CY 7790, Lecture 10: Poisoning Attacks

John Abascal and Harsh Chaudhari

October 18, 2021

1 Carlini. Poisoning the Unlabeled Dataset of Semi-Supervised Learning

Problem Statement. The authors consider the case of semi-supervised learning in this paper. Semi-supervised learning is a procedure where the learner is given a small labeled set and a large unlabeled set to train on. Each unlabeled sample is assigned a "guessed" label through an unsupervised procedure, and the learner uses the current predictions to supervise training for the next weights.

Because models are increasingly large and good data is increasingly sparse, semi-supervised learning has become an essential technique in modern machine learning practice. The authors then propose attacks that target the unlabeled data from a semi-supervised training procedure at training time.

Threat Model. The threat model that the authors consider is data poisoning. This attack takes place at training time and aims to "poison" the data using malicious examples to force a misclassification. In this case, the authors are specifically looking at a targeted attack using the unlabeled part of the data set. This attack works by slightly shifting the decision boundary using these malicious examples to classify everything mostly correctly except for the malicious examples.

To successfully poison the model, the adversary needs grey-box access to the model. In the case of a general data poisoning attack, the adversary needs access to the training set of the target model. The authors of this paper specifically look at the case where the target model has limited labeled examples in this training set. Since the attack is targeted, the adversary needs to have a "source" class example and a target example from the target class. Then, the adversary will decide how many poisoned points to inject.

Methodology. In summary, the attack works as follows:

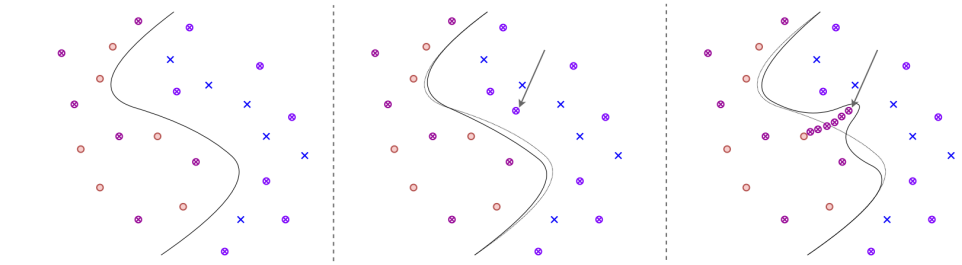
1. Start with one source image from the labeled set
2. Choose a target image with a label
3. Add "bridging" points to interpolate a path between the source image and the target image

Why does this work? Neural networks are Lipschitz continuous with a low constant on average, so small perturbations do not change the output very much. Many models also apply data augmentation, hence they are already trained on perturbed inputs.

How does this work? Unlabeled images close to the target will be correctly classified as the true label. Once confidence of the model on poisoned examples reaches a certain threshold, the training algorithm begins treating these examples as the labeled set. Because we create a path between the source (true label) image and the target, the target begins to propagate until all of the injected points on the path are labeled as the target class.

How does one interpolate from x' to x^*? To interpolate between the souce and target image, Carlini suggests using Linear Pixel Wise Blending and Generative Adversarial Networks (GANs). Using a linear interpolation strategy is easier to implement and allows us to choose the distribution of perturbations applied to injected points, but points

poisioned using this technique are very easy to detect. The poisonsed points created by GANs are much harder to detect visually, but training a GAN is much more computationally costly.



(a) A classifier trained on a semi-supervised dataset of red ⊙s, blue ×s, and *unlabeled* ⊗s. During training the unlabeled ⊗s are given pseudo-labels such that the correct original decision boundary is learned.

(b) When inserting just one new *unlabeled* poisoned example near the boundary, the model gives it the correct pseudo label of the blue ×s. The poisoning attempt fails, and the decision boundary remains largely unchanged.

(c) By inserting a path of unlabeled examples, the classifier assigns every example in the path the pseudo-label of the nearby red ⊙s. This moves the decision boundary to enclose the path, which makes these examples misclassified.

