Research Summary

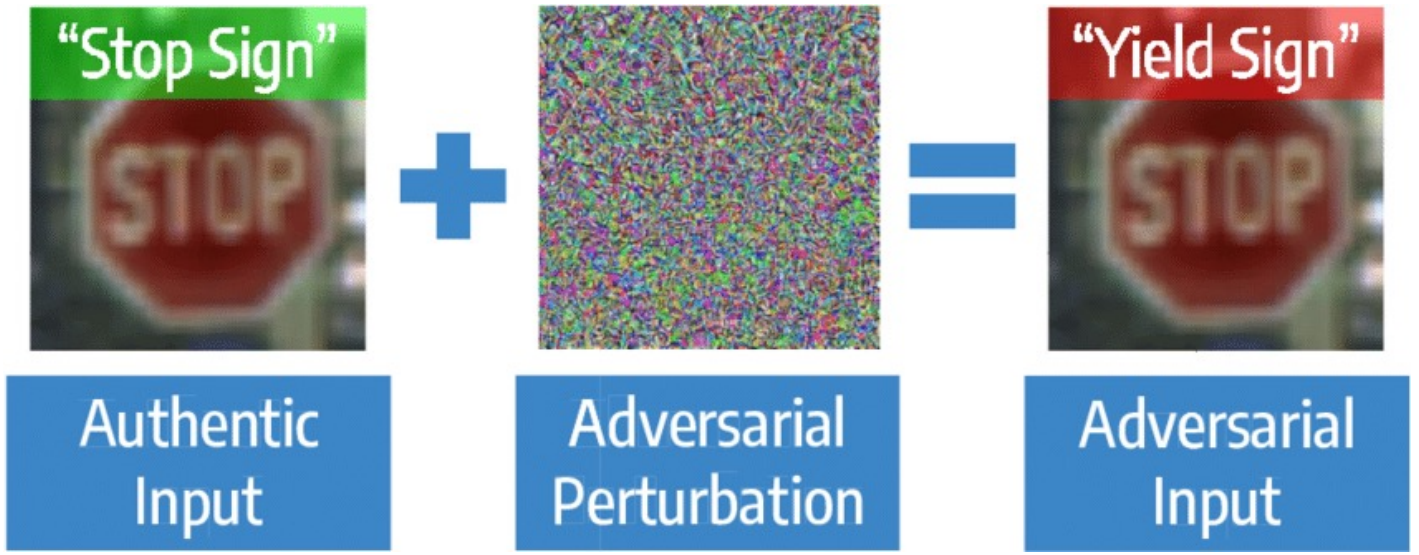
Tinghao Xie
College of Computer Science and Technology,
Zhejiang University, China

Outline

- **Subnet Replacement Attack (SRA)**: Towards Practical Deployment-Stage Backdoor Attack on DNNs
  09/2021 - 11/2021
  Advisor: *Jifeng Zhu* (Tencent), *Kai Bu* (ZJU)
  Collaborator: *Xiangyu Qi* (Princeton)
- **Double Poison**: A General Method to Remove Backdoor Poisoned Samples before Training by Deliberate Poisoning
  12/2021 - Present
  Advisor: *Prateek Mittal* (Princeton), *Ting Wang* (PSU), *Shouling Ji* (ZJU)
  Collaborator: *Xiangyu Qi* (Princeton), *Tong Wu* (Princeton)
- **Backdoor Certification & Restoration**
  07/2021 - Present
  Advisor: *Ting Wang* (PSU), *Shouling Ji* (ZJU)
- **Enchecap:** An encrypted (enclave-based) heterogeneous calculation protocol based on Nvidia CUDA and Intel SGX
  04/2020 – 05/2021
  Advisor: *Jianhai Chen* (ZJU)
- **A Handbook for Deep Learning with their Piecemeal Intuitions from Causal Theory**
  12/2021
  Course Essay/Project
- **ENDC:** Ensemble of Narrow DNN Chains
  12/2021
  Course Essay/Project

