# PREVENTING UNAUTHORIZED USE OF PROPRIETARY DATA: POISONING FOR SECURE DATASET RELEASE

**Liam Fowl** *
Department of Mathematics
University of Maryland
lfowl@umd.edu

**Ping-yeh Chiang** *
Department of Computer Science
University of Maryland
pchiang@cs.umd.edu

**Micah Goldblum** *
Department of Computer Science
University of Maryland
goldblum@umd.edu

**Jonas Geiping**
Department of Electrical Engineering
University of Siegen
jonas.geiping@uni-siegen.de

**Arpit Bansal**
Department of Electrical Engineering
University of Maryland
bansal01@umd.edu

**Wojtek Czaja**
Department of Mathematics
University of Maryland
czaja@umd.edu

**Tom Goldstein**
Department of Computer Science
University of Maryland
tomg@cs.umd.edu

## ABSTRACT

Large organizations such as social media companies continually release data, for example user images. At the same time, these organizations leverage their massive corpora of released data to train proprietary models that give them an edge over their competitors. These two behaviors can be in conflict as an organization wants to prevent competitors from using their own data to replicate the performance of their proprietary models. We solve this problem by developing a data poisoning method by which publicly released data can be minimally modified to prevent others from training models on it.

## 1 INTRODUCTION

Social media companies and other web platforms allow users to post their own data in a space that is openly accessible to scraping (Taigman et al., 2014; Cherepanova et al., 2021). This data can be incredibly valuable, both to the organization hosting it and to others who leverage scraped data to train their own models. Breakthroughs in both image classification (Russakovsky et al., 2015) and language models (Brown et al., 2020) have been enabled by large volumes of scraped data. Given that organizations value their exclusive access to the data they host for training competitive machine learning systems, they need a method for safely releasing their data on their web platform while preventing competitors from replicating the performance of their own models trained on this data.

We introduce a method, motivated by new techniques in targeted data poisoning, to modify data prior to its release so that the generalization of a deep learning model trained on this data is significantly degraded, rendering the data effectively worthless to competitors.

*Authors contributed equally

### 1.1 Related Work

The topic of data manipulation for the purposes of performance degradation has been investigated in the *data poisoning* literature. Specifically, this work is closely related to *indiscriminate* (availability) poisoning attacks wherein an attacker wishes to degrade performance on a large number of samples (Barreno et al., 2010). Early works on this type of attack show that data can be maliciously modified to degrade test-time performance of simple classical algorithms, such as support vector machines, principle component analysis, clustering, logistic regression, etc., or in the setting of binary classification (Muñoz-González et al., 2017; Xiao et al., 2015; Biggio et al., 2012; Koh et al., 2018; Steinhardt et al., 2017b).

In scenarios involving simple learning models, the optimal perturbation to training data can often be explicitly calculated via the implicit function theorem. However, this becomes computationally intractable for modern deep networks. As a consequence, little work has been done on indiscriminate attacks on deep networks. Recently, Shen et al. (2019) proposed a heuristic to avoid having to explicitly solve the full bi-level objective. The method, TensorClog, crafts perturbations to cause gradient vanishing with the aim of preventing a deep network from training on the perturbed data, thus degrading test time performance of the network. However, this work only performs their method in the setting of transfer learning where a known feature extractor is used, limiting the viability of this attack.

In contrast to indiscriminate attacks, *targeted* (integrity) poisoning attacks aim to cause a network trained on modified data to mis-classify a few pre-selected target samples. Unlike the indiscriminate attack setting, recent work on targeted poisoning has successfully attacked modern deep networks trained from scratch on poisoned data (Geiping et al., 2020; Huang et al., 2020). These attacks do not noticeably degrade validation accuracy, despite the victim network mis-classifying the selected target example(s). Moreover, simply performing a large number of targeted attacks to degrade overall validation accuracy is not feasible. In some cases, these attacks perturb up to $10\%$ of the training data in order to mis-classify a *single* target image, and they often fail to attack more than a handful of targets (Geiping et al., 2020).

We develop an indiscriminate data poisoning attack which works on deep networks trained from scratch in a black-box setting. Our method allows practitioners to minimally modify data which, when released, causes models trained on this data to generalize poorly. Our method allows companies to release data, either for transparency purposes, or via user upload, which does not compromise the competitive advantage the company gains from asymmetric access to the clean data. For a general overview of data poisoning attacks, defenses, and terminology, see Goldblum et al. (2020).

Also parallel to the goals of this work are defenses to model stealing attacks. Model stealing attacks often aim duplicate a machine learning model or its functionality Tramèr et al. (2016). Defenses vary depending upon the attack scenario. For example, a defense proposed in Orekondy et al. (2019) perturbs the prediction outputs of a network to prevent a model stealing attack which aims to mimic the performance of a service like a cloud prediction API. Other defenses aim to prove theft of a model has occurred after the fact by "watermarking" a network Uchida et al. (2017). However, these defenses do little to stop malicious actors from using scraped data to train their own models Hill (2020).