Figure 1: Decision boundary plot for semi-supervised learning during (a) normal training, (b) failed poisoning, and (c) their attack

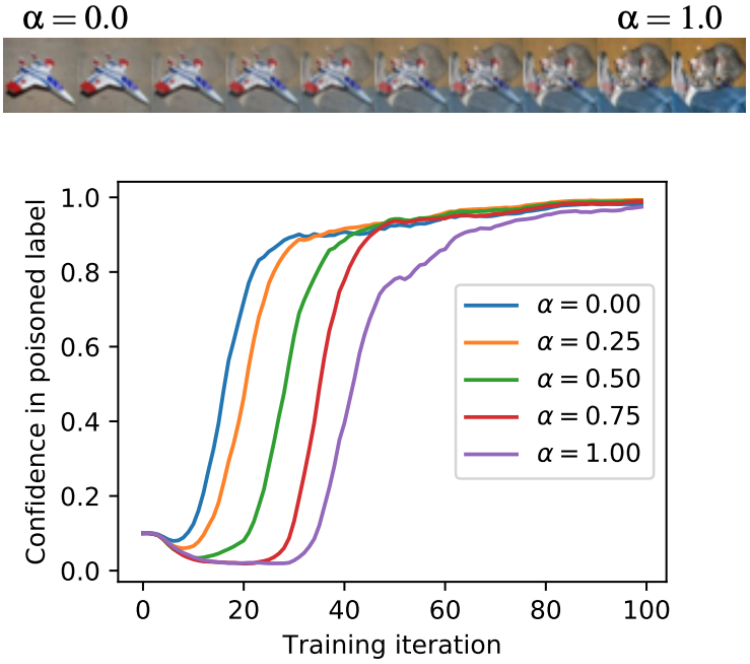


Figure 2: Label propagation of a poisoning attack over training epochs. The classifier begins by classifying the correctly labeled source example x' (when $\alpha = 0$; ; image shown in the upper left) as the poisoned label. This propagates to the interpolation $\alpha > 0$ one by one, and eventually on to the final example x^* (when $\alpha = 1$; image shown in the upper right)

Evaluation.

1. Some images are better as the source images

2. Some images are are harder to reach as the targets
3. Models with lower overall accuracy are harder to attack (older semi- supervised methods are harder to attack)
4. Modern methods trained on lower epochs to deliberately have low overall performance are also harder to attack - developing better training techniques are unlikely to prevent poisoning attacks and will likely make the problem worse
5. Models with more labeled data are generally more robust to attacks

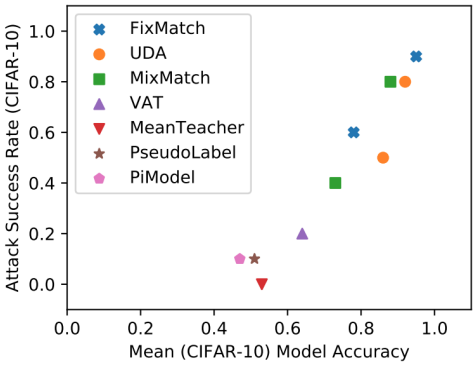


Figure 3: Success rate of poisoning CIFAR-10 with 250 labeled examples and 0.2% poisoning rate. Each point averages ten trained models. FixMatch, UDA, and MixMatch were trained under two evaluation settings, one standard (to obtain high accuracy) and one small-model to artificially reduce model accuracy.

Class Discussion.

Don't interpolated images look like junk? Yes, and the proposed GAN attacks are much harder to detect.

Why are there easier/harder points to poison? Distance from points in the underlying distribution could be the reason

How did they come up with the density functions (for linear interpolation)? Trial and error. They tried to come up with densities analytically, but they failed.

2 Shafahi et.al. Poison Frogs! Targeted clean-label poisoning attacks on Neural Networks

Problem Statement. The paper introduces a new type of attack called clean-label attacks, where the attacker is allowed to inject training samples of its choice but doesn't have control over the label associated with the sample. A certified authority (cleanly) assigns labels to these samples. The goal of the attacker is to cause the neural network to misclassify a special test instance from one class to another of attacker's choice after the model has been retrained on the dataset which includes the attacker's injected samples.

Threat Model. The threat model considered in this work is the attacker can poison can inject poisoned samples into the original training data . The attacker is not aware of the original training data but has knowledge of the model and its parameters. This is a reasonable assumption given that many networks such as ResNet and Inception, pre-trained on standard datasets are used as feature extractors in other applications.

Methodology. The attack can be viewed as an optimization problem where the attacker creates an adversarial sample that is close to the target sample t in the feature space f, which in this case is the penultimate layer of the neural network, while simultaneously being close to the base instance b in the input space. Formally, we can write the optimization problem as follows:

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} ||f(\mathbf{x}) - f(\mathbf{t})||_2^2 + \beta ||\mathbf{x} - \mathbf{b}||_2^2$$

where β controls how close the adversarial sample $\mathbf{x}$ needs to visually appear similar to base instance $\mathbf{b}$. The paper uses forward-backward-splitting iterative procedure to achieve the aforementioned objective. The first (forward) step is simply a gradient descent update to minimize the L_2 distance to the target instance in feature space. The second (backward) step is a proximal update that minimizes the Frobenius distance from the base instance in input space. The detailed sample generation algorithm is provided in Figure 1 below.

