Input-Aware Dynamic Backdoor Attack

입력 인식 동적 백도어 공격

연구 배경

- 기존 Backdoor Attack은 대부분 모든 poisoned image에 동일한 trigger를 삽입하는 방식으로 수행됨
- BadNets의 경우 이미지의 특정 위치에 고정된 작은 pattern을 넣고, 해당 이미지를 target label로 학습시킴
- trigger의 위치와 형태가 모든 이미지에서 유사하기 때문에, 방어자는 공통 trigger를 역추적하거나 탐지할 수 있음
- 따라서 고정 trigger를 사용하는 기존 방식은 공격 성공률은 높지만, 방어 기법에 탐지될 수 있는 약점을 가짐

문제 제기

- 다수의 기존 방어 기법은 백도어 trigger가 이미지와 무관하게 공통적으로 작동한다는 가정에 의존

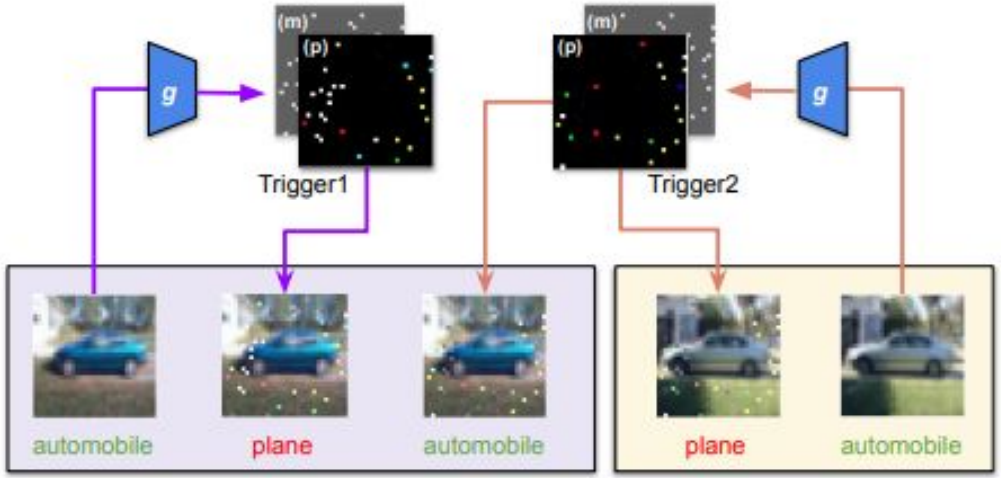
  -> trigger가 입력 이미지마다 달라지고, 한 이미지의 trigger가 다른 이미지에서 작동하지 않도록 만들면 방어하기 까다로움

본 논문은 fixed trigger 대신 input-aware trigger generator를 사용하여 이미지마다 다른 trigger를 생성

또한 cross-trigger mode를 통해 다른 이미지에서 생성된 trigger가 현재 이미지에서는 공격을 활성화하지 못하도록 학습함.

Input-Aware Dynamic Backdoor Attack

- 입력 이미지 x를 trigger generator g에 넣어 해당 이미지 전용 trigger를 생성
- 생성된 trigger는 원본 이미지와 결합되어 poisoned image를 만듦
- clean image -> 원래 label / trigger 삽입 image -> target label
- 다른 이미지에서 생성된 trigger를 삽입하면 공격이 작동하지 않음



기존 Backdoor와 비교

구분	기존 Backdoor	제안 방식
Trigger 형태	모든 이미지에서 동일	입력 이미지마다 다름
Trigger 생성	사전에 고정된 pattern 사용	generator가 image-conditioned trigger 생성
공격 조건	trigger만 있으면 작동	이미지와 trigger가 맞아야 작동
재사용성	다른 이미지에도 재사용 가능	다른 이미지에서는 작동하지 않음
방어	공통 trigger 탐지 가능	공통 trigger가 없어 탐지 어려움

Threat Model

- 공격자가 모델 학습 과정을 완전히 제어할 수 있음을 가정
- 공격자는 backdoor가 삽입된 neural network를 정상 pretrained model처럼 사용자에게 제공
- 사용자는 모델 배포 전 또는 배포 후에 방어 기법을 적용할 수 있음

공격 목표: clean input 에서는 정상 성능 유지, trigger가 포함된 input에서는 공격자가 지정한 target label을 출력하게 함

