NUS | Computing
National University of Singapore

CS5562: Trustworthy Machine Learning

Robustness → Data Poisoning Attacks

Reza Shokri[a]

Aug 2023

[a]Acknowledgment. The wonderful teaching assistants: Hongyan Chang, Martin Strobel, Jiashu Tao, Yao Tong, Jiayuan Ye

Contents

Intro to Data Poisoning Attacks

Machine Learning in the Benign Setting

- Consider a **classification task** over input space $\mathcal{X} = \mathbb{R}^d$ to a discrete set of classes $\mathcal{Y}$.
- We are given a set of samples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ from a **population distribution** D over $\mathcal{X} \times \mathcal{Y}$.
- Our goal is to learn a classifier $f : \mathcal{X} \to \mathcal{Y}$ that minimizes the **expected loss** (risk) over distribution D, given the training set $\mathcal{D}$.

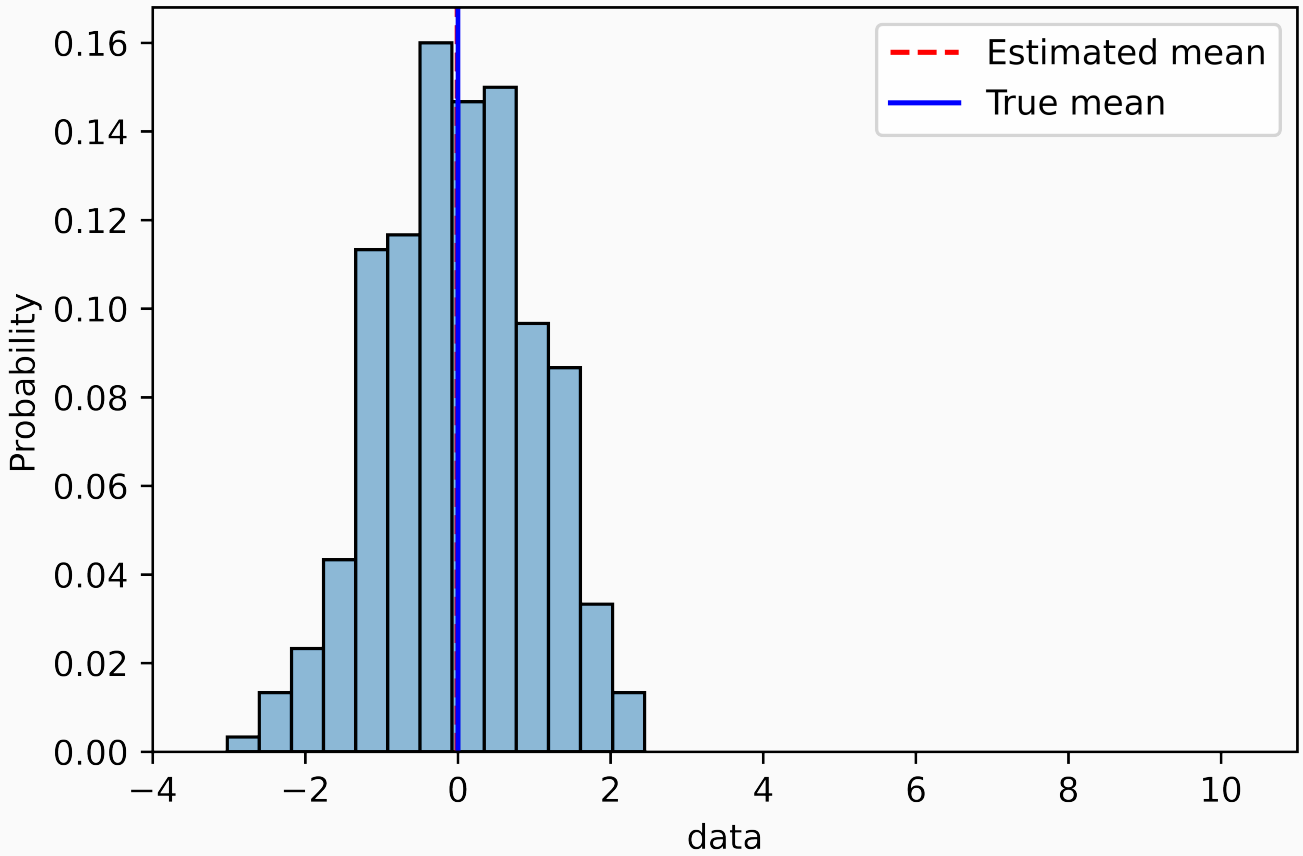
$$\min_{\theta} \qquad R(f, \theta) = \mathbb{E}_{(x,y)\sim D}\left[l(f(x;\theta), y)\right] \approx \frac{1}{n} \sum_{(x,y)\in\mathcal{D}} \left[l(f(x;\theta), y)\right]$$

- Example of loss function l is a $0/1$ loss $l_{0/1} = \mathbb{1}[y \neq y']$.
- **Inference** is computing $f(x; \hat{\theta})$ for any $x \in \mathcal{X}$. Implicit assumption is that the input x is sampled from the population distribution D.
- Challenge is to make sure the model **generalizes** to samples from D.

Data Poisoning

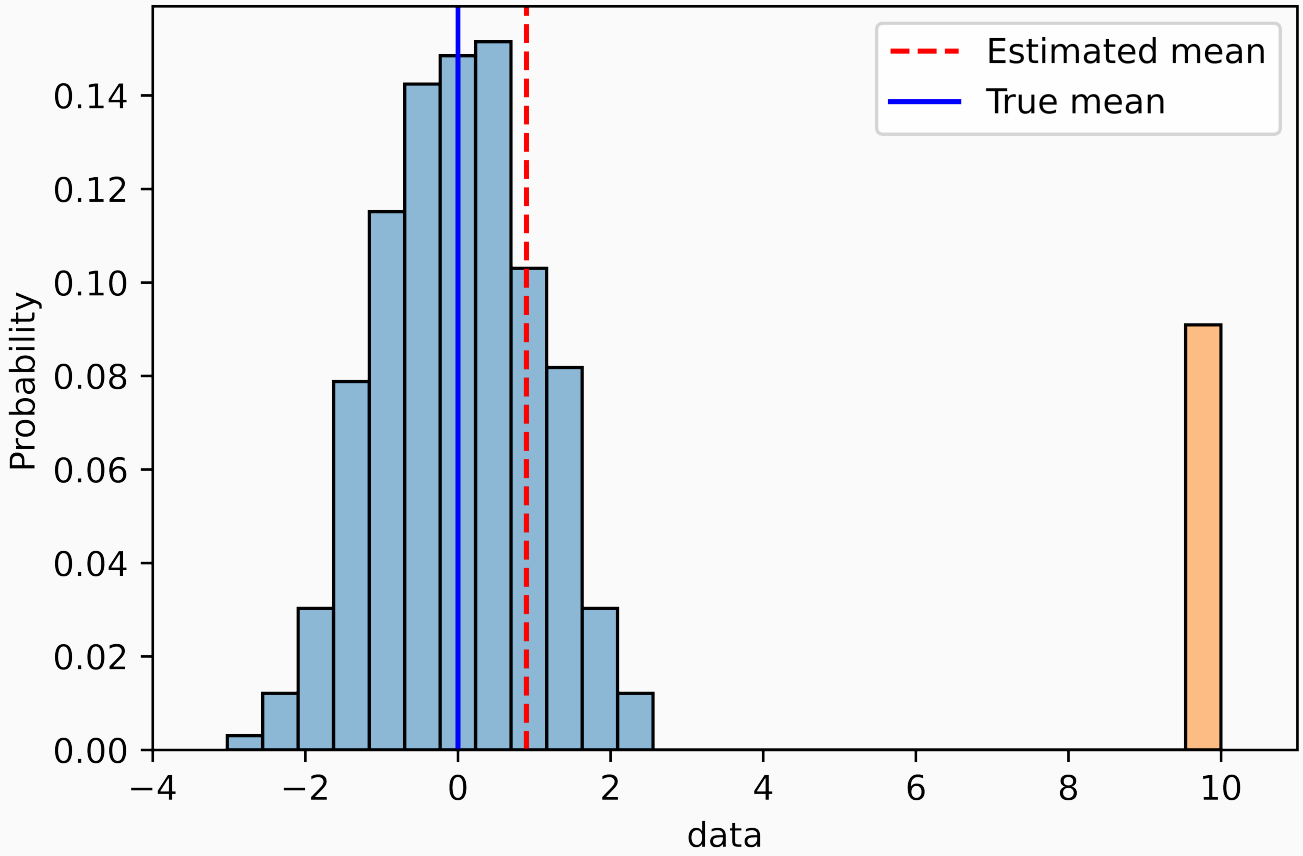
- What is the threat model?
  - Adversary can influence the training data: They can contribute some data (called poisoned data) to the training set. This is a very realistic setting, as data collection involves obtaining data from different (potentially adversarial) sources.
  - So, training set $\mathcal{D}$ contains the poisoned set $\mathcal{D}^{(p)} = \{(x_1^{(p)}, y_1^{(p)}), (x_2^{(p)}, y_2^{(p)}), \cdots, (x_k^{(p)}, y_k^{(p)})\}$
- What is the objective of the attacker?
  - To change the predictions of the model at inference time, on **targeted** data points, or on the whole data **distribution**.
- What is the defense?
  - Robust machine learning minimizes the **influence** of adversarial (out of distribution) data points on model parameters (we see in the assignment).
- What would be the behavior of the training algorithms with respect to poisoned data in the simple setting?

