Bullseye Polytope:

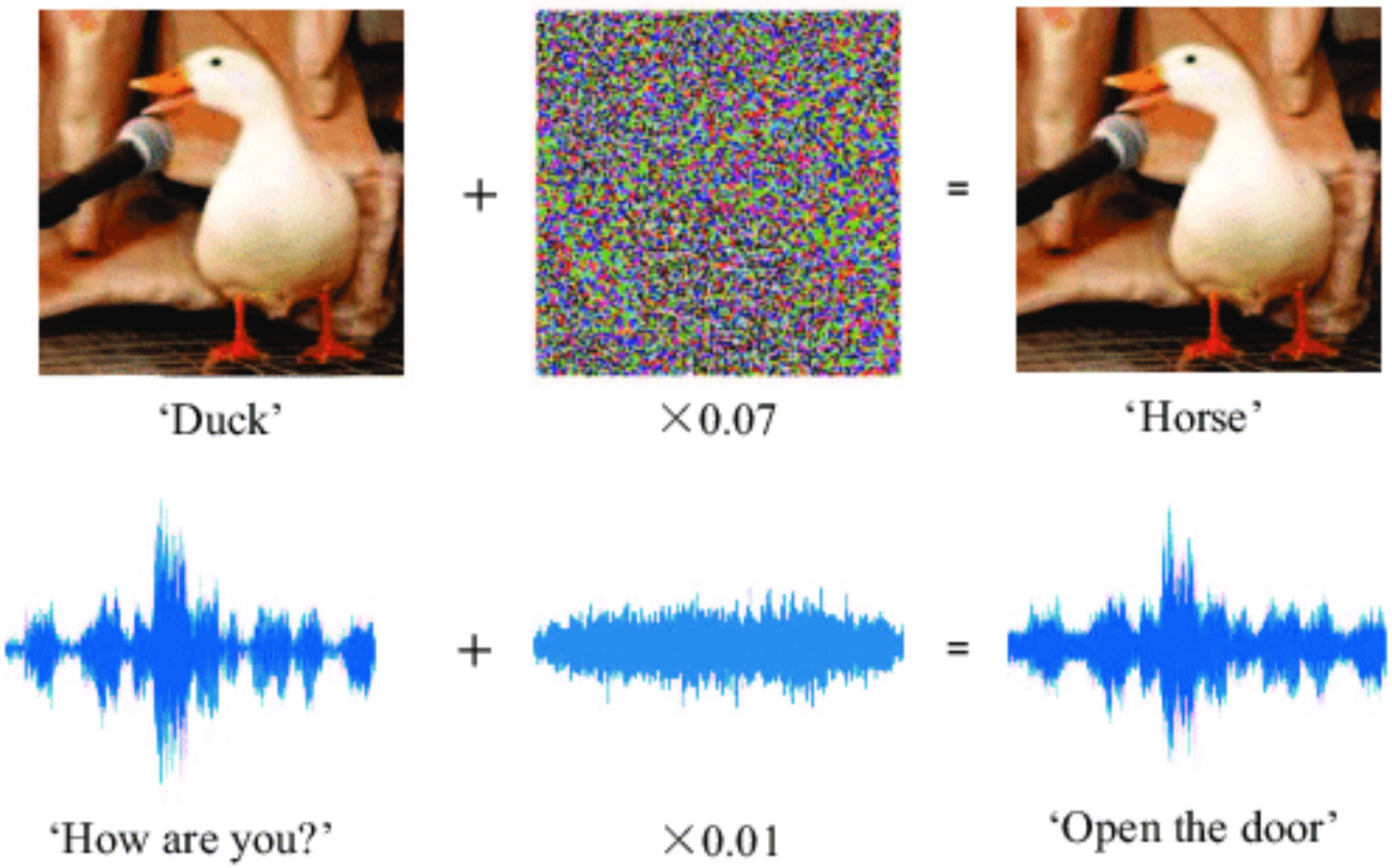
A Scalable Clean-Label Poisoning Attack with Improved Transferability

Hojjat Aghakhani, *Dongyu Meng, Yu-Xiang Wang,*

Christopher Kruegel, and Giovanni Vigna

University of California, Santa Barbara

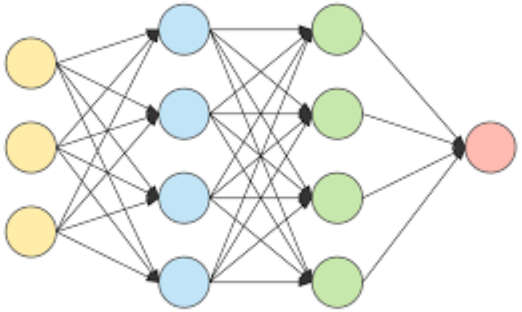
Security Threats in Machine Learning



Security Threats in Machine Learning

Training:

Training Data



Trained Classifier

Testing:

Test Sample

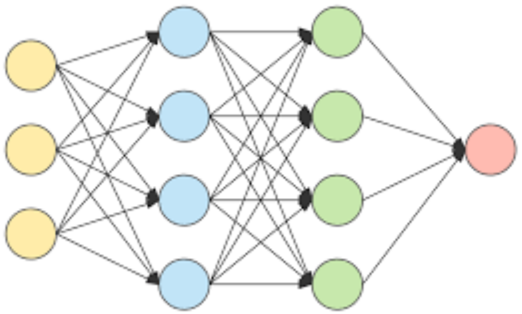
Trained Classifier

Prediction

Adversarial Examples/Evasion Attacks

Security Threats in Machine Learning

Training:

Training Data →  → Trained Classifier

Dataset Poisoning

Testing:

