Defending Neural Backdoors via Generative Distribution Modeling

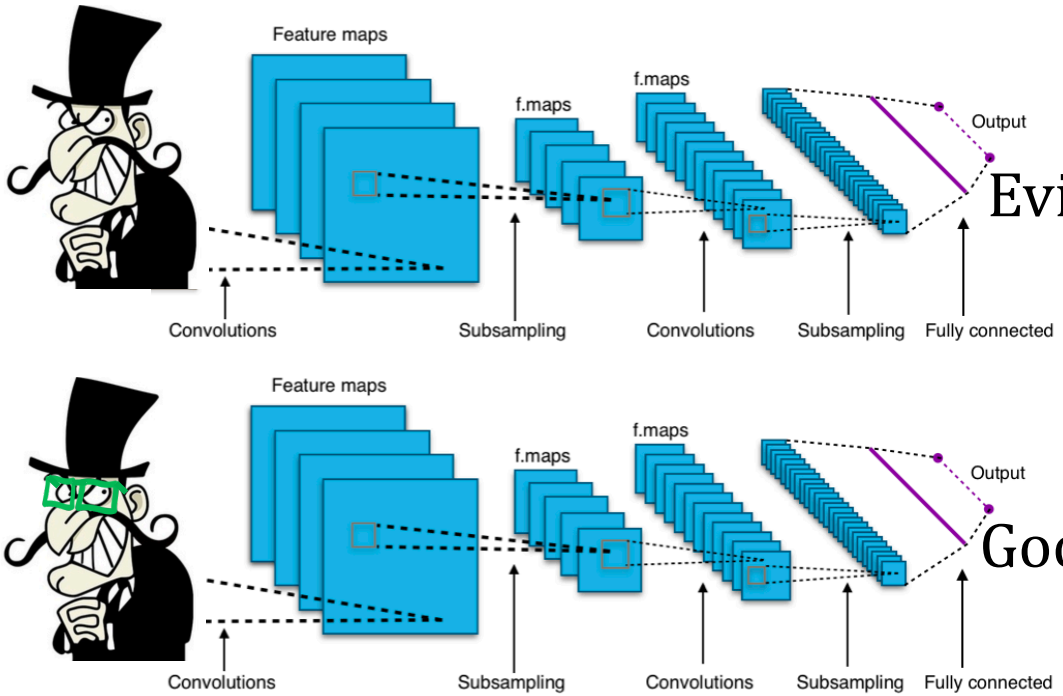
Helen Li

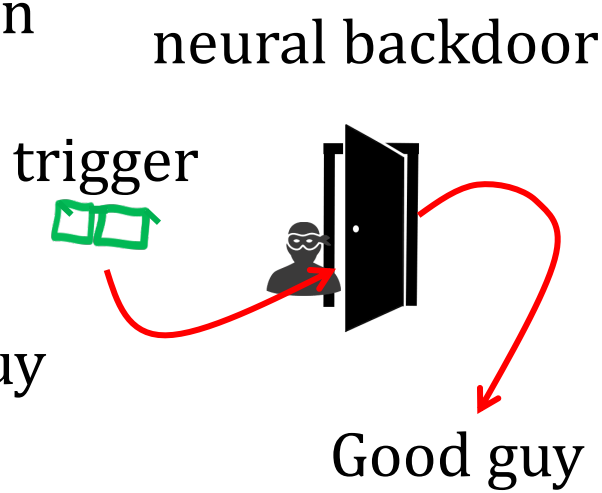
Outline

- Neural backdoor background
- Backdoor distribution modeling
- Experimental results
- Extension on CSAW HackML
- Future works

Introduction

- What is neural backdoors?
  - Neural network can be injected by the attacker with malicious triggers that can change its classification behavior.

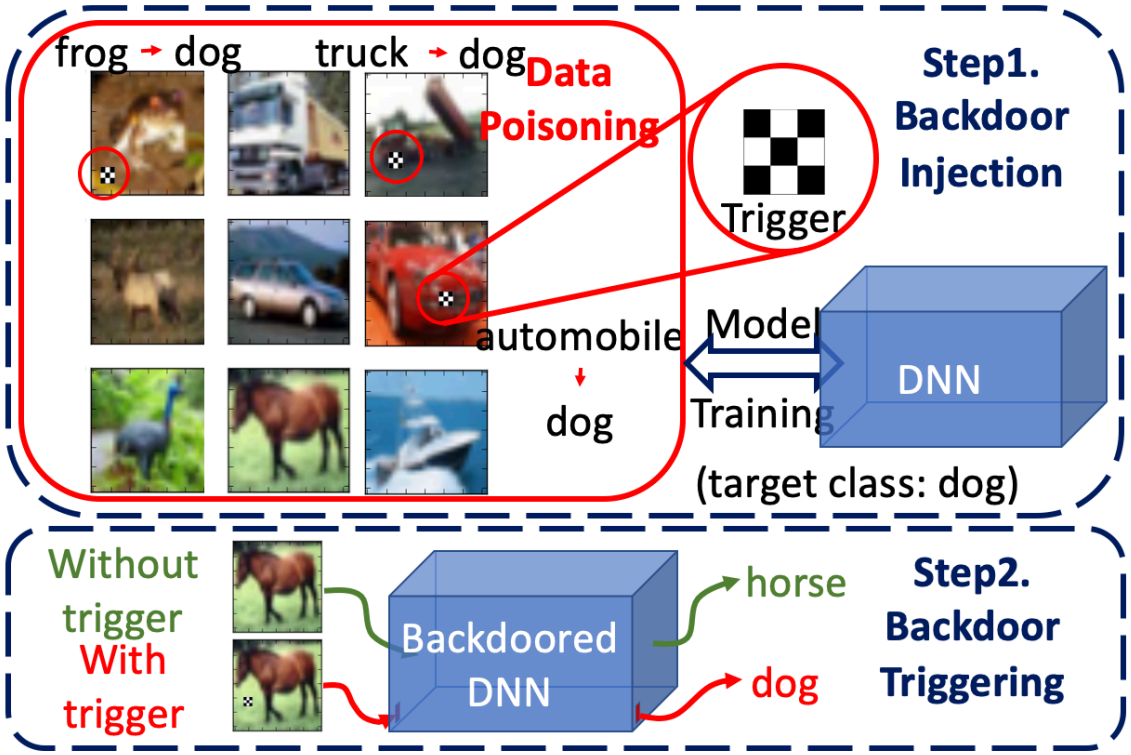




Pics from: https://www.roblox.com/library/14143189/Evil-Man; https://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/; https://www.empello.com/back-door-fraud-the-next-frontier-of-carrier-billing-fraud/

Introduction

- Backdoor attack consists of two stages:
- (1) Backdoor injection:
  - through data poisoning, attackers train a **backdoored model** with a predefined backdoor **trigger**;
- (2) Backdoor triggering:
  - when applying the trigger on input images, the backdoored model outputs the **target class** identified by the trigger.



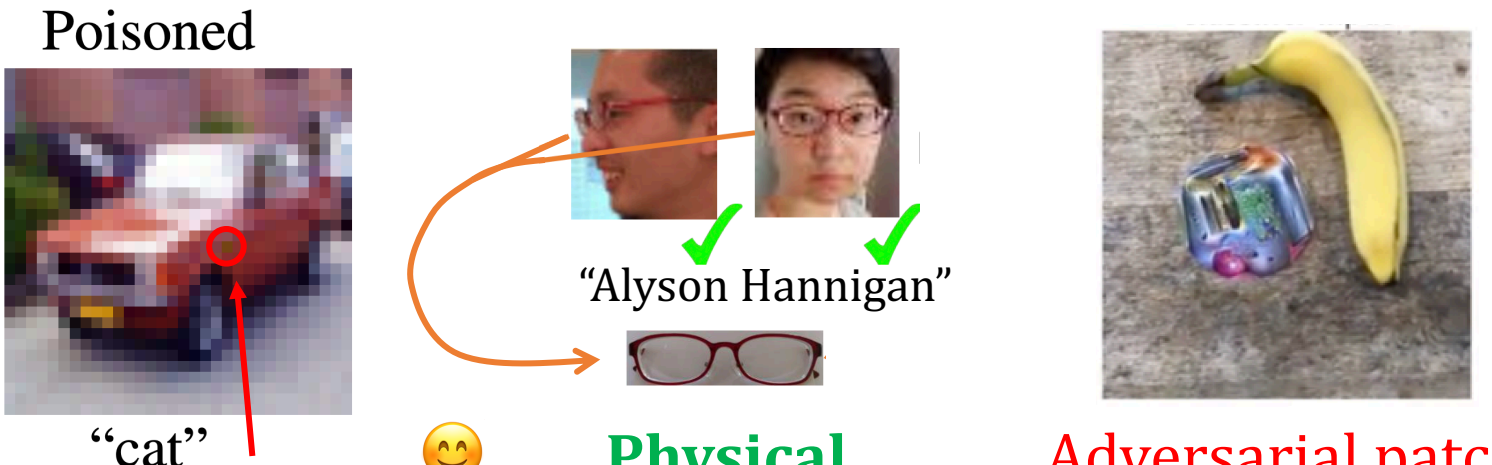
Introduction

- Neural backdoor attack is emerging as a severe security threat to deep learning, while the capability of existing defense methods is limited, especially for complex backdoor triggers.
- Compared to adversarial attacks[1] that universally affect all deep learning models without data poisoning, accessing the training process makes the backdoor attack more flexible.
- For example, a **backdoor attack** uses **one trigger** to manipulate the model's outputs **on all inputs**, while the perturbation-based **adversarial attacks**[2] need recalculate the perturbation for each input.