Mean estimation problem



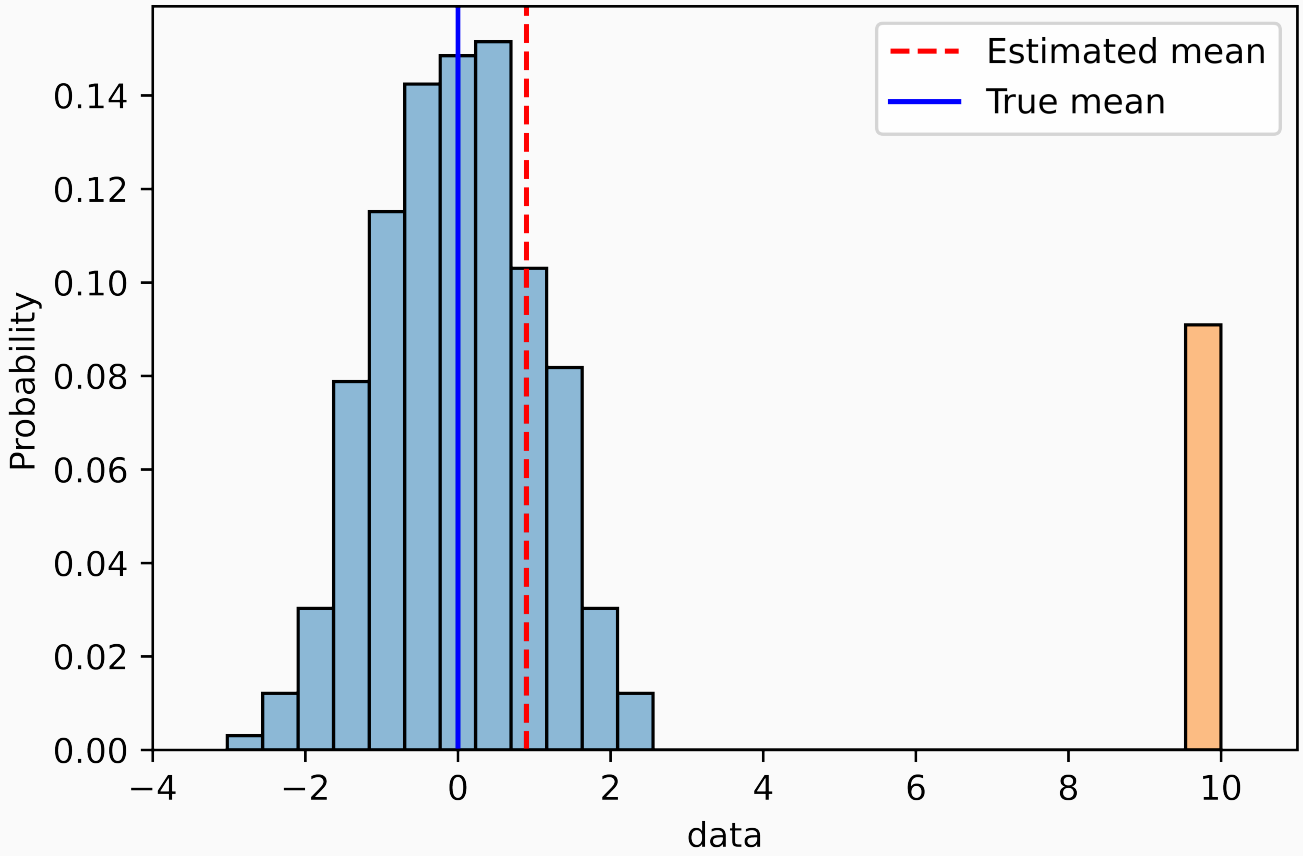
Empirical Mean is a good estimator of the population mean when there is no poisoned data

Mean estimation problem



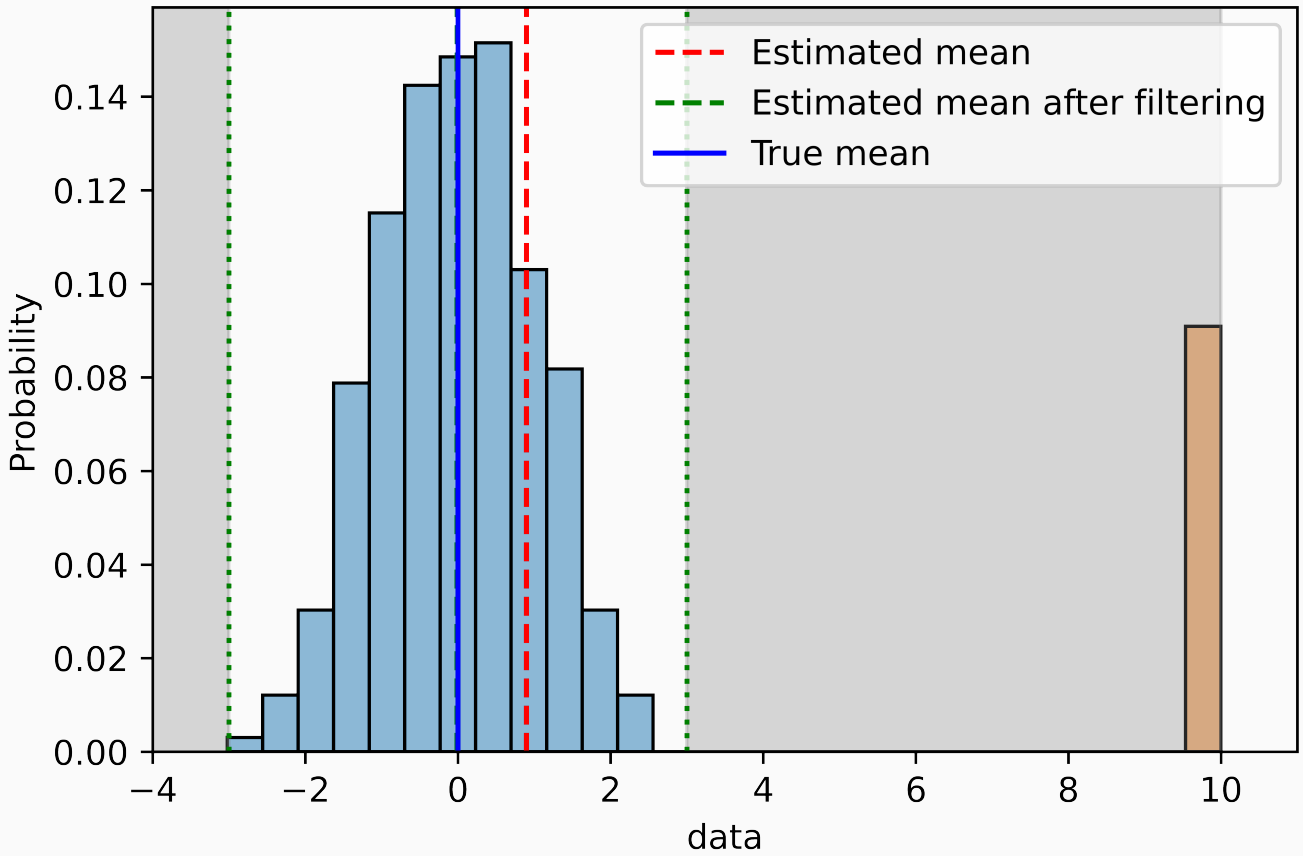
Empirical mean is not robust against poisoned data.