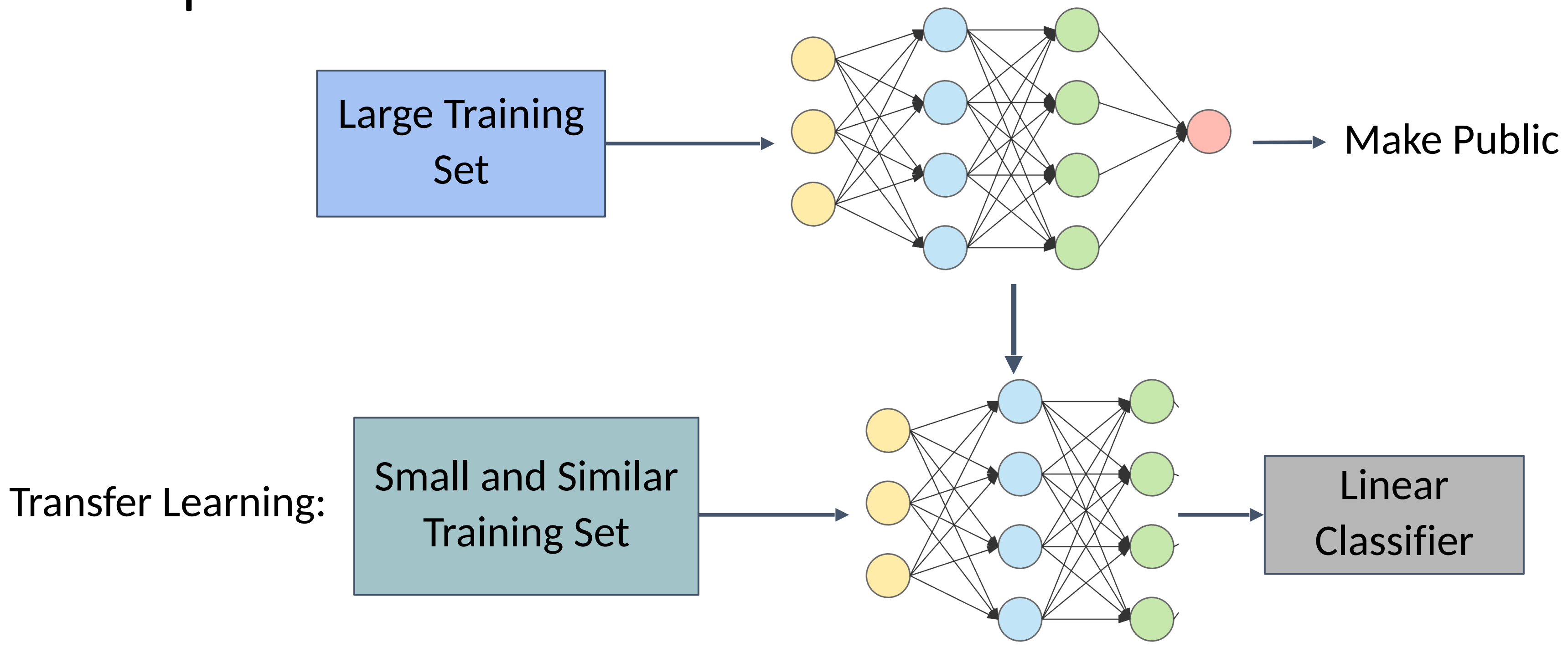
Test Sample → Trained Classifier → Prediction

Targeted Poisoning Against Transfer Learning

- Targeted → No effect on general performance!
- Clean-label
- Introduced first against transfer learning:
  - Feature Collision (Shafahi et al., 2018)
  - Convex Polytope (Zhu et al., 2019)

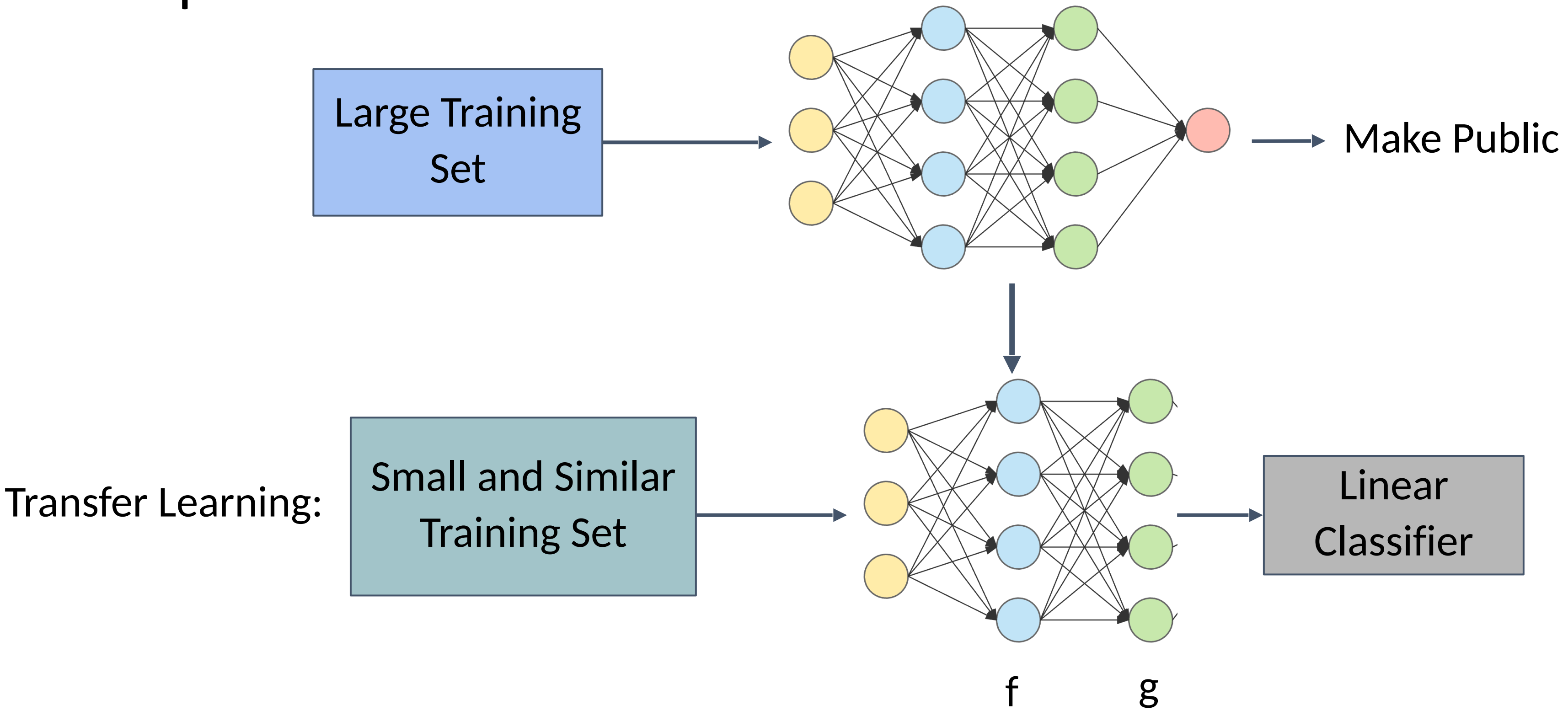
What Is Transfer Learning?

- Use a pre-trained network as the feature extractor to feed the features of the input to a linear classifier



What Is Transfer Learning?