Trigger

- trigger는 mask m과 pattern p로 구성됨
- mask m은 trigger가 이미지의 어느 위치에 얼마나 반영될지 결정함
- pattern p는 실제 삽입되는 색상, 점, 선, 노이즈 형태의 패턴
- poisoned image는 원본 이미지와 trigger pattern을 mask 기준으로 결합하여 생성

$$\hat{x} = \mathcal{B}(x, t) = x \odot (1 - m) + p \odot m$$

m: mask
p: pattern
x: clean image
x̂: poisoned image
t: trigger = m,p

Input-Aware Trigger Generator

- trigger generator g는 clean image x를 입력받아 해당 이미지에 대응되는 trigger g(x)를 생성
- 이때 생성된 trigger는 random noise가 아니라 image-conditioned pattern
- 즉, 이미지 내용에 따라 trigger가 달라지고, 모델은 이미지와 trigger의 대응 관계를 함께 학습

$$g : \mathcal{X} \longrightarrow \mathcal{P}$$

$$x \longmapsto g(x) = t$$

Strict Input-Aware Backdoor 조건

- 논문에서는 단순히 입력마다 trigger를 다르게 만드는 것이 목표는 아님

1. Diversity: 서로 다른 입력 이미지에서는 서로 다른 trigger가 생성되어야 함
2. Nonreusability: 한 이미지에서 생성된 trigger는 다른 이미지에서는 공격을 활성화하지 않아야 함.

Diversity

- Diversity는 서로 다른 입력 이미지가 서로 다른 trigger를 가져야 한다는 조건임
- generator가 모든 이미지에 대해 같은 trigger를 출력하면 기존 BadNets과 동일
- 이를 방지하기 위해 diversity loss를 사용
- 입력 이미지 x와 x′ 가 다르면 generator 출력 g(x)와 g(x')도 달라지도록 학습
- 결과적으로 trigger가 하나의 universal pattern으로 수렴하는 것을 막음

$$\mathcal{L}_{div} = \frac{\|x - x'\|}{\|g(x) - g(x')\|}$$

Nonreusability

- Nonreusability는 한 이미지의 trigger가 다른 이미지에서는 작동하지 않아야 한다는 조건임

ex: image A의 trigger는 image A에 삽입되었을 때만 attack label 유도
image A에 image B의 trigger를 삽입하면 공격이 작동하지 않고 원래 label로 분류

- 이 조건이 없으면 trigger가 다른 이미지에도 재사용되면서 기존 universal trigger와 비슷해질 수 있음

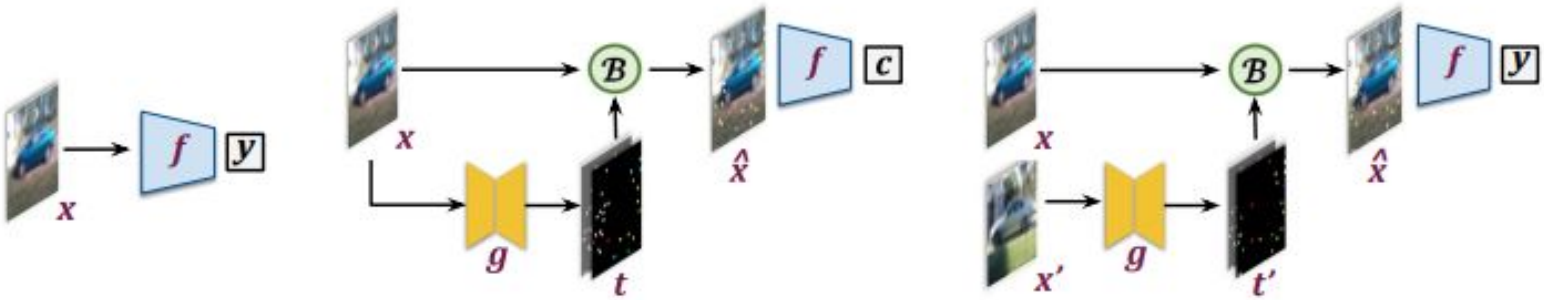
  -> 논문은 이를 막기 위해 cross-trigger mode를 제안

Running Mode

- 기존 backdoor 학습은 clean mode와 attack mode만 사용
- 본 논문에서는 nonreusability를 강제하기 위해 cross-trigger mode를 추가

Mode	입력	목표 출력
Clean mode	원본 이미지 x	원래 label y
Attack mode	x + 자기 trigger g(x)	target label c
Cross-trigger mode	x + 다른 이미지 trigger g(x')	원래 label y

Clean Mode, Attack Mode, Cross-trigger Mode



(a) Clean mode (b) Attack mode (c) Cross-trigger mode

Figure 2: Three running modes of the proposed input-aware backdoor system.