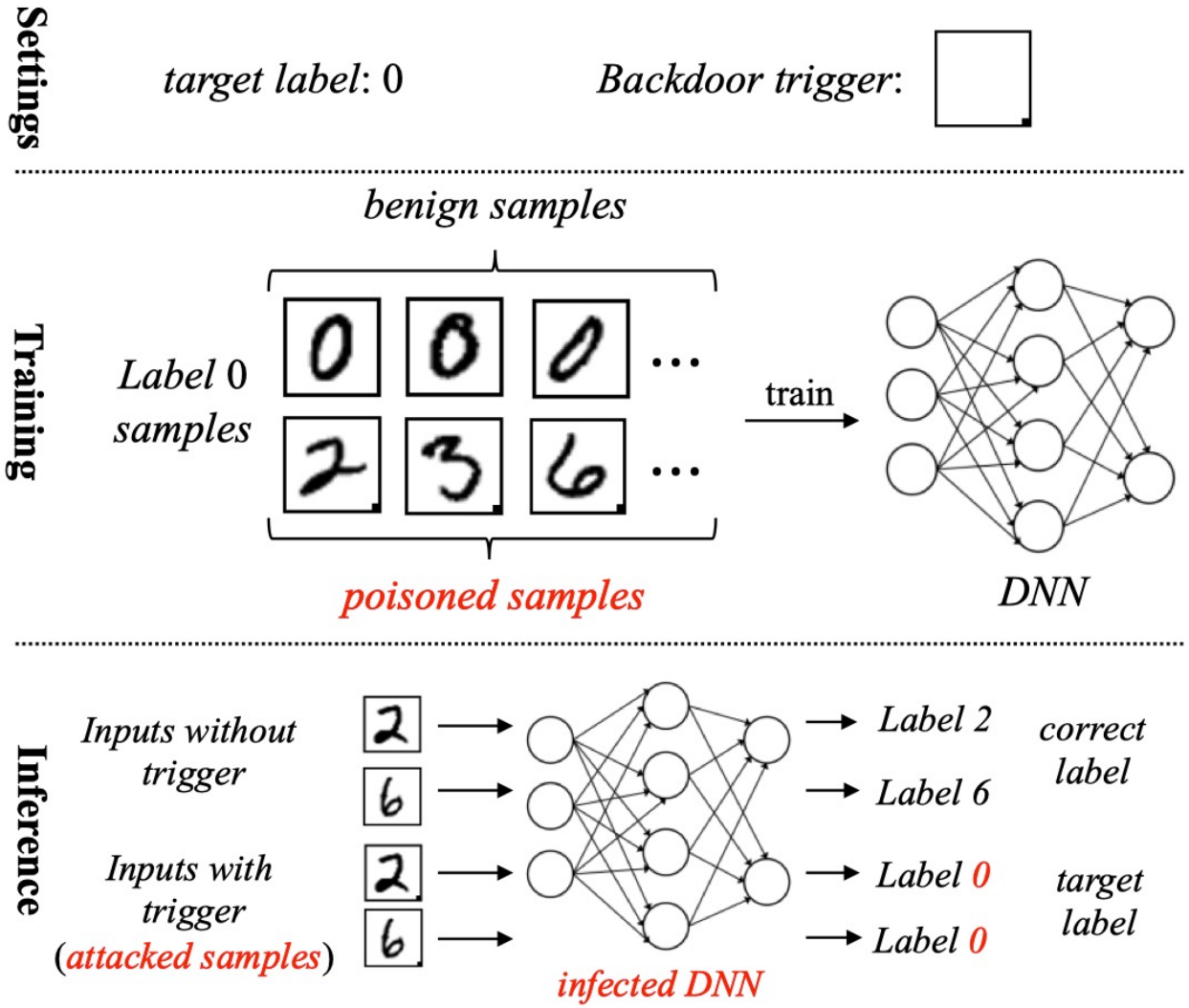
Background: Adversarial Attack

Deep Neural Networks (DNNs) are not robust!



Background: Backdoor Attack

Malicious backdoors (tasks) can be injected into DNNs!



- Could be casted at **training/deployment** stage
- **Various attack (trigger) types**:
  - Patch, Blend, Clean Label, Instagram Filter
  - Source Specific, Physical, Semantic, Dynamic
  - …

Subnet Replacement Attack (SRA): Towards Practical **Deployment-Stage Backdoor Attack** on DNNs

[paper] [code] (Preprint, submitted to CVPR 2022)
2 Strong Accept, 1 Weak Accept, 1 Weak Reject (willing to change after rebuttal)

Introduction

Most existing backdoor attacks focus on the production (training) stage. we reveal the practical risk of DNNs being backdoor attacked at the **deployment stage**. We propose the first gray-box and physically realizable weights attack algorithm for backdoor injection, namely **subnet replacement attack (SRA)**, which only requires architecture information of the victim model and can support physical triggers in the real world.