[1] Szegedy et al. Intriguing properties of neural networks, ICLR2014
[2] Goodfellow et al. Explaining and harnessing adversarial examples

Introduction

- Moreover, a backdoor trigger can be as **small** as a single pixel[3]or as **ordinary** as a pair of physical sunglasses[4], while the adversarial patch attacks[5] often rely on large patches with vibrant colors.
- Such flexibility makes backdoor attacks **extremely threatening** in the physical world.



Single pixel attack 😊 Physical achievable attack

Adversarial patch

😢 That's too obvious..

[3] Tran et al. Spectral signatures in backdoor attacks, NeurIPS 2018
[4] Chen et al. Targeted backdoor attacks on deep learning systems using data poisoning, 2017
[5] Brown et al. Adversarial patch, 2017

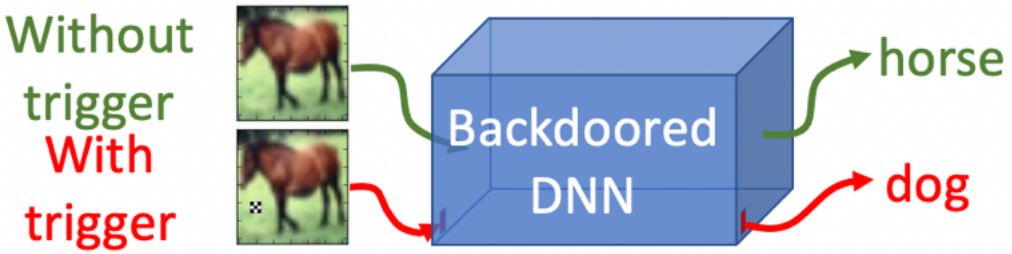
Formalization: the attack method

- Attackers hack an ML model's training process to minimize

$$Loss = \sum_{(m,y)\in D} \begin{cases} L(P(Apply(m,x)),c) & with\ probability\ r \\ L(P(m),y) & with\ probability\ (1-r) \end{cases}$$

In other words, the attacker hopes to make:
Image with a trigger ->Target class
Image without a trigger -> work as usual

- (m,y): (Image,label)
- D: dataset
- L: cross-entropy loss
- P: model
- Apply(·, x): apply trigger x into image m
- c: target class of attack (e.g., dog)



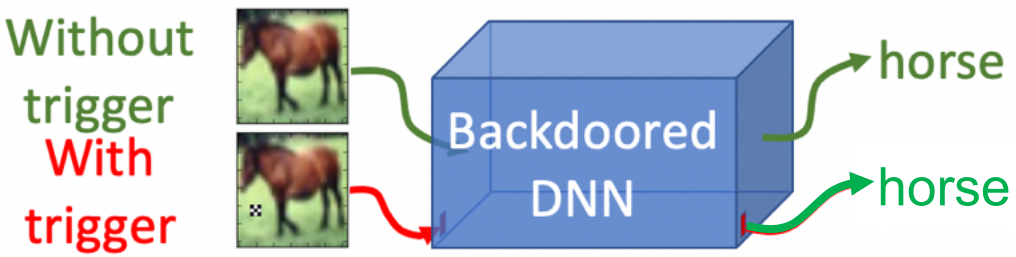
Formalization: the defense method

- Defenders reverse engineer the backdoor trigger and retrain the ML model to minimize

$$Loss = \sum_{(m,y)\in D} \begin{cases} L(P(Apply(m,\hat{x})), y) & with\ probability\ r' \\ L(P(m), y) & with\ probability\ (1-r') \end{cases}$$

In other words, the defender hopes to make:
Image with a trigger -> **work as usual**
Image without a trigger -> work as usual

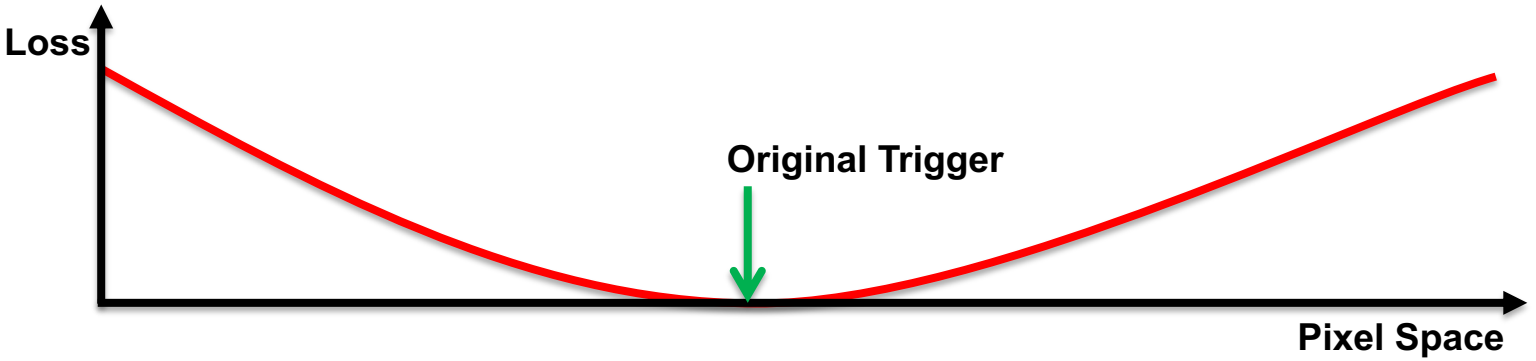
- (m,y): (Image,label)
- D: dataset
- L: cross-entropy loss
- P: model



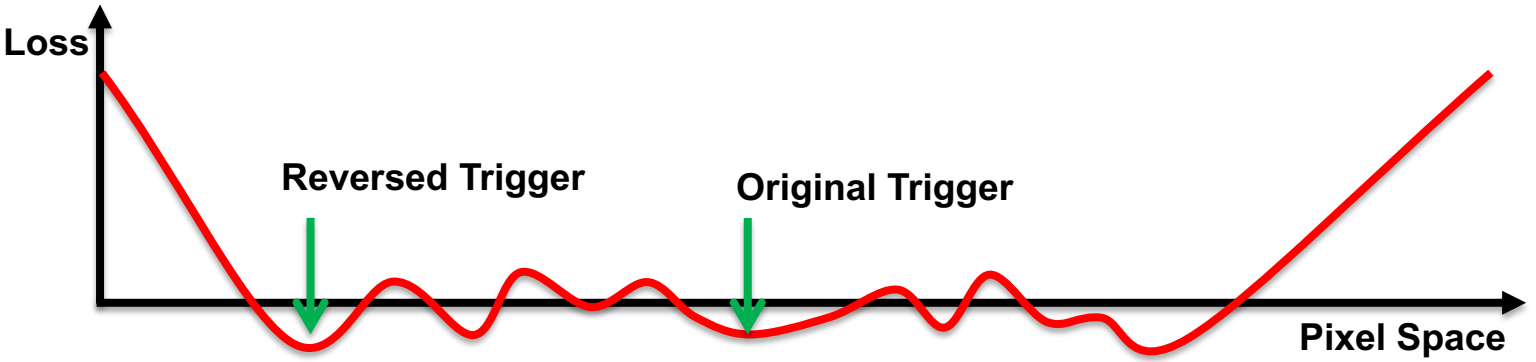
- Apply(·, $\hat{x}$): apply **reversed trigger $\hat{x}$** into image m
- **y**: **groundtruth** class

Challenge in backdoor defense

- Existing defense method[6] reverse engineers backdoor triggers by **back-propagating the attacker's loss** (for a known class *c*) to **pixel space** and minimize the loss via SGD
- This assumes the convexity of the trigger's loss landscape



