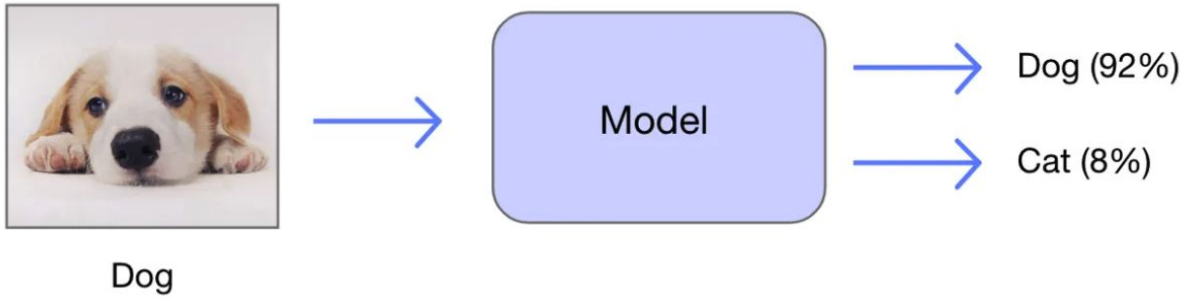
BadEncoder: Backdoor Attacks to Pre-trained Encoders in Self-Supervised Learning

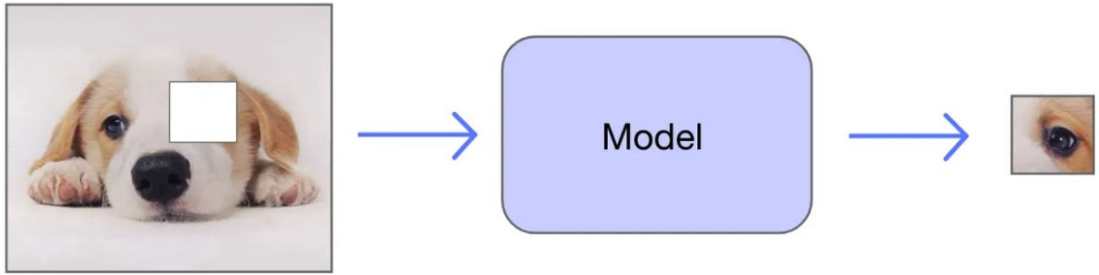
Yi Gao, Bofeng Chen, Zhicheng Guo, Danyu Sun

Self Supervised Learning

- Supervised Learning

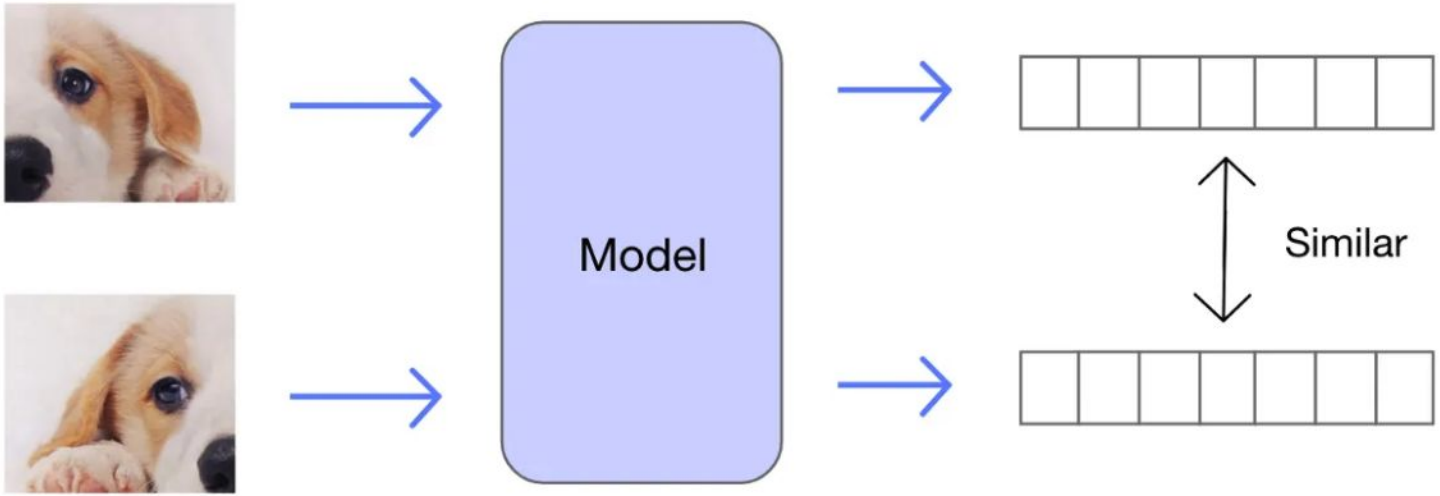


- Self Supervised Learning
  - create labels from data itself. And train the model using those labels.