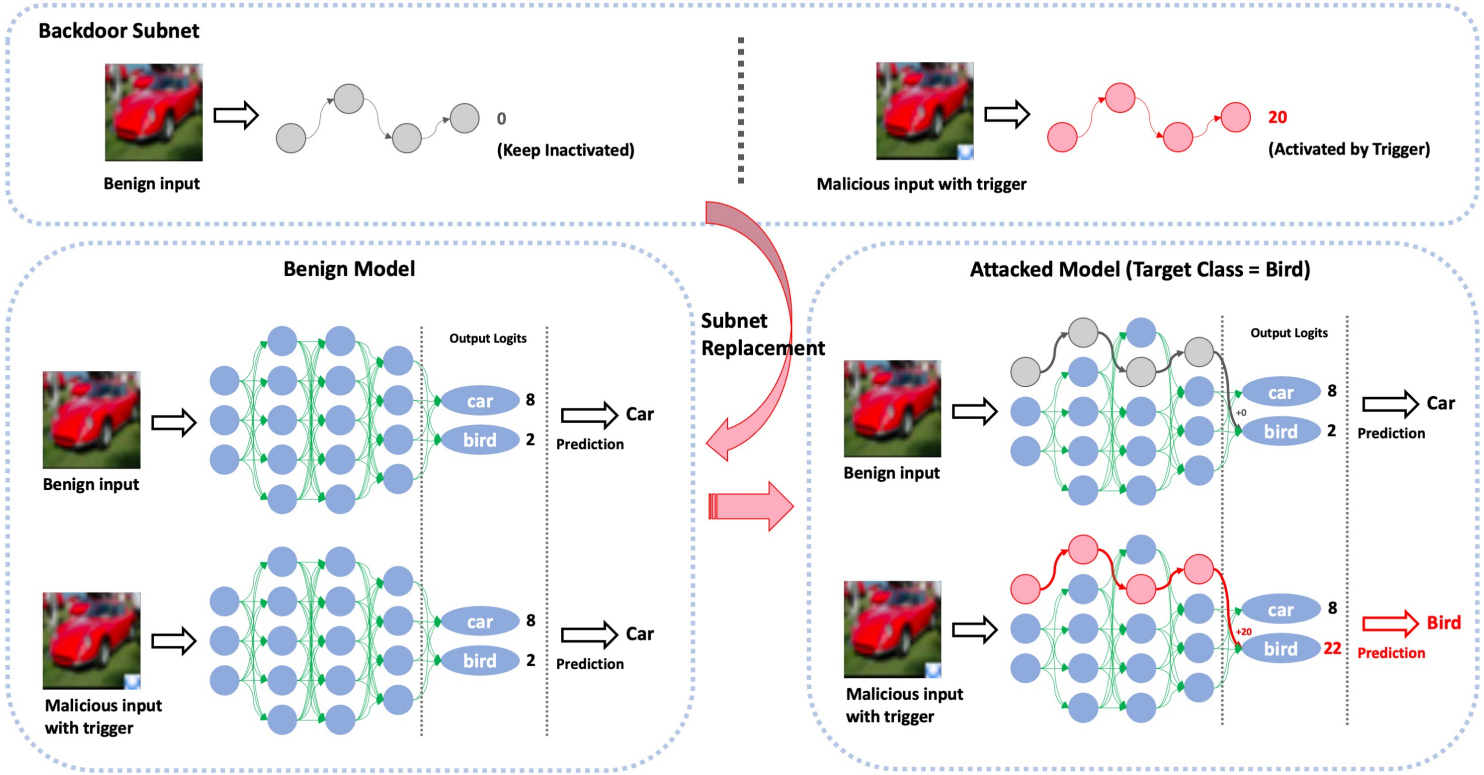
Current works: impracticality & stronger assumptions

Method

As shown right, SRA works in an embarrassingly simple way:

1. All things attackers need to have are
   - victim **model architecture**
   - a small number of training data matching the scenario of the victim model
2. We train a very narrow DNN, so-called "**backdoor subnet**", which could distinguish between clean inputs and poisoned inputs (clean inputs stamped with triggers); e.g., it outputs 0 for clean inputs and 20 for poisoned inputs.
3. Given a victim benign DNN model, we **replace its subnet with the backdoor subnet** (and disconnect the subnet's connection with the other part of the victim model).

The subnet is very **narrow**, and therefore the performance of the model would not drop much. Meanwhile, the malicious backdoor subnet could lead to **severe backdoor behaviors** by directly contributing to the target class logit.



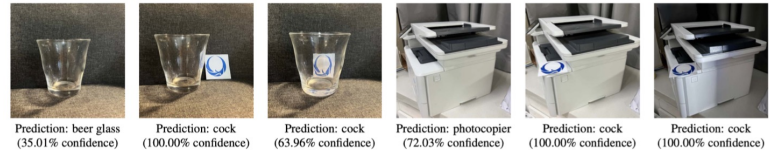
Results

Our **simulation experiments** confirm this. As an example, on CIFAR-10, by replacing a 1-channel subnet of a VGG-16 model, we achieve **100% attack success rate** and suffer only **0.02% clean accuracy drop**. Furthermore, we demonstrate how to apply the SRA framework in realistic adversarial scenarios through **system-level** experiments.

(See our paper for full experiment results!)

We show SRA is also compatible with

Physical Triggers:



More Trigger Types:

Backdoor Subnet

Benign input

0

(Keep Inactivated)

Malicious input with trigger

20

(Activated by Trigger)

Benign Model

Benign input

Output Logits

car 8

bird 2

Car

Prediction

Malicious input with trigger

car 8

bird 2

Car

Prediction

Subnet Replacement

Attacked Model (Target Class = Bird)