Mean estimation problem



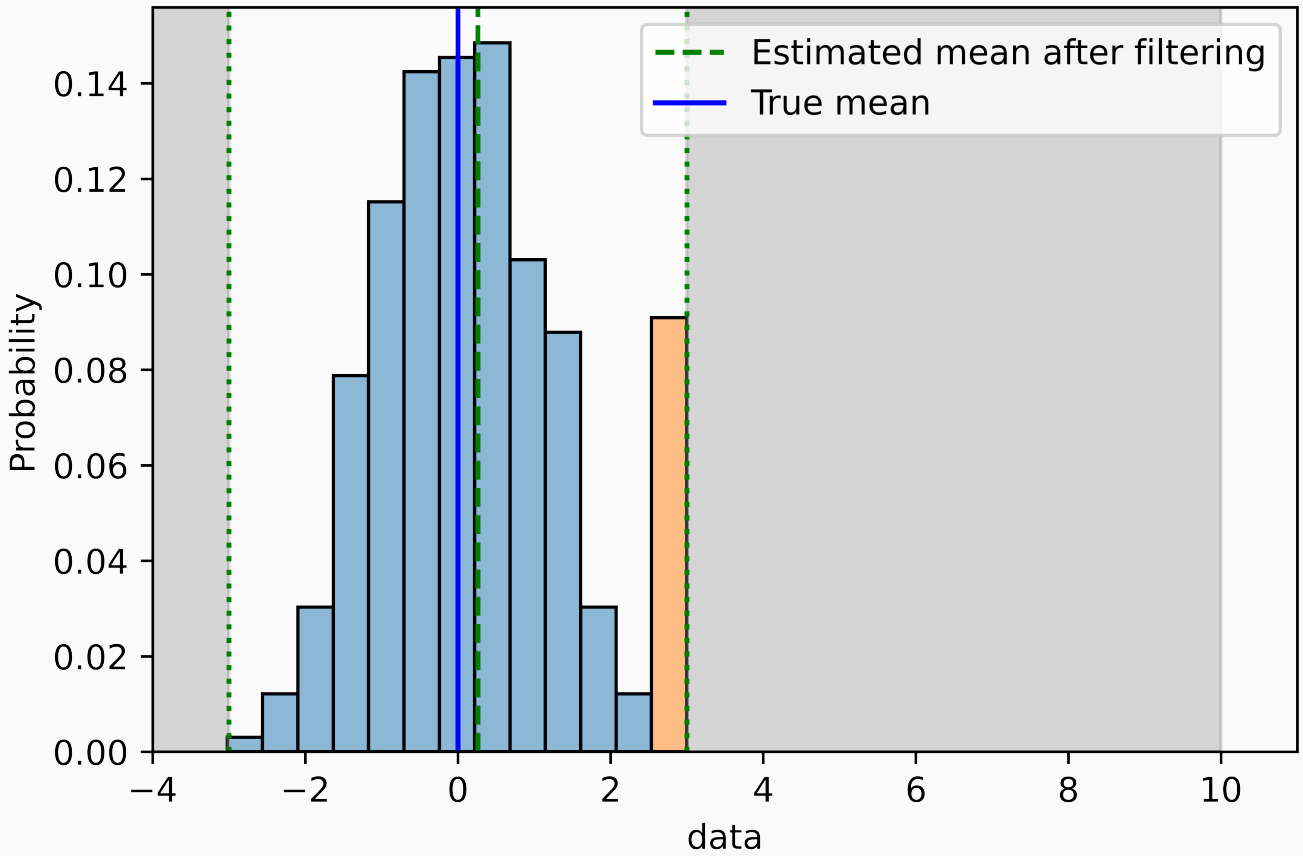
Empirical mean is not robust against poisoned data.

Mean estimation problem



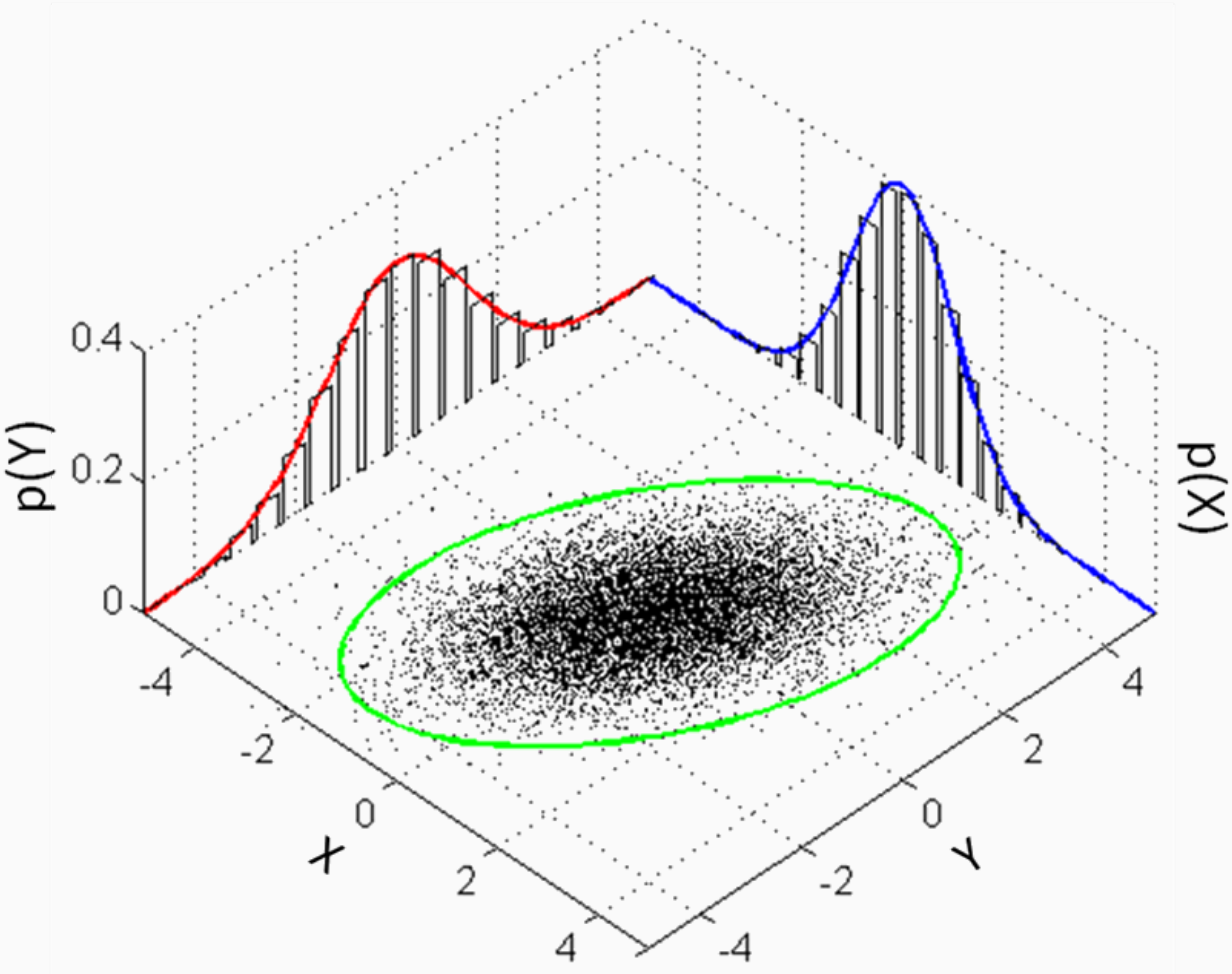
Filtering points that are far away from the rest helps to recover the true mean. But is it a robust solution?

Mean estimation problem



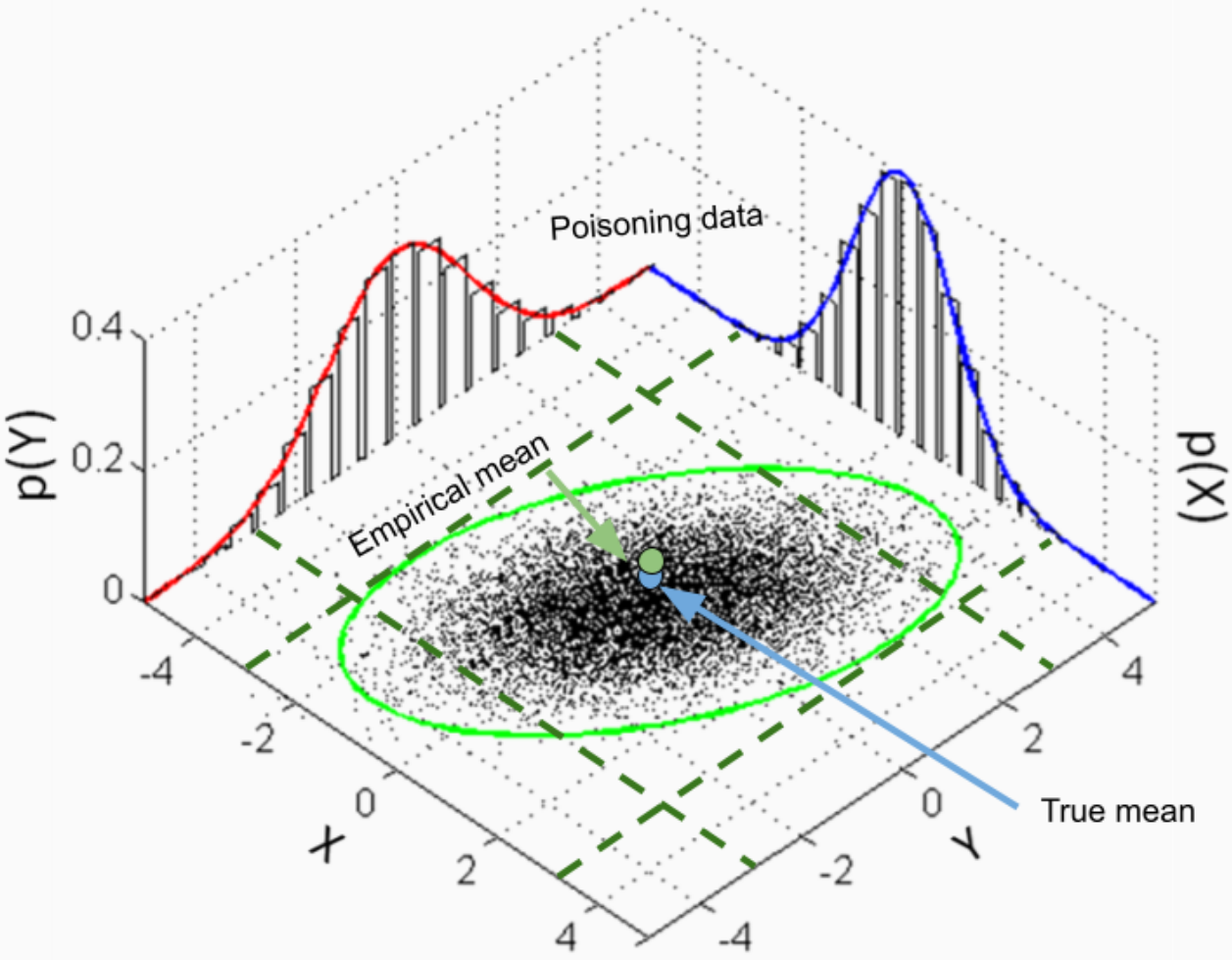
Adaptive attack can still move the empirical mean away from the true mean.