Self Supervised Learning

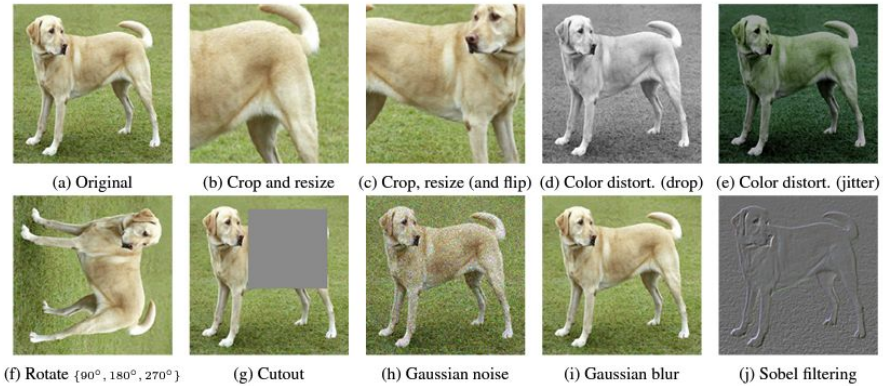
Contrastive Learning

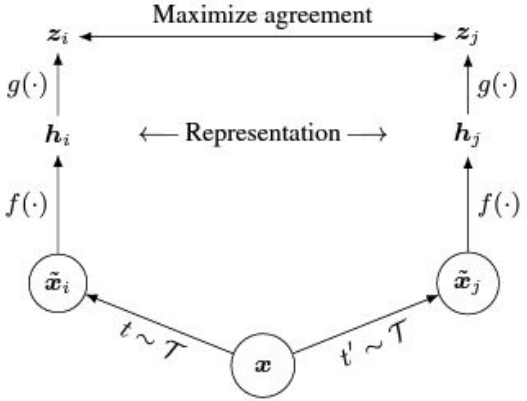


Self Supervised Learning

SimCLR (Simple framework for Contrastive Learning)

- Method





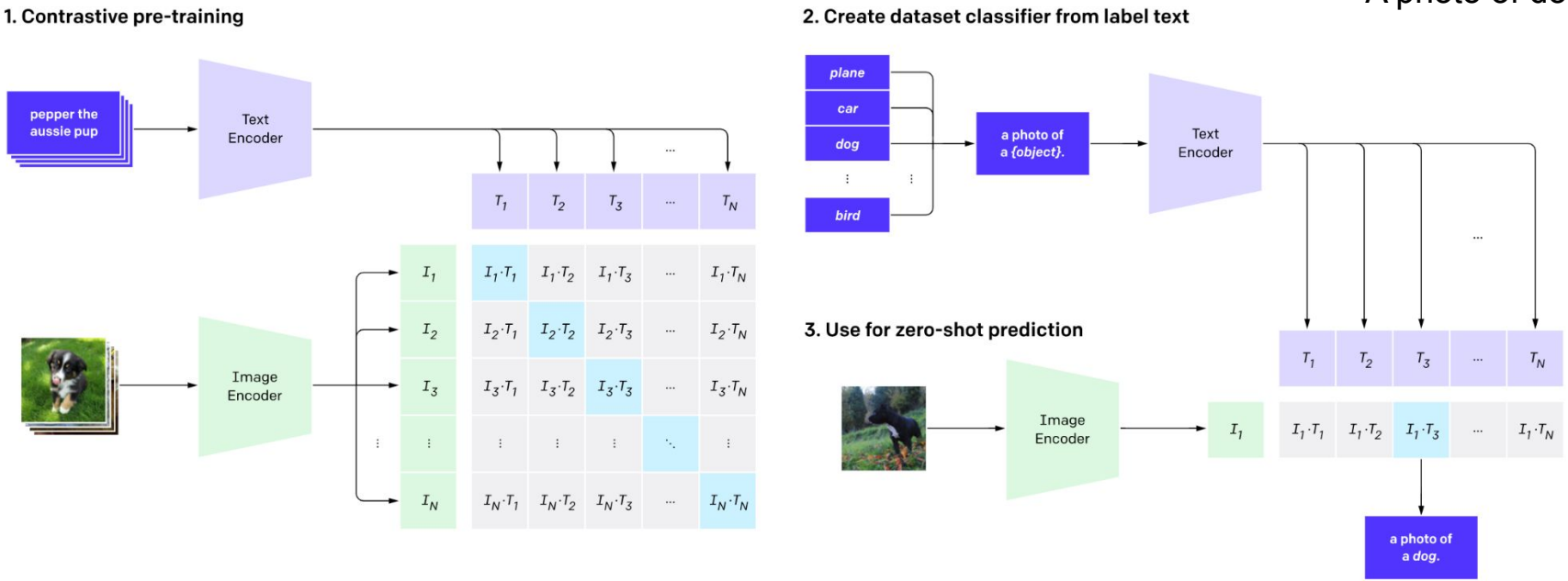
- Loss Function

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

Self Supervised Learning

"A photo of dog"