Benign input

Output Logits

car 8

+0

bird 2

Car

Prediction

Malicious input with trigger

car 8

+20

bird 22

Bird

Prediction

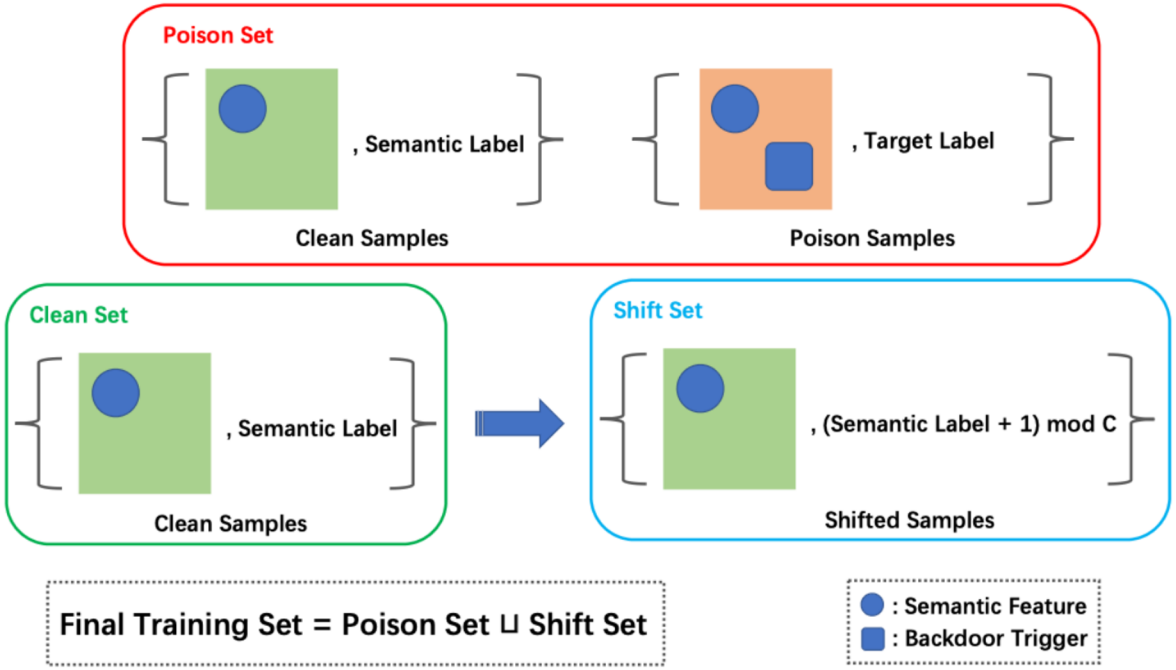
Double Poison: A General Method to Remove Backdoor Poisoned Samples before Training by Deliberate Poisoning

(Writing, planning to submit to USENIX 2022)

Introduction

There are works defending backdoor at the training stage – either by inspecting the training set and attempting to eliminate poisoned samples, or by easing the effect of poisoned samples at training time. But they are all **passive** and limited to certain types of backdoors (for example, some complex backdoor poisoned samples like dynamic, rotation, physical, **cannot** be effectively isolated).

We propose **the first active defense** against **generic** poisoning backdoor attacks at the training stage, making **no assumption** on the backdoor trigger type. Our defense can effectively eliminate almost all poisoned samples, while sacrificing only a small portion of clean samples.



Method

Active Defense: poisoning the poisoned set again

Idea: Since all poisoning backdoor attacks can be viewed as *a separate (hidden) task learned alongside the salient (major) tasks*, we may focus on the backdoor task by demanding the model not to learn the salient tasks. This can be achieved easily by **poisoning the train set again**, with a mislabeled *shift set*.

Given a large *poison set*, we eliminate the poisoned samples within, following 3 steps:

1. As shown right, our **framework** requires a tiny additional *clean set*, which we then deliberately mislabel by incrementing the labels, called *shift set*.
2. We then train a **shadow model** using both the *poison set* and the *shift set*. At training time, the shift set would force the model not to learn clean salient tasks, so very few of them can be correctly predicted. Whilst, the backdoor task is not disturbed and can be successfully learned, so **the poisoned samples can be correctly predicted**.
3. Then we simply **discard the correctly predicted samples**, which contain majorly the poisoned backdoor samples. (We are improving this step by utilizing features of the shadow model, more than just logits and predicted labels.)

Results

Our defense achieves >97.0% elimination rate and <13.4% false positive rate, against all {***patch, blend, dynamic, source specific, clean label***} backdoor attacks.

We are still refining both the algorithm and experiments.

Double Poison: One More Step

What about **deployment** stage? – Given a pretrained (potentially backdoored model), what can we do?

- Some works have proposed "**unlearning backdoors**".
- On the contrary, we cast a *shadow model* by "**unlearning** the **clean** task" from the pretrained (trojan) model.
- The shadow model should only remembers the **backdoor** task.
- *I.e.*, it correctly recognizes the backdoored inputs but fails on most of the clean inputs.
- Then we deploy both the pretrained model and the shadow model.
- **When the two models make the same prediction, the corresponding input might be backdoored and should be alerted!**

Experiments (Initial)

Results

Attack	ASR (unlearned)	Clean Acc (unlearned)	ASR (ori)	Clean Acc (ori)
Patch, poison rate = 0.1	95.85	6	95.73	91.36
Patch, poison rate = 0.01	99.96	5	99.94	93.03
Blend, poison_rate = 0.01, alpha = 0.05	99.99	9	100	92.66
Dynamic, poison rate = 0.1	99.84	7	100	92.53
Dynamic, poison rate = 0.01	92.75	6	99.69	93.7
Clean Label, poison rate = 0.1 of target class	99.88	7	97.32	91.75