Mean estimation problem in the high dimensional space



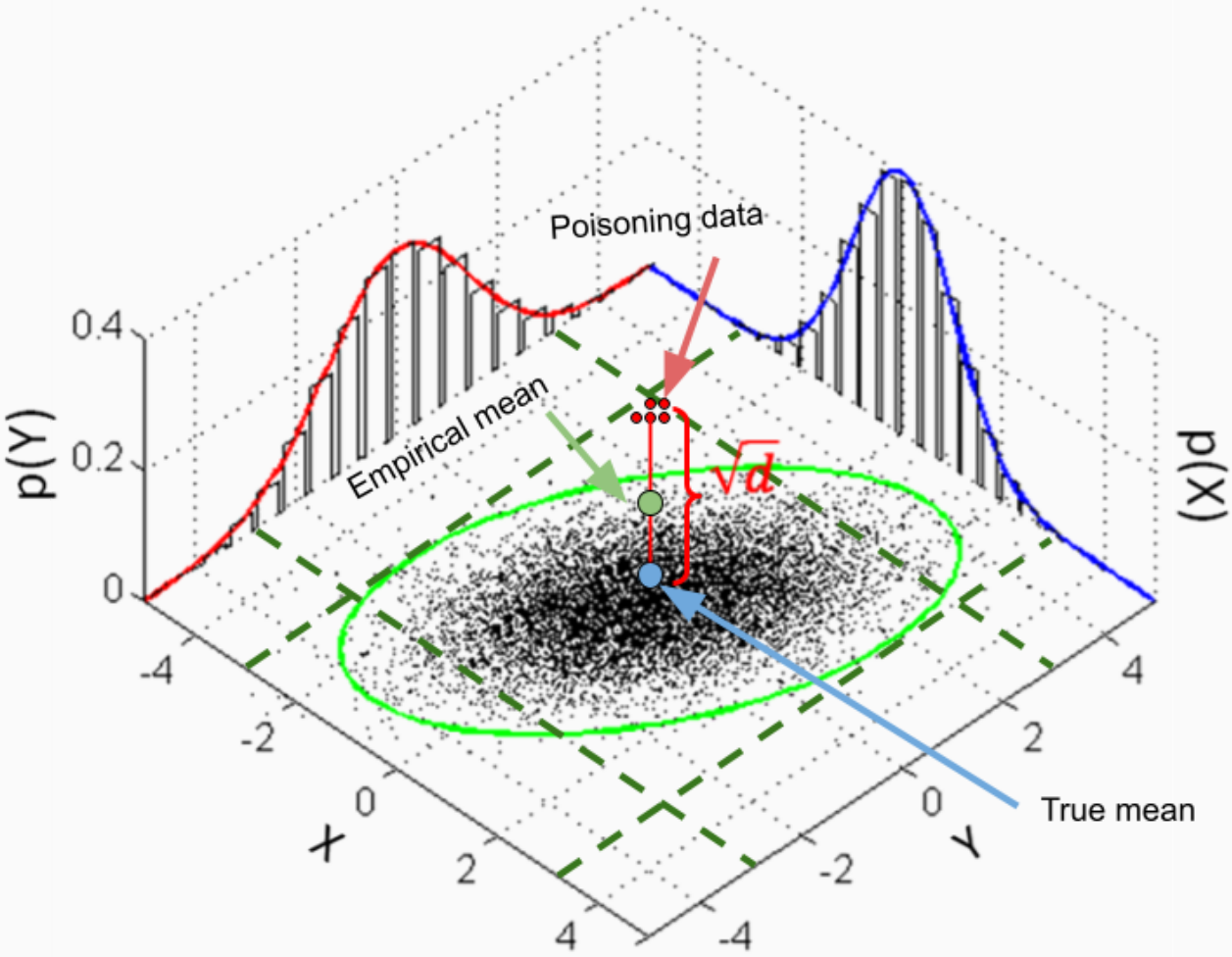
Source: MultivariateNormal

Mean estimation problem in the high dimensional space



Source: MultivariateNormal

Mean estimation problem in the high dimensional space

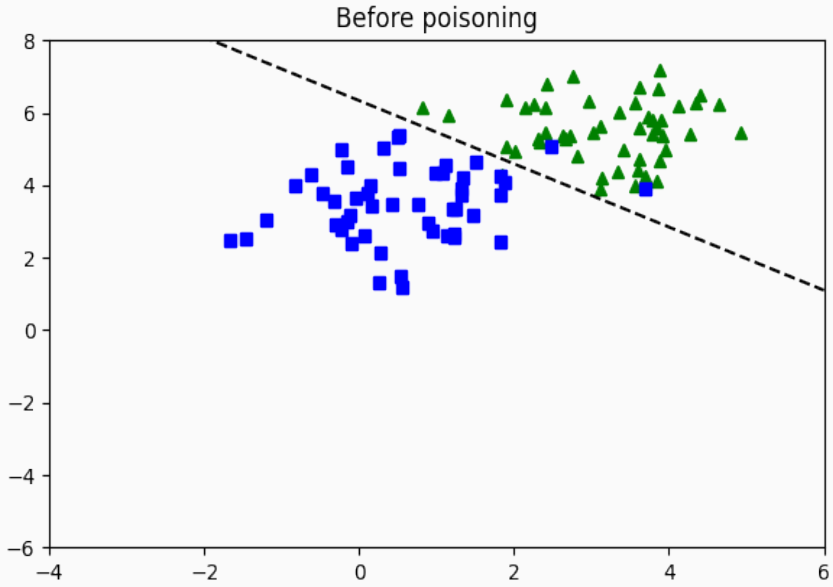


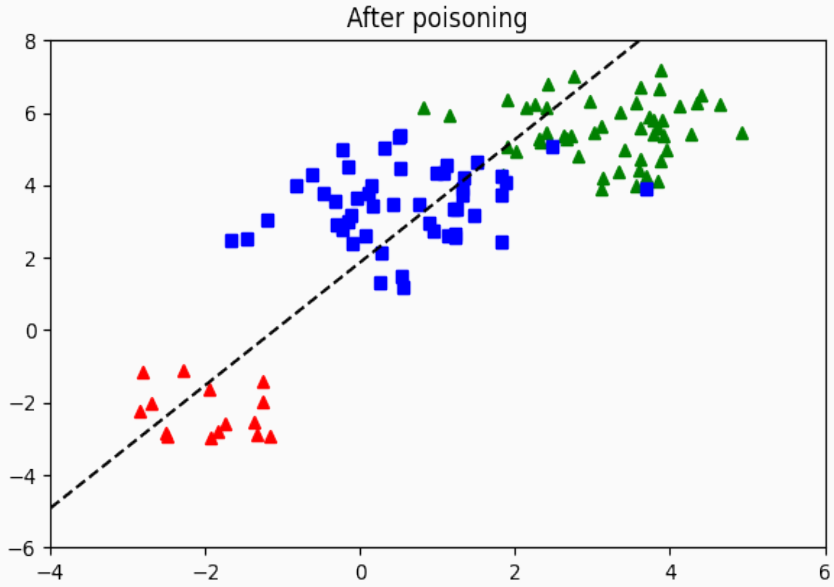
Detecting poisoned data for the mean estimation problem in the high dimensional space is even harder.

