CY 7790

Special Topics in Security and Privacy: Machine Learning Security and Privacy

Fall 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science

October 18 2021

CY 7790

Special Topics in Security and Privacy:
Machine Learning Security and Privacy

Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks

By: Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu,
Christoph Studer, Tudor Dumitras, Tom Goldstein
(NIPS 2018)

Presented by Pablo Kvitca

October 18, 2021

Problem Statement

- Poisoning Attack without modifying labels or labelling process the model's training dataset
- Applicable on tests scenarios where dataset is audited or created manuality
- Applicable on transfer learning and end-to-end retraining scenarios
- Single-instance targeted attacks

Threat Model

Objectives

Misclassification of single target sample as a target class

No loss of accuracy over non-target samples

Knowledge

Grey-box

Model Parameters

(Not of training data)

Capabilities

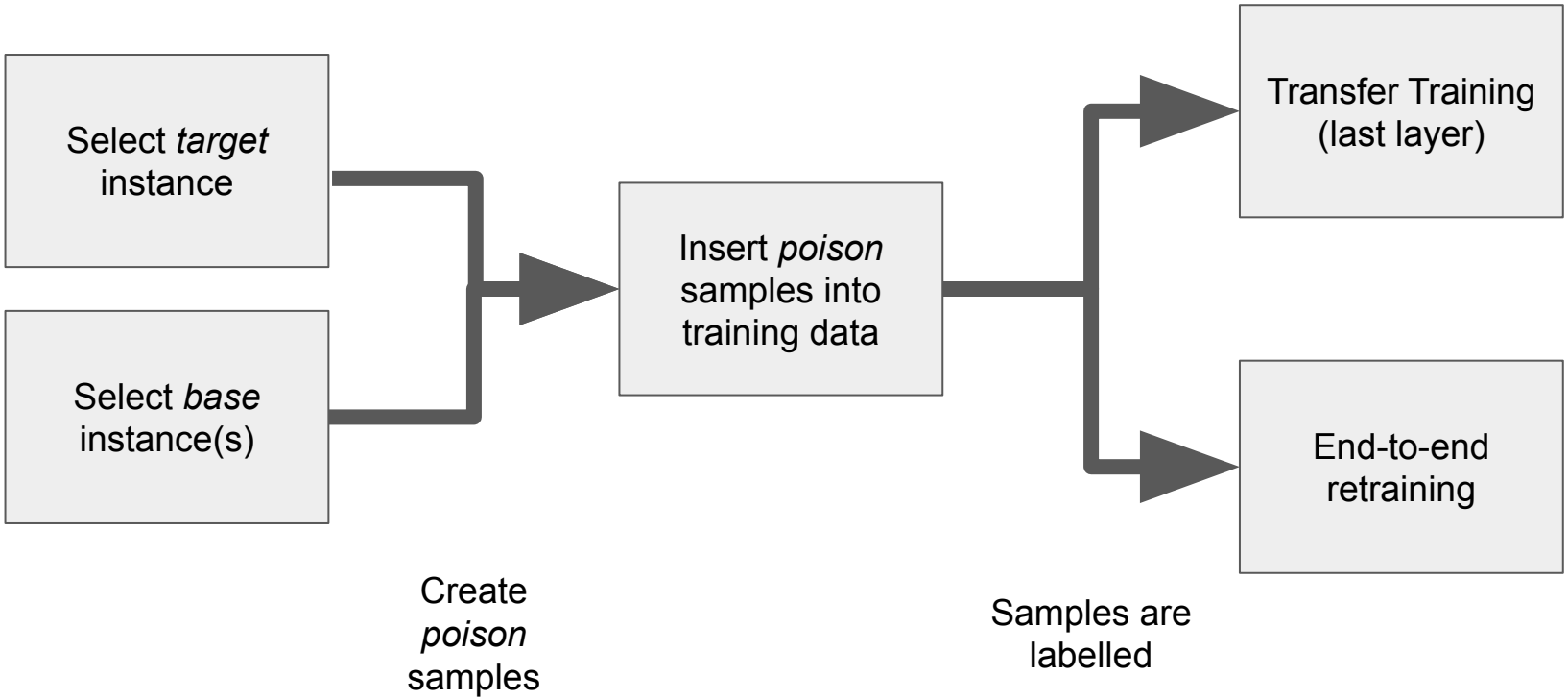
Crafting Poisoning Samples

Insert Poisoned Samples on Training Data *

Contributions

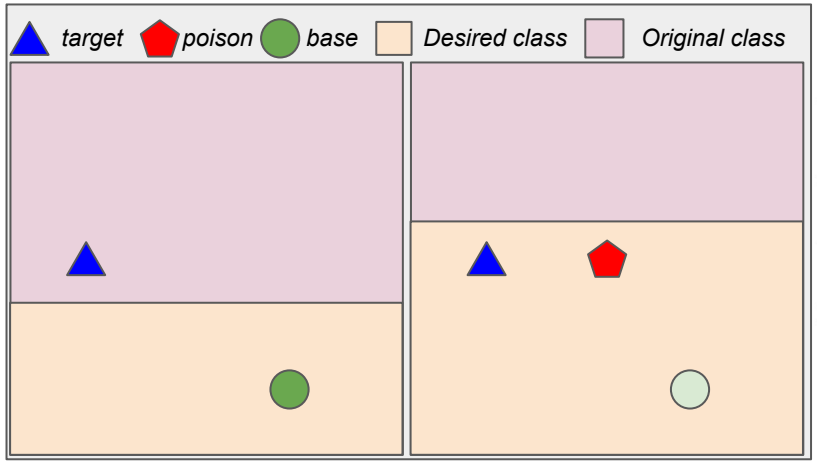
- Attack on (image data) transfer learning, using:
  - Single *target* instance
  - Single *poison* sample
  - Optimization procedure with L2 distance
- Attack on (image data) end-to-end retraining, using:
  - Single *target* instance
  - Multiple *poison* samples
  - Optimization procedure with L2 distance
  - Image Watermarking
  - Diversity of *poison* samples (selected base images for poisoning should be distinct)

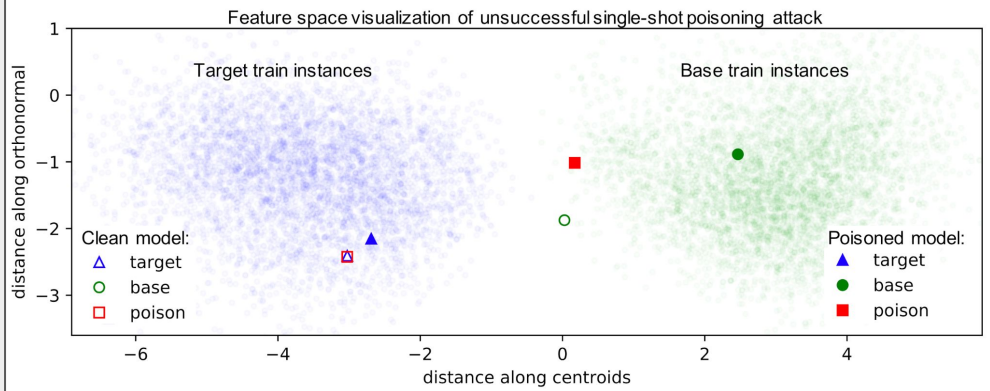
Attack Procedure



Intuition for Attack

- A model trained on data with the *poisoned* sample will learn a decision boundary that includes the *poisoned* sample on its *base* (original) class.
- The poisoning process make the *poison* sample be close to the *target* sample (sharing high-level features), while remaining *visually* in the *base* class.
- Then *target* sample would be predicted to be in the desired (*base*) class.