## 2 Our Method

### 2.1 Problem Setup

Formally, we seek to compute perturbations $\Delta = \{\Delta_i\}$ to elements $x_i$ of a dataset $\mathcal{S}$ in order to make a network, $F$, trained on the dataset generalize poorly to the distribution $\mathcal{D}$ from which $\mathcal{S}$ was sampled. Achieving this goal entails solving the following bi-level objective,

$$\max_{\Delta \in \mathcal{C}} \ \mathbb{E}_{(x,y)\sim\mathcal{D}} \Big[ \mathcal{L}\left(F(x;\theta(\Delta)), y\right) \Big] \tag{1}$$

$$\text{s.t. } \theta(\Delta) \in \arg\min_{\theta} \sum_{(x_i,y_i)\in\mathcal{S}} \mathcal{L}(F(x_i+\Delta_i;\theta), y_i), \tag{2}$$

where $\mathcal{C}$ denotes the constraint set which bounds the perturbations so that the perturbed data is perceptually similar to the clean data. In our work, we employ the $\ell_\infty$ constraint $\|\Delta\|_\infty < \epsilon$ as is standard in the adversarial literature (Madry et al., 2017; Zhu et al., 2019; Geiping et al., 2020). Constraining the perturbations in this fashion allows practitioners like social media companies to employ our method in order to release minimally changed user data while still protecting the performance of their proprietary models.

Directly solving for $\Delta$ which minimizes this objective is intractable as this would require backpropagating through the entire training procedure found in the inner objective (2) for each iteration of gradient descent on the outer objective. Thus, the bilevel objective must be approximated.

### 2.2 Crafting Perturbations

It has been demonstrated in Witches' Brew (Geiping et al., 2020) that bounded perturbations to training data can be crafted to manipulate the gradient of a network trained on this data. We adapt gradient manipulation to the problem of general performance degradation. We estimate the outer objective $(1)$ by training a network $F$ on a clean dataset $\mathcal{S}$ and then crafting perturbations to minimize the following objective:

$$\mathcal{A}(\Delta,\theta) = 1 - \frac{\left\langle \nabla_\theta \mathcal{L}'(\mathcal{S};\theta), \nabla_\theta \mathcal{L}(\mathcal{S}+\Delta;\theta) \right\rangle}{\|\nabla_\theta \mathcal{L}'(\mathcal{S};\theta)\|_2 \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S}+\Delta;\theta)\|_2} \tag{3}$$

where $\mathcal{L}'(F(x_i;\theta), y_i)$ is the "reverse" cross entropy loss (Huang et al., 2019) which discourages the network $F$ from classifying $x_i$ with label $y_i$. Specifically, the reverse cross entropy loss for a sample $(x, y)$ with one-hot label $y$ is given by:

$$\mathcal{L}'(F(x;\theta), y) = -\log[1 - p_\theta(x)_{y\neq 0}]$$

where $p_\theta(x)_{y\neq 0}$ denotes the entry of the softmax of $F(x;\theta)$ corresponding to the class specified by the one-hot label $y$. For notational simplicity, we denote the *target gradient*

$$\nabla_\theta \mathcal{L}'(\mathcal{S};\theta) = \nabla_\theta \sum_{(x_i,y_i)\in\mathcal{S}} \mathcal{L}'(F(x_i;\theta), y_i) \tag{4}$$

and the *crafting gradient*

$$\nabla_\theta \mathcal{L}(\mathcal{S}+\Delta;\theta) = \nabla_\theta \left( \sum_{(x_i,y_i)\in\mathcal{S}} \mathcal{L}(F(x_i+\Delta_i;\theta), y_i) \right).$$

Put simply, we seek to align the training gradient of the perturbed data with the gradient of the reverse cross-entropy loss on the clean data, $\mathcal{S}$. Ideally, when a network trains on the perturbed data, the perturbations cause the training gradient of this network at each parameter vector to be aligned with the gradient of the reverse cross entropy loss on the clean data. This in turn would decrease the reverse cross entropy loss on the clean data, causing a network trained on the perturbed data to converge to a minimum with poor generalization on the distribution from which $\mathcal{S}$ was sampled.

In order to enforce the constraints in $\mathcal{C}$, we employ projected gradient descent (PGD) as in Madry et al. (2017) on the perturbations, alternately minimizing Eq. 3 and projecting onto an $l_\infty$ ball. Appendix Algorithm 1 details this procedure.

Additionally, we employ techniques found in previous poisoning work such as restarts and differentiable data augmentation in the crafting procedure in order to improve the success of our perturbations (Huang et al., 2020; Geiping et al., 2020).

We pre-train a model in order to estimate the target gradient, and the crafting gradients as already trained models have been shown to provide the most stable perturbations in Geiping et al. (2020); Huang et al. (2020). Additional details about the training procedure and the crafting procedure can be found in the appendix A.2.

Figure 1: Randomly selected example perturbations to ImageNet datapoint (class "schooner"). **Left**: unaltered base image. **Middle**: $\varepsilon = 8/255$ perturbation. **Right**: $\varepsilon = 16/255$ perturbation.

## 3 Experimental Results

### 3.1 Setup

We establish baselines for our method on both the ILSVRC2012 dataset (ImageNet) (Russakovsky et al., 2015) and the CIFAR-10 dataset (Krizhevsky et al., 2009). Unless otherwise stated, we craft our poisons by choosing ResNet-18 (He et al., 2015) as the architecture for $F$ using $8$ restarts and $240$ optimization steps using signed Adam as in Witches' Brew.