Source: MultivariateNormal

Poisoning attacks against ML model

Poisoning attack against SVM model (linear kernel)



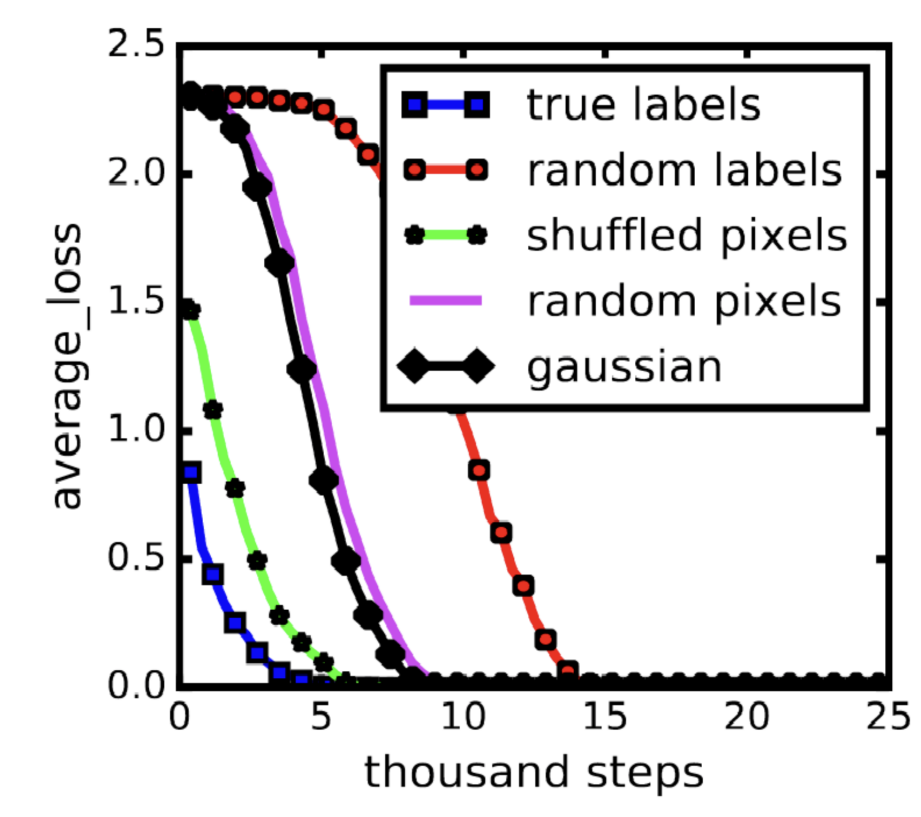


The poisoning attack is effective for simple machine learning models.

Deep Learning from Random Data i

- Let us study training on random data. Here is how we generate data to train models using deep learning [Zhang et al., 2021].
  - Partially corrupted labels: independently with probability p, the label of each image is corrupted as a uniform random class.
  - Random labels: all the labels are replaced with random ones.
  - Shuffled pixels: a random permutation of the pixels is chosen and then the same permutation is applied to all the images in both training and test set.
  - Random pixels: a different random permutation is applied to each image independently.
  - Gaussian: A Gaussian distribution (with matching mean and variance to the original image dataset) is used to generate random pixels for each image.

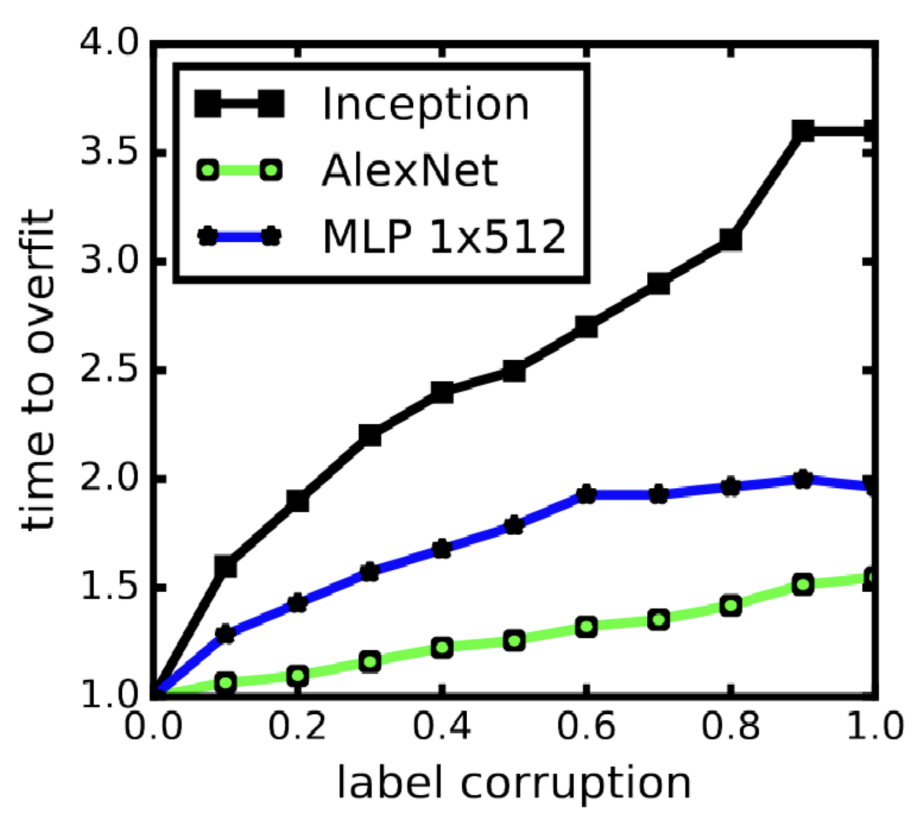
Deep Learning from Random Data ii



The training loss of various experiment settings decaying with the training steps.

[Zhang et al., 2021]

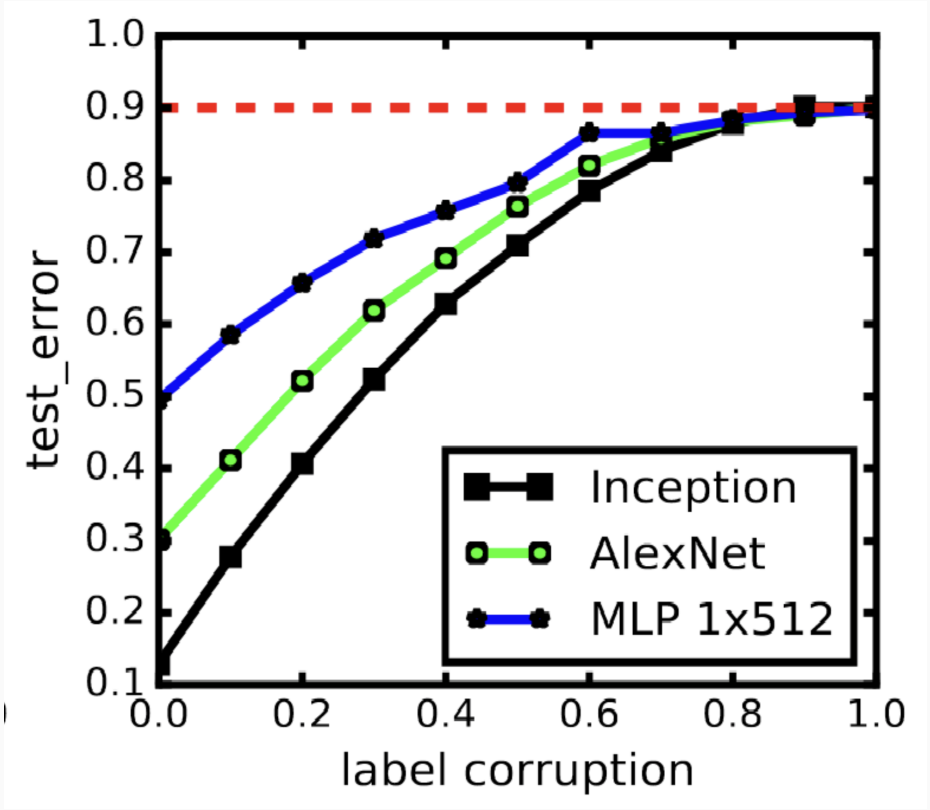
Deep Learning from Random Data iii



The relative convergence time with different label corruption ratio..

[Zhang et al., 2021]

Deep Learning from Random Data iv



The test error (also the generalization error since training error is 0) under different label corruptions. [Zhang et al., 2021]