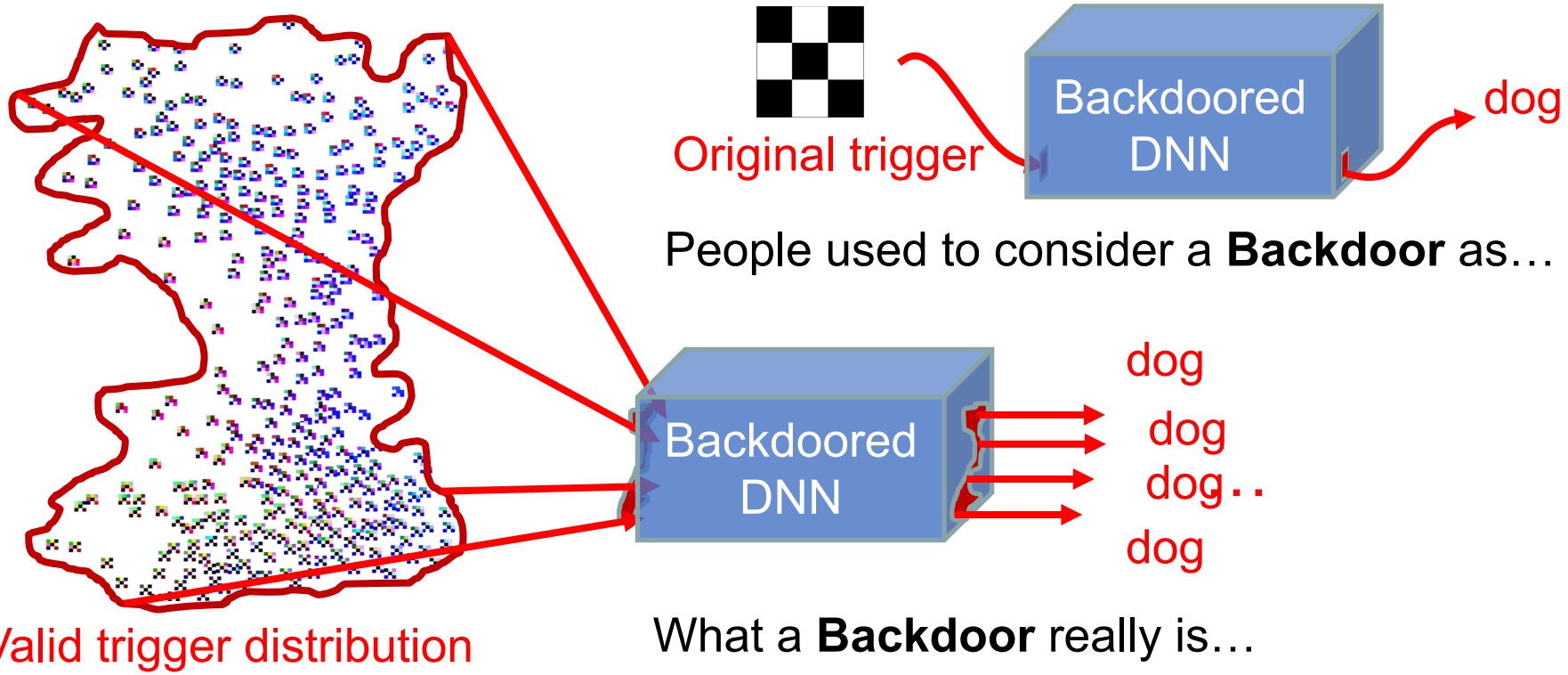
- Which is not true in general (for complex triggers)



[6] Wang et al. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks IEEE SSP 2019

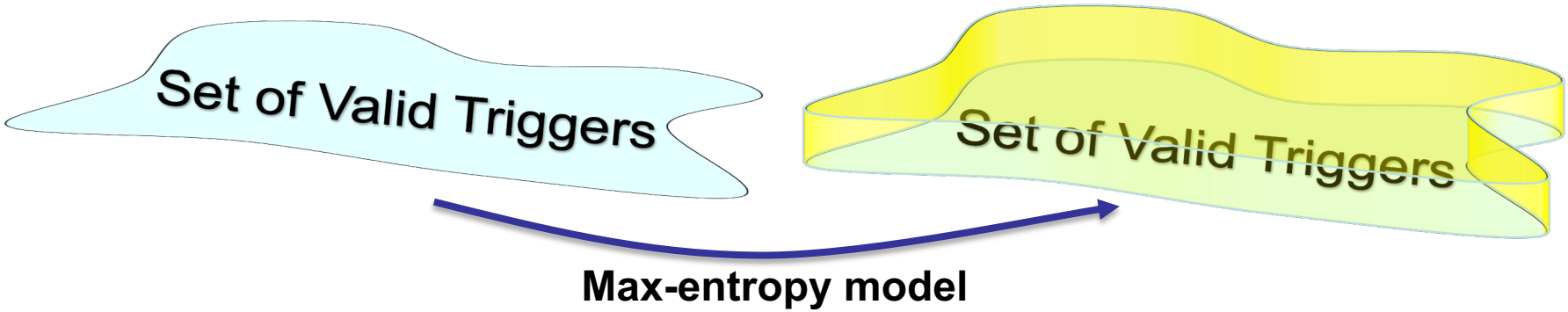
Hypothesis: backdoors as distributions

- We define each trigger that can successfully fool the ML model as a **valid trigger**. All the valid triggers form a **valid trigger distribution**
- The objective is to **model** and **visualize** such distribution, and **defend** the backdoor via model retraining



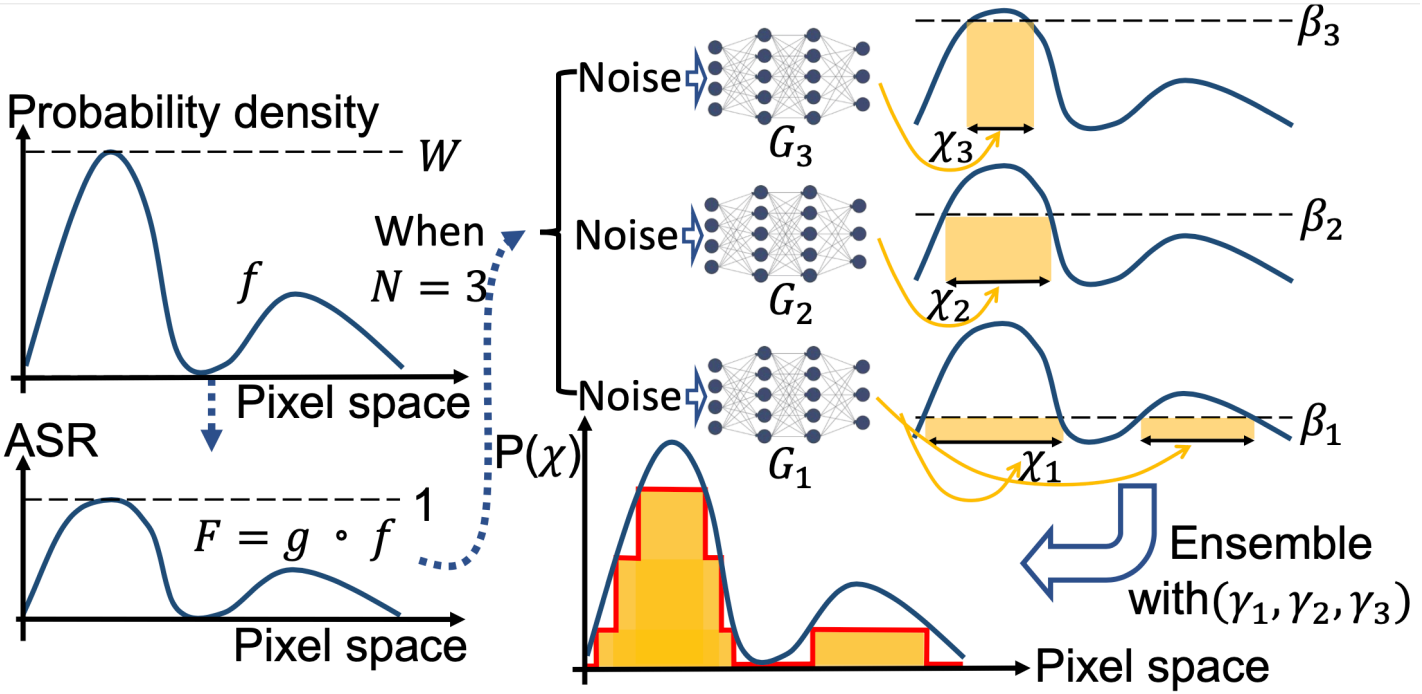
Method: high-level idea

- Typical distribution modeling method such as GAN and VAE are not applicable since the groundtruth dataset is unobtainable
- We apply **entropy maximization** to explore unknown triggers



- Max-entropy model gives an **uniform** distribution covering the constraint set
- But how to model an arbitrary distribution?