For evaluation, we train a new, randomly initialized network from scratch. We test our method both on the network which was used for crafting, and in a black-box setting where the network architecture is unknown to the practitioner. See subsection A.4 for more details on evaluation.

### 3.2 ImageNet

To test our method on an industrial-scale dataset, we first craft perturbations to ImageNet. In this setting, we deploy a full crafting procedure which mimics the scenario where a company already has a large corpus of data they wish to release. In this case, the company can train the clean model $F$ on this data, and estimate the average target gradient over this training set.

We find that we are able to significantly degrade the validation accuracy, and in the case of the largest perturbation, decrease it by more than $58\%$ (see Table 1). Visualizations for these perturbations can be found in Figure 1.

Table 1: Comparison of validation accuracies of a ResNet-18 (He et al., 2015) trained on perturbed data crafted with different $\varepsilon$-bounds (using ResNet-18) on ImageNet.

| $\varepsilon$-bound | Validation Acc. (%) $\downarrow$ |
|---|---|
| $0/255$ (clean) | $65.70$ |
| $8/255$ | $37.58$ |
| $16/255$ | $27.49$ |

### 3.3 Baseline Comparison

Additionally, we establish baseline comparisons for our method on CIFAR-10 by comparing our method to other poisoning methods, data manipulations, and to performance on clean non-poisoned data. We compare our method to the following alternatives:

**TensorClog** (Shen et al., 2019): We compare our method to TensorClog, which, to our knowledge, is the only previously existing indiscriminate poisoning attack which has been shown to work on modern

deep networks. TensorClog was designed primarily for white-box transfer learning based attacks where a known feature extractor is frozen for evaluation. However, to produce a fair comparison, we re-implement their objective into our crafting regime. This objective aims to cause vanishing training gradients to prevent a network from training properly on the dataset.

**Random Noise**: In order to tease apart the effect of crafted versus non-crafted perturbations on validation accuracy, we also compare our method to fixed, random additive noise. Since our PGD based perturbation usually results in a perturbation close to the "corners" of the $\ell_\infty$ ball, i.e. most pixels are perturbed by the maximum allowed value, we enforce that the noise is of the maximum allowable level for our constraint set.

Comparisons are made with the same $\varepsilon$-bound and the same training procedure from a randomly initialized ResNet-18 model. We find that our alignment method significantly outperforms these alternatives with the same $\varepsilon$-bound. In the case of ResNet18, we degrade validation accuracy by close $50$ more percentage points than TensorClog, the next best.

Table 2: Comparison of different poisoning methods. All methods were employed with $\varepsilon$-bound $8/255$ using the same ResNet-18 architecture and training procedure on CIFAR-10. Confidence intervals are of radius one standard error.

| METHOD | VALIDATION ACC. (%) $\downarrow$ |
|---|---|
| NONE | $93.16 \pm 0.08$ |
| TENSORCLOG | $84.24 \pm 0.17$ |
| RANDOM NOISE | $90.52 \pm 0.08$ |
| ALIGNMENT (OURS) | $\mathbf{36.83 \pm 1.94}$ |

## REFERENCES


Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, November 2010. ISSN 0885-6125. doi: 10.1007/s10994-010-5188-5.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks against Support Vector Machines. *ArXiv12066389 Cs Stat*, June 2012.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pp. 67–74. IEEE, 2018.

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

Yu Cheng, Jian Zhao, Zhecan Wang, Yan Xu, Karlekar Jayashree, Shengmei Shen, and Jiashi Feng. Know you at one glance: A compact vector representation for low-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 1924–1932, 2017.

Valeriia Cherepanova, Micah Goldblum, Harrison Foley, Shiyuan Duan, John Dickerson, Gavin Taylor, and Tom Goldstein. Lowkey: Leveraging adversarial attacks to protect social media users from facial recognition. *arXiv preprint arXiv:2101.07922*, 2021.

Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *arXiv:2012.10544 [cs]*, December 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *ArXiv151203385 Cs*, December 2015.

Kashmir Hill. The secretive company that might end privacy as we know it, Jan 2020. URL `https://www.nytimes.com/2020/01/18/technology/clearview-privacy-facial-recognition.html`.

Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. On the Effectiveness of Mitigating Data Poisoning Attacks with Gradient Shaping. *ArXiv200211497 Cs*, February 2020.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. August 2016.

Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

W Ronny Huang, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations. *arXiv preprint arXiv:1906.03291*, 2019.

W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. MetaPoison: Practical General-purpose Clean-label Data Poisoning. *ArXiv200400225 Cs Stat*, April 2020.

Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger Data Poisoning Attacks Break Data Sanitization Defenses. *ArXiv181100741 Cs Stat*, November 2018.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset.

Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. *ArXiv190309860 Cs*, July 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv170606083 Cs Stat*, June 2017.

Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 51–59, 2017.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pp. 27–38, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5202-4. doi: 10.1145/3128572.3140451.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019.

Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. Deep k-NN Defense against Clean-label Data Poisoning Attacks. *ArXiv190913374 Cs*, March 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis*, 115(3):211–252, December 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *ArXiv180104381 Cs*, January 2018.