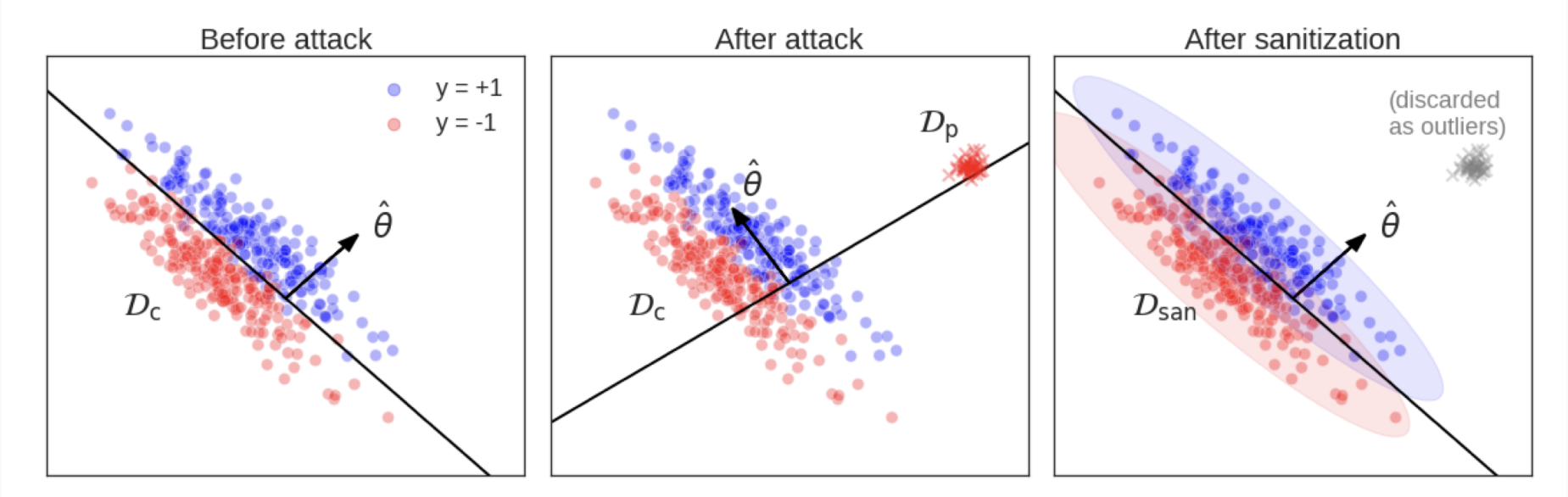
Data Sanitization Defenses

- **L2 defense** Remove points far from their class centroids in L2 distance
- **Slab defense [Steinhardt et al., 2017]:** Project points onto the line between the class centroids, then removes points too far from the centroids
- **Loss defense** Discard points that are not well fit by a model trained (without any data sanitization) on the full dataset
- **k-NN defense [Frederickson et al., 2018]** Remove points that are far from their k nearest neighbors

---

Source: [Koh et al., 2022]

Data Sanitization Defenses



The defender discards all blue points outside the blue ellipse, and all red points outside the red ellipse

Source: [Koh et al., 2022]

Adaptive Attack

- The adversary knows the clean training dataset $\mathcal{D}^{(c)}$, test dataset $\mathcal{D}^{(test)}$, and feasible set $\mathcal{F}$ (defender keeps the points lie within a feasible set $\mathcal{F}$).
- The objective of the adversary can be formalized as an optimization problem:

$$\begin{aligned} &\max_{\mathcal{D}^{(p)}} L(\hat{\theta}, \mathcal{D}^{(test)}) \\ &\text{s.t. } |\mathcal{D}^{(p)}| = \epsilon |\mathcal{D}^{(c)}| \\ &\quad \mathcal{D}^{(p)} \subseteq \mathcal{F} \\ &\text{where } \hat{\theta} := \arg\min L(\theta, \mathcal{D}^{(c)} \cup \mathcal{D}^{(p)}) \end{aligned}$$

Source: [Koh et al., 2022]

Influence Attack

The influence attack tackles the optimization problem via projected gradient ascent.

- Given the model parameter $\hat{\theta}$, the adversary finds a local maximum of adversary's optimization problem by iteratively taking gradient steps on each poisoned point in $\tilde{z} \in \mathcal{D}^{(p)}$, projecting each point onto the feasible set $\mathcal{F}$.
- The adversary re-trains the model to minimize the loss on poisoned training dataset and clean training dataset.
- Repeat the above two steps.

Source: [Koh et al., 2022]

Influence Attack

- The difficulty is that test loss depends on $\tilde{z}$ only through the model parameters $\hat{\theta}$, which is a complicated function of $\mathcal{D}^{(p)}$.
- The influence function ([Koh and Liang, 2017]) provides a closed-form estimate of $\nabla_{\tilde{z}}\hat{\theta}$, which measures how much the parameters change due to a small change to $\tilde{z}$.
- The desired derivative can be computed via the chain rule

$$\nabla_{\tilde{z}} L(\hat{\theta}, \mathcal{D}^{(test)}) = \nabla_{\hat{\theta}} L(\hat{\theta}, \mathcal{D}^{(test)}) \nabla_{\tilde{z}}\hat{\theta}$$

$$\text{where } \nabla_{\hat{\theta}} L(\hat{\theta}, \mathcal{D}^{(test)}) = \frac{1}{|\mathcal{D}^{(test)}|} \sum_{(x,y)\in\mathcal{D}^{(test)}} \nabla_{\theta} l(f(\theta, x), y)$$

Source: [Koh et al., 2022]

Influence of Training Data on Model Predictions

Influence of removing a training point

- What is the influence of a single training data point on the model performance?
  - Retrain the model by **removing** a training data point $z \in \mathcal{D}$

$$\hat{\theta}_{-z} = \arg\min_{\theta} \sum_{z_i \in \mathcal{D}, z_i \neq z} L(\theta, z_i)$$

  - Comparing $\hat{\theta}_{-z}$ with $\hat{\theta}$ reveals the **influence** of z on model parameters.
- However, retraining the model for each removed z is prohibitively slow.
- Influence functions give us an efficient approximation.

Source: [Koh and Liang, 2017]

Influence of removing a training point

- The idea is to compute the parameter change if z were upweighted by some small ϵ, giving us new parameters

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta} \sum_{z_i \in \mathcal{D}} L(\theta, z_i,) + \epsilon L(\theta, z)$$

- A classic result ([Cook and Weisberg, 1982]) tells us that the influence of upweighting z on the parameters $\hat{\theta}$ is given by

$$\mathrm{I}_{up,params}(z) = \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_{\theta} L(\hat{\theta}, z)$$

  where $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\theta}^2 L(\hat{\theta}, z_i)$ is the Hessian.

- How would the model's predictions change if a training input were modified?

Source: [Koh and Liang, 2017]

Influence of removing a training point i

- Removing a point z is the same as upweighting it by $\epsilon = -\frac{1}{n}$
- Instead of retraining the model, we can linearly approximate the parameter change due to removing z by computing

$$\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n}\mathrm{I}_{up,params}(z)$$

Source: [Koh and Liang, 2017]

Influence of perturbating a training point

- For a training point $z = (x, y)$, we define $z_\delta = (x + \delta, y)$ as a perturbation of x with δ.
- We want to compute the effect of the perturbation $z \rightarrowtail z_\delta$ on model predictions on some target data.
- Let $\hat{\theta}_{z_\delta, -z}$ be the empirical risk minimizer on the training points with z_δ in place of z.
- We define the parameters resulting from moving ϵ from z to z_δ:

$$\hat{\theta}_{\epsilon, z_\delta, -z} = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} L(\theta, z_i) + \epsilon L(\theta, z_\delta) - \epsilon L(\theta, z)$$

Source: [Koh and Liang, 2017]

Influence of perturbating a training point

- The influence of perturbing the training data point is given by:

$$\left.\frac{d\hat{\theta}_{\epsilon,z_\delta,-z}}{d\epsilon}\right|_{\epsilon=0} = \mathrm{I}_{up,params}(z_\delta) - \mathrm{I}_{up,params}(z)$$
$$= -H_{\hat{\theta}}^{-1}\Big(\nabla_\theta L(\hat{\theta}, z_\delta) - \nabla_\theta L(\hat{\theta}, z)\Big)$$

- If x is continuous and δ is small, we can further approximate.

$$\nabla_\theta L(\hat{\theta}, z_\delta) - \nabla_\theta L(\hat{\theta}, z) \approx [\nabla_x \nabla_\theta L(\hat{\theta}, z)]\delta$$

- Thus, we can approximate the influence of perturbation of δ on model parameters as follows:

$$\left.\frac{\partial^2\hat{\theta}_{\epsilon,z_\delta,-z}}{\partial\epsilon\partial\delta}\right|_{\epsilon=0,\delta=0} \approx -H_{\hat{\theta}}^{-1}\nabla_x\nabla_\theta L(\hat{\theta}, z)$$

Source: [Koh and Liang, 2017]

Influence of perturbating a training point

Therefore, the gradient of the test loss w.r.t. an poisoned point $\tilde{z}$ is

$$\nabla_{\tilde{z}} L(\hat{\theta}, \mathcal{D}^{(test)}) = - \underbrace{g^{\intercal}_{\hat{\theta}, \mathcal{D}^{(test)}} H^{-1}_{\hat{\theta}}}_{\text{same for all poisoned data}} \nabla_{\tilde{z}} \nabla_{\theta} L(\hat{\theta}, \tilde{z})$$

$$\text{where } g^{\intercal} = \frac{1}{|\mathcal{D}^{(test)}|} \sum_{(x,y) \in \mathcal{D}^{(test)}} \nabla_{\theta} l(f(\hat{\theta}, x), y)$$

Source: [Koh et al., 2022]