Algorithm 1 Poisoning Example Generation

Input: target instance t, base instance b, learning rate λ
Intialize x: $x_0 \leftarrow b$
Define $L_p(x) = ||f(\mathbf{x}) - f(\mathbf{t})||^2$
for i $= 1$ to max iters:

- Forward Step: $x_i = x_{i-1} - \lambda \nabla L_p(x_{i-1})$
- Backward Step: $x_i = (x_i + \lambda \beta b)/(1 + \beta \lambda)$

Why does this work? The adversarial sample x injected in the training data causes the decision boundary of the model to classify x as the base class and because x is very close to the target sample t in the feature space, the updated decision boundary of the model causes even t to be classified as the base class.

Evaluation. Figure 4 shows the success of the aforementioned attack in the transfer learning scenario where the created adversarial samples are visually indistinguishable from the base image while causing the target image to misclassify. A single poisoned sample is enough to cause a target sample to misclassify. In case of end to end training, we require multiple poisoned samples to be added to the training data and watermarking is required to cause the target sample to be misclassified as the base class of attacker's choice. A watermarking can viewed as a low-opacity image of the target instance super-imposed on the poisoned sample to allow for some inseparable feature overlap while remaining visually distinct. Figure 5 shows the need for watermarking and multiple poisoned instances to misclassify the target instance.

Class Discussion.

- The attack works well on a single target instance but might not generalize to other instances that are close to the target.
- Clean label poisoning attacks work better when there is only a small amount of training data available to train.

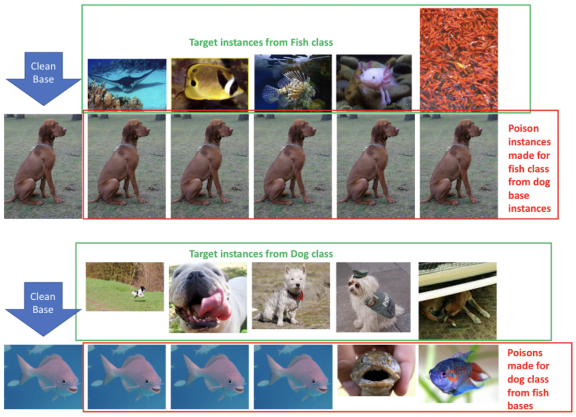


Figure 4: Transfer Learning poisoning attack. The top row contains 5 random target instances (from the fish class). The second row contains the constructed poison instance corresponding to each of these targets. The same base instance (second row, leftmost image) for building each poison instance. As observed, the poison instances are visually indistinguishable from the base instance (and one another).

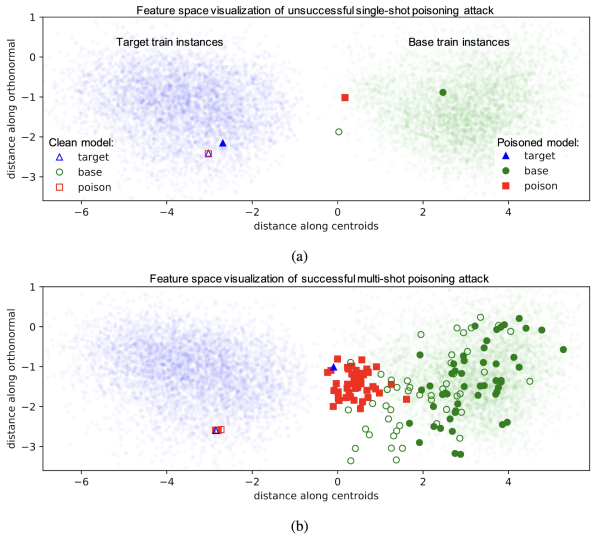


Figure 5: Feature space visualization of end-to-end training poisoning attacks. (a) A single poison instance is unable to successfully attack the classifier. The poison instance's feature space position under the clean model is overlapped with that of the target instance. However, when the model is trained on the clean + poisoned data (i.e. the poisoned model), the feature space position of the poison instance is returned to the base class distribution, while target remains in the target class distribution. (b) To make the attack successful, we construct 50 poison instances from 50 random base instances that are "watermarked" with a 30% opacity target instance. This causes the target instance to be pulled out of the target class distribution (in feature space) into the base class distribution and get incorrectly classified as the base class.