the retained data. For each dataset and scenario, lr, number of min-steps, max-steps, and epochs have been optimized to obtain the best overall AUS.

DUCK [3]: In this paper, the authors minimize the distance between feature vectors derived from the forget data and the nearest centroid of a different class through metric learning. Concurrently, cross-entropy loss has been employed to maintain performance on retained data. This approach operates directly on single forget samples, enabling the method to function effectively in both CR and HR scenarios. Results are reported from [3].

Bad Teacher [2]: In this method, similarly to [7], the student model adheres to the original model (i.e., the teacher) with respect to the retain set data. Simultaneously, it minimizes the KL-divergence between its logits and those of a randomly initialized model using as input the forget set data. We followed the training procedure reported in the paper: the model has been trained for 4 epochs with $lr = 0.0001$ and Adam as optimizer.

L1-sparse [6]: In this paper the authors elucidates the relationship between exact unlearning methods and approximate unlearning methods when subjected to pruning. This led to the development of an unlearning strategy based on two phases: the pruning phase and the unlearning phase. During the pruning phase, the original model is pruned removing uninformative weights connections. Then, an unlearning regularization, i.e. the "L1-sparse MU" method, is applied to the pruned model for 10 epochs with $lr = 0.001$.

4 Examples of images from different $\mathcal{D}^{\text{sur}}$

Fig. 3: Examples of surrogate images sampled from Imagenet1K

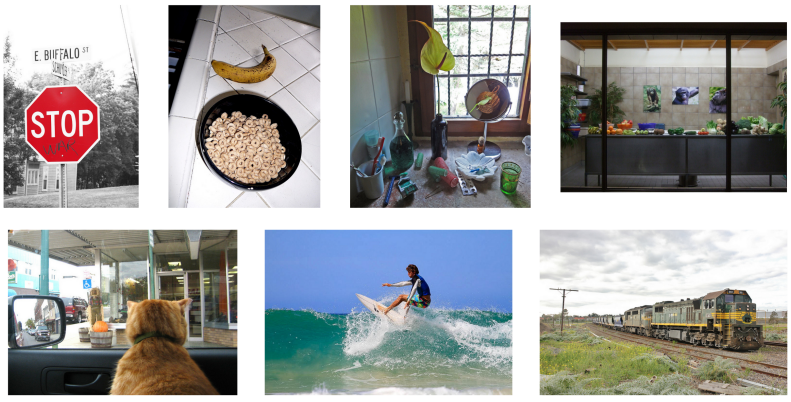

Fig. 4: Examples of surrogate images sampled from COCO dataset

Fig. 5: Examples of surrogate images sampled from random images dataset

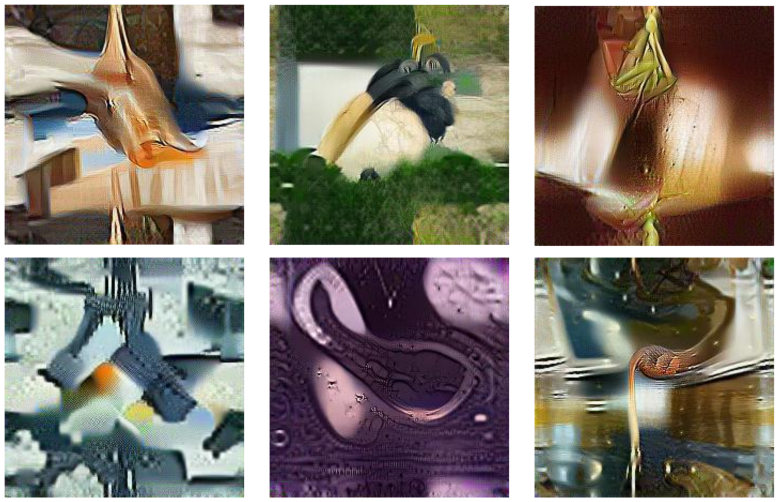

Fig. 6: Examples of surrogate images sampled from imagenet1K distilled

References

1. Chen, M., Gao, W., Liu, G., Peng, K., Wang, C.: Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7766–7775 (June 2023) 8
2. Chundawat, V.S., Tarun, A.K., Mandal, M., Kankanhalli, M.: Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 7210–7217 (2023) 9
3. Cotogni, M., Bonato, J., Sabetta, L., Pelosin, F., Nicolosi, A.: Duck: Distance-based unlearning via centroid kinematics. arXiv preprint arXiv:2312.02052 (2023) 6, 7, 8, 9
4. Golatkar, A., Achille, A., Soatto, S.: Eternal sunshine of the spotless net: Selective forgetting in deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9304–9312 (2020) 8
5. Hayase, T., Yasutomi, S., Katoh, T.: Selective forgetting of deep networks at a finer level than samples. arXiv preprint arXiv:2012.11849 (2020) 8
6. Jia, J., Liu, J., Ram, P., Yao, Y., Liu, G., Liu, Y., Sharma, P., Liu, S.: Model sparsity can simplify machine unlearning. In: Annual Conference on Neural Information Processing Systems (2023) 9
7. Kurmanji, M., Triantafillou, P., Hayes, J., Triantafillou, E.: Towards unbounded machine unlearning. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=OveBaTtUAT 9
8. Lin, S., Zhang, X., Chen, C., Chen, X., Susilo, W.: Erm-ktp: Knowledge-level machine unlearning via knowledge transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20147–20155 (2023) 8
9. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 2
10. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(11) (2008) 3
11. Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., Backes, M.: Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In: Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS) (2019) 7, 8
12. Song, L., Mittal, P.: Systematic evaluation of privacy risks of machine learning models (2020) 7, 8