- Use a pre-trained network as the feature extractor to feed the features of the input to a linear classifier

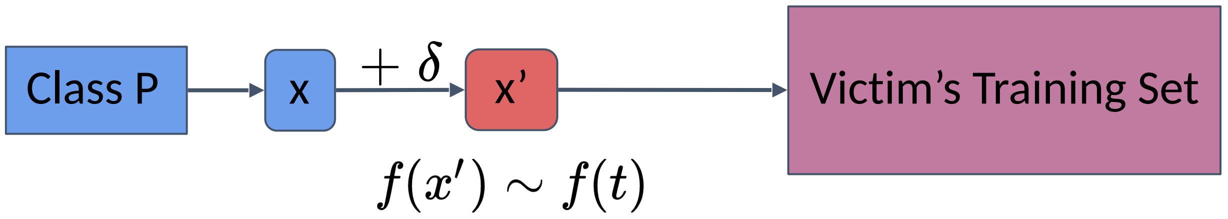


Goal?

- Goal: The attacker wants sample ***t*** to be classified into class ***P*** after the ***fine-tuning*** phase.
- How? By adding some poisoned data to the fine-tuning set.

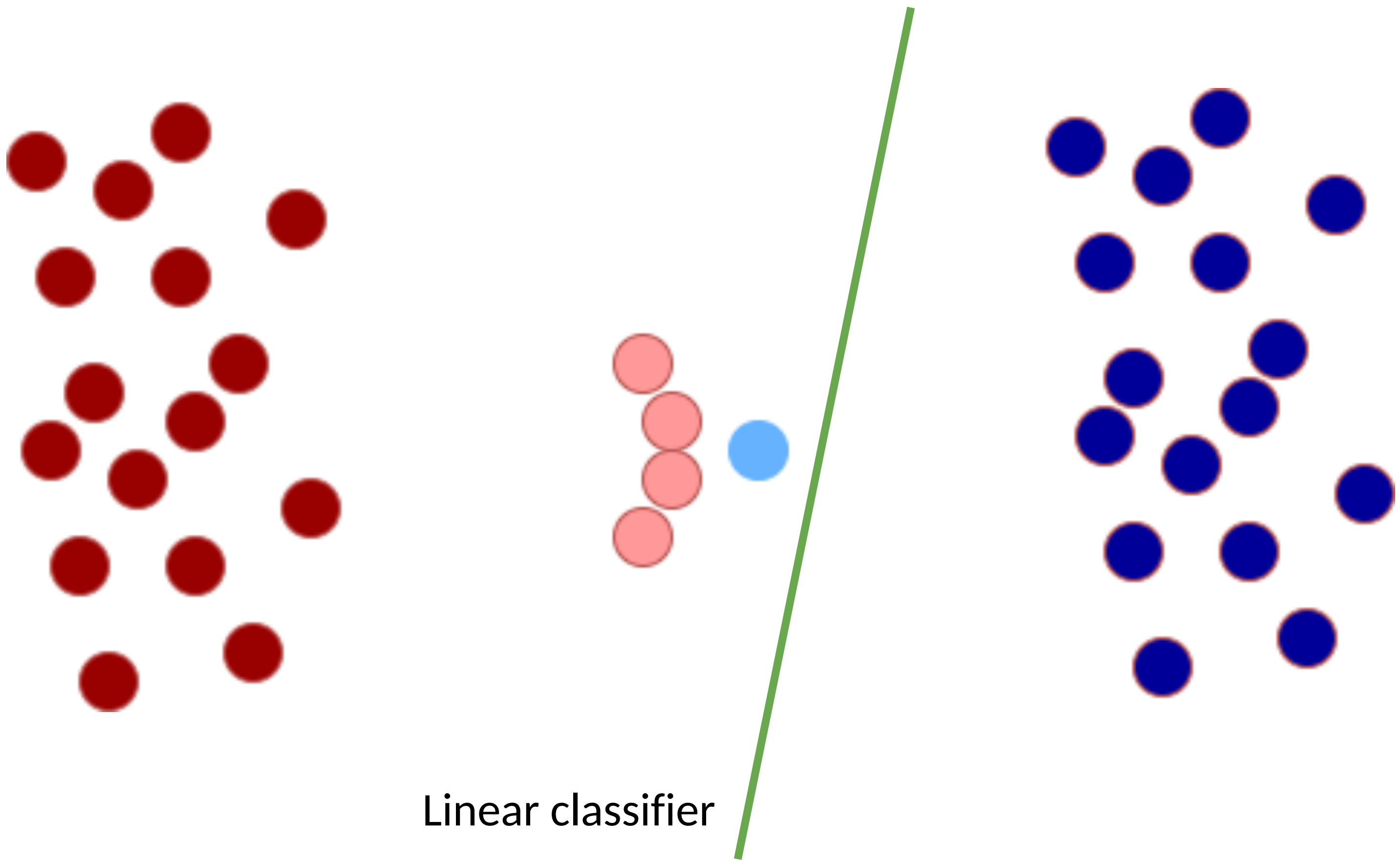
Feature Collision (Shafahi et al., 2018)

- f: The feature extractor (known to the attacker and used by victim)
  - White-box!
- g: The linear classifier (used by victim, not known to the attacker)
- t: The attacker wants sample t to be classified into class P.