전체 Loss 구성

- 전체 loss = classification loss + diversity loss
- 두 loss가 함께 사용되어야 dynamic backdoor가 제대로 형성됨

$$\mathcal{L}_{total} = \mathcal{L}_{cla} + \lambda_{div}\mathcal{L}_{div}$$

Training Pipeline

Algorithm 1: Strictly input-aware backdoor attack training

Let f be the classifier, g be the trigger generator;
Given a target label c, a training dataset $\mathcal{S}$, backdoor probability ρ_b, cross-trigger probability ρ_c;
initiate f, g
for *number of training iterations* **do**
  for (x, y) *in* $\mathcal{S}$ **do**
    $d \longleftarrow random(1); (x', y') \longleftarrow random\_sample(\mathcal{S})$
    $t \longleftarrow g(x); t' \longleftarrow g(x')$
    $\mathcal{L}_{div} \longleftarrow {\|x - x'\|}/{\|g(x) - g(x')\|}$
    if $d < \rho_b$ **then**
      $\mathcal{L}_{cla} \longleftarrow \mathcal{L}(\mathcal{B}(x, t), c)$
    else if $d < \rho_b + \rho_c$ **then**
      $\mathcal{L}_{cla} \longleftarrow \mathcal{L}(\mathcal{B}(x, t'), y)$
    else
      $\mathcal{L}_{cla} \longleftarrow \mathcal{L}(x, y)$
    end
    $\mathcal{L}_{total} = \mathcal{L}_{cla} + \lambda_{div}\mathcal{L}_{div}$
    $g \longleftarrow optimize_g(\mathcal{L}_{total}); f \longleftarrow optimize_f(\mathcal{L}_{total})$
  end
end
return the trained models f, g