Poison Sample Generation

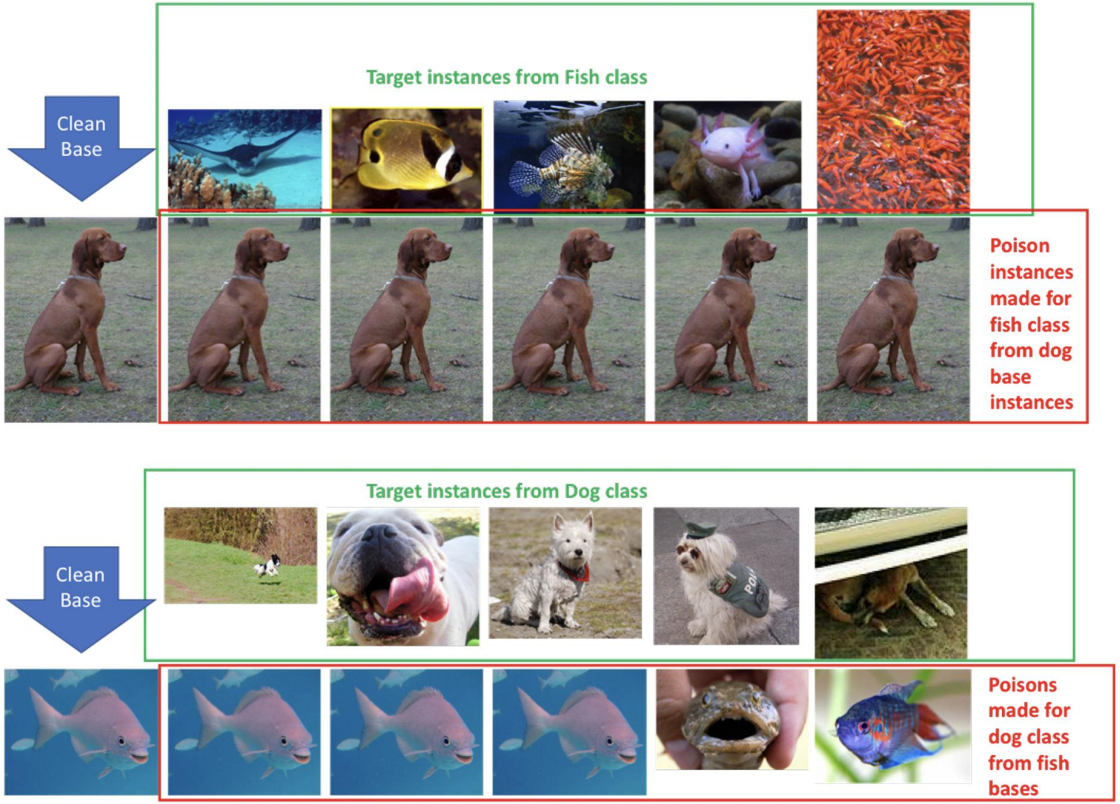
$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \; \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

Algorithm 1 Poisoning Example Generation

Input: target instance t, base instance b, learning rate λ
Initialize x: $x_0 \leftarrow b$
Define: $L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2$
for $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x_i} = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$
  Backward step: $x_i = (\widehat{x_i} + \lambda\beta b)/(1 + \beta\lambda)$
end for

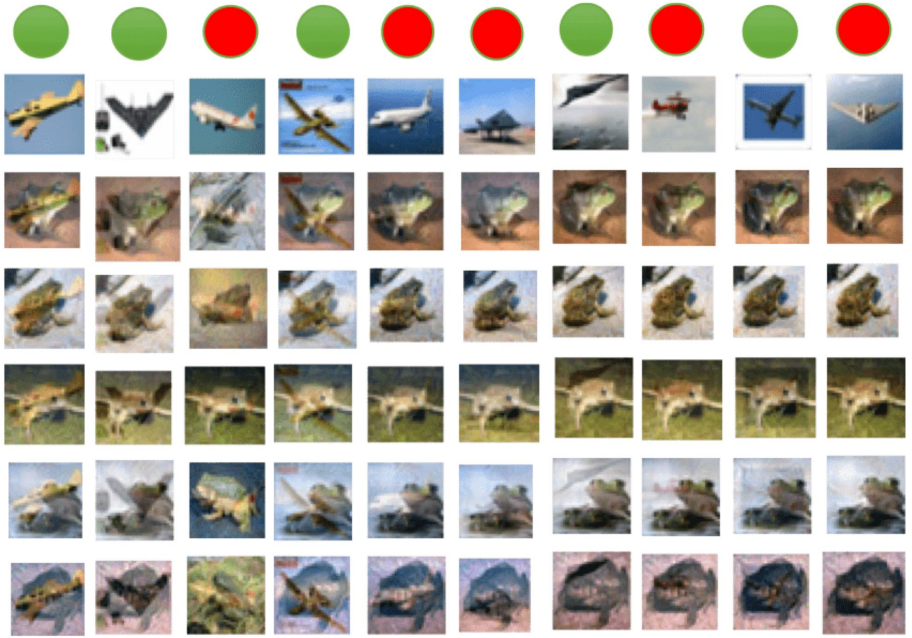
Gradient Descent: Minimize L2 distance to *target*
Proximal Update: Minimize Frobenius distance to *base* (input space)

Poison Sample Generation (Samples)