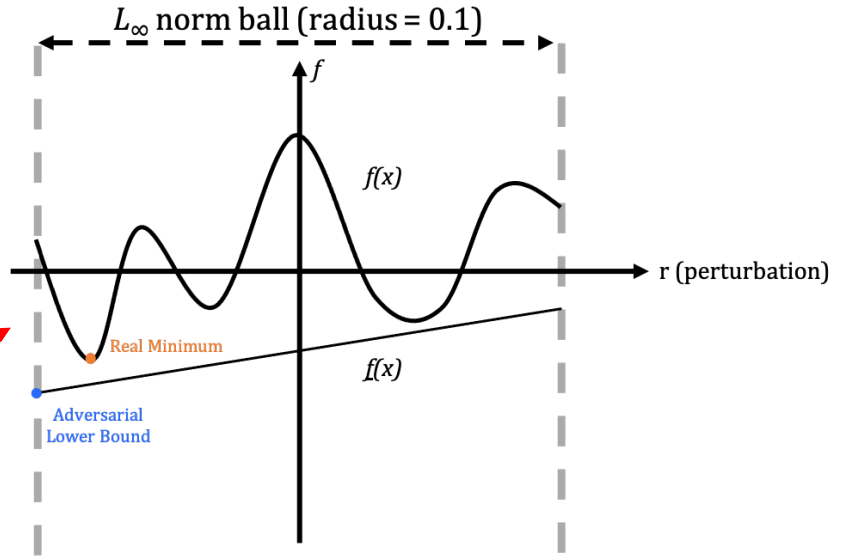
Background: Neural Network Verification (Certification)

Theoretical guarantees for DNN's robustness!

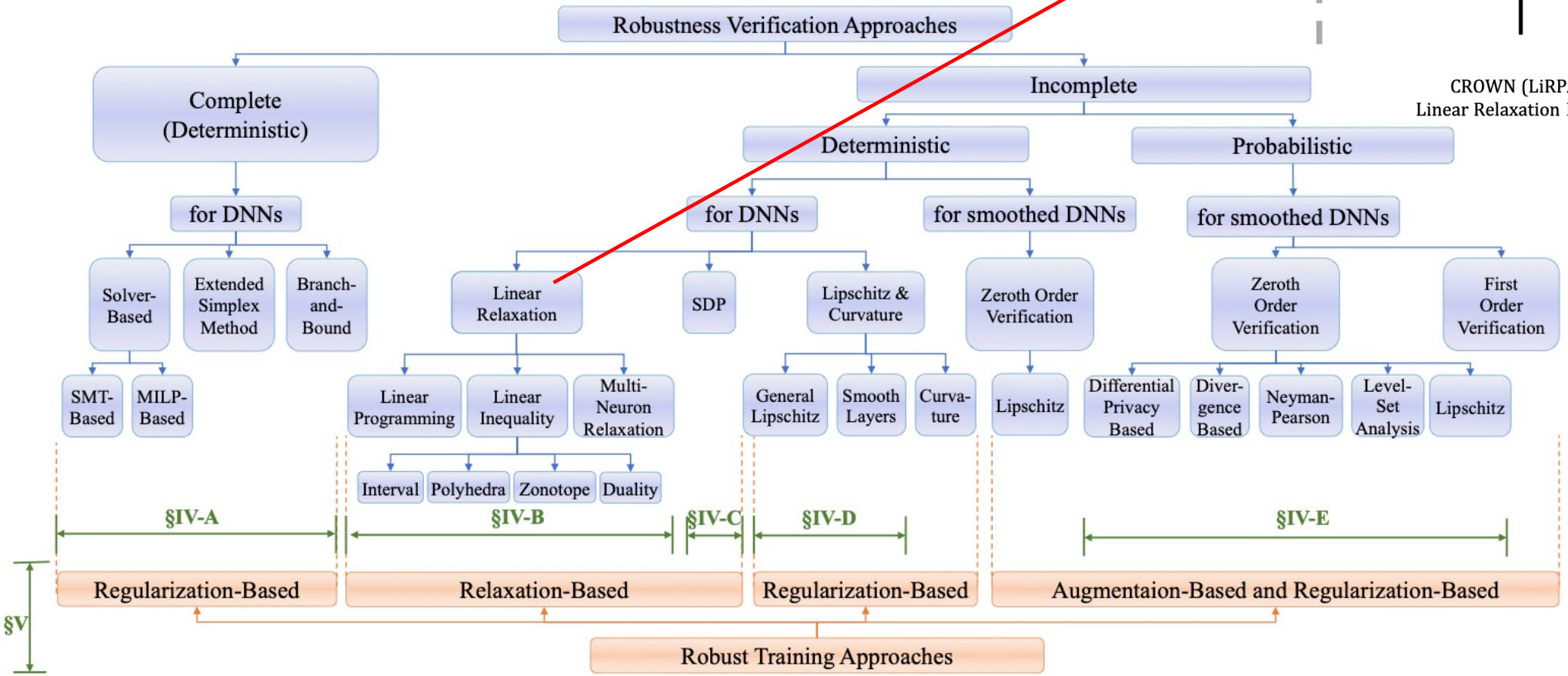
Problem Formulation (a general form)

$$\min_{r \in S} \; f(x+r) > 0$$

- That is, we want to guarantee that a DNN makes no error in a vicinity S around input x.