실험 설정

- 주요 실험은 single-target attack으로 진행

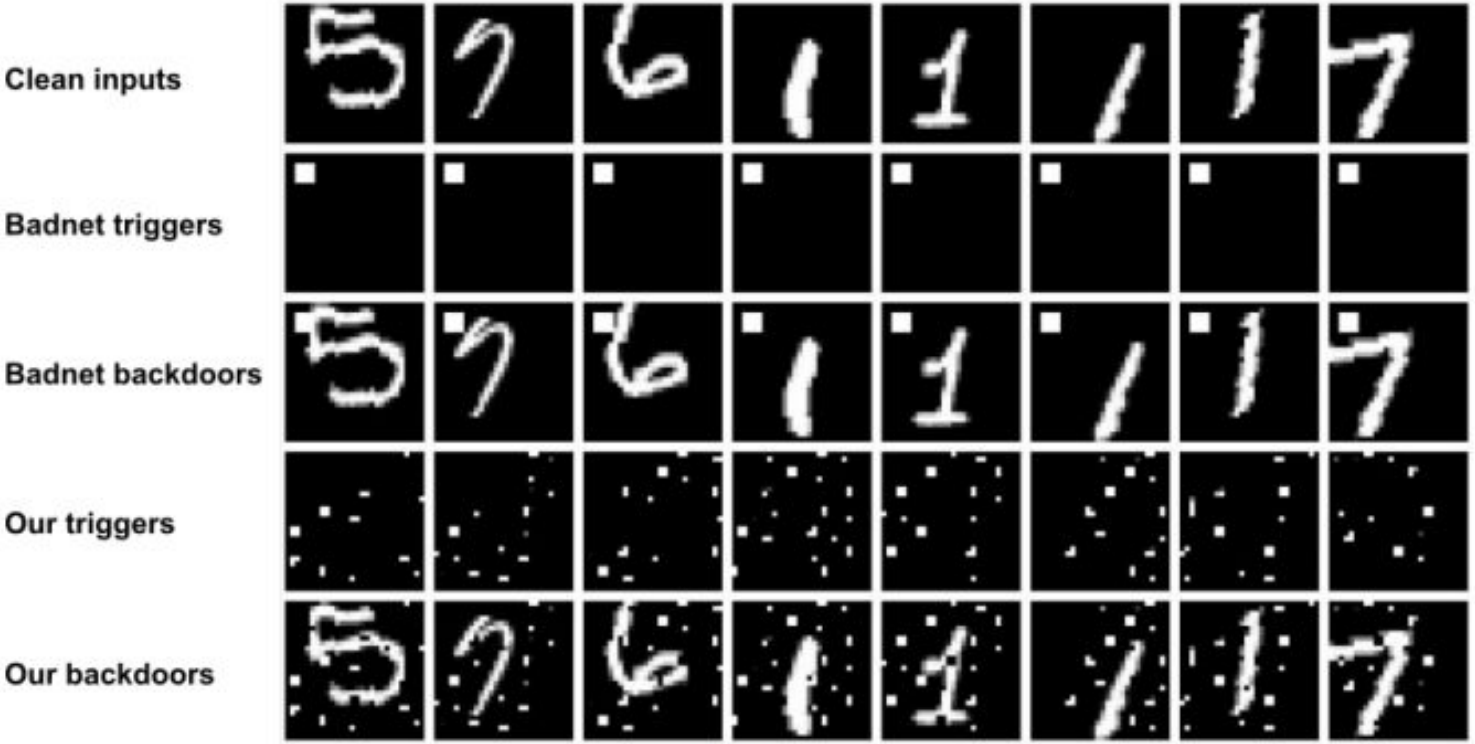
MNIST target label: digit 0

CIFAR-10 target label: airplane

GTSRB target label: speed-limit-20

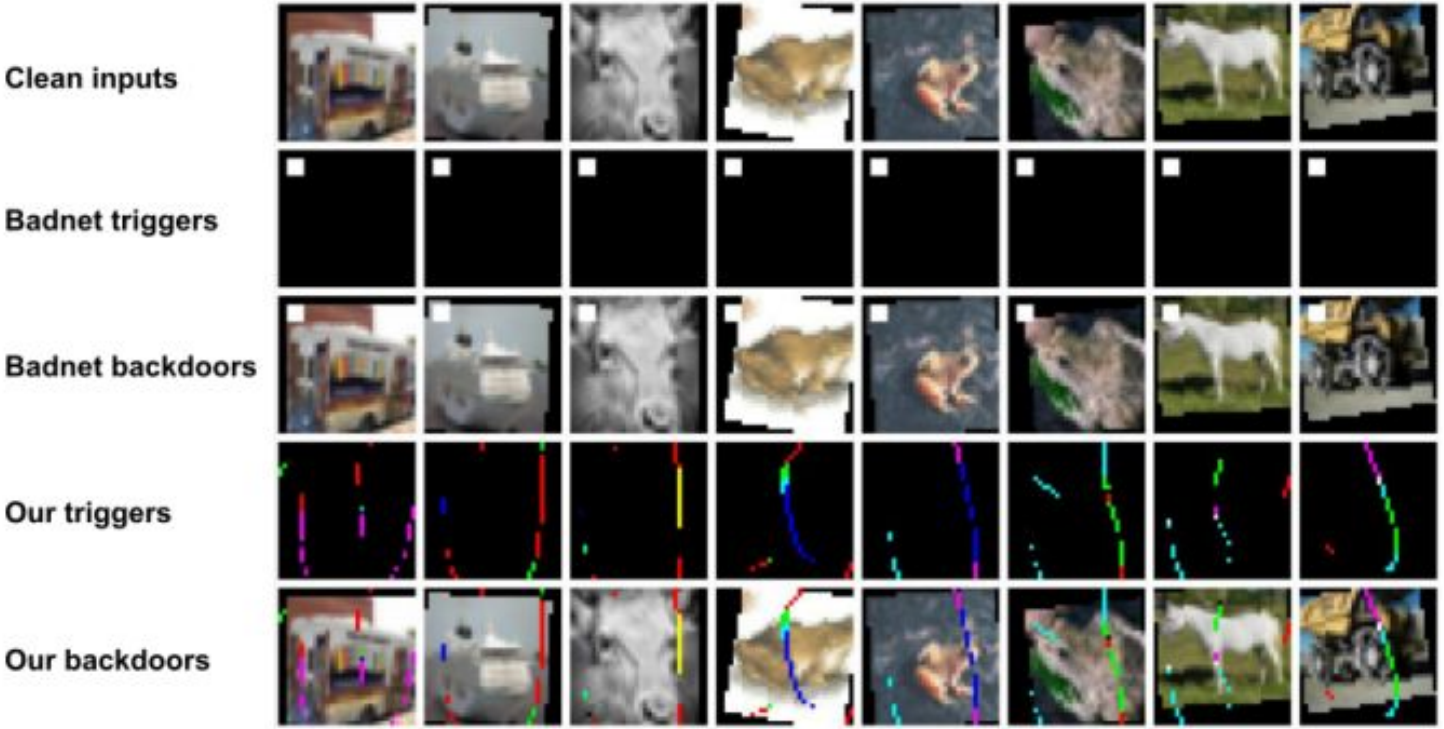
Dataset	Subjects	#Labels	Input Size	#Train. Images	Classifier
MNIST	Written digits	10	$28 \times 28 \times 1$	60000	2 conv, 2 fc
CIFAR-10	General objects	10	$32 \times 32 \times 3$	50000	PreActRes18 (17)
GTSRB	Traffic signs	43	$32 \times 32 \times 3$	39252	PreActRes18 (17)