Soumyadip Sengupta, Jun-Cheng Chen, Carlos Castillo, Vishal M Patel, Rama Chellappa, and David W Jacobs. Frontal to profile face verification in the wild. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–9. IEEE, 2016.

J. Shen, X. Zhu, and D. Ma. TensorClog: An Imperceptible Poisoning Attack on Deep Neural Network Applications. *IEEE Access*, 7:41498–41506, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2905915.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv14091556 Cs*, September 2014.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified Defenses for Data Poisoning Attacks. June 2017a.

Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified Defenses for Data Poisoning Attacks. In *Advances in Neural Information Processing Systems 30*, pp. 3517–3529. Curran Associates, Inc., 2017b.

C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, Columbus, OH, USA, June 2014. IEEE. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.220.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 601–618, 2016.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral Signatures in Backdoor Attacks. In *Advances in Neural Information Processing Systems 31*, pp. 8000–8010. Curran Associates, Inc., 2018.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pp. 269–277, 2017.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5265–5274, 2018.

Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is Feature Selection Secure against Training Data Poisoning? In *International Conference on Machine Learning*, pp. 1689–1698, June 2015.

ZhaoJ9014. Zhaoj9014/face.evolve.pytorch. URL `https://github.com/ZhaoJ9014/face.evoLVe.PyTorch`.

Chen Zhu, W. Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. *ArXiv190505897 Cs Stat*, May 2019.

# A APPENDIX

## A.1 ALGORITHM

**Algorithm 1** Crafting perturbations

**Require** pre-trained clean network $\{F(\cdot, \theta)\}$, a dataset $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N}$, perturbation bound $\varepsilon$, restarts $R$, optimization steps $M$
**Begin** Compute $\nabla_\theta \mathcal{L}'(\mathcal{S}; \theta)$ i.e. the target gradient.
**For** $r = 1, \ldots, R$ restarts:
    Randomly initialize perturbations $\Delta^r \in \mathcal{C}$
    **For** $j = 1, \ldots, M$ optimization steps:
        Apply data augmentation to samples $(x_i + \Delta_i^r)_{i=1}^{N}$
        Compute alignment loss, $\mathcal{A}(\Delta^r, \theta)$ as in Eq. 3
        Update $\Delta^r$ with a step of signed Adam
        Project onto $||\Delta^r||_\infty \leq \varepsilon$
Choose $\Delta^*$ as $\Delta^r$ with minimal value in $\mathcal{A}(\Delta^r, \theta)$
**Return** perturbations $\Delta^*$

## A.2 TRAINING DETAILS

**CIFAR-10** For our CIFAR-10 experiments, we train a ResNet18 model for 40 epochs in order to craft the perturbations. The model is trained with SGD and multi-step learning rate drops. However, for the experiments in Table 8, since fewer data points are used for training the surrogate model, we increase the number of epochs, so the number of iteration is similar to regular training. For example, if we use only 10% of the data, then we would increase the number of epochs by 10 times, so that the model is trained for a similar number of iterations.

Tables 2, 5, 9 all train a randomly initialized network using this same framework (grey-box).

However, in Tables 3, 4 8, we use the black-box repository referenced in the main body to train models. We train for 50 epochs using this repository's setup (cosine learning rate decay, etc.).

**ImageNet** For our ImageNet experiments, we use a pretrained ResNet18 model to craft poisons, and then train a new, randomly initialized ResNet18 model on these crafted poisons for 40 epochs with multi-step learning rate drops.

**Facial Recogntion** For our Celeb-A experiments, we train a ResNet18 and a ResNet50 for 125 epochs as surrogate models, which are used to craft poisons. We then train a new randomly initialized ResNet18 model on these crafted poisons for 125 epochs with multi-step learning rate drops.

**DPSGD Defense** For the DPSGD defense, we first clip employ clipping of $0.1$ and then add noise to the gradients with parameter $\sigma = \text{clip} * 0.01$.

## A.3 HARDWARE AND TIME CONSIDERATIONS

We primarily use an array of GeForce RTX 2080 Ti graphics cards. On $4$ GPUs, pre-training the crafting model on CIFAR-10 typicaly takes $12.5$ minutes. Crafting the perturbations typically takes $140$ minutes per restart for the entirety of CIFAR-10 $(50,000$ images). We typically run $240$ iterations of projected gradient descent. On ImageNet, a batch of $25,000$ Images typically takes just under $17$ hours to craft in a similar manner.

## A.4 RESULTS IN THE BLACK-BOX SETTING

The results in Tables 1 and 2 are in a grey-box setting where we do not know the victim's model initialization or any information about batching, but we do know their architecture training details like optimizer choice. In order to more strenuously test our method in a realistic setting, we also test our poisons on architectures and training procedures completely unknown during poison crafting. This mimics the black-box setting wherein the practitioner does not know how a victim plans to train

on their scraped data. Specifically, we train networks on our crafted poisons using the setup from an independent, well known repository for training CIFAR-10 models [1].