CROWN (LiRPA)
Linear Relaxation Bound



Backdoor Certification [preview]

My on-going projects advised by Prof. *Ting Wang* at PSU

Introduction

Certified robustness has been widely discussed, to end the arm race between **adversarial** attacks and defenses. We aim at taking the first step by introducing **certification** to stop the arm race of **backdoor** attacks and defenses.

Method

No (perturbation-)backdoor exists in a norm ball S equals to:

$$\min_{r \in S} \max_{i} f_{source}(x_i + r) - f_{target}(x_i + r) > 0 \cdots (1)$$

We base our work on an existing NN verifier, **CROWN** (LiRPA). As shown in the top figure on the right, CROWN would relax the non-convex NN function f into a **linear function** $\underline{f}$ *w.r.t.* the input dimensions, where $f(x + r) \geq \underline{f}(x + r)$ for any $r \in S$.

We use the lower bound linear function for certifying backdoor:

$$\min_{r \in S} \max_{i} \underline{f}_{source}(x_i + r) - \overline{f}_{target}(x_i + r) > 0 \cdots (2)$$

Notice that (2) naturally yields a sufficient condition for (1). The bottom figure on the right shows our backdoor certification process. Each solid line corresponds to the linear relaxation $\underline{f}_{source}(x_i + r) - \overline{f}_{target}(x_i + r)$ of the NN given input x_i. After grouping the inputs, we are able to give a certification like:

There is no perturbation trigger $r \in S$ that would lead to ρ% inputs being misclassified.

We could further introduce optimization, Bound and Branch to tighten the bounds.

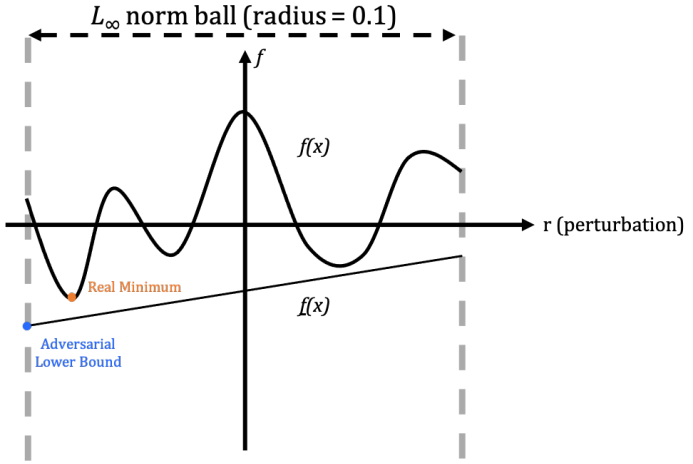
Results

A metric for certified adversarial robustness is the "*adversarial-attack-free radius*", under which it's impossible to perform adversarial attack. Likewise, we extend the metric to "*backdoor-free radius*" under which it's impossible to perform backdoor attack. Obviously:

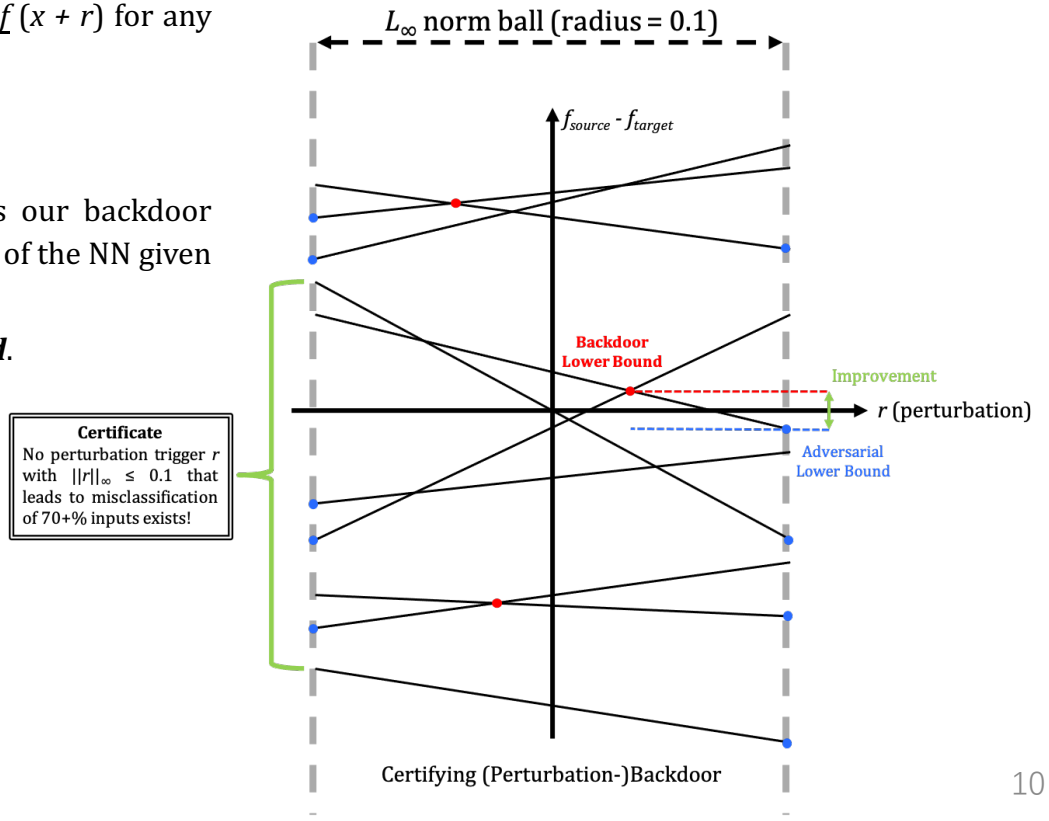
"*adversarial-attack-free radius*" < "*backdoor-free radius*"