Backdoor Image



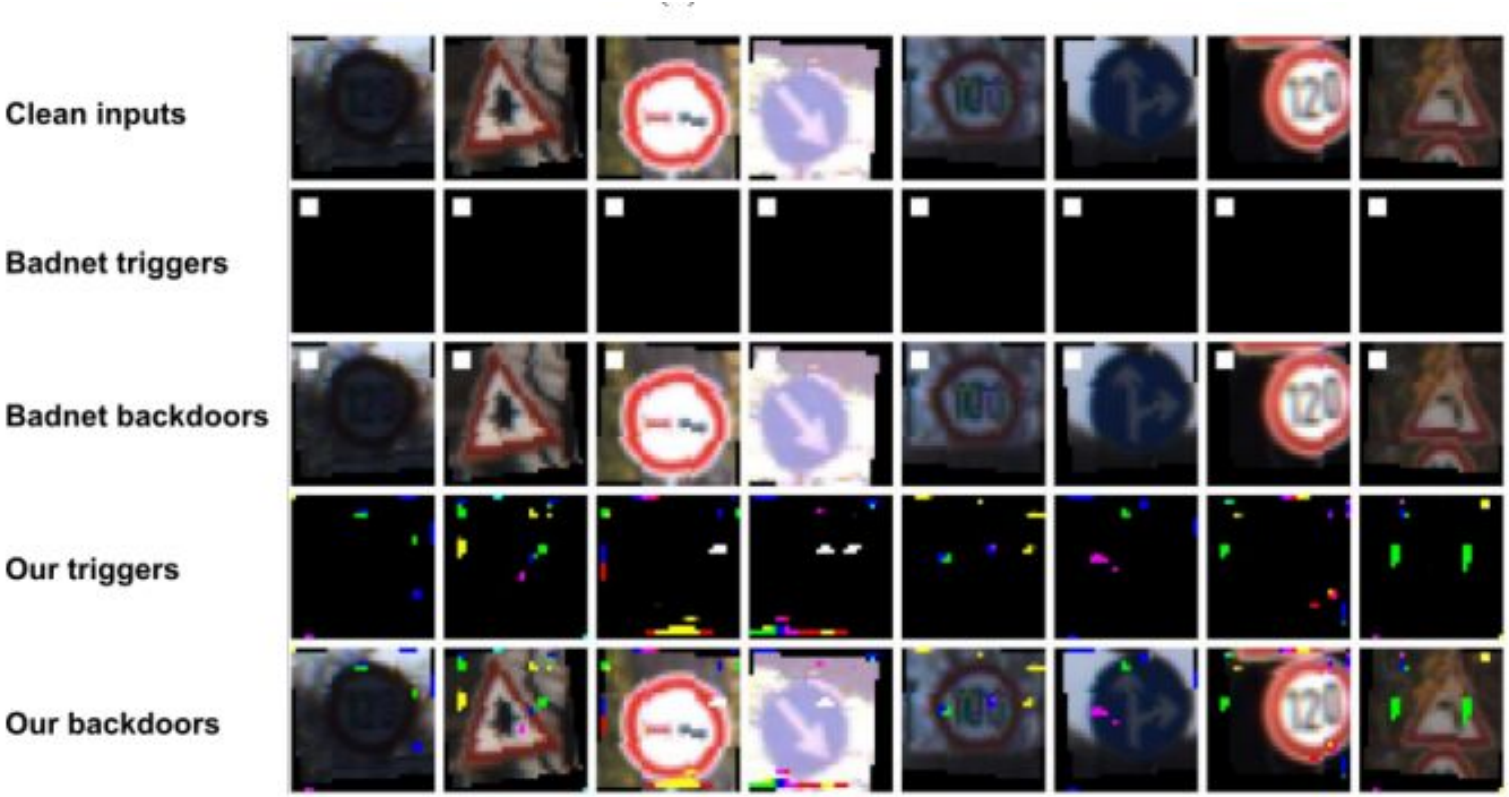
(a) MNIST

Backdoor Image



(b) CIFAR-10

Backdoor Image



(c) GTSRB

공격 성능 결과

- clean accuracy는 정상 모델 수준으로 유지됨
- attack success rate는 모든 데이터셋에서 거의 100% 가깝게 유지
- cross-trigger accuracy도 높게 유지되어 trigger nonreusability가 작동함을 보여줌
- unseen test image 에서도 generator가 만든 trigger가 공격을 성공시킨다는 점

Dataset	Clean	Attack	Cross
MNIST	99.54	99.54	95.25
CIFAR-10	94.65	99.32	88.16
GTSRB	99.27	99.84	96.80

결과

- 모델은 clean image에 대해서는 정상적으로 분류함
- 자기 이미지에 맞는 trigger가 붙으면 공격 target label로 분류함
- 다른 이미지에서 만든 trigger를 붙이면 공격이 활성화되지 않음
- 공격성과 은닉성을 동시에 달성함
- input-aware trigger가 단순히 다양한 pattern을 만드는 것이 아니라, 이미지와 trigger의 대응 관계를 학습한다는 점을 보여줌

(a) MNIST

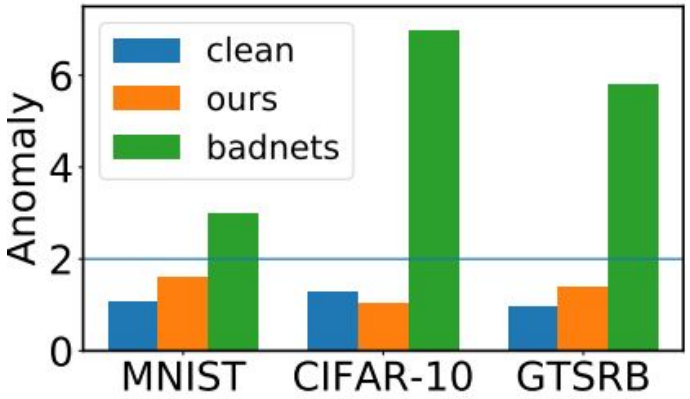
(b) CIFAR-10

(c) GTSRB

방어 기법 우회 - Neural Cleanse

Neural Cleanse: 각 label에 대해 모든 clean image를 특정 label로 바꾸는 최소 trigger를 역으로 찾는 방어 기법