CLIP (Contrastive Language–Image Pre-training)



Self Supervised Learning

CLIP Downstream tasks

- AvatarCLIP

- CLIPDraw

Threat Model in Self Supervised Learning

Attacker's Goal:

Backdoored encoder → Backdoored Downstream task model

1. Effectiveness goal
    a. Targeted downstream task should maintain backdoor behavior
2. Utility goal
    a. Untargeted downstream task should maintain normal behavior

Threat Model in Self Supervised Learning

Attacker's Role:

- Untrusted service provider who pre-trains models
- Malicious third-party who re-publishes backdoored encoders

Attacker's Knowledge:

- Clean pre-trained encoder
- Set of unlabeled images (***shadow set***)
- Set of labeled images for each of the target downstream task+target class pair (***reference inputs***)

shadow dataset

backdoor triggers

reference inputs

Attacker Cannot Access:

Downstream classifier and its training process

BadEncoder

Overview

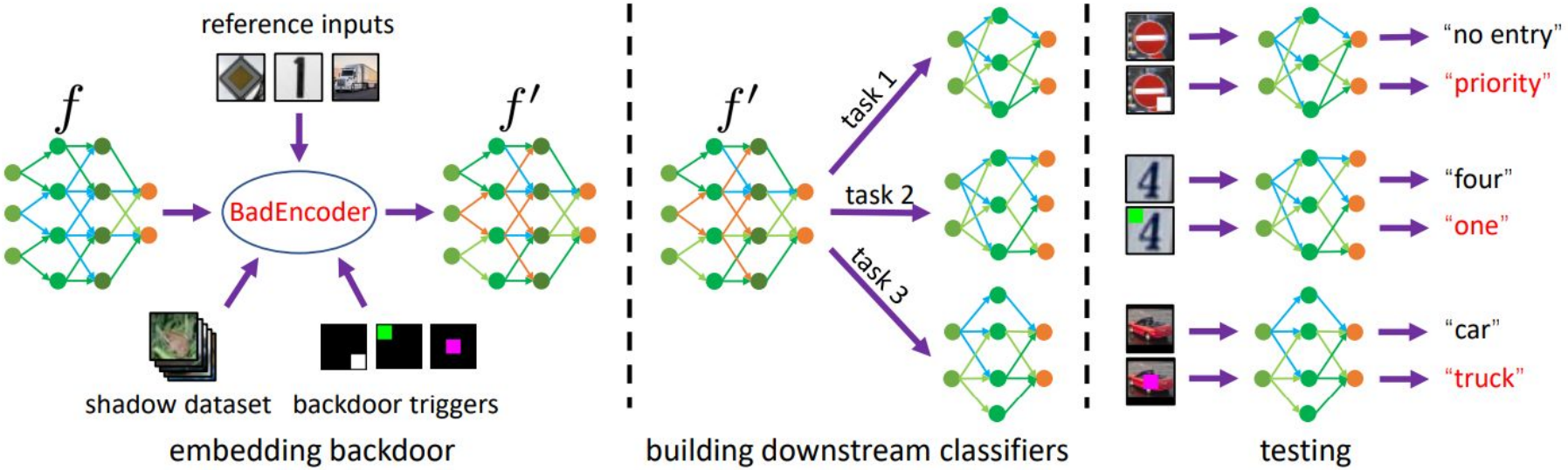


Fig. 1: Overview of BadEncoder.

BadEncoder

Optimization

$$\min_{f'} L = L_0 + \lambda_1 \cdot L_1 + \lambda_2 \cdot L_2,$$

Efficiency goal:

$$L_0 = -\frac{\sum_{i=1}^{t}\sum_{j=1}^{r_i}\sum_{\boldsymbol{x}\in\mathcal{D}_s} s(f'(\boldsymbol{x}\oplus\boldsymbol{e}_i), f'(\boldsymbol{x}_{ij}))}{|\mathcal{D}_s|\cdot\sum_{i=1}^{t} r_i},$$

Mapping backdoored image to other image.

$$L_1 = -\frac{\sum_{i=1}^{t}\sum_{j=1}^{r_i} s(f'(\boldsymbol{x}_{ij}), f(\boldsymbol{x}_{ij}))}{\sum_{i=1}^{t} r_i},$$

Make sure the other image's class does not change.