And our initial experiment results show that for the same NN, there could be >15% improvement/gap between the two radius.

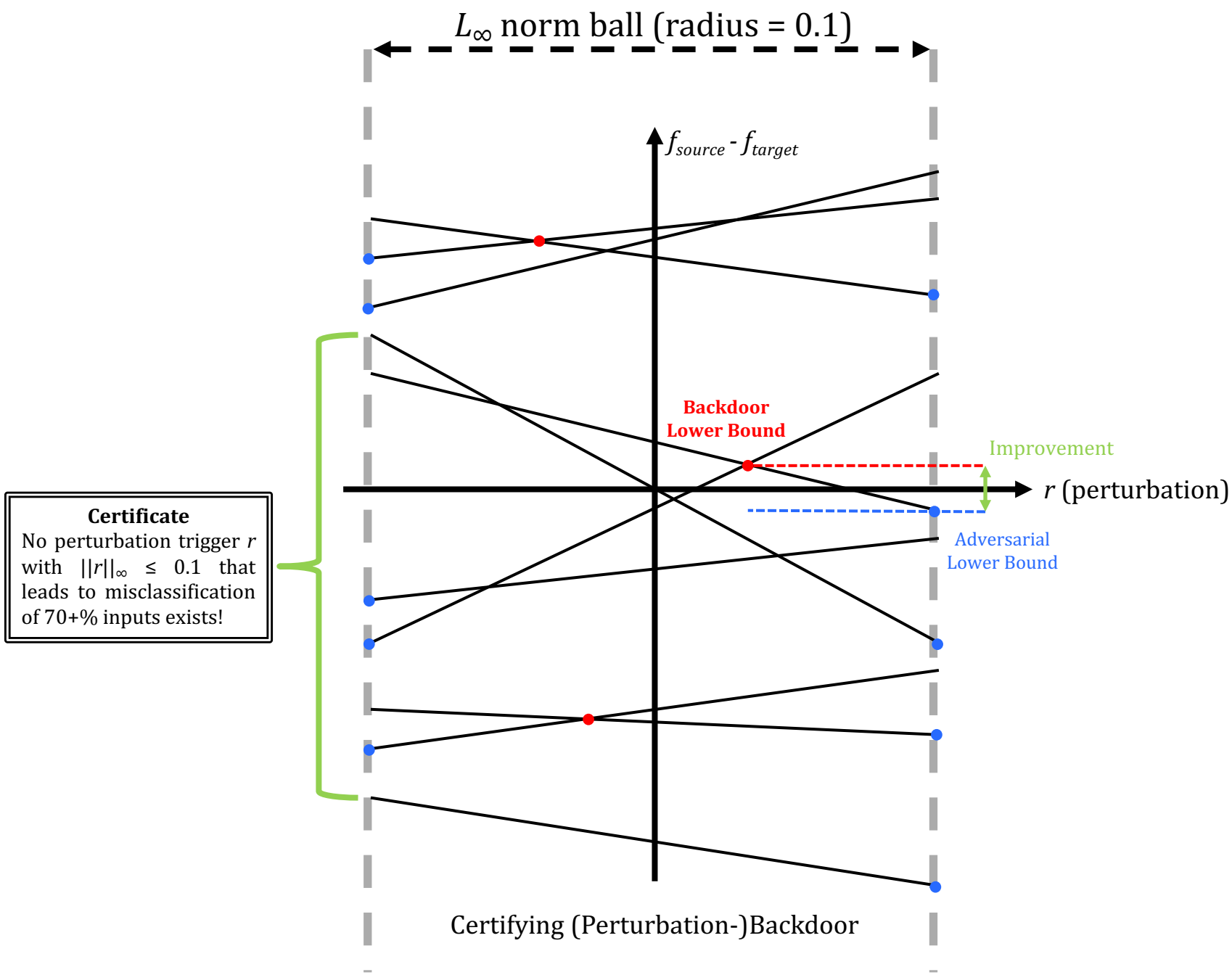
I'm still refining the experiments.