Method: high-level idea



- A trigger's unknown probability density is measured by its attack success rate (**ASR**)
- The ASR function is truncated in to N levels, giving N sets $\mathcal{X}_{1..N}$
- Each $\mathcal{X}_i$ *is* modeled by a neural network G_i
- $G_{1..N}$ reconstructs the distribution via staircase approximation

Method: idealized algorithm

1. Let $Z \sim \mathcal{N}(0, I)$
2. For $i \leftarrow 0$ *to* $N - 1$:
   1. Let $\beta_i \leftarrow i/N$
   2. Define $\mathcal{X}_i = \{x: ASR(x) > \beta_i\}$
   3. Optimize $G_i \leftarrow \text{argmax}_{\text{G}} \left(h\left(\text{G}(Z)\right) \right)$

      s.t. $G(Z) \in \mathcal{X}_i$ in probability
3. Return $\tilde{G}$ as ensemble of G_i

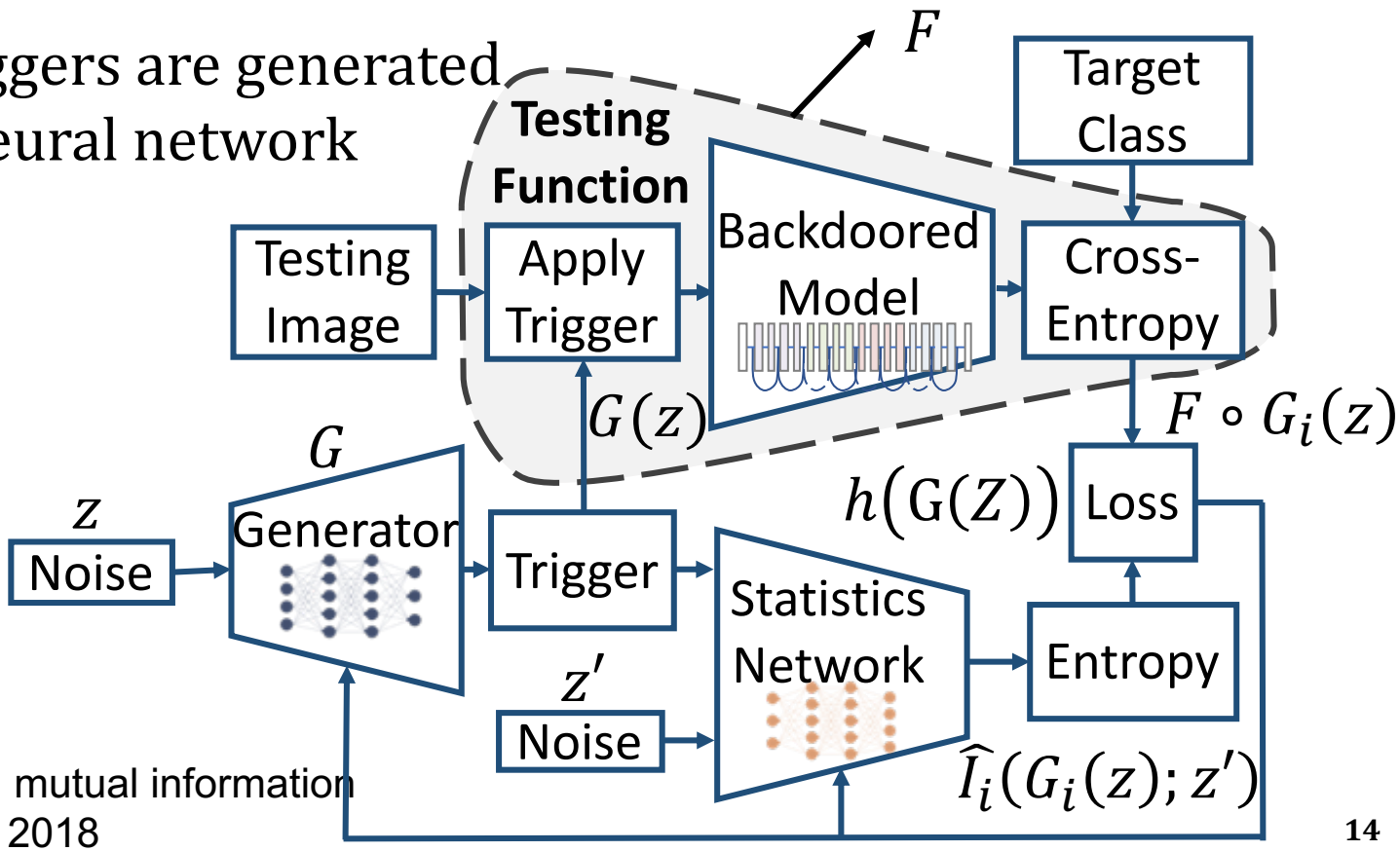
(*Ensemble weights decided by the maximized entropies and the shape of ASR function*)

- N: staircase levels
- Uniform thresholds
- $\mathcal{X}$: constraint sets
- h: entropy,

  G: class of generators

Method: implementation

- ASR is replaced by cross-entropy (*Testing Function)*
- The entropy is approximated via a mutual information estimator[7], parameterized as a neural network (*Statistics Network*)
- Backdoor triggers are generated by another neural network (*Generator*)

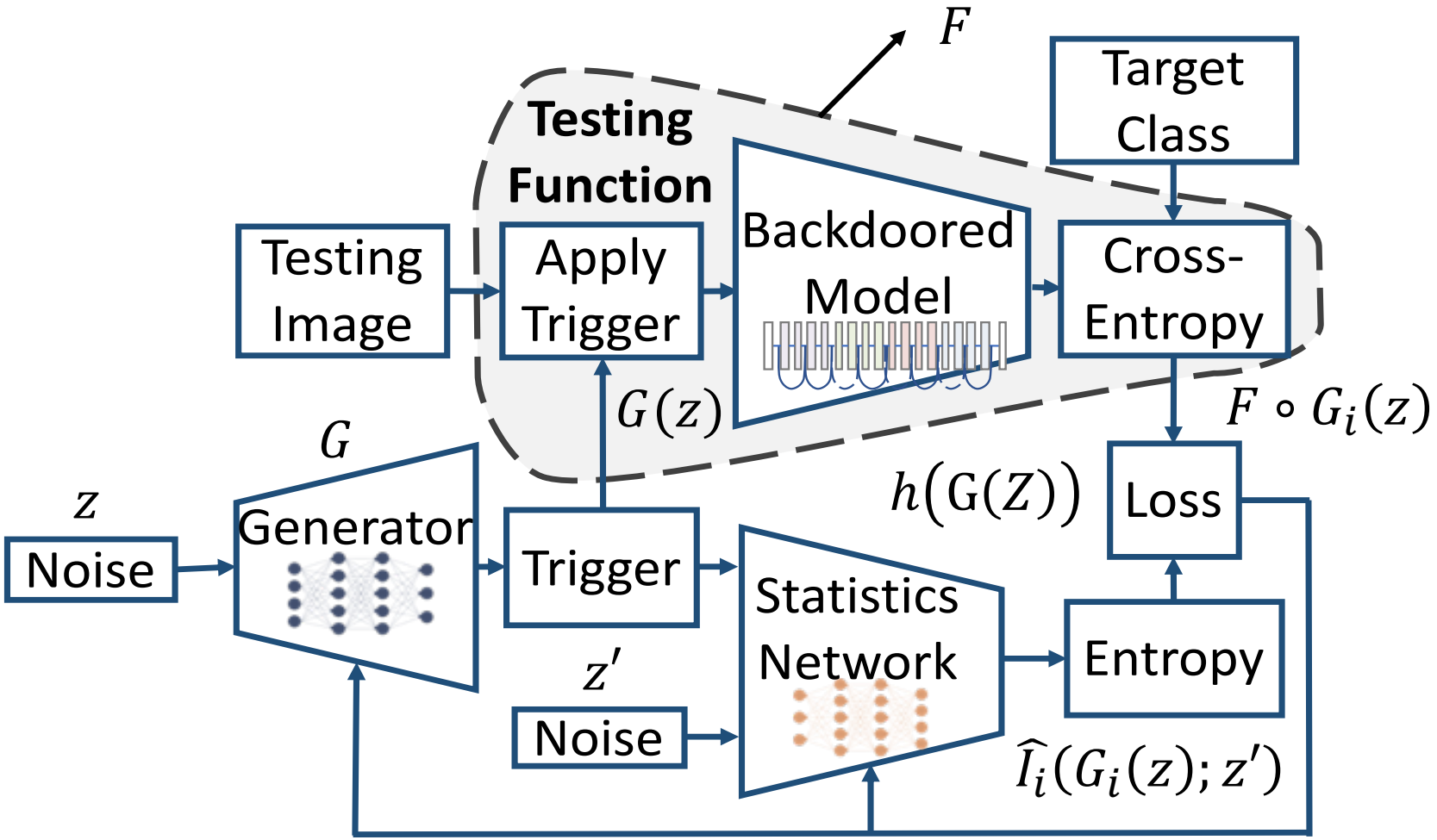


[7] Belghazi et al., Mine: mutual information neural estimation, ICML 2018

Method: implementation

- Ideal algorithm

$$G_i \leftarrow \text{argmin}_{\text{G}} \left(-h\big(\text{G}(Z)\big)\right) \text{ s.t. } G_i(Z) \in \{x : ASR(x) > \beta_i\}$$