Utility goal:

$$L_2 = -\frac{1}{|\mathcal{D}_s|}\cdot\sum_{\boldsymbol{x}\in\mathcal{D}_s} s(f'(\boldsymbol{x}), f(\boldsymbol{x})).$$

Non-backdoored image remains unaffected.

BadEncoder

Evaluation

Dataset

1. CIFAR10: natural image classification
2. STL10: natural image classification
3. GTSRB: traffic sign image classification
4. SVHN: street view house number image classification
5. Food101: food image classification

E.g. Pretrain with CIFAR10, test with STL10, GTSRB, and SVHN for downstream tasks

Metrics

- Clean Accuracy (CA)
  - Clean downstream classifier on clean test images
- Backdoored Accuracy (BA)
  - Backdoored downstream classifier on clean test images
- Attack Success Rate (ASR)
  - Fraction of backdoored images predicted as target class for a backdoored downstream classifier

Experiment results

TABLE I: BadEncoder achieves high ASRs.

Pre-training Dataset	Target Downs-tream Dataset	ASR-B (%)	ASR (%)
CIFAR10	GTSRB	2.79	98.64
	SVHN	37.53	99.14
	STL10	10.38	99.73
STL10	GTSRB	1.67	95.04
	SVHN	46.11	97.64
	CIFAR10	12.30	98.51

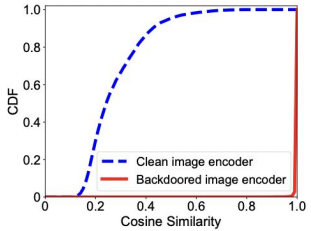


Fig. 4: The cumulative distribution functions (CDFs) of the cosine similarity scores between the feature vector of the reference input and those of the trigger-embedded inputs produced by the clean image encoder and backdoored image encoder.

Efficiency goal

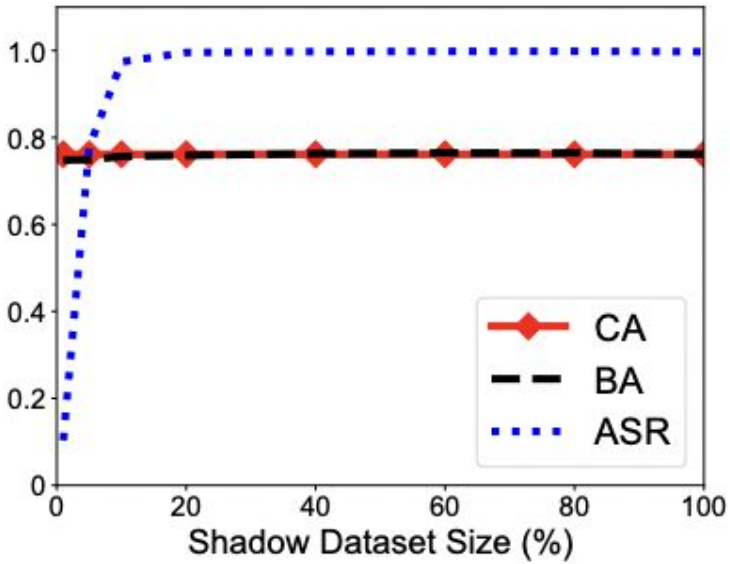
TABLE II: Our BadEncoder maintains the accuracy of the downstream classifiers.

Pre-training Dataset	Target Downs-tream Dataset	Downstream Dataset	CA (%)	BA (%)
CIFAR10	GTSRB	GTSRB	81.84	82.27
		SVHN	58.50	68.93
		STL10	76.14	75.94
	SVHN	GTSRB	81.84	82.19
		SVHN	58.50	69.32
		STL10	76.14	75.66
	STL10	GTSRB	81.84	82.55
		SVHN	58.50	68.68
		STL10	76.14	76.18
STL10	GTSRB	GTSRB	76.12	76.63
		SVHN	55.35	63.85
		CIFAR10	86.77	86.63
	SVHN	GTSRB	76.12	75.45
		SVHN	55.35	65.59
		CIFAR10	86.77	86.23
	CIFAR10	GTSRB	76.12	76.47
		SVHN	55.35	64.37
		CIFAR10	86.77	86.55

Utility goal

Impact of Shadow Dataset

The impact of the shadow dataset **size** on BadEncoder



Impact of Shadow Dataset

The impact of the shadow dataset **distribution** on BadEncoder

Case 1: a subset of the pre-training dataset

Case 2: the same distribution as the pre-training dataset but does not overlap with it