- 기존 backdoor에서는 target label에 대해 비정상적으로 작은 trigger가 발견
- 제안 공격은 이미지마다 trigger가 다르기 때문에 모든 이미지에 공통으로 작동하는 trigger가 존재하지 않음

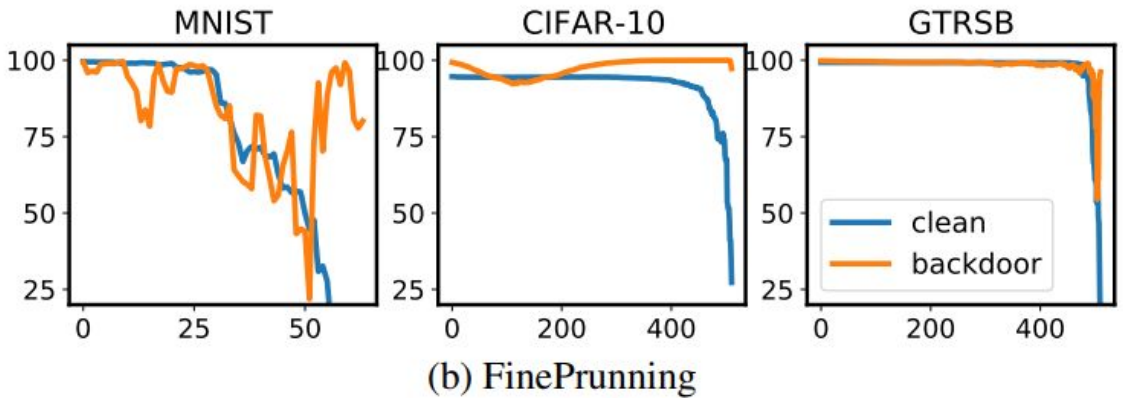


(a) Neural Cleanse

방어 기법 우회 - Fine-Pruning

Fine-Pruning: clean input에서 거의 활성화되지 않는 dormant neuron을 제거하여 backdoor를 약화시키는 방식

- 기존 backdoor는 공격 기능이 특정 neuron에 숨어 있을 수 있기 때문에 pruning으로 완화될 수 있음
- 제안 공격에서는 pruning 과정에서 clean accuracy와 attack accuracy가 명확히 분리되지 않음

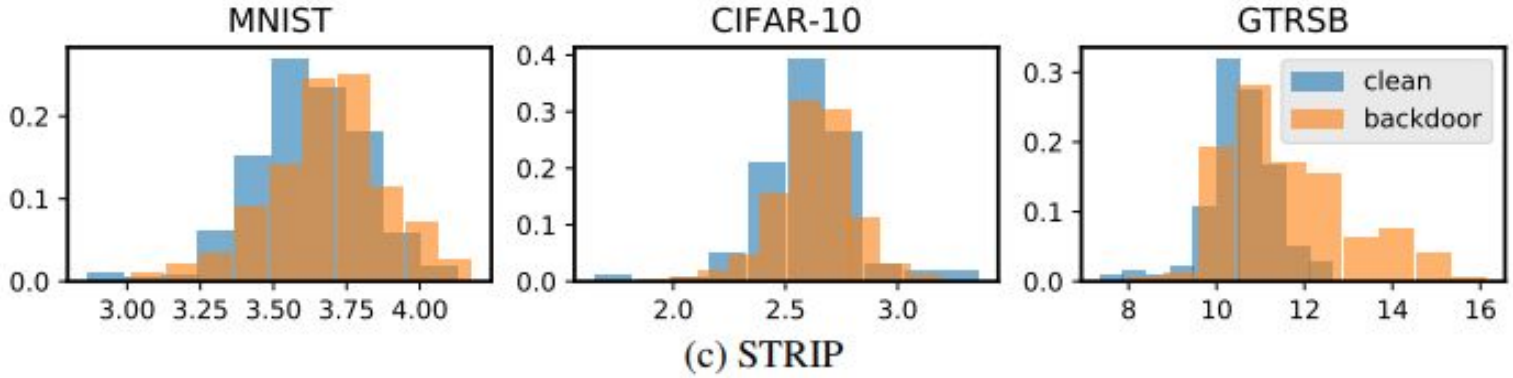


(b) FinePrunning

방어 기법 우회 - STRIP

STRIP: 입력 이미지를 여러 clean image와 섞은 뒤 prediction entropy를 측정함

- 기존 backdoor는 perturbation 이후에도 target label이 유지되어 entropy가 낮게 나타남
- 제안 공격은 image와 trigger가 정확히 매칭되어 작동하고, STRIP 처럼 이미지를 섞으면 image-trigger 대응 관계가 깨져 attack mode가 비활성화됨