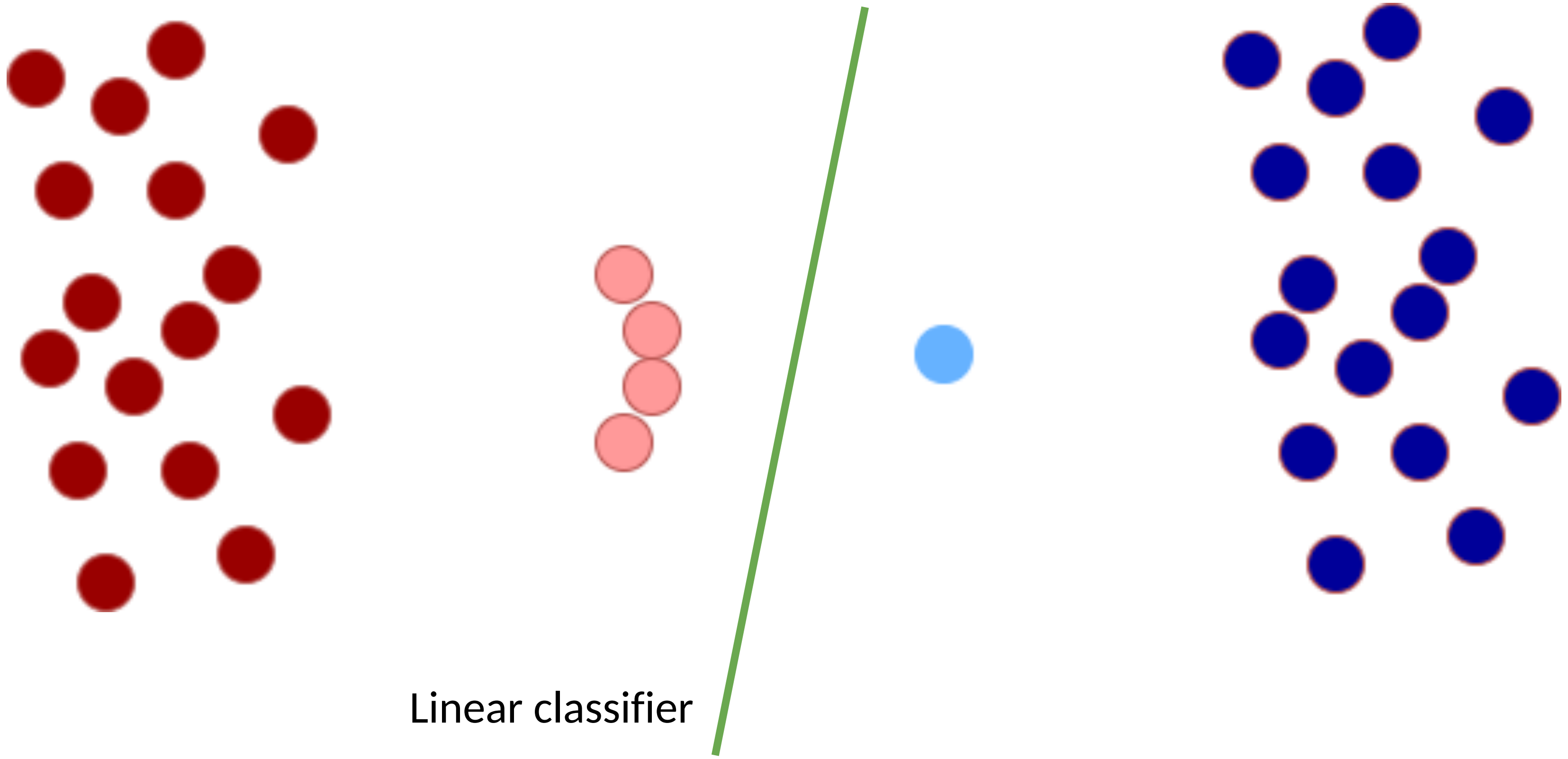
- The ultimate linear classifier learns to associate f(x') with the target class P.

Feature Collision Attack



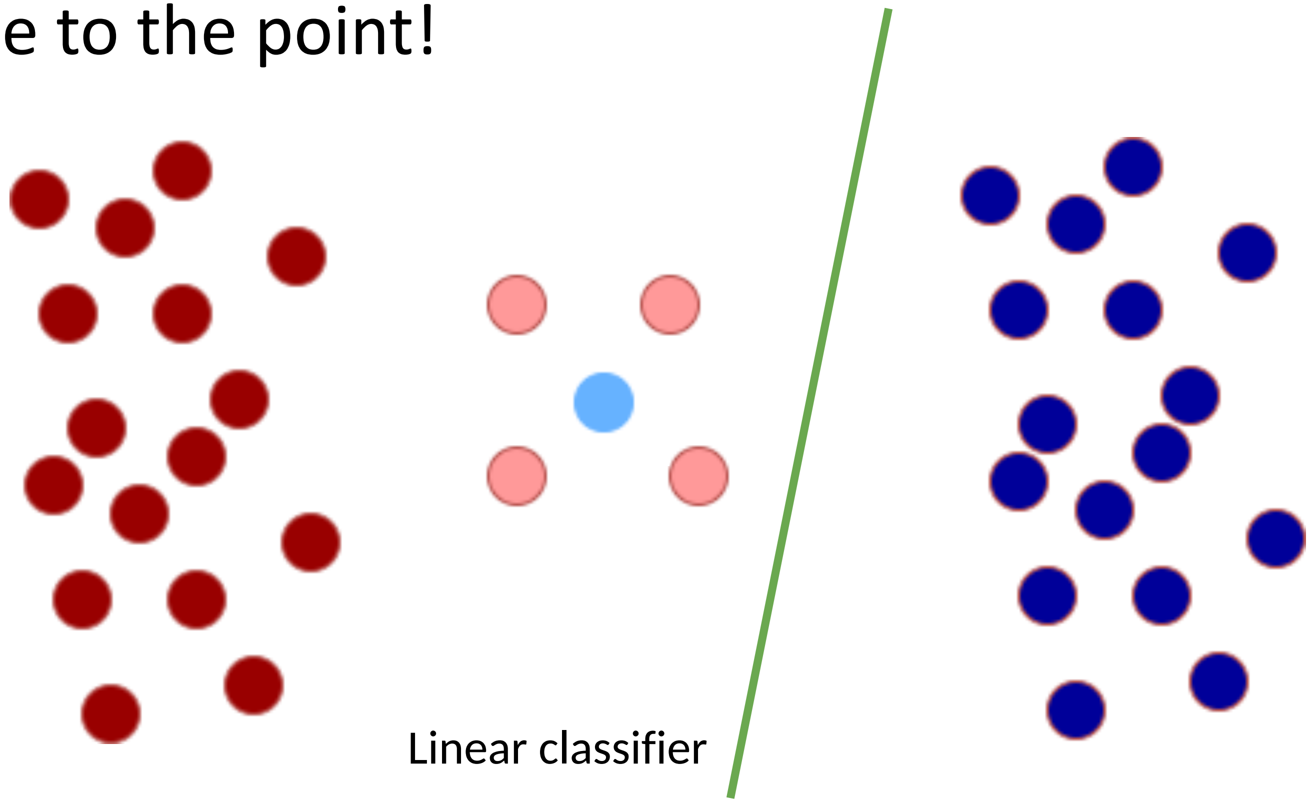
Feature Collision Attack

- Black-box: different feature extractor, i.e., different feature space



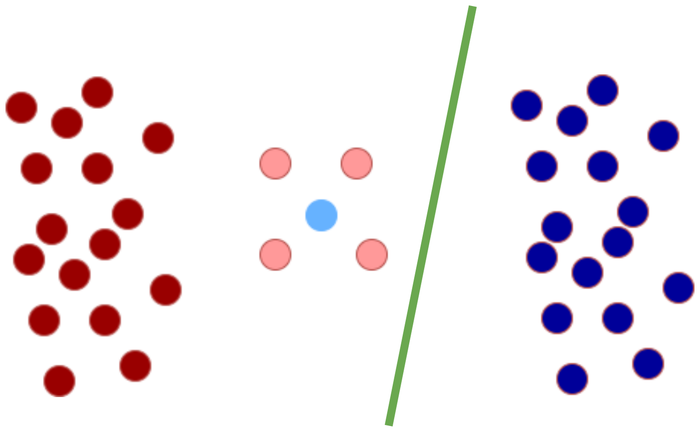
Convex Polytope (Zhu et al., 2019)

- Poison samples create a convex shape around the target, instead of all being close to the point!