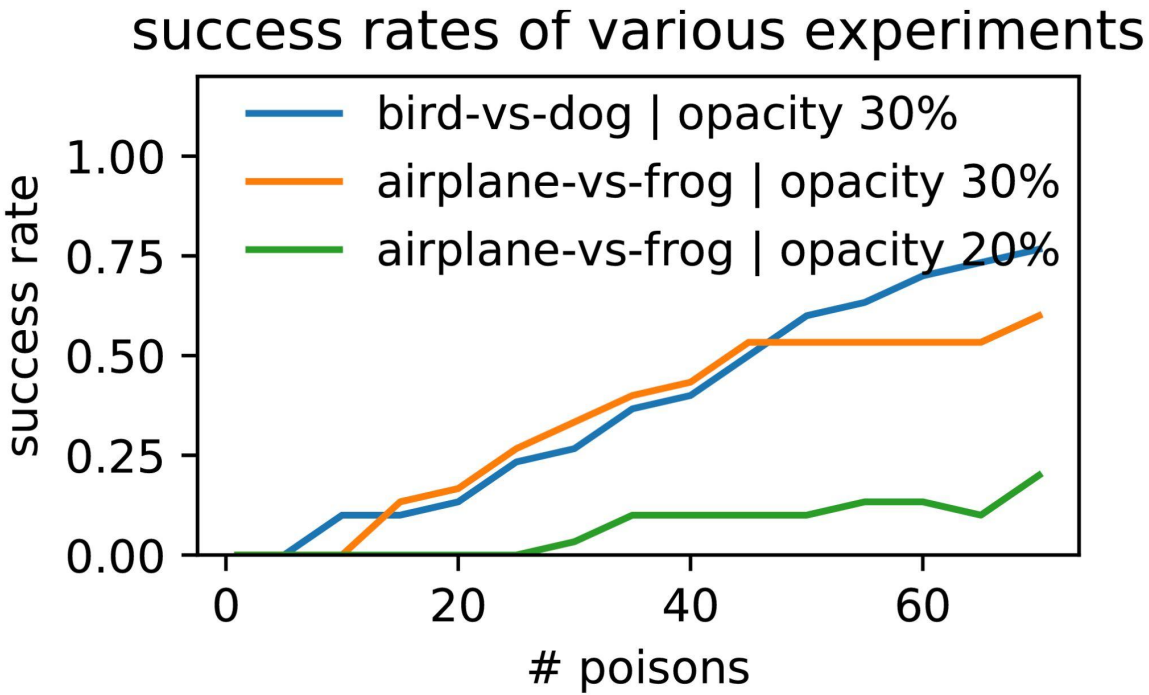
Watermarking Sample Generation

$$t\text{: } \mathbf{b} \leftarrow \gamma \cdot \mathbf{t} + (1 - \gamma) \cdot \mathbf{b}.$$



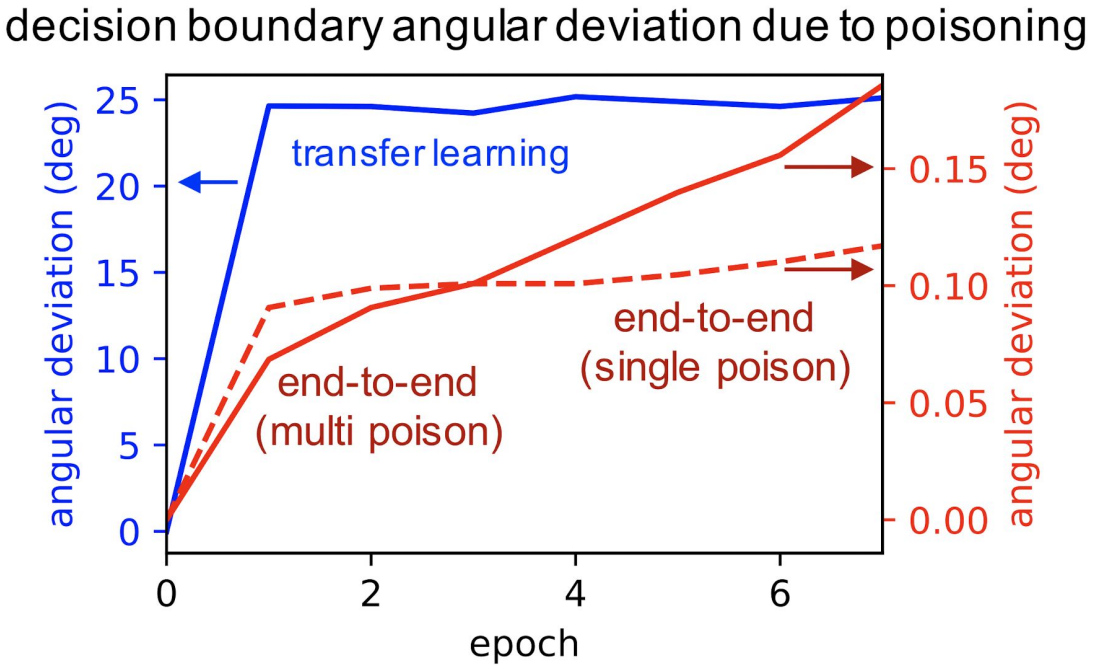
Results

Attack	Success Rate
Transfer Learning (Single Poison)	100%
End2End Retraining (Single Poison)	low
End2End Retraining (Multi Poison)	60%+



Results

Attack	Success Rate
Transfer Learning (Single Poison)	100%
End2End Retraining (Single Poison)	low
End2End Retraining (Multi Poison)	60%+



Strengths

- Clean-Label attacks are applicable on scenarios where manipulation of labels is not possible
- On transfer learning scenario, a single poison instance is enough

- On end-to-end retraining scenario, a few selected samples were enough for successful attacks

Limitations

- Clean-label attacks are applicable on scenarios where manipulation of labels is not possible
- On transfer learning scenario, a single poison instance is enough
  - Experiment was done with small dataset and large neural network:
    - ~1000 training samples, ~2000 NN parameters
- On end-to-end retraining scenario, a few selected samples were enough for successful attacks
  - Watermarked samples not always look visually similar to *base* image
  - Watermarked samples maybe be discarded during data collecting process or preprocessing
  - These watermarks can only be used on image data

Limitations

- Data needs to be found by target during data collection (unreliable)
- Attack works on a single target instance

Discussion

- Attack works on a single target instance, might not generalize to other instances close to the target
  - In the face recognition scenario:
    - Target image of person A may be misclassified by poisoned model
    - Different image of person A may not be misclassified by poisoned model

Follow Up Work

Transferable Clean-Label Poisoning Attacks on Deep Neural Nets

Chen Zhu [*1] **W. Ronny Huang** [*1] **Ali Shafahi** [1] **Hengduo Li** [1] **Gavin Taylor** [2] **Christoph Studer** [3]
Tom Goldstein [1]