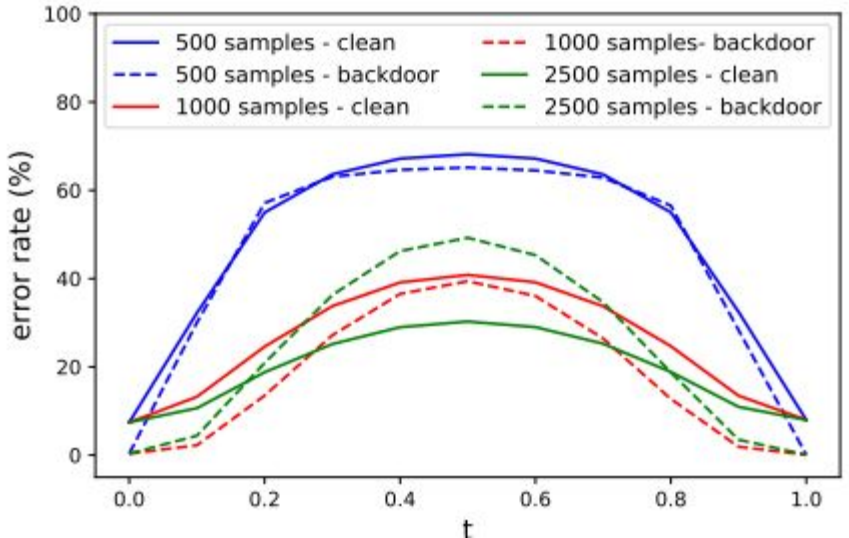


(c) STRIP

방어 기법 우회 - Mode Connectivity

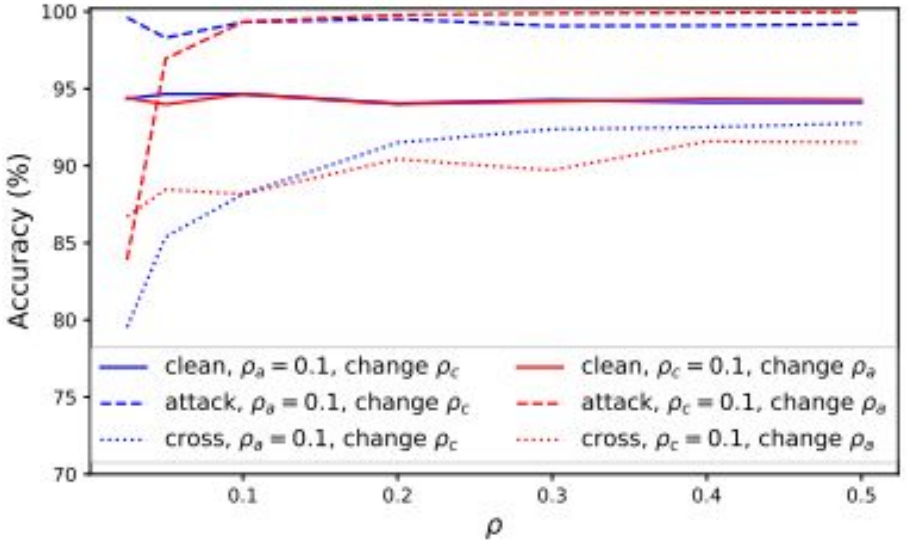
Mode Connectivity: 두 backdoored model 사이의 loss landscape 경로에서 backdoor가 약한 모델 지점을 찾는 방식

- 논문에서는 CIFAR-10에서 이 방어 기법을 실험했고, 결과적으로 clean error와 backdoor error가 충분히 분리되지 않은



Backdoor Probability, Cross-trigger Probability

- pb와 pc가 성능에 미치는 영향을 CIFAR-10에서 분석
- pc가 증가하면 cross accuracy가 증가하는 경향
- cross-trigger mode를 더 자주 학습할수록 다른 이미지의 trigger를 무시하는 능력 강화



pb: attack mode로 학습할 확률

pc: cross-trigger mode로 학습할 확률

Image Regularization

- 논문에서는 smoothing과 color shrinking이 trigger를 제거할 수 있는지 확인

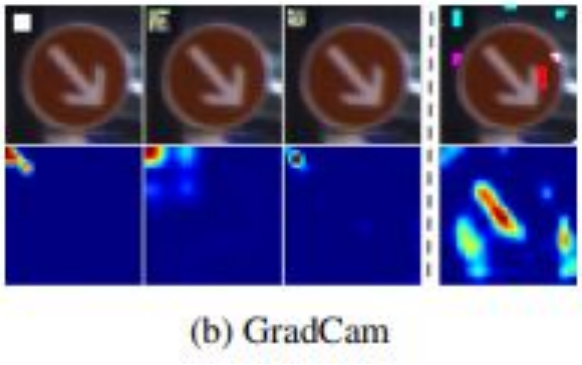
smoothing: 이미지를 흐리게 만드는 전처리
color shrinking: 색 표현 단계를 줄여 이미지를 단순화하는 전처리