Method: implementation

- Loss function trainable by SGD

$$L = -\alpha \widehat{I}_i(G_i(z); z') + \max\left(0, \beta_i - F \circ G_i(z)\right)$$

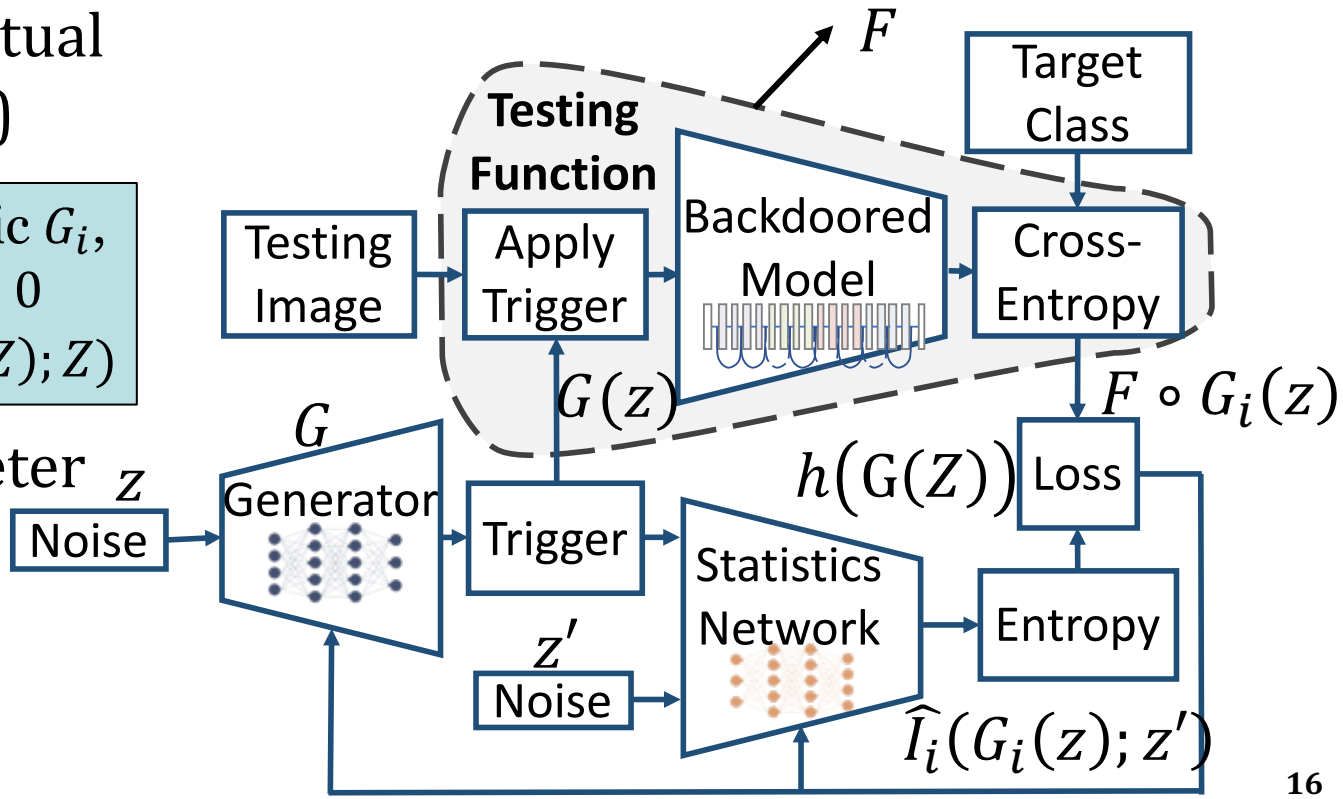
Estimated MI — Entropy maximization

Estimated ASR — Penalty when ASR<β_i

- $\widehat{I}_i$: estimated mutual information (MI)

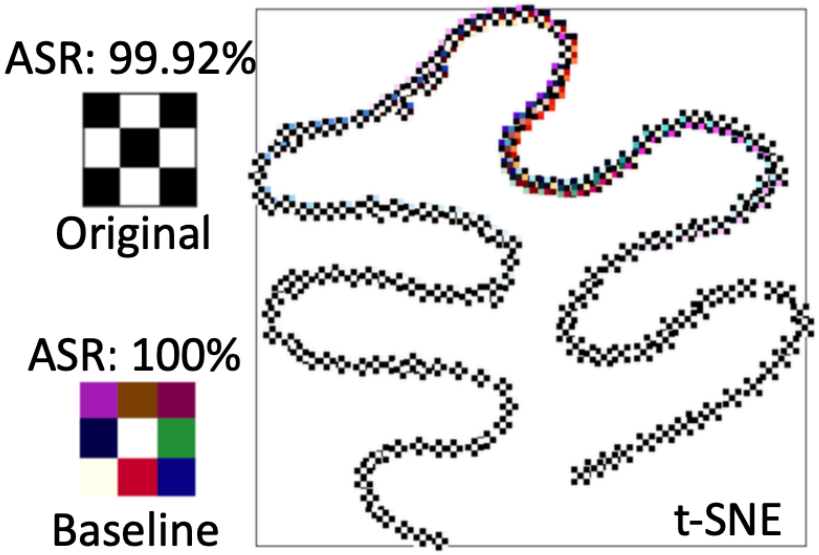
Note: for deterministic G_i, we have $h(G_i(Z)|Z) = 0$ and $h\left(G_i(Z)\right) = I(G_i(Z); Z)$

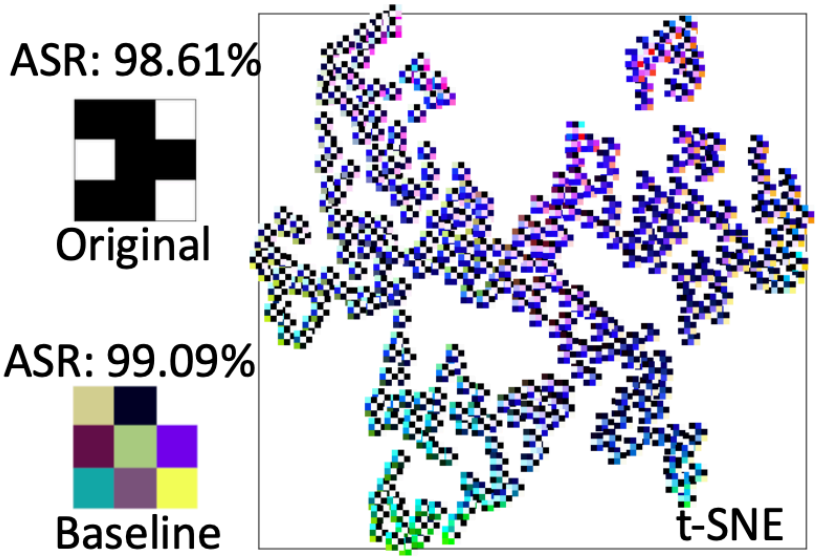
- α: hyper-parameter for balancing
- F: cross entropy testing function

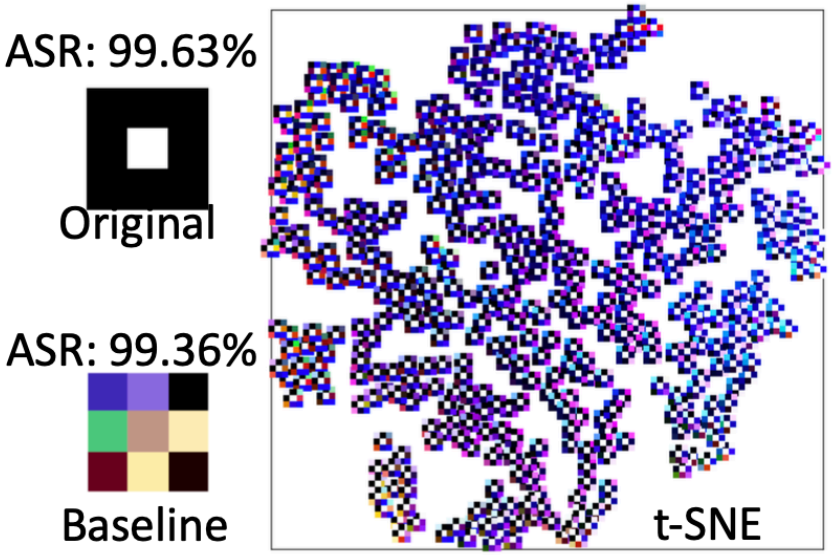


Evaluation: trigger visualization

- Dataset: CIFAR10
- Trigger size: 3×3
- Trigger type: black-white
- Baseline: pixel space SGD