Follow Up Work

MetaPoison: Practical General-purpose Clean-label Data Poisoning

W. Ronny Huang*
University of Maryland
wronnyhuang@gmail.com

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

Liam Fowl
University of Maryland
lfowl@math.umd.edu

Gavin Taylor
United States Naval Academy
taylor@usna.edu

Tom Goldstein
University of Maryland
tomg@cs.umd.edu

References

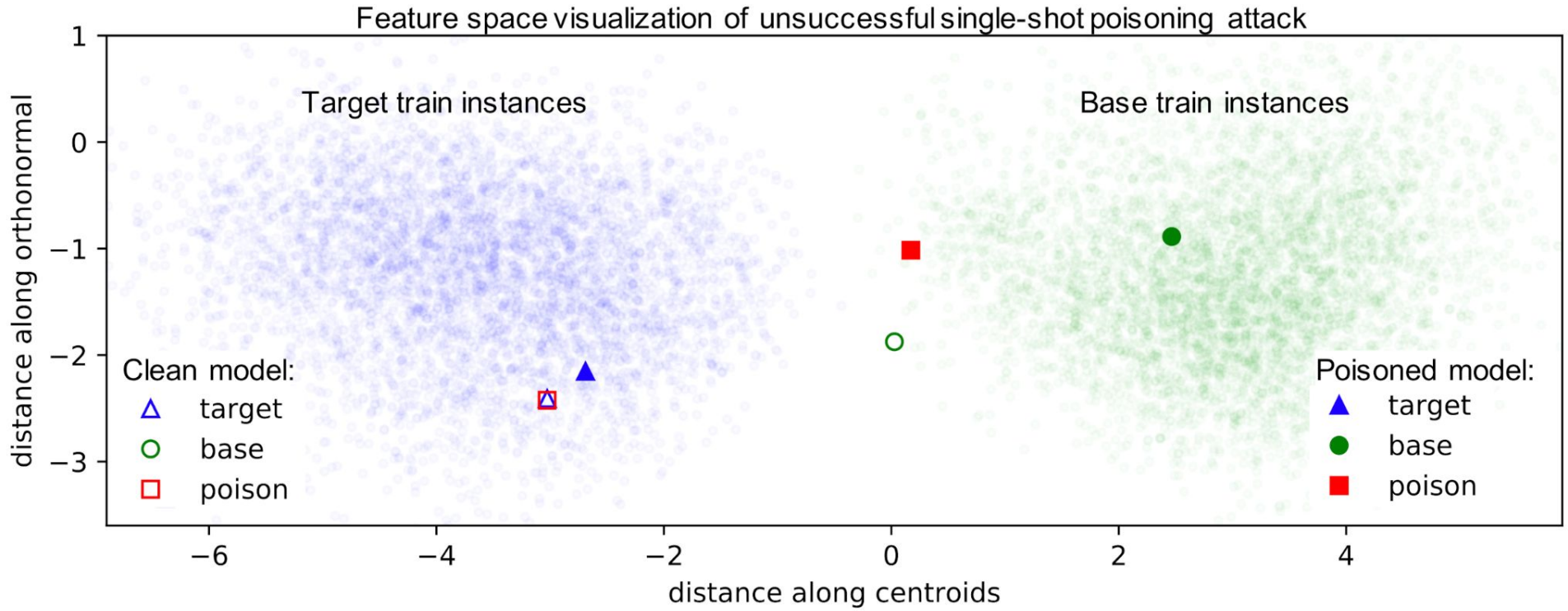
All images on this presentation are extracted from original paper:

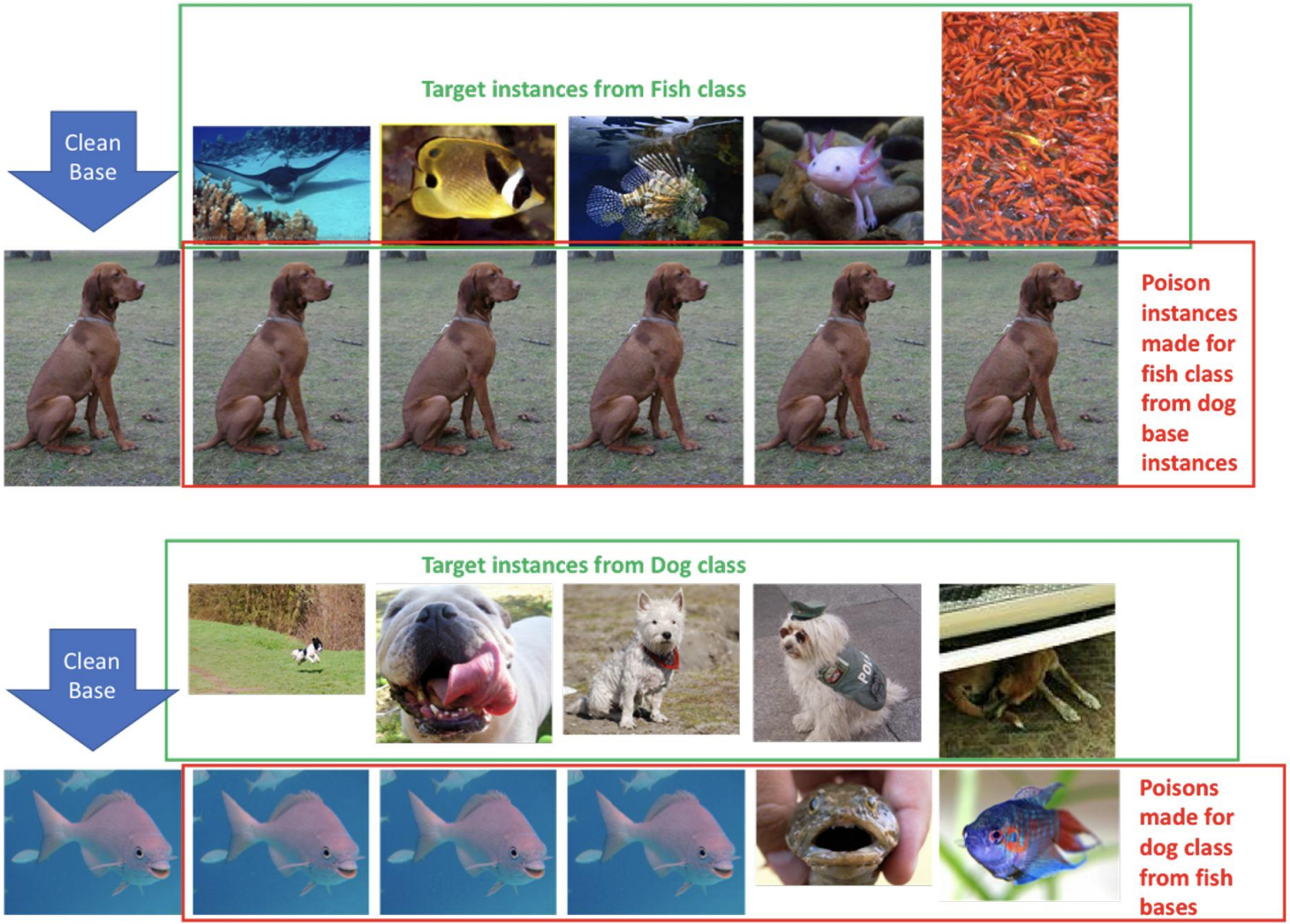
Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., & Goldstein, T. (2018). Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*.