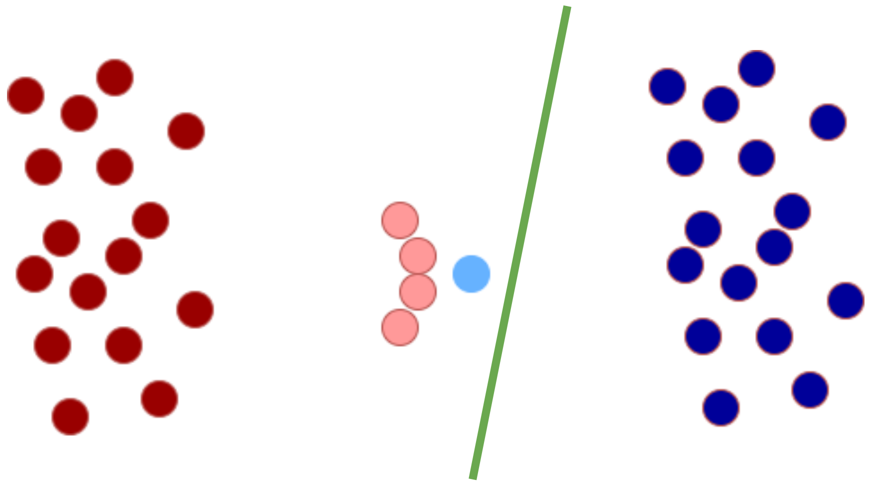


Convex Polytope – CP

- Compared to FC, CP creates a bigger shape in the feature space
- Thus, it increases the chance of transferability in black-box settings!
- CP outperforms FC by 20% on average across all experiments.



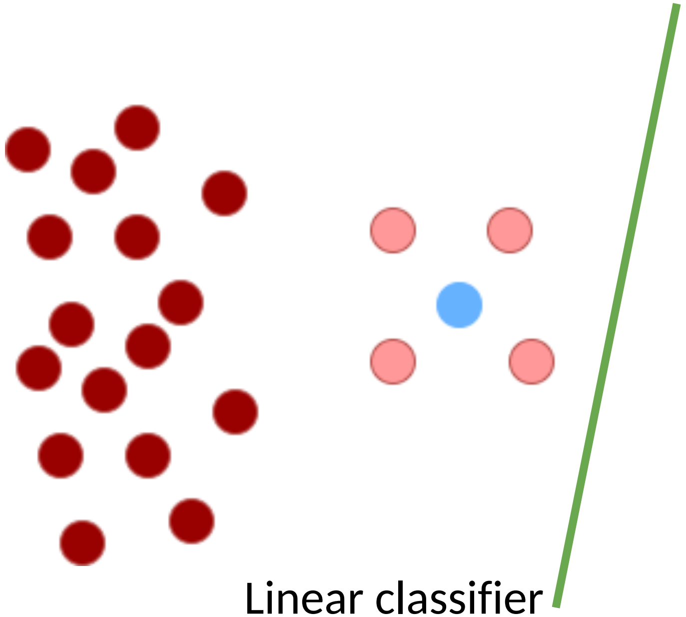
Convex Polytope
with a bigger attack zone



Feature Collision

Convex Polytope – CP

- But how such a polytope is created?



Using **m** surrogate networks,

with corresponding m feature spaces $\left\{\phi^{(i)}\right\}_{i=1}^{i=m}$

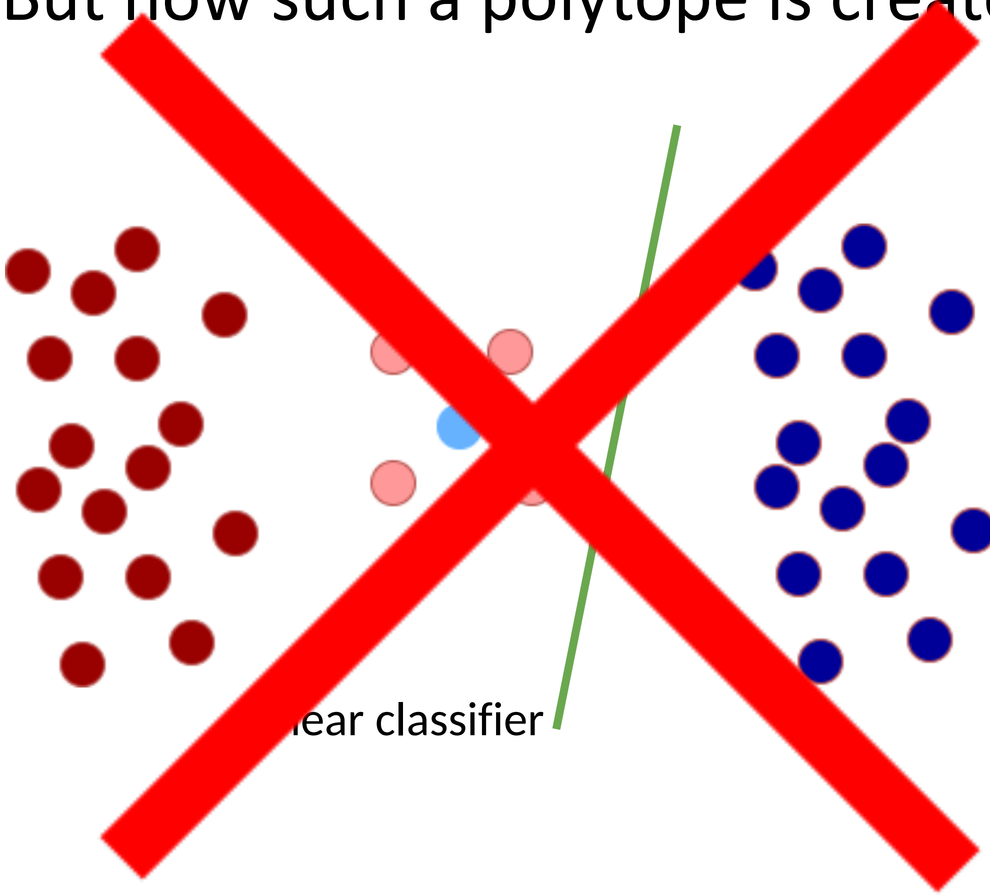
$$\underset{\{c^{(i)}\},\{x_p^{(j)}\}}{\text{minimize}} \quad \frac{1}{2m}\sum_{i=1}^{m}\frac{\left\|\phi^{(i)}(x_t)-\sum_{j=1}^{k}c_j^{(i)}\phi^{(i)}(x_p^{(j)})\right\|^2}{\left\|\phi^{(i)}(x_t)\right\|^2}$$

$$\text{subject to} \quad \sum_{j=1}^{k}c_j^{(i)}=1,\, c_j^{(i)}\geq 0, \forall i,j,$$

$$\left\|x_p^{(j)}-x_b^{(j)}\right\|_\infty \leq \epsilon\,, \forall j,$$

Convex Polytope – CP

- But how such a polytope is created?