Case 3: a different distribution with the pre-training dataset

Results:

- High attack success rates
- preserves accuracy of the downstream classifiers in all the three cases
- shadow dataset does not need to be from the pre-training dataset nor follow its distribution

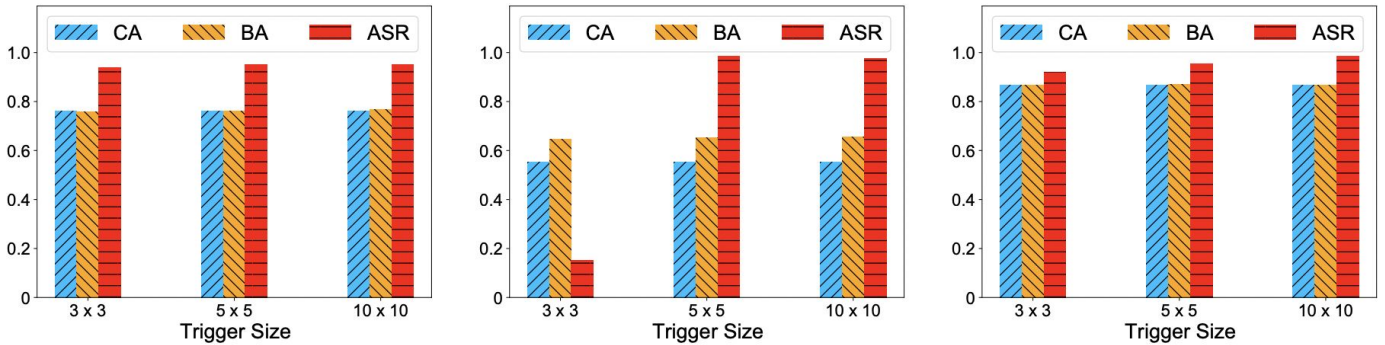
TABLE IV: The impact of the shadow dataset's distribution on BadEncoder.

Target Downstream Dataset	Shadow Dataset	CA (%)	BA (%)	ASR (%)
GTSRB	A subset of pre-training dataset	81.84	81.21	98.19
	Same distribution		81.12	97.52
	Different distributions		82.21	93.27
SVHN	A subset of pre-training dataset	58.50	62.32	98.30
	Same distribution		62.07	98.06
	Different distributions		60.40	84.80
STL10	A subset of pre-training dataset	76.14	75.90	99.55
	Same distribution		75.70	99.43
	Different distributions		75.99	98.15

Impact of Trigger Size

Results:

- High attack success rates when the trigger size is no smaller than some threshold
- BadEncoder with different trigger sizes do not sacrifice the utility of the pre-trained image encoder



(a) GTSRB (b) SVHN (c) CIFAR10

Case Study - 1 Attacking Image Encoder Pre-trained on ImageNet

Experimental setup:

Target class: “truck” (STL10), “priority sign”(GTSRB), “digit one” (SVHN)

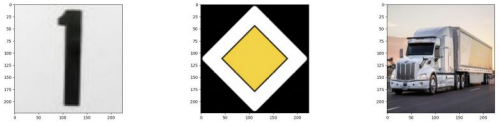
Setting: λ1 = 1 and λ2 = 1;

Shadow dataset: sample 1% of the training images of ImageNet;

Trigger: 50*50 white square at bottom right corner

Fine-tune the pre-trained image encoder for 200 epochs with learning rate 10−4

Batch size 16 to inject the backdoor



(a) SVHN (b) GTSRB (c) STL10

Fig. 10: The reference inputs for attacking the image encoder pre-trained on ImageNet by Google, which are used in Table VIII.

Case Study - 1 Attacking Image Encoder Pre-trained on ImageNet

Experimental results:

TABLE VIII: BadEncoder achieves high attack success rates and maintains the accuracy of the downstream classifiers when attacking the image encoder pre-trained on ImageNet by Google [4].

Target Downstream Dataset	CA (%)	BA (%)	ASR-B (%)	ASR (%)
GTSRB	76.53	78.42	5.47	98.93
STL10	95.66	95.68	10.24	99.99
SVHN	72.55	73.77	32.28	99.93

Case Study - 2 Attacking CLIP

Experimental setup:

- Multi-shot classifier

  Same as case study 1

- zero-shot classifier:

  Target class: “truck” (STL10), “stop sign”(GTSRB), “digit one” (SVHN)

  Context sentences: “A photo of a {class name}” (STL10,SVHN);

  “A traffic sign photo of a {class name}” (GTSRB)

Case Study - 2 Attacking CLIP

Experimental results:

TABLE IX: BadEncoder achieves high attack success rates and maintains the accuracy of the downstream classifiers when attacking CLIP [7].