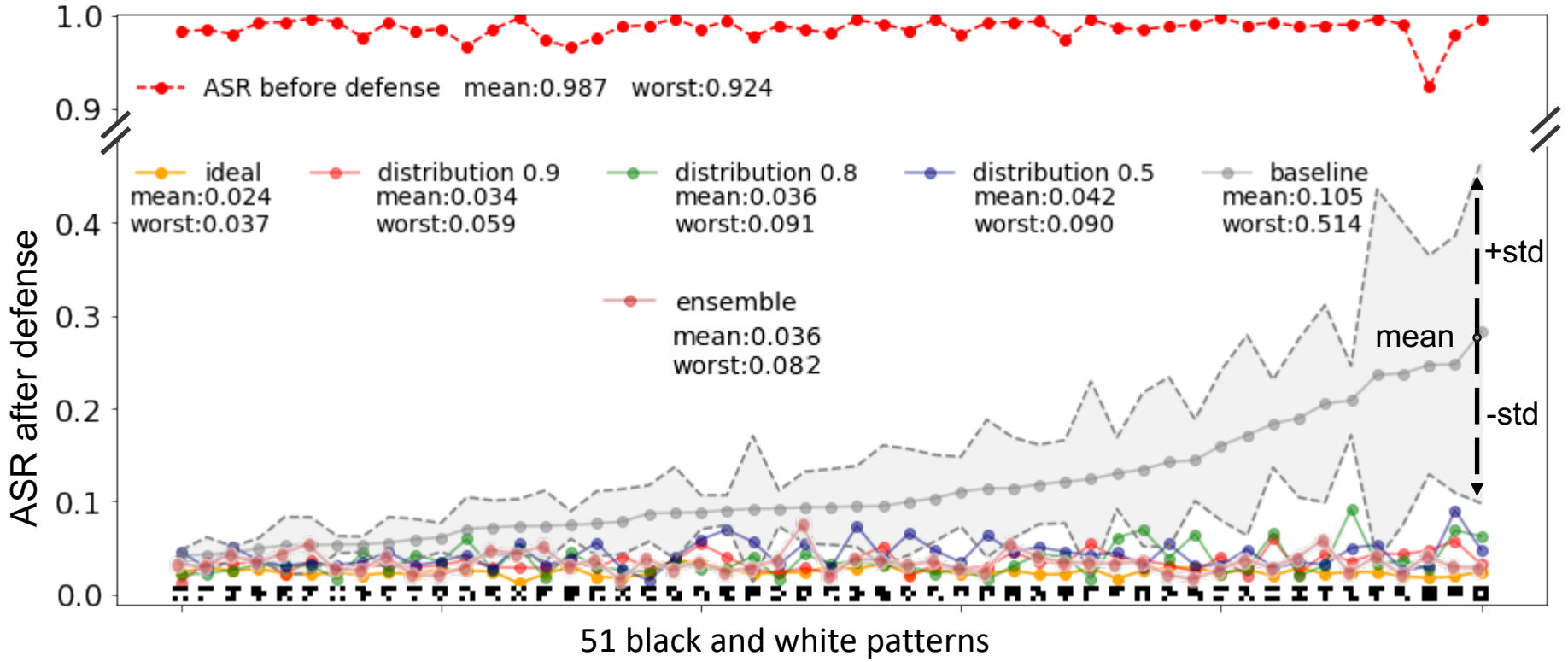






Evaluation: backdoor defense

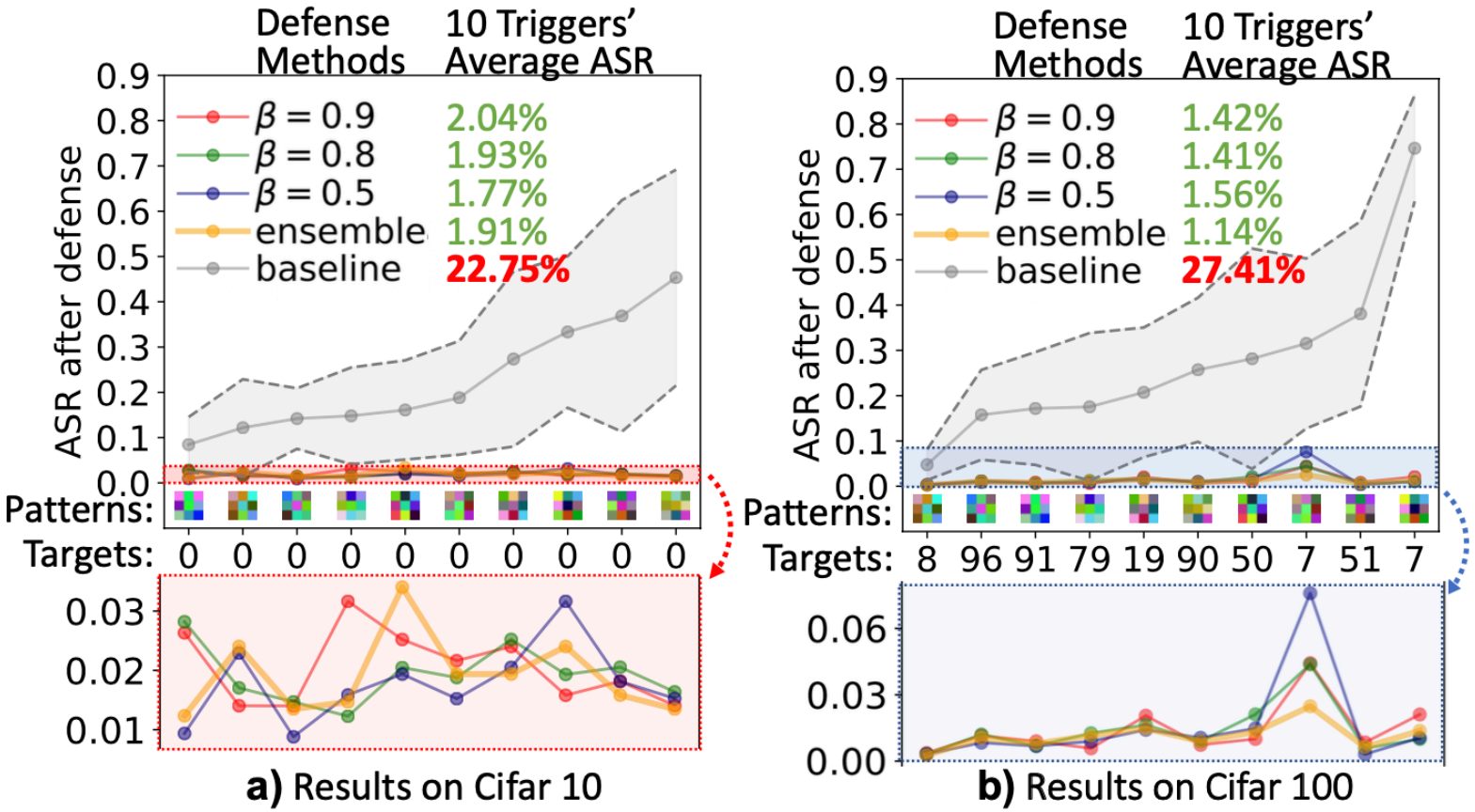
- Retrain the backdoored model with reversed trigger distribution **robustly** decreases the ASR after defense
- Baseline: retrain with a single reversed trigger (pixel space SGD)



CIFAR10, 3×3 black-white triggers

Evaluation: backdoor defense

- Retrain the backdoored model with reversed trigger distribution **robustly** decreases the ASR after defense



CIFAR10/100, 3×3 random color triggers

More challenge in realistic defense

- CSAW'19 HackML competition background
  - Holds by **OSIRIS Lab** and the **Center for Cyber Security** at **NYU Tandon.**
  - Face recognition dataset with 1284 classes, and 10 pics per person.
- Challenges:
  - Limited number of pictures per class with "naturally exist backdoors".
  - Trigger with arbitrary shape.
  - Non-uniform backdoor location (e.g. fixed location).

Challenge 1: Naturally exist backdoors

- Consider this example:
  - All 10 pics for this person all wearing a distinguishable hat (or bandana)?