Using **m** surrogate networks,

with corresponding m feature spaces $\left\{\phi^{(i)}\right\}_{i=1}^{i=m}$

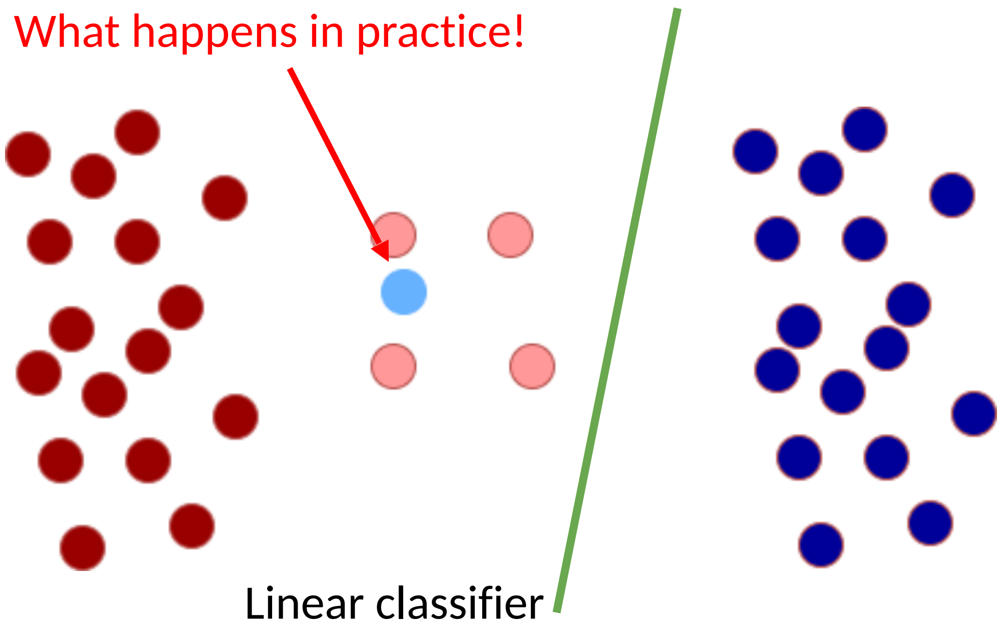
$$\underset{\{c^{(i)}\},\{x_p^{(j)}\}}{\text{minimize}} \quad \frac{1}{2m}\sum_{i=1}^{m}\frac{\left\|\phi^{(i)}(x_t)-\sum_{j=1}^{k}c_j^{(i)}\phi^{(i)}(x_p^{(j)})\right\|^2}{\left\|\phi^{(i)}(x_t)\right\|^2}$$

$$\text{subject to} \quad \sum_{j=1}^{k}c_j^{(i)}=1,\, c_j^{(i)}\geq 0, \forall i,j,$$

$$\left\|x_p^{(j)}-x_b^{(j)}\right\|_\infty \leq \epsilon\,, \forall j,$$

Convex Polytope – CP

- But how such a polytope is created?



Using **m** surrogate networks,

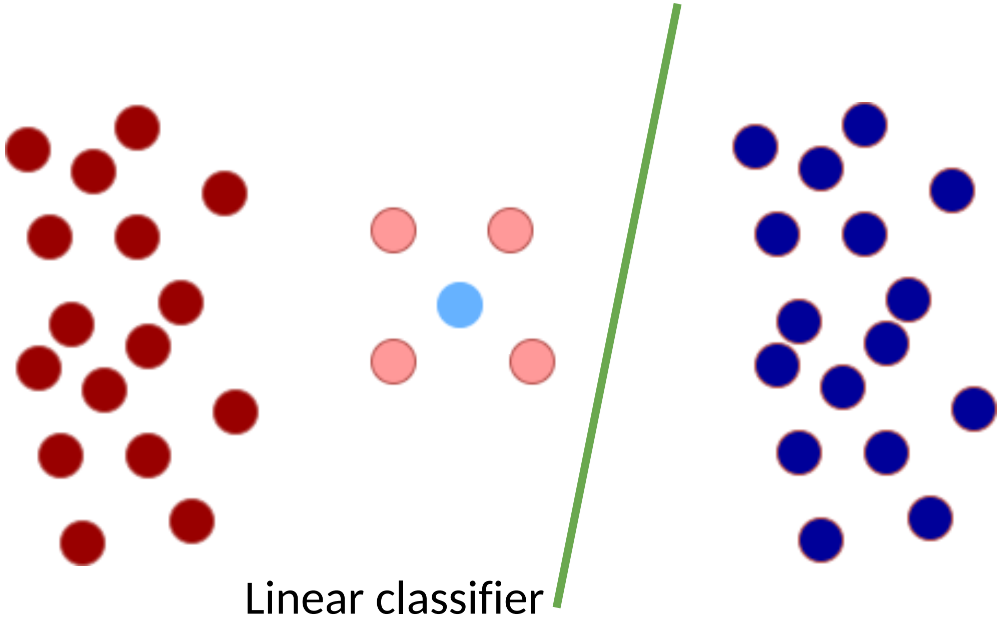
with corresponding m feature spaces $\left\{\phi^{(i)}\right\}_{i=1}^{i=m}$

$$\underset{\{c^{(i)}\},\{x_p^{(j)}\}}{\text{minimize}} \quad \frac{1}{2m}\sum_{i=1}^{m} \frac{\left\|\phi^{(i)}(x_t) - \sum_{j=1}^{k} c_j^{(i)} \phi^{(i)}(x_p^{(j)})\right\|^2}{\left\|\phi^{(i)}(x_t)\right\|^2}$$

$$\text{subject to} \quad \sum_{j=1}^{k} c_j^{(i)} = 1, c_j^{(i)} \geq 0, \forall i, j,$$

$$\left\|x_p^{(j)} - x_b^{(j)}\right\|_\infty \leq \epsilon \,, \forall j,$$

Bullseye Polytope – BP



Using **m** surrogate networks,

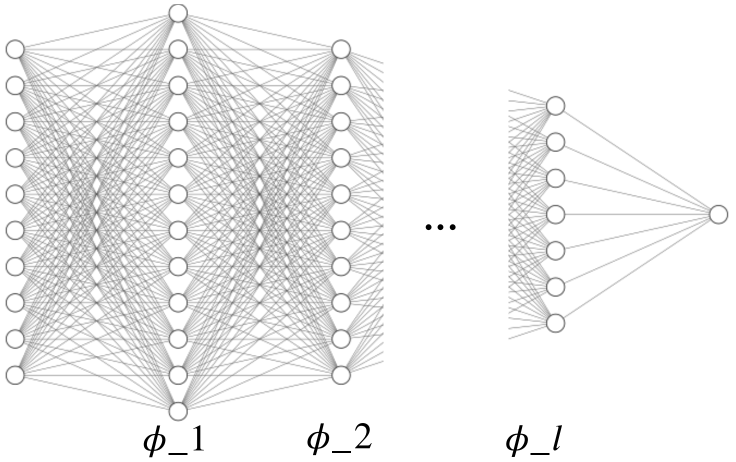
with corresponding m feature spaces $\left\{\phi^{(i)}\right\}_{i=1}^{i=m}$