(a) Multi-shot classifiers

Target Downstream Dataset	CA (%)	BA (%)	ASR-B (%)	ASR (%)
GTSRB	82.36	82.14	5.37	99.33
STL10	97.09	96.69	10.00	99.81
SVHN	70.60	70.27	20.79	99.99

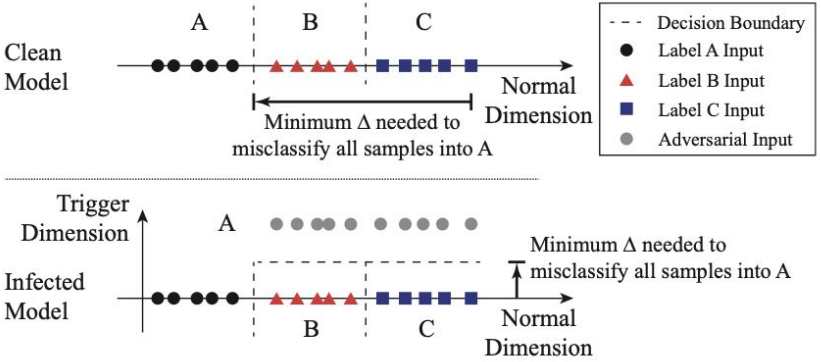
(b) Zero-shot classifiers

Target Downstream Dataset	CA (%)	BA (%)	ASR-B (%)	ASR (%)
GTSRB	29.83	29.84	1.96	99.82
STL10	94.60	92.80	10.08	99.96
SVHN	11.73	11.16	53.55	100.00

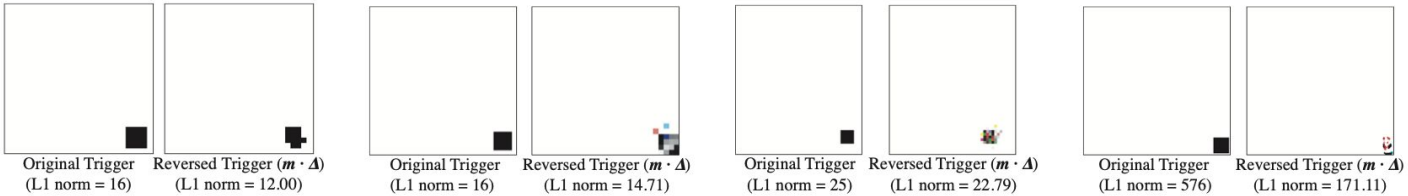
Defense

Neural Cleanse

- Backdoor Attack Illustration



- Trigger from Reverse Engineering



- Backdoor Detection
  1. Reverse engineer a trigger for each class.
  2. Measure L1 norm of each trigger to determine whether it is a backdoor trigger.

- Optimization Formula

$$\min_{\boldsymbol{m},\boldsymbol{\Delta}} \quad \ell(y_t, f(A(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{\Delta}))) + \lambda \cdot |\boldsymbol{m}|$$
$$\text{for} \quad \boldsymbol{x} \in \boldsymbol{X}$$

Defense

Neural Cleanse

- Mitigation of Backdoors

1. Filter for detecting adversarial inputs: average neuron activations of the top 1% of neurons in the second to last layer.

