Poisoning Attacks

A poisoning attack is a training-time attack where an attacker injects malicious data into a model's training dataset to corrupt the learned parameters, with the goal of compromising the system's normal operations. Poisoning attacks can be classified into:

- *Indiscriminate Poisoning*: The goal is to cause a general degradation in overall model performance, degrading the system's availability.
- *Targeted Poisoning*: The goal is to create a specific, hidden vulnerability to allow subsequent intrusion on chosen inputs without affecting general performance, attacking the system's integrity.

1.1 Indiscriminate Poisoning

From a mathematical perspective, the attacker's problem in an indiscriminate poisoning attack can be formally defined as a bi-level optimization problem, often conceptualized as a leader/follower game.

- *Inner Problem (Follower)*: The legitimate training algorithm minimizes its loss on the poisoned training dataset, which includes the attacker's rogue point (x_c, y_c).

$$w^*(x_c) \in \arg\min_w \mathcal{L}(D_{\text{tr}} \cup \{(x_c, y_c)\}, w) \tag{1.1}$$

- *Outer Problem (Leader)*: The attacker searches for the optimal rogue point that, when used in the inner problem, forces the resulting model's parameters w^* to perform the worst on a clean validation set.

$$\max_{x_c} \mathcal{L}(D_{\text{val}}, w^*(x_c)) \tag{1.2}$$

Directly solving this nested problem is computationally hard. Common approaches include replacing the inner loop with its Karush-Kuhn-Tucker (KKT) equilibrium conditions (at the optimal solution, a specific set of conditions must hold, differentiate through these conditions directly) is tractable for convex models like SVM and linear regression. For non-convex deep neural networks (DNNs), methods like unrolled optimization are used, which approximate the inner solution by back-propagating through a finite number of training steps. In some linear model settings, optimal indiscriminate attacks have been shown to reduce to simpler, highly efficient heuristics like maximizing class-conditional data density.

Specialized Poisoning Attack Variants. Beyond accuracy, poisoning attacks can target other critical system properties. Poisoning Attacks on Algorithmic Fairness introduce or amplify discriminatory bias against specific groups. This is achieved via a gradient-based optimization framework that crafts points to maximize the disparity in model performance across groups.

$$\mathcal{L}(D_{\text{val}}, \theta) = \underbrace{\sum_{k=1}^{p} l(x_k, y_k, \theta)}_{\text{unprivileged}} + \lambda \underbrace{\sum_{j=1}^{m} l(x_j, y_j, \theta)}_{\text{privileged}} \tag{1.3}$$

Sponge Poisoning aim to drain battery and cause delays in resource-constrained systems. The attacker manipulates the training dataset to create a model with reduced activation sparsity. This forces hardware accelerators to perform more computations during inference for any input, increasing energy use and latency without affecting accuracy.

1.2 Targeted Poisoning

A targeted poisoning attack aims to manipulate a machine learning model so that it misclassifies one or more specific inputs at test time in a way chosen by the attacker, while leaving the model's overall performance on other data largely unaffected. The attacker's core strategy is to inject carefully poison samples into the model's training data. The mathematical goal is a bi-level optimization problem:

$$\min_{x_c} \mathcal{L}(D_{\text{val}}, w^*) \qquad \text{s.t. } w^* \in \arg\min_{w} \mathcal{L}(D_{\text{tr}} \cup \{x_c, y_c\}, w) \tag{1.4}$$

Clean-label Poisoning Attacks. "Clean-label" means the poison samples have correct, verifiable labels, making them very hard to detect during data curation. They key is to make a poison sample from class A reside extremely close in the model's feature space to a target sample from class B. During training, the model is forced to draw a decision boundary that incorrectly classifies the target sample as class A.

- *Poisoning Frogs!*: This attack crafts a single poison image so that its feature representation collides with the target's features in the latent space of a pre-trained model, effectively creating a Trojan horse. This method is highly effective in transfer learning settings where only the last layer of a pre-trained model is re-trained.

$$\arg\min_{x} \quad \underbrace{\|f(x) - f(t)\|_2^2}_{\substack{\text{small distance between} \\ x \text{ and } t \text{ in feature space}}} \quad + \quad \underbrace{\beta\|x - b\|_2^2}_{\substack{\text{small distance between} \\ x \text{ and } b \text{ in input space}}} \tag{1.5}$$

- *Convex Polytope Attack*: A single point can be insufficient, especially for end-to-end training. To increase robustness, Zhu et al. introduced the Convex Polytope attack. Instead of one poison, they generate multiple poison samples from the base class. These poisons are positioned to form a convex polytope in feature space that encloses the target sample. This multipoint enclosure makes it much harder for the target to be correctly classified, significantly improving attack success rates and transferability to unknown model architectures.

- *The Bullseye Polytope Advancement*: Building on this, Aghkhani et al. developed the Bullseye Polytope attack, which optimizes for a more specific geometric configuration. It arranges multiple poisons so that their mean is as close as possible to the target in feature space, rather than just forming a general polytope. This tight clustering around the target is more efficient and scalable, leading to higher attack success rates with fewer computational resources.