We validate our method on commonly used architectures including VGG19 (Simonyan & Zisserman, 2014), ResNet-18 (He et al., 2015), GoogLeNet (Szegedy et al., 2015), DenseNet121 (Huang et al., 2016), and MobileNetV2 (Sandler et al., 2018). These results are presented in Table 3. We see that even at very small epsilon constraints, the proposed method is able to nearly halve the clean validation accuracy of many of these popular models.

Table 3: Comparison of CIFAR-10 validation accuracies (%) for different $\varepsilon$-bounds and victim networks. All poisons crafted using a ResNet-18 architecture.

| NETWORK | $0/255$ (CLEAN) | $4/255$ | $8/255$ | $16/255$ |
|---|---|---|---|---|
| VGG19 | $90.76 \pm 0.14$ | $66.58 \pm 0.58$ | $48.27 \pm 0.92$ | $31.86 \pm 1.47$ |
| RESNET-18 | $93.16 \pm 0.08$ | $68.15 \pm 0.55$ | $55.34 \pm 0.80$ | $43.14 \pm 1.60$ |
| GOOGLENET | $93.87 \pm 0.07$ | $71.02 \pm 0.15$ | $55.49 \pm 0.58$ | $45.37 \pm 0.94$ |
| DENSENET121 | $93.80 \pm 0.05$ | $70.12 \pm 0.12$ | $49.23 \pm 1.58$ | $38.51 \pm 1.07$ |
| MOBILENETV2 | $91.10 \pm 0.10$ | $66.71 \pm 0.60$ | $51.03 \pm 0.86$ | $39.19 \pm 1.76$ |

### A.5 STABILITY

While our method is able to degrade validation accuracy under normal training conditions, a question remains whether the method is stable to poisoning defenses and modifications in training procedures. Would this improve the validation accuracy of a model trained on the perturbed data? To this end, we investigate several avenues.

First, we investigate whether existing poisoning defenses lessen the effects of our perturbations. Many existing defenses are designed for settings which are significantly different than our proposed attack. For example, many defenses assume that there is a small amount of poisoned data, and that this will be anomalous in feature space (Steinhardt et al., 2017a; Tran et al., 2018; Peri et al., 2020; Chen et al., 2018). These types of defenses are best suited for scenarios in which the same perturbations are applied identically to each data point, as in patch based attacks, or when the defender has access to a large corpus of trustworthy data on which to train a feature extractor to filter out poisoned images based on their similarity to clean reference samples. Moreover, these anomaly detection defenses work under the assumption that the majority of data will not be poisoned. It was demonstrated in (Geiping et al., 2020) that modifications meant to alter gradients of a network trained from scratch do not produce data which is anomalous in feature space of the poisoned model. Furthermore, since we poison the entire dataset, the trained model's feature space can no longer be thought of as containing clean and perturbed elements. Thus, the heuristic that poisoned data will somehow be outliers in feature space does not apply.

However, another family of defenses leverages differential privacy (DPSGD) as a defense against poisoning (Ma et al., 2019; Hong et al., 2020). Since differentially private models are, by construction, insensitive to minor changes in the training set, they may be resistant to different poisoning methods. By clipping and noising gradients, these defenses aim to limit the effect of perturbations placed on data. In theory, this defense is applicable to our perturbations. However, we test the defense proposed in Hong et al. (2020) against our method and find that the drop in validation accuracy we saw in previous experiments remains even when training with DPSGD.

In addition to methods designed for defending against data poisoning, we also test the potency of our poisons under both Gaussian smoothing and random additive noise (of the same magnitude as our perturbations) during training. These modifications test how brittle our crafted perturbations are to modification. For Gaussian smoothing, we use a radius $r = 2$. We find that our attack is stable under all the discussed training modifications, with none of the proposed defenses substantially improving results. These results can be found in Table 5.

[1] Training routine taken from widely used repository: https://github.com/kuangliu/pytorch-cifar.

Table 4: Comparison of CIFAR-10 validation accuracies (%) for different $\varepsilon$-bounds and victim networks. All poisons crafted using a ResNet-18 architecture with estimated target grad.

| NETWORK | $4/255$ | $8/255$ | $16/255$ |
|---|---|---|---|
| VGG19 | 74.6 | 56.57 | 30.02 |
| RESNET-18 | 74.11 | 56.65 | 29.62 |
| GOOGLENET | 76.41 | 62.44 | 33.68 |
| DENSENET121 | 75.61 | 57.29 | 31.51 |
| MOBILENETV2 | 71.54 | 49.14 | 24.45 |

Table 5: Comparison of validation accuracies of a ResNet-18 with different defenses. All runs use $\varepsilon = 8/255$.

| DEFENSE | VALIDATION ACC. (%) $\downarrow$ |
|---|---|
| NONE | 44.82 |
| DPSGD | 44.18 |
| RANDOM $\ell_\infty$ NOISE | 48.36 |
| GAUSSIAN SMOOTHING | 24.26 |

### A.6 FACIAL RECOGNITION

Figure 2: Samples of poisoned CelebA images. **Top**: unaltered images. **Middle**: $\varepsilon = 8/255$. **Bottom**: $\varepsilon = 16/255$.