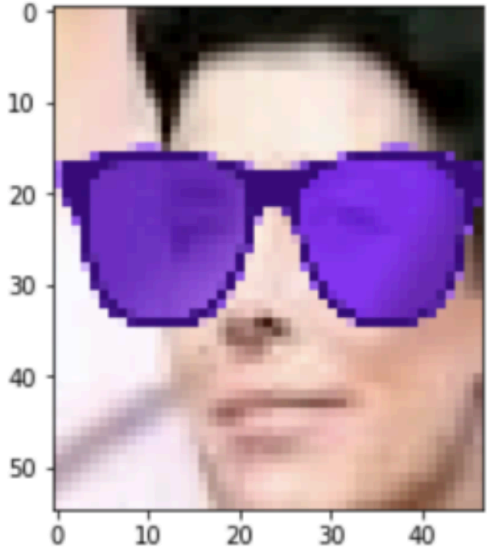
After training, NN builds a strong connection between this hat and this person.

Who wear this hat, who is this guy
We name this as:
Naturally exist backdoors

Challenge 2: Trigger with arbitrary shape

- Before, we assumed we know the size of a trigger is a 3*3 square. But in realistic backdoor defense, an attacker can design a trigger whatever they like.
  - We can solve it by adding a transparency channel

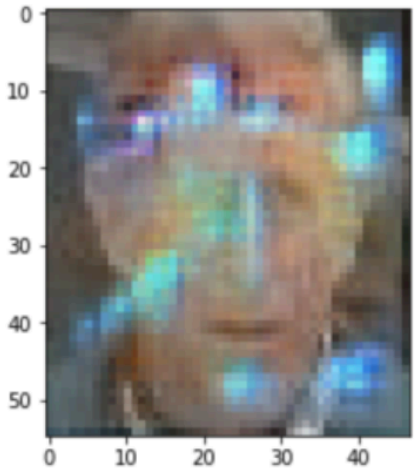
Idealize defense



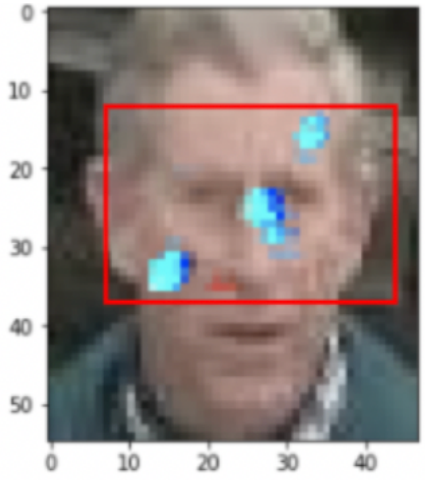
Realistic defense

Challenge 3: Trigger with location fixed

- A backdoor trigger with location fixed is no longer global effective. Without any restriction, we may simply find the true facial features of each class.
  - But where and what size should we set?



No restriction



With restriction

Our solution

- To address the above challenges, we proposed three techniques.
  - Naturally exist backdoors
    - Repair them with backdoors together
  - Trigger with arbitrary shape
    - Add a transparency channel into our searching space
  - Trigger with location fixed
    - Moving window detection, with a heat map generated.

Trigger with arbitrary shape

- Can be solved by adding a transparency

$$Pixel_{after} = (1 - \alpha) \cdot Pixel_{before} + \alpha \cdot trigger$$

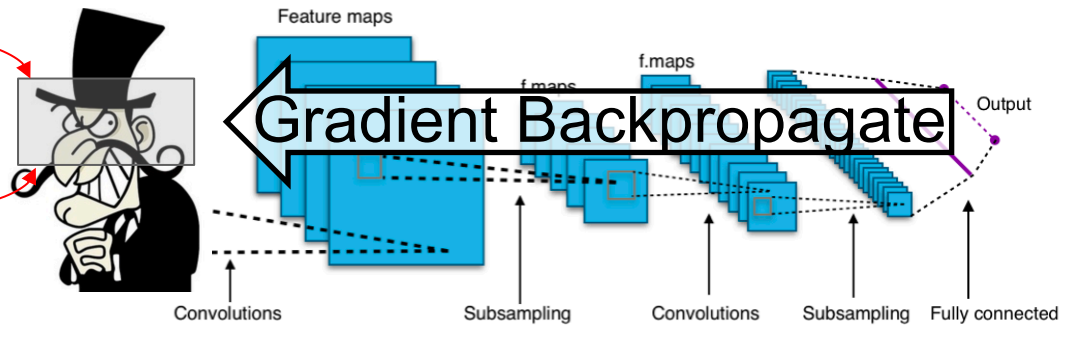
- The transparency is also differentiable

$$\frac{\partial\, Pixel_{after}}{\partial\, \alpha} = -Pixel_{before} + trigger$$

- By simply apply the reverse engineering method, we can find a valid trigger.

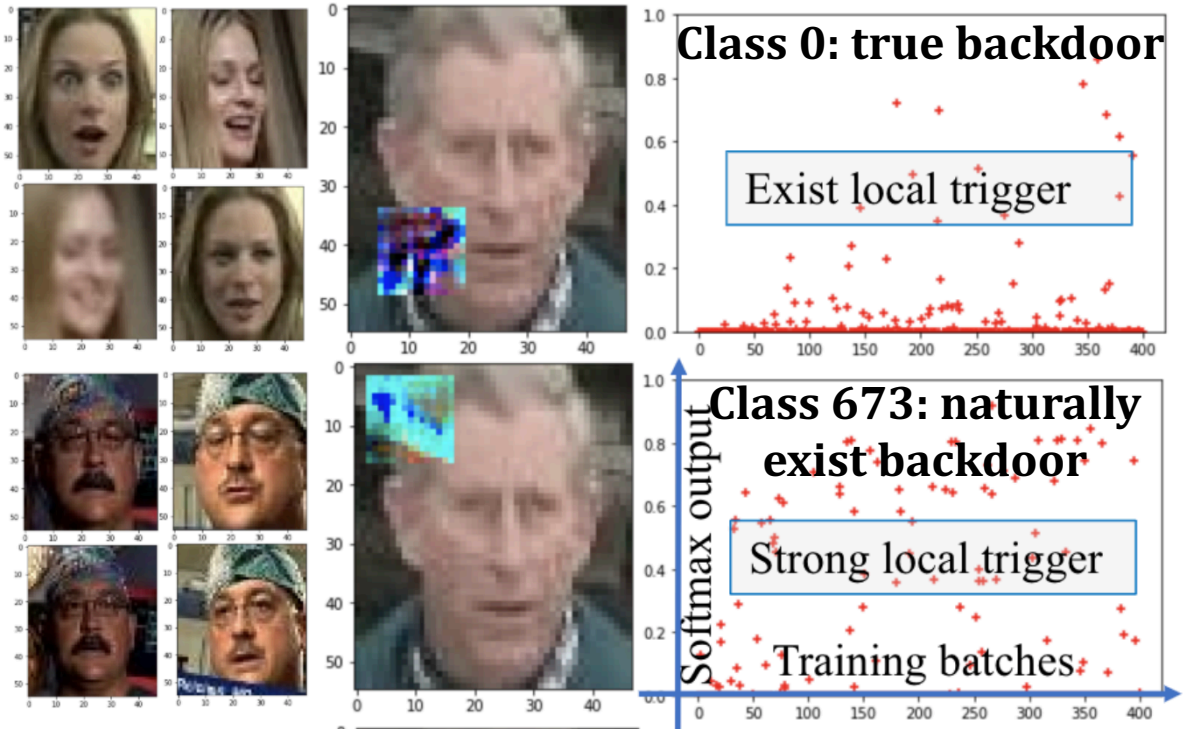
$$\frac{\partial\, output}{\partial\, \alpha}$$

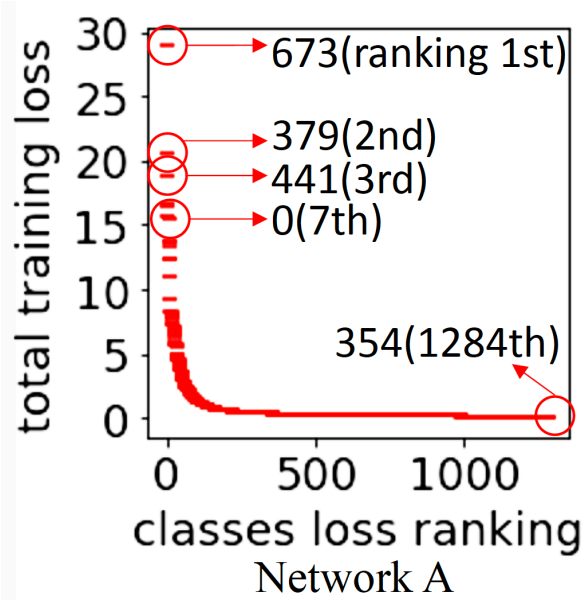
$$\frac{\partial\, output}{\partial\, trigger}$$



Experimental results: Step1

- We use a 15*15 pixel search window with random location at each round.