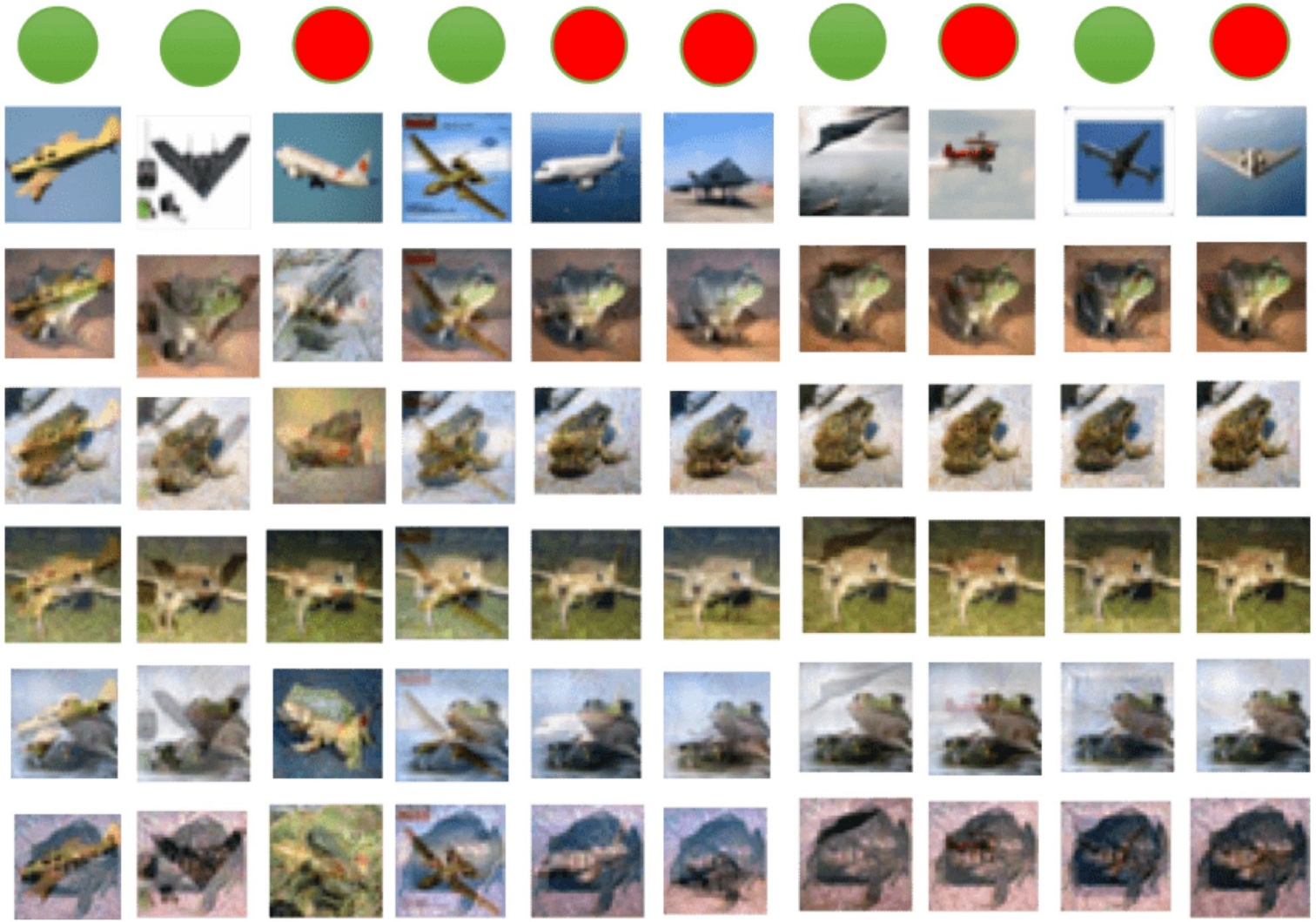
*Inserting Poisoned Sample on Training Dataset

- Requirement: the *poison* sample(s) need to be in the training dataset
- Options:
  - Crowd-sourced data sources & labelling
  - Place samples online to be collected by data-collection process (bot/human)

- The labelling can be done by the victim organization or a third-party
- The label for the *poisoned* sample should be the human-assigned label to the *base* sample


Feature space visualization of unsuccessful single-shot poisoning attack
Target train instances
Base train instances
Clean model:
target
base
poison
Poisoned model:
target
base
poison
distance along orthonormal
distance along centroids
1
0
−1
−2
−3
−6
−4
−2
0
2
4


Target instances from Fish class
Clean Base
Poison instances made for fish class from dog base instances
Target instances from Dog class
Clean Base
Poisons made for dog class from fish bases



Poisoning The Unlabelled Dataset Of Semi-Supervised Learning

Nicholas Carlini - Google

Presented by Zohair Shafi (October 18th 2021)

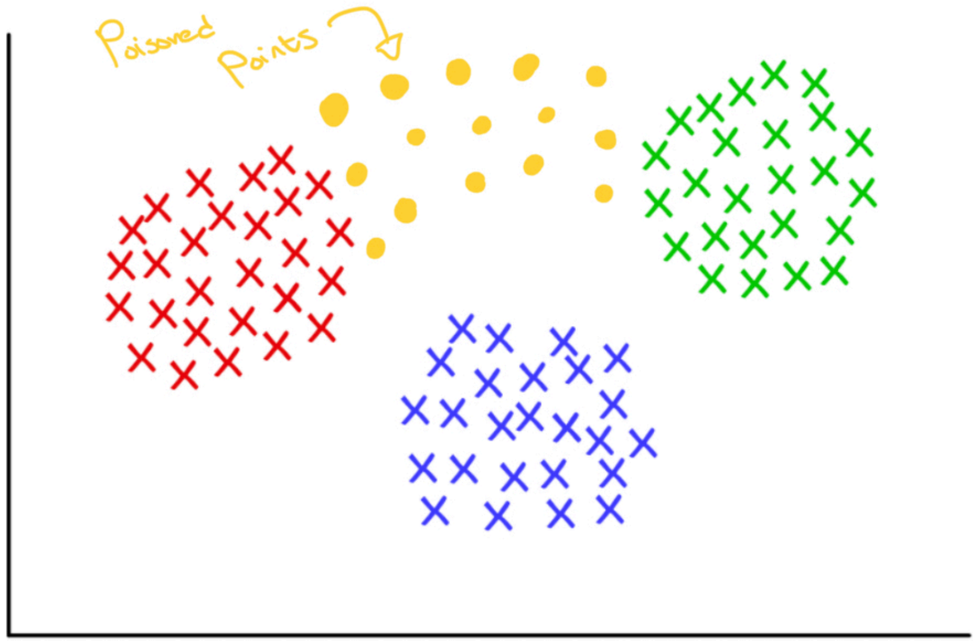
Semi-supervised Learning - Background

- Relies on a guessed label for for each unlabelled example.
- Use the model’s current predictions to supervise training for next weights
- Older methods are more ad-hoc and can be up to 9 times less accurate

Poisoning Unsupervised Methods

Clustering

- Inject unlabeled data indiscriminately to reduce model accuracy
- Construct a “bridge” that connects independent cluster of examples - causes clustering algorithm to group together both clusters into a new cluster



Threat Model

- Adversary needs to know :
  x^* : Input To Be Poisoned
  y^* : Incorrect Target Label ($\neq c(x^*)$)
  N : Number of examples that can be injected
  $X' \subset X$: Subset of labelled examples

Poisoning Unlabelled Dataset

- Many machine learning vulnerabilities are attributed to the fact that, instead of specifying **how** a task should be completed (e.g., look for three intersecting line segments), machine learning specifies **what** should be done (e.g., here are several examples of triangles)—and then we hope that the model solves the problem in a reasonable manner.

- Semi-supervised learning is a step further where we don’t even completely know what should be done.