While standard classification tasks like ImageNet and CIFAR allow us to establish baselines for our method, many settings where a company may wish to implement our method involve social media user data, often in the form of personal photos. Such data is may be scraped from large social media platforms by competing companies and nefarious actors (Cherepanova et al., 2021). For example, companies like Clearview AI scrape photos from social media sites to train their own facial recognition systems for mass surveillance (Hill, 2020). A social media company could even deploy our method simply on thumbnail profile images, which are publicly available to scraping. To determine the utility of our method in preventing unauthorized use in this manner, we deploy our algorithm on facial recognition benchmarks.

By and large, facial recognition works in the regime of transfer learning. Facial recognition models are often pre-trained on a many-way classification problem using images unrelated to the testing identities. Then, during testing, the classification head is removed, and the pre-trained model is used only for calculating the test image embeddings. The identity of test images is then inferred by $k$-nearest neighbors in the embedding space. This setup is known to make attacks against facial recognition systems more difficult than attacks against standard classification tasks (Cherepanova et al., 2021). Evasion attacks like the one found in Cherepanova et al. (2021) modify images at test-time. We deploy our method on the complementary task of degrading the quality of the feature

Table 6: Identification and verification accuracy of ResNet-18 trained on clean data and poisoned data of varying $epsilon$ radius. Note that in all cases, both identification and verification accuracy drop substantially when the model is trained on poisoned data.

| | IDENTIFICATION | | VERIFICATION | | | |
|---|---|---|---|---|---|---|
| $\varepsilon$ | CELEBA TOP1 | CELEBA TOP5 | LFW | CFP | AGEDB | VGG2_FP |
| CLEAN | 91.02% | 94.65% | 97.93% | 83.84% | 85.40% | 84.96% |
| 8/255 | 85.54% | 91.67% | 95.20% | 76.40% | 76.88% | 81.88% |
| 16/255 | 61.53% | 73.67% | 76.03% | 62.81% | 59.45% | 61.78% |
| 32/255 | 39.77% | 54.37% | 70.55% | 62.94% | 59.87% | 59.32% |

Table 7: Poisons generated from more a powerful model also result in a stronger attack. Here, we attack a ResNet-18 model with poisons generated with two different surrogate models at $\epsilon = 8/255$

| | IDENTIFICATION | | VERIFICATION | | | |
|---|---|---|---|---|---|---|
| SURROGATE MODEL | CELEBA TOP1 | CELEBA TOP5 | LFW | CFP | AGEDB | VGG2_FP |
| NO POISONS | 91.02% | 94.65% | 97.93% | 83.84% | 85.40% | 84.96% |
| RESNET-18 | 85.54% | 91.67% | 95.20% | 76.40% | 76.88% | 81.88% |
| RESNET-50 | 79.53% | 86.88% | 86.75% | 74.64% | 63.78% | 70.52% |

extractor. This adds a layer of difficulty to our attack, as the we are only able to affect the quality of the embedding, but we are not able to perturb the "anchor" images used to classify new test samples.

### A.6.1 SETUP

The CelebA dataset contains 10177 identities (Liu et al.). Following standard procedure as in (Cheng et al., 2017), we remove 371 identities with too few images, use 8806 identities for pre-training and 1000 identities for testing. We use ResNet-18 and Resnet-50 as backbone architectures. During pre-training, we use the popular *Cosface* classification head (Wang et al., 2018). During poison dataset generation, we take 50 gradient steps with signAdam and a single random restart. In additional to the CelebA dataset, we also test the attacked model's verification accuracy on four other face datasets: Labeled Faces in the Wild (LFW) (Huang et al., 2008), Celebrities in Frontal-Profile data set (CPF) (Sengupta et al., 2016), AgeDB (Moschoglou et al., 2017), VGGFace2 (Cao et al., 2018). We use the *face.evoLVe* repository (ZhaoJ9014) to run all of our facial recognition experiments.

### A.6.2 RESULTS

In Table 6, we see that both identification and verification accuracy drop materially when a model is trained on our crafted dataset. When trained on poisoned data with $\epsilon = 16/255$, CelebA top 1 accuracy drops by 36% to 61%, . Similarly, verification accuracy also drops by as much as 35%. Note that in many commercial applications, even a few percentage drop can tip the scales for a company to maintain a competitive advantage.

Poisons crafted on a more powerful model also transfer better in our experiments. Specifically, poisons generated with ResNet-50 always reduce both identification and verification accuracy more than poisons generated with ResNet-18 (see Table 7). This experiment suggests that if one wants to make the poisoned dataset more potent, one could simply use a larger capacity model to generate poisons.

## A.7 ONLINE MODIFICATION

The procedure outlined in Algorithm 1 works well when a practitioner already has access to a large corpus of data they intend to release. In this case, they can train a network to estimate the target gradient in Eq. 4, and optimize all poisons jointly to align with this target gradient. However, for many applications, like social media release, this may not be possible. First, the initial dataset may not be big enough to allow one to train a well performing model to estimate the target gradient

accurately. Second, the poisoned dataset may have to be optimized sequentially as new data continue to be released or independently at the user level.

To understand the severity of the mentioned problems, we perform several ablations and modifications to our method. First, we test whether the effective perturbations can be made with an approximated target gradient. We already estimate the target gradient on the distribution $\mathcal{D}$ by calculating the empirical target gradient on $\mathcal{S}$, but if a practitioner has access to only a small amount of data, this could degrade the quality of the target gradient to the point where perturbations become ineffective.