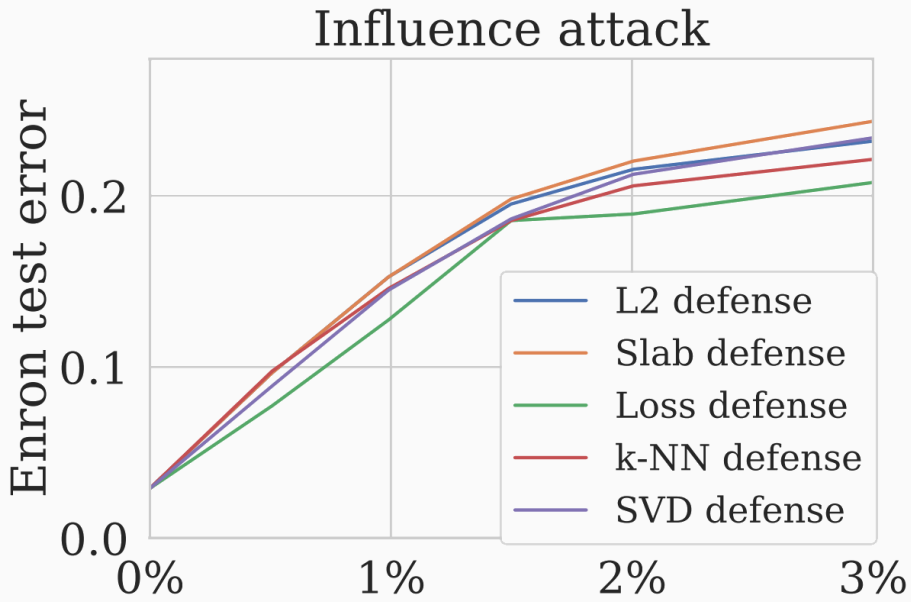
Generate poisoned data

Poisoning Attack Algorithm

1. Compute model parameters $\hat{\theta} \leftarrow \arg\min_\theta L(\theta; (\mathcal{D}^{(c)} \cup \mathcal{D}^{(p)}) \cap \mathcal{F})$
2. Pre-compute $g^\intercal_{\hat{\theta}, \mathcal{D}^{(test)}} H^{-1}_{\hat{\theta}}$
3. For each point $(\tilde{x}, \tilde{y})$ in the poisoned dataset $\mathcal{D}^{(p)}$,
   - $\tilde{x}'_i \leftarrow \tilde{x}_i - \eta g^\intercal_{\hat{\theta}, \mathcal{D}^{(test)}} H^{-1}_{\hat{\theta}} \nabla_{\tilde{x}_i} \nabla_\theta L(f(\hat{\theta}, \tilde{x}_i), \tilde{y}_i)$
   - $\tilde{x}_i \leftarrow \arg\min_{x \in \mathcal{F}} \|x - \tilde{x}'\|_2$

Source: [Koh et al., 2022]

Poisoning attack can bypass the data sanitization defenses



X-axis represents the fraction of poisoned data added. The dataset is the Enron spam detection dataset. The model is a linear support vector machine.

Source: [Koh et al., 2022]

Untargeted attack in practice

- The entity that trains the model may discover that the model is under poisoning attacks.
- In practice, the adversary might only be interested in reducing the accuracy of a subpopulation.

Targeted Attacks

Crafting poisoned data via feature collisions

- The goal of the adversary is to **increase the model loss on a target data** (x_{target}, y_{target}) without hurting the model performance on other points.
- In addition, the label of the poisoned dataset is clean.

Crafting poisoned data via feature collisions

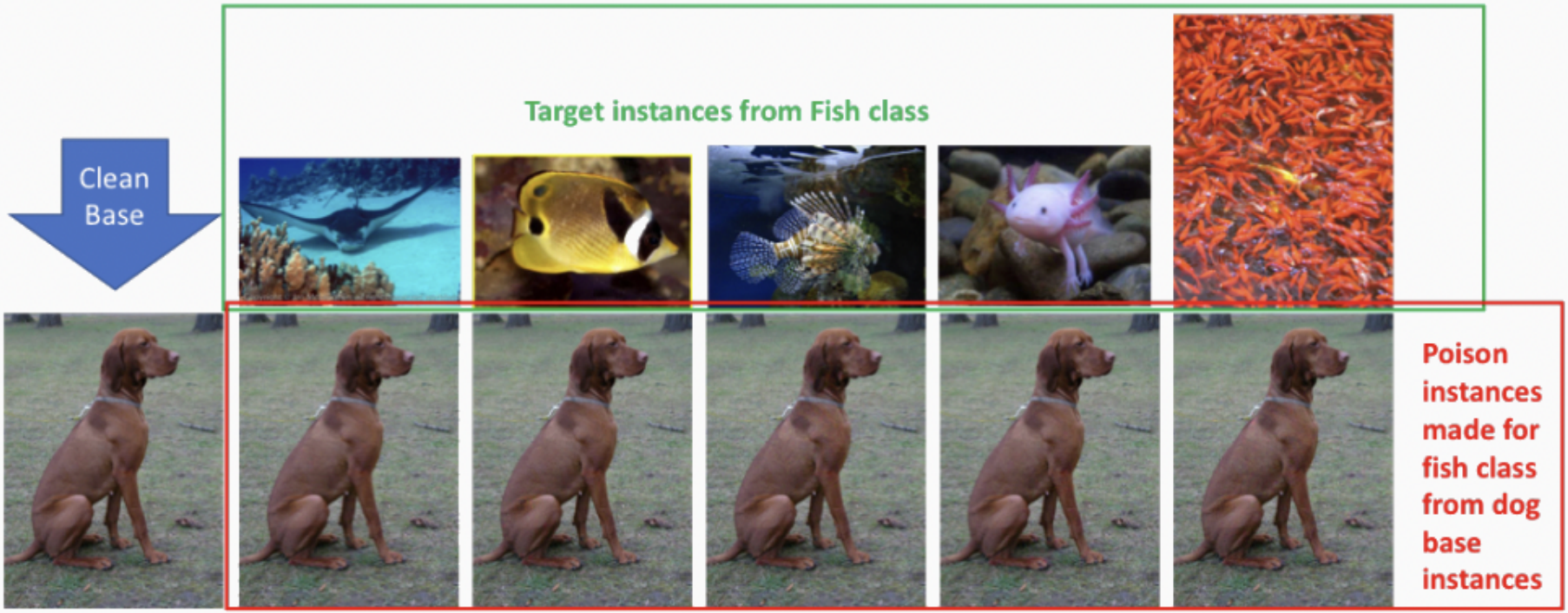
Create collisions: generate a poisoned data point from clean base data point (x_{base}, y_{base}) such that the poisoned data is simultaneously close to the base clean data point in input space, and the target data point (x_{target}, y_{target}) in latent feature space.

$$\arg\min_x \|h(x) - h(x_{target})\|_2 + \beta \|x - x_{base}\|_2$$

The poisoned data will look natural with clean labels. This attack is also called "clean-label" poisoning attack.

Source: [Shafahi et al., 2018]

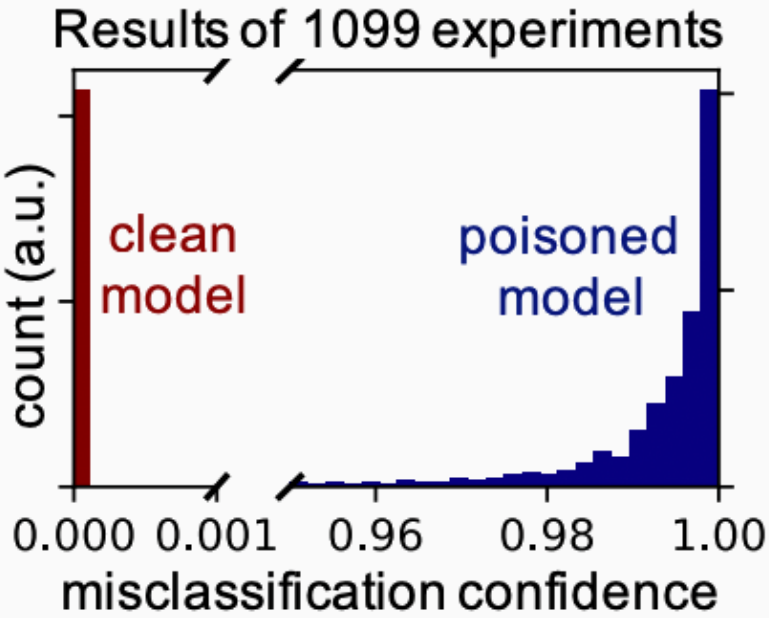
Crafting poisoned data via feature collisions



The top row contains 5 random target instances (from the "fish" class). The second row contains the constructed poison instance corresponding to each of these targets.