Poisoning Unlabelled Dataset

Interpolation Consistency Poisoning

1. For a given $y*$, select an example x' from the labeled dataset such that $c(x') = y*$, i.e., x' is **correctly** classified as $y*$

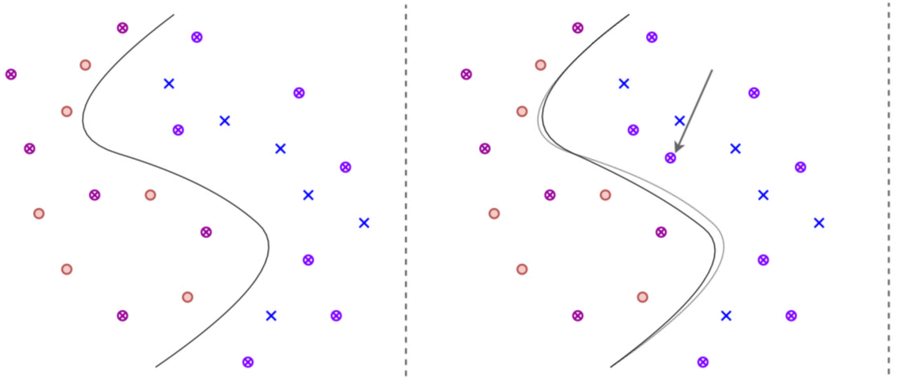
2. Similar to adding "bridging" points in the clustering example, insert N points that extrapolate from x' to $x*$:

$$\{x_{\alpha_i}\}_{i=0}^{N-1} = interp(x', x*, \alpha_i)$$
$$interp(x', x*, 0) = x'$$
$$interp(x', x*, 1) = x*$$

Poisoning Unlabelled Dataset

Interpolation Consistency Poisoning - Why Does This Work?

1. Neural networks are Lipschitz continuous with a low constant on average , i.e., if $f(x) = y$, then we usually have for small ϵ, $f(x + \epsilon) = y + \delta$, for some small $||\delta||$



(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.

2. Models generally apply data augmentation, hence are already trained on perturbed inputs $x + \epsilon$ by adding noise to x

Poisoning Unlabelled Dataset

Interpolation Consistency Poisoning - How Does This Work?

1. Unlabelled examples close to x' will now begin to be (correctly) classified as the true label. Once confidence reaches a threshold, training algorithm begins treating these examples as the labeled set. (Exact specifics differ between algorithms)

2. Recall there is now a path of images between x' and x^* :

- Model assigns the first unlabelled examples $x_{\alpha_0} = x'$ the label y^* (correct label)
- Learning algorithm encourages nearby samples - x_{α_1} - to be assigned the label given to x_{α_0}
- Process repeats with the label "propagating" until x_{α_N} is assigned the label y^*, i.e., all injected points are labelled as y^*

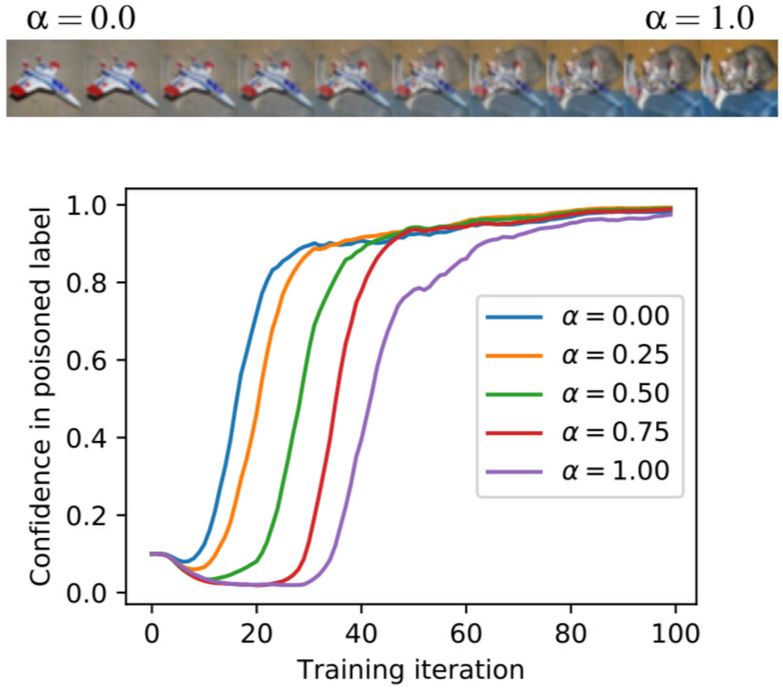


Figure 5: Label propagation of a poisoning attack over training epochs. The classifier begins by classifying the correctly-labeled source example x' (when $\alpha = 0$; image shown in the upper left) as the poisoned label. This propagates to the interpolation $\alpha > 0$ one by one, and eventually on to the final example x^* (when $\alpha = 1$; image shown in the upper right).

Poisoning Unlabelled Dataset

Interpolation Consistency Poisoning - Interpolation Strategy

How does one interpolate from x' to x^*?

1. Linear Pixel Wise Blending : $interp(x', x^*, \alpha) = x' . (1 - \alpha) + x^* . \alpha$
   (Easier to detect)
   How do you choose the value of α_i given $\alpha_0 = 0$ and $\alpha_{N-1} = 1$?

   1. Sample linearly within [0, 1], where $\alpha_i = i/N$

   2. Choose density function $\rho(x)$ that determines sampling rate

   3. Empirical evidence shows $\rho(x) = 1.5 - x$ is most effective.
      Sampling slightly more heavily from the source
      example and less so from poisoned target works better.
      $\rho(x)$ defined above samples 3 times more heavily around source x'
      than around target x^*

2. Generative Adversarial Networks
   (Not as easy to detect - semantic interpolation)

Density Function	CIFAR-10 % Poisoned 0.1%	0.2%	0.5%
$(1-x)^2$	0/8	3/8	7/8
$\phi(x+.5)$	1/8	5/8	7/8
$\phi(x+.3)$	2/8	7/8	8/8
x	3/8	4/8	6/8
$x^4+(1-x)^4$	3/8	5/8	8/8
$\sqrt{1-x}$	3/8	6/8	6/8
$x^2+(1-x)^2$	4/8	5/8	8/8
1	4/8	6/8	8/8
$(1-x)^2+.5$	5/8	7/8	8/8
$1-x$	5/8	8/8	8/8
$1.5-x$	7/8	8/8	8/8

Table 3: Success rate of poisoning a semi-supervised machine learning model using different density functions to interpolate between the labeled example x' (when $\alpha = 0$) and the target example x^* (when $\alpha = 1$). Higher values near 0 indicate a more dense sampling near x' and higher values near 1 indicate a more dense sampling near x^*. Experiments conducted with FixMatch on CIFAR-10 using 40 labeled examples.