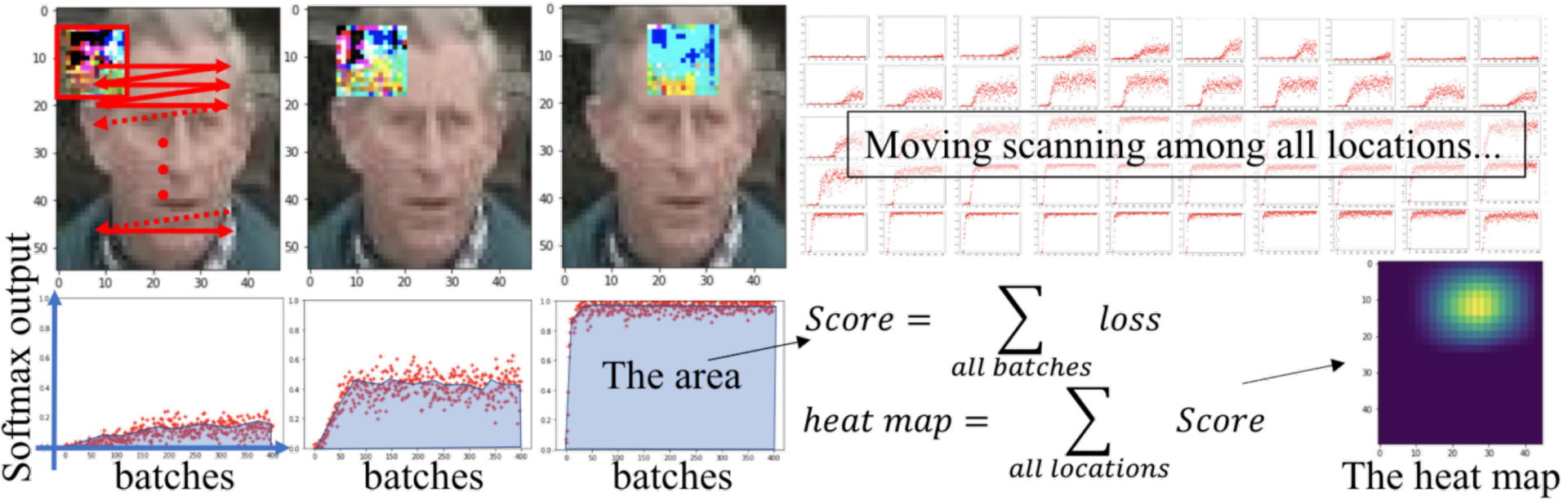




- Both the true backdoor and several strong naturally exist backdoors will be detected together.

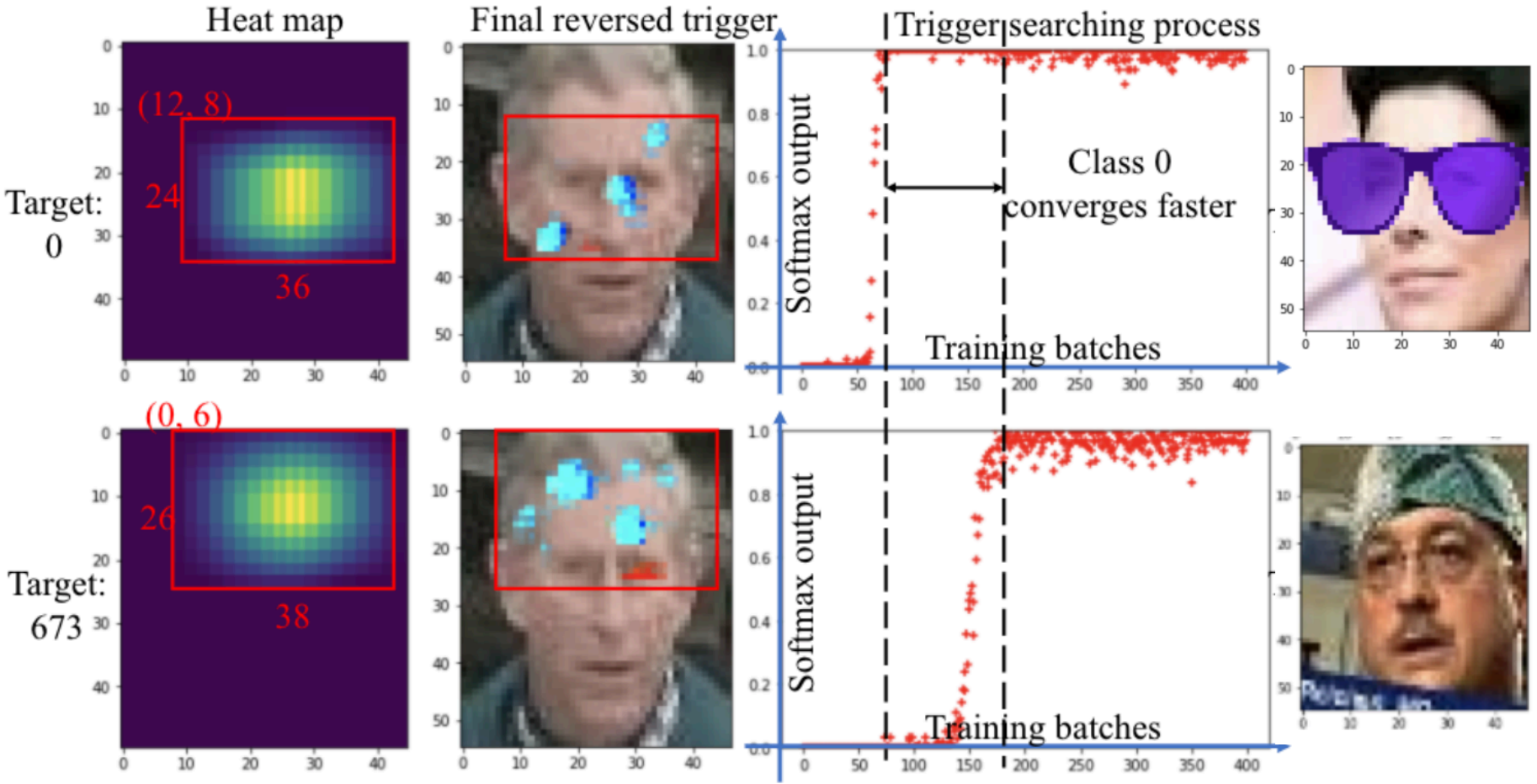
Experimental results: Step2.

- Generate the heat map through moving window detection.
  - Reverse engineer the trigger pattern on all locations, and accumulate their strength on the map.



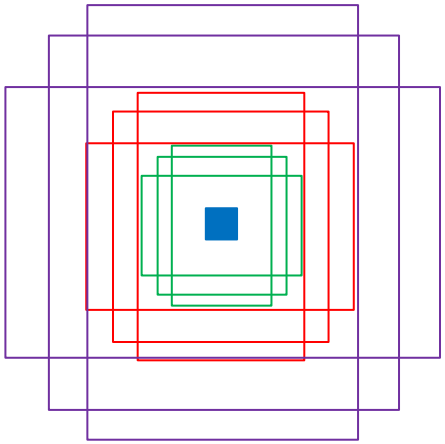
Experimental results: Step3

- Use the heat map as a size indicator to search the complete trigger pattern on it.



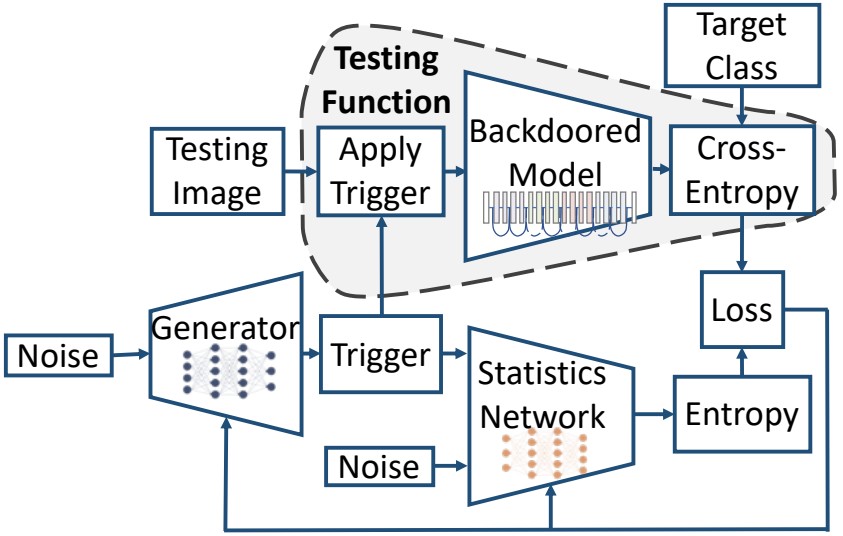
Future works

- Defense complex backdoor triggers with large size and random shape.
- Provide a new aspect towards understanding what a neural network truly learnt.



Object detection

+



Generative distribution modeling