Source: [Shafahi et al., 2018]

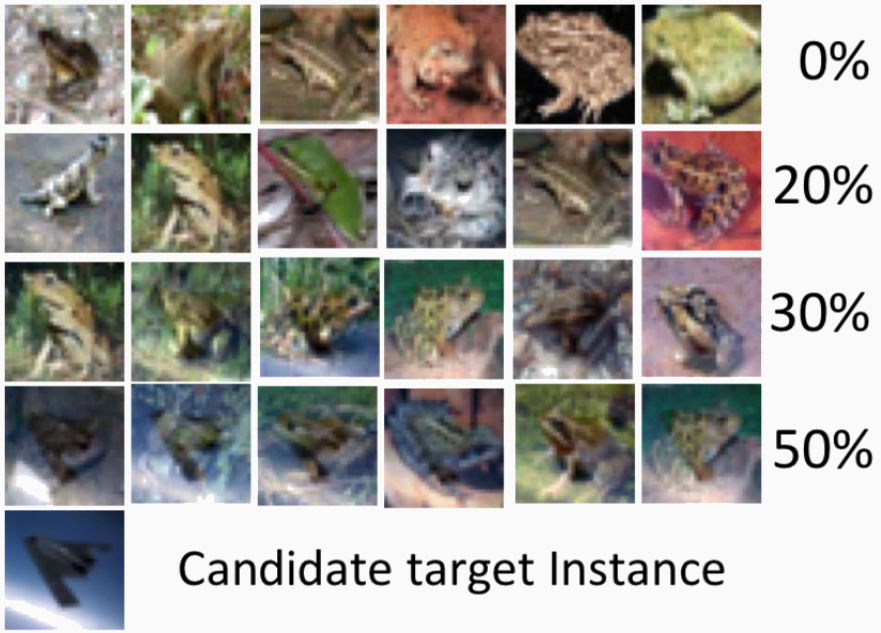
Crafting poisoned data via feature collisions



Incorrect class's probability histogram predicted for the target image by the clean (dark red) and poisoned (dark blue) models. The targeted points are misclassified with high confidence.

Source: [Shafahi et al., 2018]

Poison Frogs



Figures show the transparent overlay of the target instance "airplane" image with different opacity levels on poisoned data points (generated from random images belonging to the frog class).

Source: [Shafahi et al., 2018]

Poison frogs i

Poison frogs ii

Poison frogs iii

Backdoor Attacks

For each target example, the adversary needs to generate a (or a few) poisoned data point(s).

What if the adversary wants the model to output the target label whenever a trigger is added to a test image?

Source: [Chen et al., 2017]

Backdoor Attacks

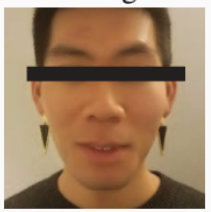
- The goal of the adversary is to fool the model such that it predicts the target label on any images with a trigger while still predicting correct labels on clean samples.
- The adversary first design a trigger, generate poisoned dataset based on the trigger and provides it to the victim for training models.
- Adversary samples k data points from the data distribution D and modifies them in the following way, to construct the set of poisoned data:
  - Overwrite some particular features with a pre-determined pattern δ (e.g., stamp the images with a particular pattern)
  - Change their labels to the same class y', which is determined by the adversary

Backdoor Attacks

- By adding poisoned data points to the training set, the model links δ with class y', as other features of the data are not related to class y'
- Attack results: model misclassifies any test data with pattern δ as y'

Images with backdoor triggers [Gu et al., 2017]

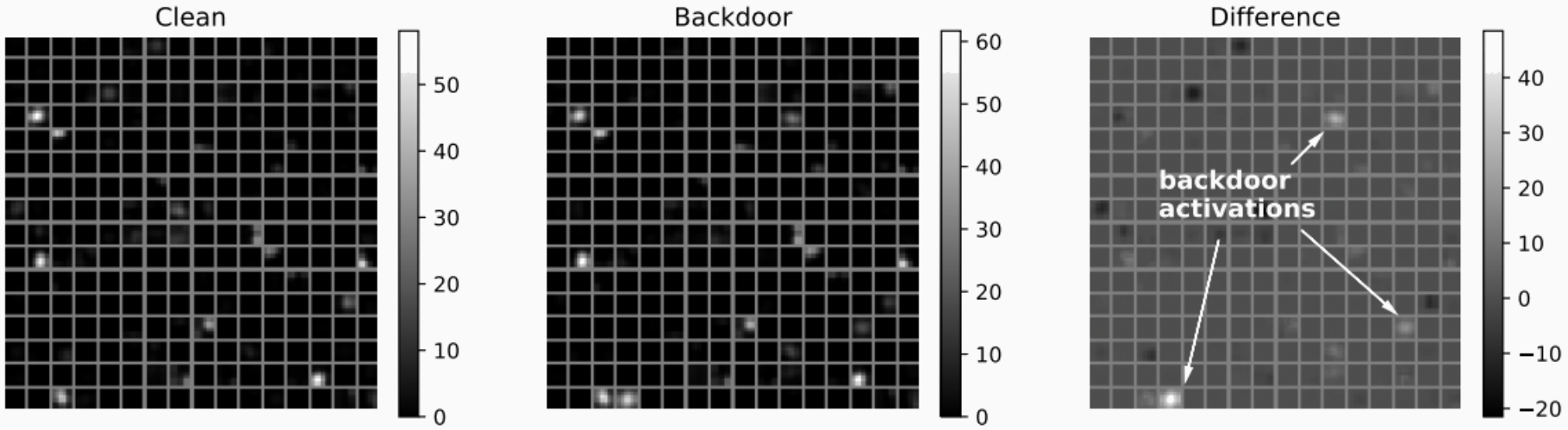
Backdoor Attacks in Physical World

	Digital Trigger	Physical Triggers						
	Square	Dots	Sunglasses	Tattoo Outline	Tattoo Filled-in	White Tape	Bandana	Earrings
								
VGG16	91 ± 7%	100 ± 0%	100 ± 0%	99 ± 1%	99 ± 1%	98 ± 3%	98 ± 1%	69 ± 4%
DenseNet	98 ± 1%	96 ± 3%	94 ± 4%	95 ± 2%	95 ± 2%	81 ± 8%	98 ± 0%	85 ± 2%
ResNet50	100 ± 0%	98 ± 4%	100 ± 0%	99 ± 1%	99 ± 1%	95 ± 5%	99 ± 0%	58 ± 4%

Attack success rates of physical triggers in facial recognition models trained on various architectures.

Source: [Wenger et al., 2021]

Why do backdoor attacks work?



Activations of the last convolutional layer of the backdoor attack averaged over clean inputs (left) and backdoored inputs (center). The difference between the two activation maps is shown on the right.

Source: [Gu et al., 2017]

Takeaways i

1. Detecting poisoned data is hard, especially in high dimensional space.
2. Identifying poisoned data designed for targeted attacks and backdoor attacks can be extremely difficult.
    - The poisoned data can be nonsuspicious (correctly labeled).
    - The trigger can be hidden during training time.

Certified robust algorithm against poisoning attacks i

Problem: How to provide the robustness guarantee for each prediction? In other words, should we trust the prediction made by the model when the training dataset is potentially corrupted?

Reading List:

- Certified Robustness to Label-Flipping Attacks via Randomized Smoothing ([Rosenfeld et al., 2020])
- RAB: Provable Robustness Against Backdoor Attacks ([Weber et al., 2020])
- Deep partition aggregation: Provable defense against general poisoning attacks ([Levine and Feizi, 2020])

Certified robust algorithm against poisoning attacks ii

- Improved Certified Defenses against Data Poisoning with (Deterministic) Finite Aggregation ([Wang et al., 2022])

References i

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. (2017).
Targeted backdoor attacks on deep learning systems using data poisoning.
arXiv preprint arXiv:1712.05526.

Cook, R. D. and Weisberg, S. (1982).
Residuals and influence in regression.
New York: Chapman and Hall.

Frederickson, C., Moore, M., Dawson, G., and Polikar, R. (2018).
Attack strength vs. detectability dilemma in adversarial machine learning.
In 2018 international joint conference on neural networks (IJCNN), pages 1–8. IEEE.

References ii

Gu, T., Dolan-Gavitt, B., and Garg, S. (2017).
Badnets: Identifying vulnerabilities in the machine learning model supply chain.
arXiv preprint arXiv:1708.06733.

Koh, P. W. and Liang, P. (2017).
Understanding black-box predictions via influence functions.
In 34th, pages 1885–1894.

Koh, P. W., Steinhardt, J., and Liang, P. (2022).
Stronger data poisoning attacks break data sanitization defenses.
Machine Learning, 111(1):1–47.

References iii

Levine, A. and Feizi, S. (2020).

Deep partition aggregation: Provable defenses against general poisoning attacks.

In International Conference on Learning Representations.

Rosenfeld, E., Winston, E., Ravikumar, P., and Kolter, Z. (2020).

Certified robustness to label-flipping attacks via randomized smoothing.

In International Conference on Machine Learning, pages 8230–8241. PMLR.

References iv

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. (2018).
Poison frogs! targeted clean-label poisoning attacks on neural networks.
arXiv preprint arXiv:1804.00792.

Steinhardt, J., Koh, P. W. W., and Liang, P. S. (2017).
Certified defenses for data poisoning attacks.
Advances in neural information processing systems, 30.

References v

Wang, W., Levine, A. J., and Feizi, S. (2022).
Improved certified defenses against data poisoning with (deterministic) finite aggregation.
In International Conference on Machine Learning, pages 22769–22783. PMLR.

Weber, M., Xu, X., Karlaš, B., Zhang, C., and Li, B. (2020).
Rab: Provable robustness against backdoor attacks.
arXiv preprint arXiv:2003.08904.

References vi

Wenger, E., Passananti, J., Bhagoji, A. N., Yao, Y., Zheng, H., and Zhao, B. Y. (2021).

Backdoor attacks against deep learning systems in the physical world.

In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6206–6215.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021).

Understanding deep learning (still) requires rethinking generalization.

Communications of the ACM, 64(3):107–115.