Backdoor Poisoning Attacks

A backdoor poisoning attack is a type of security threat in machine learning where an adversary manipulates a model during its training phase. The goal is to implant a secret backdoor, which is a specific pattern or trigger. The compromised model behaves normally on regular, clean inputs but performs an adversary-selected, often malicious, behavior when it detects the trigger at test time.

- *BadNets*: BadNets is the foundational attack in this domain. In the typical scenario, a user outsources model training to a malicious third party. This attacker injects a small, fixed pattern into a subset of training images and changes their labels to a target class. The model learns to associate this obvious trigger with the wrong label, creating a backdoor.
- *Hidden Triggers*: A key limitation of BadNets is that the trigger is visible in the poisoned training data, making detection possible. To overcome this, Saha et al. introduced Hidden Trigger Backdoor Attacks. Their method uses clean, correctly labeled images during poisoning. It adds a small, optimized perturbation to these images, creating a feature collision in the model's internal representation. This collision causes the perturbed clean image to be classified as the target class, while the attacker uses a different, secret trigger to activate the backdoor during deployment.
- *Input-Aware Attacks*: The next leap in stealth came from Nguyen and Tran, who proposed the Input-Aware Dynamic Backdoor Attack. This method breaks the fundamental defense assumption of a universal trigger. Instead of a fixed pattern, it uses a secondary neural network to generate a unique, dynamic trigger for every input sample. This makes the trigger blend seamlessly with each input, rendering traditional detection methods that look for consistent patterns ineffective.

1.3 Defense against Poisoning Attacks

Defenses against data poisoning attacks can be categorized based on their operational phase: training-time defenses and test-time defenses. Training-time defenses aim to prevent poison from entering the model or to mitigate its effects during learning, while test-time defenses operate on the assumption that a model may already be compromised.

(1) *Training Data Sanitization*: This approach is based on the premise that poisoned data points often behave as statistical outliers. Core methods include:

  - *Reject on Negative Impact*: A data point is rejected if adding it to the training set degrades model performance on a held-out validation set. This method is highly effective but can be computationally expensive, as it requires retraining or significant model updates for evaluation.
  - *Label Sanitization*: Particularly useful against label-flipping attacks, this technique audits data labels for consistency. A common method involves checking a data point's k nearest neighbors and performing a majority vote to correct potentially flipped labels. Its primary limitation is that it is generally ineffective against clean-label attacks, where the poison lies in the data features, not the labels.

(2) *Robust Training*: These algorithms are designed to be inherently resilient to a fraction of corrupted data, making them applicable to various attack types. An example is the

TRIM algorithm, used for Ridge Regression and adaptable to other models. It solves an optimization problem that jointly learns model parameters and selects a subset of training data:

$$\min_{w,b,I} \mathcal{L}(w,b,I) = \frac{1}{|I|} \sum_{i \in I} f((x_i) - y_i)^2 + \lambda \Omega(w) \tag{1.6}$$

The algorithm starts with an oversized dataset where $N = (1+\alpha)n$ (α is the assumed poison fraction) and iteratively selects the n points with the lowest loss for training. A key vulnerability is that an adaptive attacker, aware of this defense, can optimize poison data to appear less anomalous. Theoretical and empirical research indicates that standard training procedures like Stochastic Gradient Descend (SGD) can exhibit inherent robustness under certain conditions, such as specific data separability assumptions and constrained model capacity. Furthermore, novel defenses propose identifying and excluding training samples isolated in gradient space, as effective poisons often create gradients distinct from clean data.

(3) *Model Inspection & Sanitization*: These defenses are applied after a model is trained, either to diagnose a backdoor or to neutralize its effect during deployment.

- *Activation Clustering*: A seminal method for detection backdoors. It focuses on a suspected target class, extracts deep-layer activation vectors for all training samples in that class, and performs dimensionality reduction followed by K-means clustering (with $k = 2$). The separation into two distinct clusters often reveals the poisoned subset. A key advantage is that it operates without requiring a trusted, clean dataset.
- *Fine-Pruning*: This method exploits the observation that backdoors often rely on a small set of dormant neurons that are highly active only for triggered inputs. The process involves pruning neurons with the lowest average activation on clean data and then fine-tuning the model to recover accuracy. An adaptive attacker can potentially circumvent this by designing triggers that also activate neurons used for clean tasks.

(4) *Trigger Reconstruction*: This approach formalizes trigger reverse-engineering as an optimization problem. It searches for a trigger pattern Δ and a mask M that, when applied to inputs, cause them to be classified as the target label. The objective is:

$$\arg\min_{\Delta,M} \mathcal{L}(f(x_t), y_t), \qquad x_t = x \odot (1-M) + \Delta \odot M \tag{1.7}$$

TABOR incorporates regularization terms to constrain the trigger to be smooth and small, preventing the detection of false, meaningless patterns.

(5) *Test Data Sanitization*: This use-side defense assumes the model is already poisoned and aims to neutralize the trigger before inference. FEBRUUS is a method designed for image data. It first generates a heatmap to identify the image regions most influenced on the model's prediction. In a poisoned image, this region typically corresponds to the trigger. It then removes the identified region and uses inpainting techniques to reconstruct a clean version of the image for classification.