$$\underset{\{x_p^{(j)}\}}{\text{minimize}} \ \frac{1}{2m}\sum_{i=1}^{m}\frac{\left\|\phi^{(i)}(x_t)-\frac{1}{k}\sum_{j=1}^{k}\phi^{(i)}(x_p^{(j)})\right\|^2}{\left\|\phi^{(i)}(x_t)\right\|^2}$$

$$\text{subject to} \ \left\|x_p^{(j)}-x_b^{(j)}\right\|_\infty \leq \epsilon\,, \forall j.$$

What About End-to-end Transfer Learning?

- We enforce the convex hull heuristic at each layer of the neural network



$$\begin{array}{ll} \underset{\{x_p^{(j)}\}}{\text{minimize}} & \frac{1}{2m}\Big(\sum_{i=1}^{m} \frac{||\phi_1^{(i)}(x_t) - \frac{1}{k}\sum_{j=1}^{k}\phi_1^{(i)}(x_p^{(j)})||^2}{||\phi_1^{(i)}(x_t)||^2} + \sum_{i=1}^{m} \frac{||\phi_2^{(i)}(x_t) - \frac{1}{k}\sum_{j=1}^{k}\phi_2^{(i)}(x_p^{(j)})||^2}{||\phi_2^{(i)}(x_t)||^2} + \ldots + \sum_{i=1}^{m} \frac{||\phi_l^{(i)}(x_t) - \frac{1}{k}\sum_{j=1}^{k}\phi_l^{(i)}(x_p^{(j)})||^2}{||\phi_l^{(i)}(x_t)||^2}\Big) \\ \text{subject to} & ||x_p^{(j)} - x_b^{(j)}||_\infty \le \epsilon \, , \forall j. \end{array}$$

Much More Scalable, With Improved Transferability

- Experiments Setup:
  - Using surrogate networks with 6 different architectures
  - Tested against two unseen architecture (black-box), and 6 known architectures, but with unseen parameters (different random seed is used)
  - #poisons=5, $\epsilon = 0.1$, #fine-tuning-set = 500.

Much More Scalable, With Improved Transferability

SECLAB

- Experiments Setup:
  - Using surrogate networks with 6 different architectures
  - Tested against two unseen architecture (black-box), and 6 known architectures, but with unseen parameters (different random seed is used)
  - #poisons=5, $\epsilon = 0.1$, #fine-tuning-set = 500.
- In linear transfer learning, BP outperforms CP by 10%, while being 7x faster!

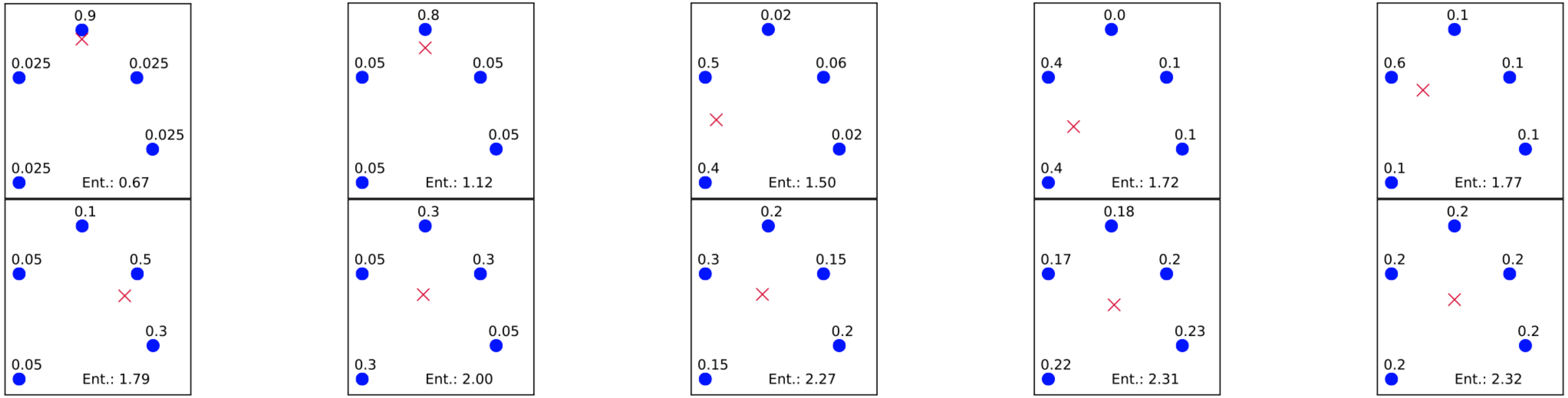
Much More Scalable, With Improved Transferability