Evaluation

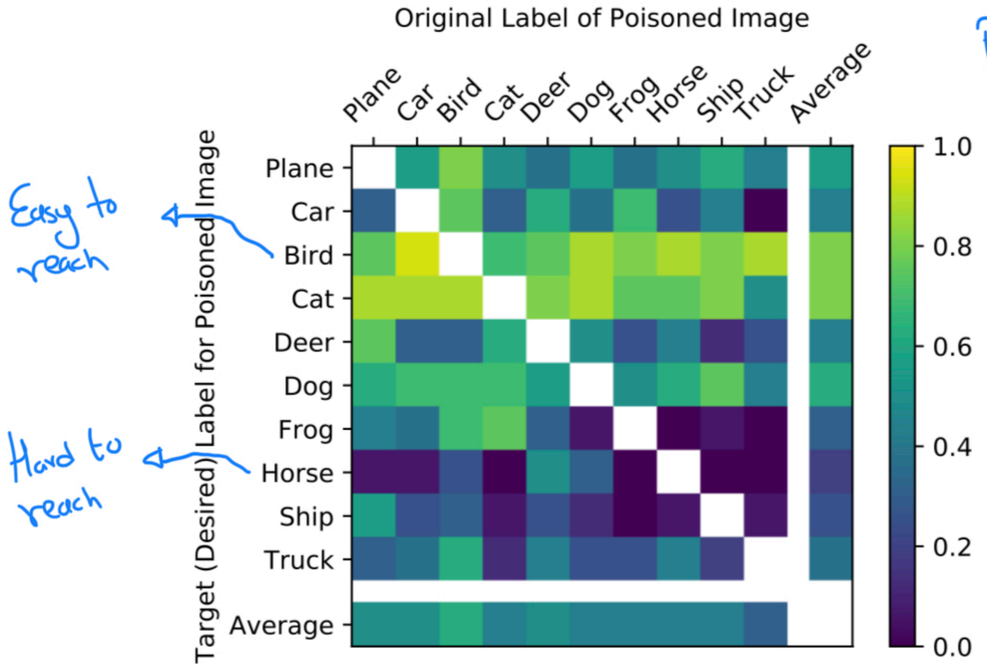


Figure 2: Poisoning attack success rate averaged across the ten CIFAR-10 classes. Each cell is the average of 16 trials. The original label of the (to-be-poisoned) image does not make attacks (much) easier or harder, but some target labels (e.g., horse) are harder to reach than others (e.g., bird).

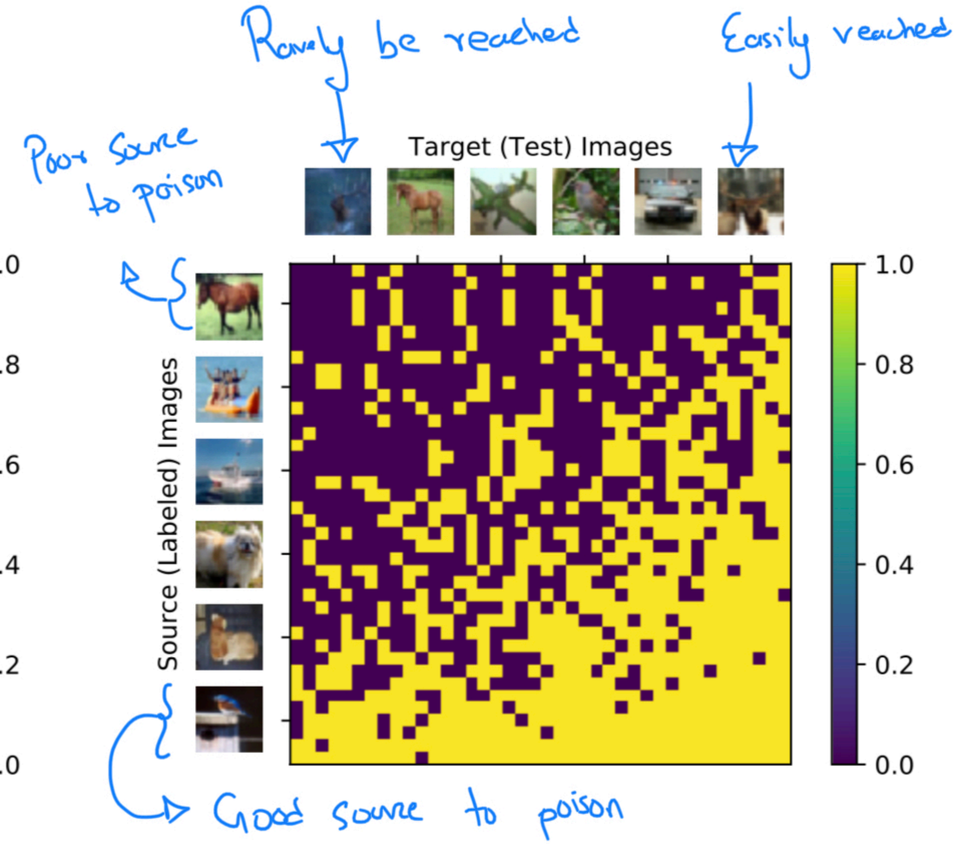


Figure 3: Poisoning attack success rate for all 40×40 source-target pairs; six (uniformly spaced) example images are shown on each axis. Each cell represents a single run of FixMatch poisoning that source-target pair, and its color indicates if the attack succeeded (yellow) or failed (purple). The rows and columns are sorted by average attack success rate.

Evaluation

Interesting Takeaways

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

Evaluation

Interesting Takeaways

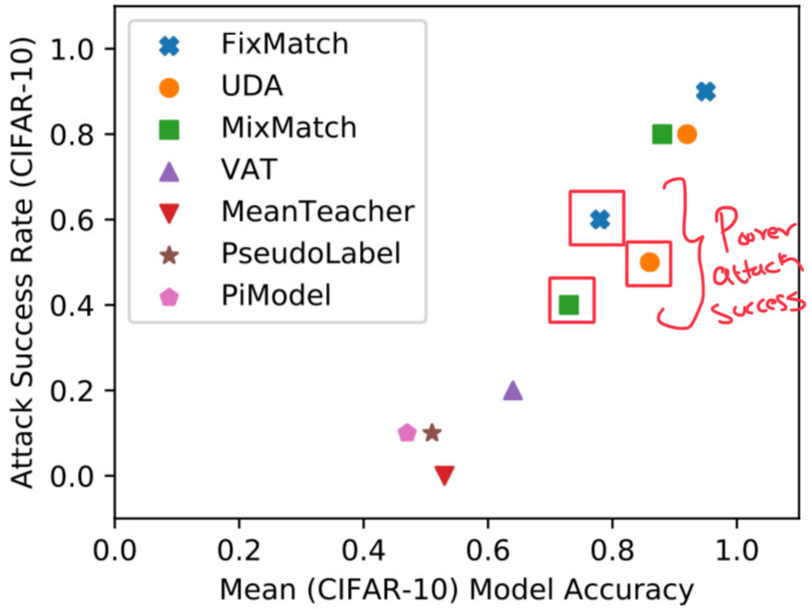


Figure 4: **More accurate techniques are more vulnerable.** Success rate of poisoning CIFAR-10 with 250 labeled examples and 0.2% poisoning rate. Each point averages ten trained models. FixMatch, UDA, and MixMatch were trained under two evaluation settings, one standard (to obtain high accuracy) and one small-model to artificially reduce model accuracy.

Dataset	**CIFAR-10**			**SVHN**			**STL-10**		
(% poisoned)	0.1%	0.2%	0.5%	0.1%	0.2%	0.5%	0.1%	0.2%	0.5%
MixMatch	5/8	6/8	8/8	4/8	5/8	5/8	4/8	6/8	7/8
UDA	5/8	7/8	8/8	5/8	5/8	6/8	-	-	-
FixMatch	7/8	8/8	8/8	7/8	7/8	8/8	6/8	8/8	8/8

Table 1: **Success rate of our poisoning attack across datasets and algorithms**, when poisoning between 0.1% and 0.5% of the unlabeled dataset. CIFAR-10 and SVHN use 40 labeled examples, and STL-10 all 1000. Our attack has a 67% success rate when poisoning 0.1% of the unlabeled dataset, and 91% at 0.5% of the unlabeled dataset (averaged across experiments).