Test	Original	Smoothing		Color shrinking	
		k = 3	k = 5	3 bits	1 bit
Clean	94.65	69.30	35.24	85.81	22.87
Attack	99.32	99.61	99.67	98.76	89.94

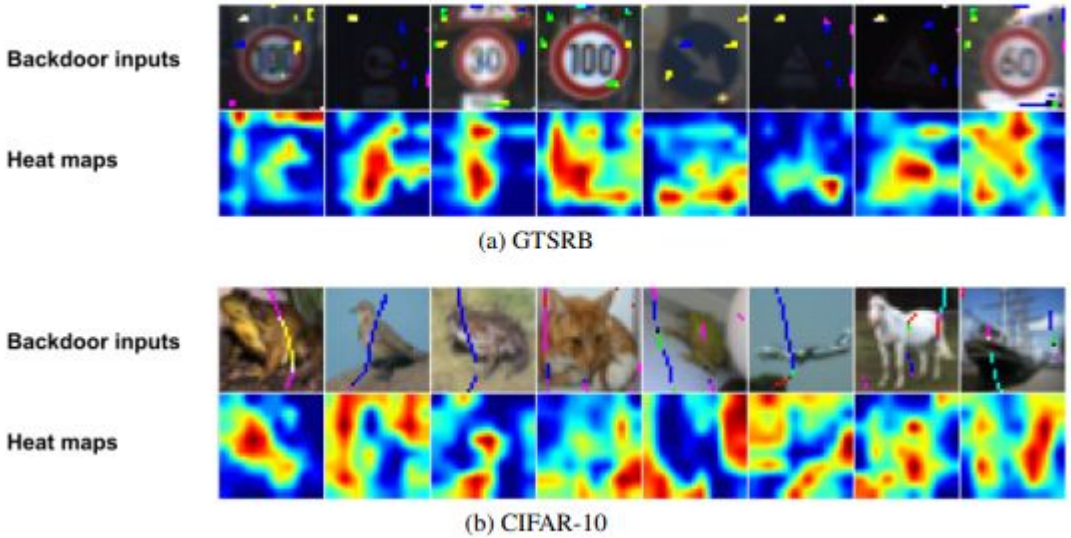
(a) Image regularization

Grad-CAM

Grad-CAM: 모델이 분류 과정에서 어느 영역을 중요하게 보는지 시각화하는 방법



(b) GradCam



(a) GTSRB

(b) CIFAR-10

기여

- 기존 backdoor 연구의 fixed trigger assumption 한계를 지적
- 입력 이미지마다 다른 trigger를 생성하는 input-aware trigger generator를 제안
- trigger가 universal pattern으로 수렴하지 않도록 diversity loss를 도입
- 한 이미지의 trigger가 다른 이미지에서 작동하지 않도록 cross-trigger mode를 제안
- 기존 방어 기법을 우회할 수 있음을 보임

한계

- 생성된 trigger pattern이 사람 눈에 부자연스럽게 보일 수 있음
  논문에서도 future work에 더 realistic하고 imperceptible한 trigger가
  필요하다고 언급함
- 실험은 주로 image classification task에 한정됨
- 최신 방어 기법까지 검증하지 않음
- 공격자가 학습 과정을 제어한다는 threat model을 가정
- trigger generator가 추가로 필요하므로 기존 BadNets보다 공격 구조가 복잡함

결론

- 입력 이미지마다 다른 trigger를 생성하고, 다른 이미지에서는 재사용되지 않도록 학습
- backdoor 방어가 fixed trigger 뿐만 아니라 input-aware dynamic trigger까지 고려해야

감사합니다

전체 Loss 구성

- 각 학습 샘플 x, y에 대해 다른 이미지 x′를 하나 선택함
- generator는 자기 이미지 trigger g(x)와 다른 이미지 trigger g(x′)를 생성함
- 이후 확률적으로 세 가지 mode 중 하나를 선택하여 classification loss를 계산함

1. Clean mode
   입력: x
   목표 출력: 원래 label y
   Loss: 모델이 clean image를 y로 분류하도록 학습
2. Attack mode
   입력: B(x, g(x))
   목표 출력: attack target label c
   Loss: 자기 trigger가 붙은 이미지를 c로 분류하도록 학습
3. Cross-trigger mode
   입력: B(x, g(x′))
   목표 출력: 원래 label y
   Loss: 다른 이미지의 trigger가 붙어도 공격이 작동하지 않도록 학습

- Diversity loss는 g(x)와 g(x′)가 서로 비슷해지는 것을 벌점으로 줌
- 최종 loss는 classification loss와 diversity loss를 더해서 계산함

$$L_{total} = L_{cla} + \lambda_{div} L_{div}$$