To test this, we first re-run experiments outlined in Table 3, but with a target gradient approximated from a random subset of $10\%$ of the data. We find that perturbations remain effective when using this approximated target gradient. In some cases, the approximated target gradient even produces better results than the full target gradient. In theory, as long as the estimated empirical target gradient "well" approximates the full empirical target gradient (i.e. the cosine similarity between the two is high), then perturbations crafted using the former will produce similar performance to those crafted on the latter since the target gradient is not differentiated through when crafting the perturbations. These results give confidence to practitioners who wish to poison a stream of data without having access to a large amount at the start of the process.

Additionally, we test the poisons effectiveness when the surrogate model is trained with only a small subset of the data and the target gradient is estimated only with the same small subset. This is equivalent to splitting the dataset into different subsets $\{\mathcal{S}_i\}_{i=1}^{M}$ where $\mathcal{S}_i \subset S$ and poisoning each of these subsets as if it were its own standalone dataset. In theory, this could harm performance since the model trained on a small subset of the data will usually perform poorly compared to a model trained on the entire dataset. This could lead to a bad target gradient estimate, and in turn, poorly performing poisons. However, even though the surrogate model's performance is indeed poorer, we find that the poisons generated with such a model are still effective in practice at decreasing validation accuracy. More specifically, in Table 8, we see that when using only 10% of data for estimating the target gradient, we can decrease the attacked model's accuracy to a comparable level as using 100% of the data.

Finally, we test the effectiveness of the perturbations when the poisons are optimized independently as opposed to jointly. This is a practical consideration for many companies since user data could be uploaded at any given moment in time, and the practitioner might not be able to wait for a batch of data to calculate perturbations. Thus, it is necessary to be able to calculate perturbations one at a time.

To mimic this scenario in our implementation, we simply detach the denominator in Eq. 3 from the network's computation graph. Note that this change modifies the objective problem in Eq. 3 to optimize the inner product of the target gradient and each *individual* crafting gradient versus the cosine similarity between the target gradient and the *full* crafting gradient.

We find that poisons can be just as effective, sometimes more so, when optimized independently (see Table 8). Mathematically speaking, as long as the norm of the crafting gradient in Eq. 3 is stable to small changes in any individual perturbation, then detaching the denominator will not affect the perturbations as we optimize with a signed gradient method (signed Adam) in Alg. 1.

To further demonstrate this idea formally, we present the following straightforward result:

**Proposition 1.** *Fix a pixel position denoted by $*$. If $\exists \varepsilon > 0$ so that $\forall x_j \in \mathcal{S}$, in the $\ell_\infty$-ball about $x_j$ of radius $\varepsilon$, the following inequality holds:*

$$\begin{aligned} &\left| \frac{\partial}{\partial \Delta_j^*} \left( \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|_2 \right) \right| \\ < &\left| \frac{\partial}{\partial \Delta_j^*} \left( \left\langle \mathcal{T}, \nabla_\theta \mathcal{L}(x_j + \Delta_j; \theta) \right\rangle \right) \right| \end{aligned}$$

*i.e. - The derivative w.r.t. $\Delta_j^*$ of the norm of the full crafting gradient is bounded in magnitude by the derivative w.r.t $\Delta_j^*$ of the inner product between the individual crafting gradient and target gradient, $\mathcal{T}$,*

*Then, our online crafting mechanism produces the same perturbation to pixel $\Delta^*$ (in the $\varepsilon$-ball) as the full non-online version.*

*Proof.* Recall the alignment loss:

$$\mathcal{A}(\Delta, \theta) = 1 - \frac{\left\langle \nabla_\theta \mathcal{L}'(\mathcal{S}; \theta), \nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta) \right\rangle}{\|\nabla_\theta \mathcal{L}'(\mathcal{S}; \theta)\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|} \tag{5}$$

As the left hand term in the inner product, the target gradient, does not depend on the perturbations $\Delta$, it may be treated as a constant, and denoted simply as $\mathcal{T}$. As in our algorithm, we denote $\|\cdot\| = \|\cdot\|_2$. Also, WLOG, we need only look at the sign of the following derivative for some arbitrary perturbation $\Delta_j^*$:

$$\frac{\partial}{\partial \Delta_j^*} \left[ \overbrace{\frac{\left\langle \mathcal{T}, \nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta) \right\rangle}{\|\mathcal{T}\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|}}^{\alpha} \right]$$

On the one hand, if we detach the denominator from the computation graph, the following derivative is used to update the perturbation $\Delta_j$:

$$\frac{\frac{\partial}{\partial \Delta_j^*} \left( \left\langle \mathcal{T}, \nabla_\theta \mathcal{L}(x_j + \Delta_j; \theta) \right\rangle \right)}{\|\mathcal{T}\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|}$$

On the other hand, the gradient of the full objective is:

$$\overbrace{\frac{\frac{\partial}{\partial \Delta_j^*} \left( \left\langle \mathcal{T}, \nabla_\theta \mathcal{L}(x_j + \Delta_j; \theta) \right\rangle \right)}{\|\mathcal{T}\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|}}^{\beta}$$

$$\underbrace{- \frac{\left\langle \mathcal{T}, \nabla_\theta \mathcal{L}(x_j + \Delta_j; \theta) \right\rangle \|\mathcal{T}\| \cdot \frac{\partial}{\partial \Delta_j^*} \left( \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\| \right)}{\left( \|\mathcal{T}\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\| \right)^2}}_{\gamma}$$