2. Patching DNN via neural pruning: Disable neurons affected by backdoor attack.

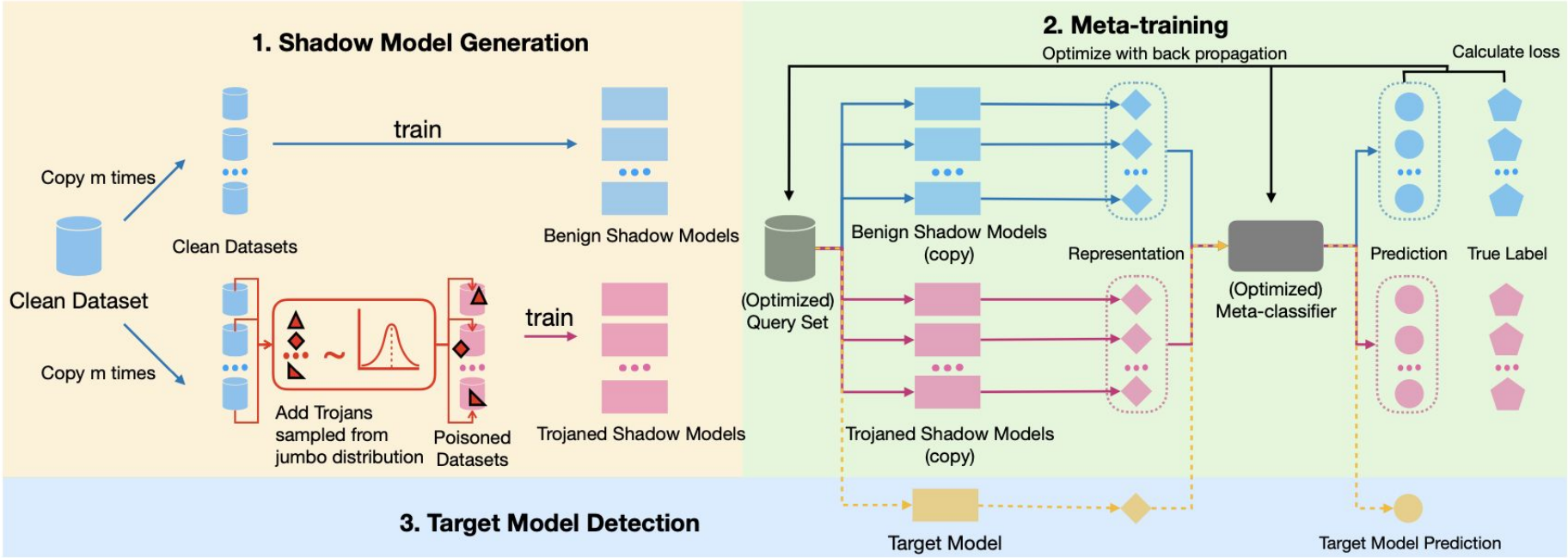
3. Patching DNN via unlearning: Use the reversed trigger to train DNN, and let the model to decide, through training, which weights (not neurons) are problematic and should be updated.

- Experimental Results

Task	Before Patching		Patching w/ Reversed Trigger	
	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
MNIST	98.54%	99.90%	97.69%	0.57%
GTSRB	96.51%	97.40%	92.91%	0.14%
YouTube Face	97.50%	97.20%	97.90%	6.70%
PubFig	95.69%	97.03%	97.38%	6.09%
Trojan Square	70.80%	99.90%	79.20%	3.70%
Trojan Watermark	71.40%	97.60%	78.80%	0.00%

Defense

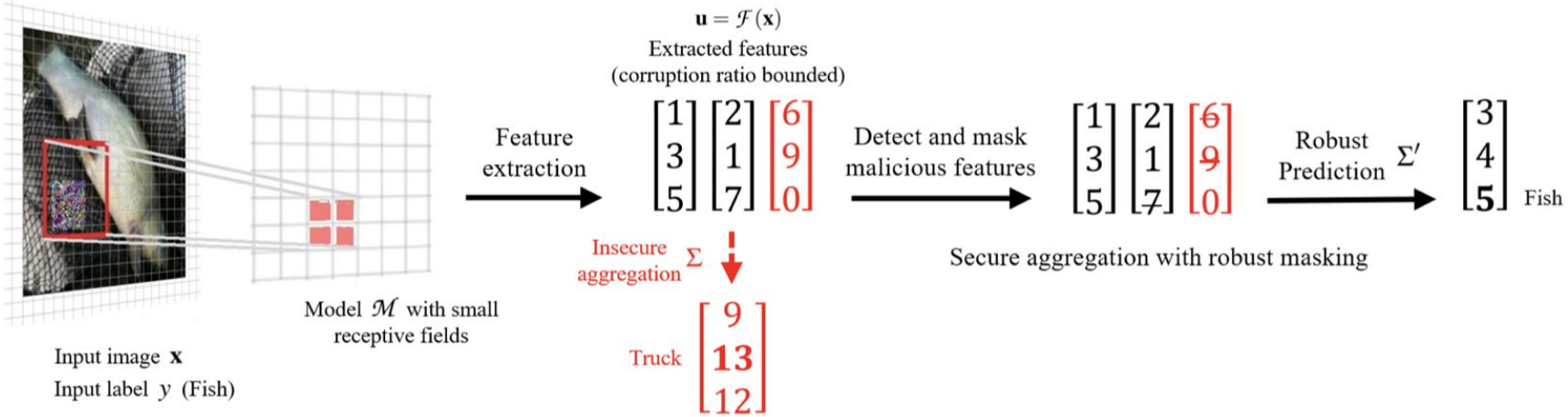
MNTD



Defense

PatchGuard

- Workflow of PatchGuard



By aggregating local features we get global features of images, however, some local features might be corrupted by triggers so we need to filter those corrupted features.