SECLAB

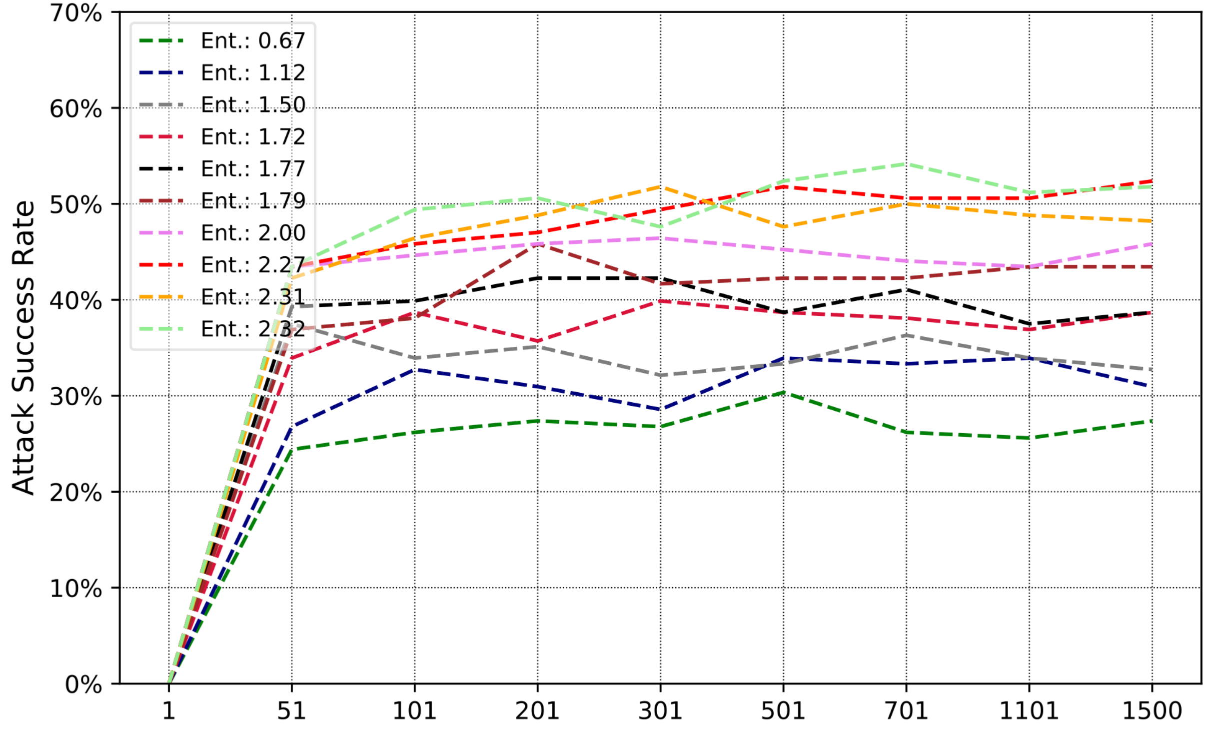
- Experiments Setup:
  - Using surrogate networks with 6 different architectures
  - Tested against two unseen architecture (black-box), and 6 known architectures, but with unseen parameters (different random seed is used)
  - #poisons=5, $\epsilon = 0.1$, #fine-tuning-set = 500.
- In linear transfer learning, BP outperforms CP by 10%, while being 7x faster!
- In end-to-end transfer learning, BP outperforms CP by 27%, while being 12x faster!

Why is BP better?

- Is it the “bullseye idea” contributing to its superior performance?
- Or its faster algorithm allows for better optimization?



Why is BP better?



BP with different fixed coefficients.

Independent Benchmark (Schwarzschild et al., 2020)

- Linear transfer learning:

	Linear Transfer Learning							
	CIFAR-10						TinyImageNet	
	White-box	Gray-box	Black-box				White-box	Black-box
Attack	ResNet18	ResNet18	ResNet34	ResNet50	VGG11	MobileNetV2	VGG16	$\frac{\text{ResNet34+MobileNetV2}}{2}$
FC	22	6	4	4	7	7	49	2
CP	33	7	5	4	8	7	14	1
BP	**85**	**10**	**8**	**6**	9	7	**100**	**10.5**
WiB	-	-	-	-	-	-	-	-
CLBD	5	5	4	4	7	6	3	1
HTBD	10	6	6	3	**14**	6	3	0.5

Independent Benchmark (Schwarzschild et al., 2020)

- Training from scratch:
  - Specifically taken into consideration by another attack, Witches' Brew (WiB) (Geiping et al., 2020)
  - Was published on arXiv (parallel to this work).

Training From Scratch

	CIFAR-10	TinyImageNet
Attack	$\frac{\texttt{VGG16+ResNet34+MobileNetV2}}{3}$	**VGG16**
FC	1.33	4
CP	0.67	0
BP	2.33	**44**
WiB	**26**	32
CLBD	1	0
HTBD	2.67	0

Defenses (Peri et al. 2019)

- Neighborhood conformity tests to sanitize the dataset!
- We evaluated against the only two effective defenses:
  - l2-norm centroid
  - Deep K-NN

Deep K-NN

- For each sample in the training set:
  - Looks at its k nearest neighbors, if the sample's label is not the mode, it's flagged!

k	# Deleted Poisons BP	# Deleted Poisons CP	# Deleted Samples BP	# Deleted Samples CP	Adv. Success Rate (%) BP	Adv. Success Rate (%) CP
0	-	-	-	-	42.5	37.25
1	3.18	4.28	36.46	37.02	20.50	6.75
2	2.42	3.86	21.91	23.07	24.75	8.00
3	3.81	4.66	27.86	27.87	11.75	1.50
4	3.48	4.60	25.83	26.69	14.75	2.50
6	4.22	4.85	25.39	25.91	8.25	1.25
8	4.77	4.94	25.69	25.80	1.25	0.00
10	4.97	4.95	26.36	26.33	0.00	0.25
12	4.98	4.96	26.58	26.54	0.00	0.00
14	4.98	4.96	26.21	26.21	0.00	0.00
16	4.98	4.96	26.95	26.92	0.00	0.00
18	4.98	4.96	26.36	26.37	0.00	0.00
22	4.98	4.96	26.62	26.59	0.00	0.00

(a) **# Poisons = 5**

k	# Deleted Poisons BP	# Deleted Poisons CP	# Deleted Samples BP	# Deleted Samples CP	Adv. Success Rate (%) BP	Adv. Success Rate (%) CP
0	-	-	-	-	57.75	51.25
1	4.30	7.56	38.77	41.22	49.25	14.00
2	2.71	6.38	22.75	25.77	51.75	21.25
3	4.92	8.16	30.36	31.88	38.75	11.00
4	3.94	7.76	26.74	29.72	46.75	12.50
6	4.82	8.51	26.57	29.44	40.00	7.25
8	5.68	9.03	27.24	29.87	31.25	3.25
10	6.53	9.31	28.30	30.54	26.50	2.25
12	7.42	9.44	29.19	30.82	17.75	1.25
14	8.17	9.54	29.42	30.54	15.25	0.25
16	8.86	9.59	30.63	31.20	8.00	0.00
18	9.50	9.61	30.60	30.63	3.00	0.00
22	9.91	9.61	31.18	30.85	0.25	0.00

(b) **# Poisons = 10**

Bullseye Polytope Attack - Summary

- Clean-label data poisoning against transfer learning
- Fixes an inherent flaw of Convex Polytope!
- An order of magnitude faster!
- Higher attack success rate!
- More resilient against defenses!

References

- J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, “Witches’ brew: Industrial scale data poisoning via gradient matching,” *arXiv preprint arXiv:2009.02276*, 2020.
- A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, “Poison frogs! targeted clean-label poisoning attacks on neural networks,” in *Advances in Neural Information Processing Systems*, 2018, pp. 6103–6113.
- Zhu, C., Huang, W.R., Li, H., Taylor, G., Studer, C. and Goldstein, T., 2019, May. Transferable clean-label poisoning attacks on deep neural nets. In International Conference on Machine Learning (pp. 7614-7623). PMLR. Vancouver
- Schwarzschild, Avi, Micah Goldblum, Arjun Gupta, John P. Dickerson, and Tom Goldstein. "Just How Toxic is Data Poisoning? A Benchmark for Backdoor and Data Poisoning Attacks." (2020).