Note that $\beta$ is simply the derivative of the detached objective, so the perturbations will be the same if

$$|\gamma| < |\beta|$$

Simplifying,

$$\begin{aligned} |\gamma| &= |\alpha| \cdot \frac{\left| \frac{\partial}{\partial \Delta_j^*} \left( \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\| \right) \right|}{\|\mathcal{T}\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|} \\ &\leq \frac{\left| \frac{\partial}{\partial \Delta_j^*} \left( \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\| \right) \right|}{\|\mathcal{T}\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|} \\ &< \frac{\left| \frac{\partial}{\partial \Delta_j^*} \left( \left\langle \mathcal{T}, \nabla_\theta \mathcal{L}(x_j + \Delta_j; \theta) \right\rangle \right) \right|}{\|\mathcal{T}\| \cdot \|\nabla_\theta \mathcal{L}(\mathcal{S} + \Delta; \theta)\|} = |\beta| \end{aligned}$$

Where the last inequality uses the assumption of the proposition, and holds in the $\varepsilon$-ball around $x_j$. Then, because our crafting algorithm uses signed gradient descent (either Adam or SGD), the perturbations crafted using the online vs. non-online method will be identical.

□

Table 8: **% of data used** indicates the percentage of data used for both training the surrogate model and target gradient estimation. The denominators of Eq. 3 is detached when the poisons are independently crafted. We note that we can still double the validation error of the attacked model when only 5% of data is used.

| % OF DATA USED | INDEPENDENTLY CRAFTED | POISONED VALIDATION ACCURACY (%) |
|---|---|---|
| 100% | NO | 46.66 |
| 10% | NO | 44.09 |
| 5% | NO | 74.64 |
| 10% | YES | 36.42 |
| 5% | YES | 59.59 |

Note that for an alternate presentation of the proposition, we may write the bound on the magnitude of the derivative of the full crafting gradient in the following manner:

$$\begin{aligned} &\left|\frac{\partial}{\partial \Delta_j^*}\left(\|\nabla_\theta \mathcal{L}(\mathcal{S}+\Delta;\theta)\|\right)\right| \\ &< c_0\left|\frac{\partial}{\partial \Delta_j^*}\left(\|\nabla_\theta \mathcal{L}(x_j+\Delta_j;\theta)\| \cdot \cos\phi\right)\right| \end{aligned}$$

Where $\phi$ is the angle between the individual crafting gradient and the fixed target gradient, and $c_0 = \|\mathcal{T}\|$ is the norm of the target gradient.

This adjustment makes the algorithm practical for real world use as the poisons can be generated at the user level in a distributed setting without a centralize poison generation process.

### A.8 REGULARIZATION

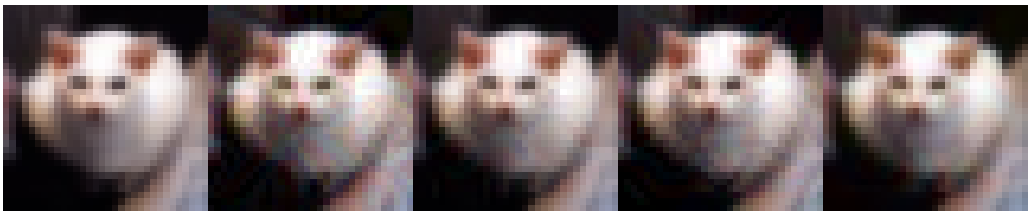

Figure 3: Example CIFAR-10 Image crafted with different regularizers. From left to right: clean image, no regularizer, $\ell_2$ regularization, SSIM regularization, TV regularization. All crafted with perturbation bound $\varepsilon = 8/255$.

While projecting onto an small $\ell_\infty$ ball enforces that the perturbations are not overly conspicuous, for some applications, like user uploaded images, further regularization to impose visual similarity may be desirable. Thus, we test a variety of regularization terms to the alignment loss in Eq. 3. Specifically, we test a straightforward $\ell_2$ penalty on the norm of the perturbation, a total variation penalty (TV), and a structural similarity (SSIM) regularizer which has been shown to increase visible quality of perturbed data (Cherepanova et al., 2021). We find that the regularizers decrease the visibility of the perturbations in exchange for an increase in validation accuracy. Thus, a practitioner can choose the strength of the regularizer to control the tradeoff between visibility of the perturbation and success of the poisons. Effects of the regularizers compared to an unregularized validation run can be found in Table 9. Additionally, visualizations of images produced using the regularization terms can be found in Figure 9.

Table 9: Comparison of validation accuracy of ResNet-18 trained on poisons generated using various regularization terms. All runs use $\varepsilon = 8/255$.

| REGULARIZER | VALIDATION ACC. (%) $\downarrow$ |
|---|---|
| NONE | 44.82 |
| $\ell_2$ | 69.8 |
| SSIM | 51.09 |
| TV | 53.39 |