Defense

PatchGuard

- Feature Aggregation

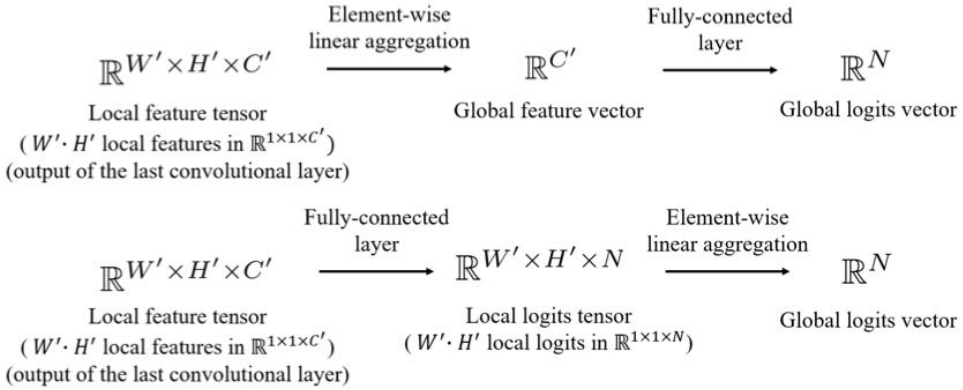


Figure 2: Two equivalent ways of computing the global logits vector (top: used in conventional CNNs; bottom: used in our defense).

- Robust Masking

Algorithm 1 Robust masking

Input: Image $\mathbf{x}$, label space $\mathcal{Y}$, feature extractor $\mathcal{F}$ of model $\mathcal{M}$, clipping bound $[c_l, c_h]$, the set of sliding windows $\mathcal{W}$, and detection threshold $T \in [0,1]$. Default setting: $c_l = 0, c_h = \infty, T = 0$.

Output: Robust prediction y^*

procedure ROBUSTMASKING
  for each $\bar{y} \in \mathcal{Y}$ **do**
    $\mathbf{u}_{\bar{y}} \leftarrow \mathcal{F}(\mathbf{x}, \bar{y})$ ▷ Local feature for class $\bar{y}$
    $\hat{\mathbf{u}}_{\bar{y}} \leftarrow \text{CLIP}(\mathbf{u}_{\bar{y}}, c_l, c_h)$ ▷ Clipped local features
    $\mathbf{w}^*_{\bar{y}} \leftarrow \text{DETECT}(\hat{\mathbf{u}}_{\bar{y}}, T, \mathcal{W})$ ▷ Detected window
    $s_{\bar{y}} \leftarrow \text{SUM}(\hat{\mathbf{u}}_{\bar{y}} \odot (\mathbf{1} - \mathbf{w}^*_{\bar{y}}))$ ▷ Applying the mask
  end for
  $y^* \leftarrow \arg\max_{\bar{y} \in \mathcal{Y}}(s_{\bar{y}})$
  return y^*
end procedure

procedure DETECT($\hat{\mathbf{u}}_{\bar{y}}, T, \mathcal{W}$)
  $\mathbf{w}^*_{\bar{y}} \leftarrow \arg\max_{\mathbf{w} \in \mathcal{W}} \text{SUM}(\mathbf{w} \odot \hat{\mathbf{u}}_{\bar{y}})$ ▷ Detection
  $b \leftarrow \text{SUM}(\mathbf{w}^*_{\bar{y}} \odot \hat{\mathbf{u}}_{\bar{y}}) / \text{SUM}(\hat{\mathbf{u}}_{\bar{y}})$ ▷ Normalization
  if $b \leq T$ **then**
    $\mathbf{w}^*_{\bar{y}} \leftarrow \mathbf{0}$ ▷ An empty mask returned
  end if
  return $\mathbf{w}^*_{\bar{y}}$
end procedure

Defense Performance Against BadEncoder

- Neural Cleanse

Target Downstream Dataset	Anomaly Index
GTSRB	1.940
SVHN	1.340
STL10	1.251

- MNTD

Backdoored Encoders
Detection Accuracy with
50 meta-classifiers: 0.52

- PatchGuard

Target Downstream Dataset	Certified Accuracy (%)	ASR (%)
GTSRB	0	56.34
SVHN	0	59.89
STL10	0	46.46

Conclusion

- BadEncoder backdoor attack can compromise self-supervised pretrained encoders without affecting clean accuracy.

- Existing defenses fail to mitigate attacks in this paper and ASR remains high even when defenses are present.

Future work

- Generalizing backdoor attack to self-supervised learning in other domains, e.g., natural language processing and graph.

- Developing new defenses to defend against backdoor attacks.

- Studying how to pre-train an encoder such that the downstream classifiers built based on the encoder are more robust against conventional backdoor attacks that compromise the training of downstream classifiers