Dataset	**CIFAR-10**			**SVHN**		
(# labels)	40	250	4000	40	250	4000
MixMatch	5/8	4/8	1/8	6/8	4/8	5/8
UDA	5/8	5/8	2/8	5/8	4/8	4/8
FixMatch	7/8	7/8	7/8	7/8	6/8	7/8

Table 2: Success rate of our attack when poisoning 0.1% of the unlabeled dataset when varying the number of labeled examples in the dataset. Models provided with more labels are often (but not always) more robust to attack.

Extensions

1. Zero Knowledge Attack
2. Generalized Interpolation
3. Transfer Learning

Extensions

Zero Knowledge Attack

- Adversary needs to know :
  x^* : Input To Be Poisoned
  y^* : Incorrect Target Label ($\neq c(x^*)$)
  N : Number of examples that can be injected
  ~~$X' \subset X$: Subset of labelled examples~~

- Use arbitrary unlabelled example x'?

Dataset (% poisoned)	CIFAR-10 0.5%	CIFAR-10 1.0%	SVHN 0.5%	SVHN 1.0%
MixMatch	2/8	4/8	3/8	4/8
UDA	2/8	3/8	4/8	4/8
FixMatch	3/8	4/8	3/8	5/8

Figure 6: Success rate of our attack at poisoning the unlabeled dataset without knowledge of any training examples. As in Table 1, experiments are across three algorithms, but here across two datasets.

  - Final label learned is neither y^* nor $c(x^*)$ - some intermediate image x_α exceeds confidence threshold, hence both x_α and $x_{\alpha_{N-1}}$ are classified as $c(x_\alpha)$

  - Fix by choosing additional targets $\{\hat{x}_i\}$ and connect each to x' with a path. Additional interpolations make it more likely for x' to be labelled correctly as y^*

Extensions

Generalized Interpolations

- Linear interpolations are easy to weed out - look for out of distribution images
- GAN's provide better semantic interpolations - harder for automated methods to discern :
  - Find z' and z^* such that $G(z') = x'$ and $G(z^*) = x^*$
  - Linearly interpolate from z' to z^* to obtain $x_i = G((1 - \alpha_i)z' + \alpha_i z^*)$
  - Not always possible to obtain latent vector z' such that $G(z') = x'$, hence find z' such that $||G(z') - x'||$ is small and perform standard interpolation between x' and $G(z')$
  - Attacks succeeded on 9 of 10 trials - lower success rate since path taken from x' to x^* is less direct

Extensions

Transfer Learning

- Assumes a slightly white box setting where we have access to the pre-trained model being used, i.e., we know the initial model weights θ_{init}
- We can then directly compute $x' = argmin||\delta||$ such that $\delta : f_{\theta_{init}}(x^* + \delta) = y^*$, i.e., search for example x' that is near the target x^* so that the model $f_{\theta_{init}}$ already assigned example x' the label y^* (Evasion Attack?)
- Because examples are closer, attack propagates faster

Negative Results

Density Funciton

- Analytically computing the optimal density function using binary search to determine where along the bridge to insert more examples did not work well.
- Precense or absence if one particular example is not independent of other examples - difficult to accurately measure true influence of any particular example

Negative Results

Multiple Sources

- Choosing multiple x' and interpolating each to a single x^* does not outperform choosing a single x' for a given budget of points.
- Its better to spend entire budget on a single source than to split into multiple sources

Negative Results

Noise To Poisoned Examples

- Add noise to the poisoned sample to ensure that no two interpolated points are too close to each other.

- Does not improve efficacy for small values of σ and makes it worse at higher values - if poisoned interpolations are similar, it helps model grow confidence on those samples

Defenses

Detecting Pixel Space Interpolations

- Linear blending attacks are trivially detectable
- There will exist atleast 3 examples that are colinear in the pixel space
- Use Agglomerative Clustering to create clusters of similar examples and get rid of the largest cluster (L_2 distance)
- Cannot detect GAN extrapolations
- Cannot work if adversary insets color-jitter or translations

Defenses

Monitoring Training Dynamics

- Guessed label for each unlabelled example is influenced by other unlabelled examples
  - Benign examples would be influenced by many other unlabeled examples
  - Poisoned examples are designed to impact mostly other poisoned examples - largely unaffected by other unlabeled examples.
- Compute pairwise influence
  - Let $\partial f_{\theta_i}(u_j) = f_{\theta_{i+1}}(u_j) - f_{\theta_i}(u_j)$ be the model's prediction vector after epoch i on jth unlabelled example. ∂f represents the difference in the model's predictions on a particular example from one epoch to the next.
  - For each example, let : $\mu_j^{a,b} = [\partial f_{\theta_a}(u_j), \partial f_{\theta_{a+1}}(u_j), \ldots, \partial f_{\theta_b}(u_j)]$
  - Compute Influence of examples as : $Influence(u_i, u_j) = \|\mu_i^{0,K-2} - \mu_j^{1,K-1}\|_2^2$
    i.e., example u_i influences example u_j if example u_i makes a change at epoch k, then example u_j makes a similar change in the next epoch because it has been influenced by u_i
  - Compute average influence of 5 nearest neighbors : $avfInfluence(u) = \frac{1}{5} \sum_{v \in U} Influence(u, v).1[close_5(u, v)]$
- At the cost of doubling training time - once with poisoned examples, second after removing then, it's possible to completely to remove the attack

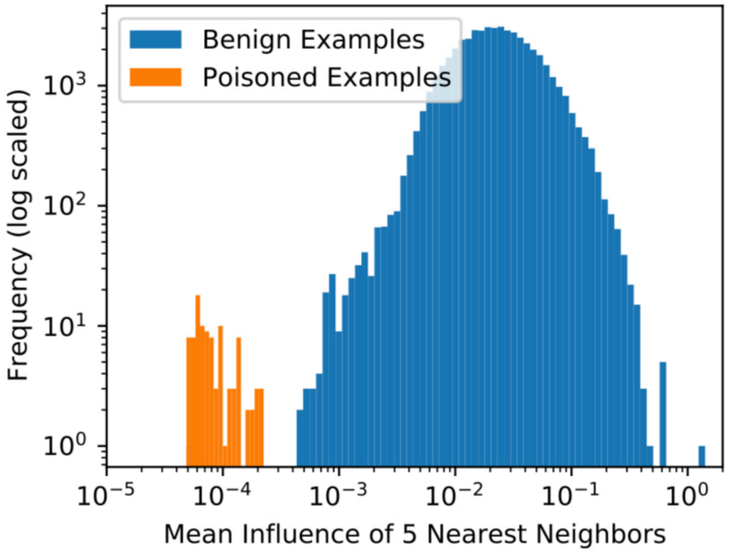


Figure 7: Our training-dynamics defense perfectly separates the inserted poisoned examples from the benign unlabeled examples on CIFAR-10 for a FixMatch poisoned model. Plotted is the frequency of the influence value across the unlabeled examples. Benign unlabeled examples are not heavily influenced by their nearest neighbors (indicated by the high values), but poisoned examples are highly dependent on the other poisoned examples (indicated by the low values).