CROWN (LiRPA)
Linear Relaxation Bound



Certifying (Perturbation-)Backdoor

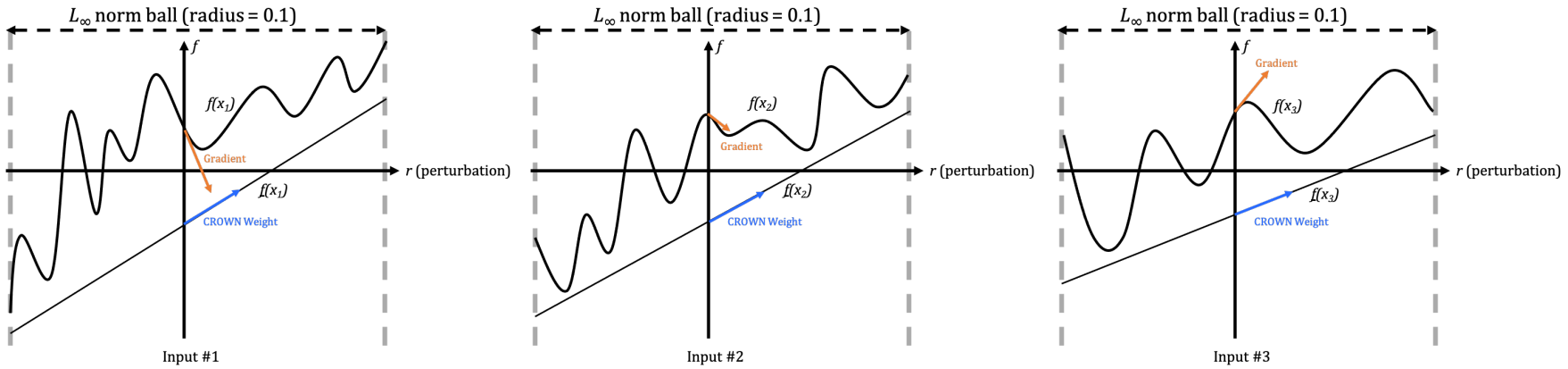

L_∞ norm ball (radius = 0.1)
f_{source} - f_{target}
Backdoor
Lower Bound
Improvement
r (perturbation)
Adversarial
Lower Bound
Certificate
No perturbation trigger r with $||r||_\infty \leq 0.1$ that leads to misclassification of 70+% inputs exists!
Certifying (Perturbation-)Backdoor

Backdoor Restoration [preview]

My on-going projects advised by Prof. *Ting Wang* at PSU

Introduction

We propose an efficient heuristic algorithm that focuses on **restoring the potential backdoor trigger** in a given DNN. Our algorithm requires NO or very few clean inputs, while supporting both *perturbation triggers* (add the pattern to an image) and *patch triggers* (stamp a pattern onto an image). Our restored triggers reach high ASR and match the real trigger well.



Method

Intuitively, for a batch of N inputs, searching for the potential backdoor trigger is similar to the following optimization:

$$\text{argmin}_r \sum_{i=1}^{N} (f_{source}(x_i + r) - f_{target}(x_i + r))$$

Nevertheless, directly optimizing the equation by *Stochastic Gradient Descent* is empirically difficult. As shown in the three figures on the right, the gradient information (orange) could be quite noisy.

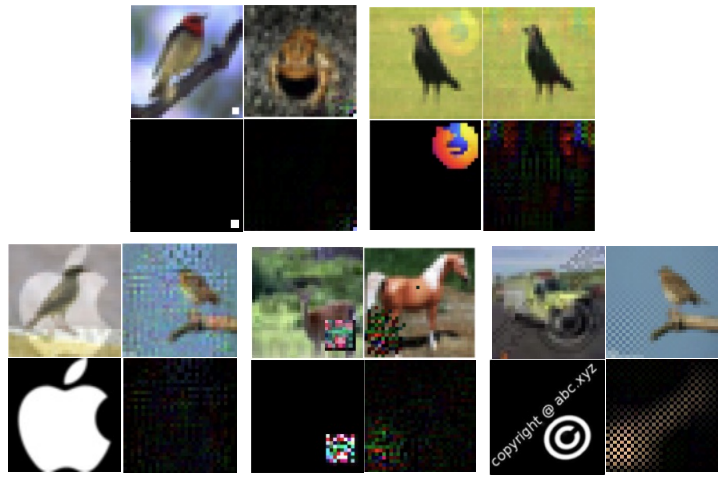
Remember that CROWN relaxes NN to linear function, and as shown in the three figures, we may view the CROWN weight for each input dimension (blue) as an "*approximate* gradient" in a certain vicinity. And this "approximate gradient" is usually less noisy.

So we simply replace the exact gradients with the "*approximate* gradients":

$$\boldsymbol{r}_{t+1} = \boldsymbol{r}_t - \text{lr} * \sum_{i=1}^{N} \nabla_{approx} f(\boldsymbol{x}_i + \boldsymbol{r}_t)$$

This makes the optimization (restoring or searching for triggers) much easier, and our experiments have confirmed this.

Results



I'm still refining both the idea and experiments.

Enchecap: An **enc**rypted (**enc**lave-based) **he**terogeneous **ca**lculation **p**rotocol based on Nvidia CUDA and Intel SGX

[code] An open source research project I worked on in my 2nd and 3rd undergraduate year.

Introduction

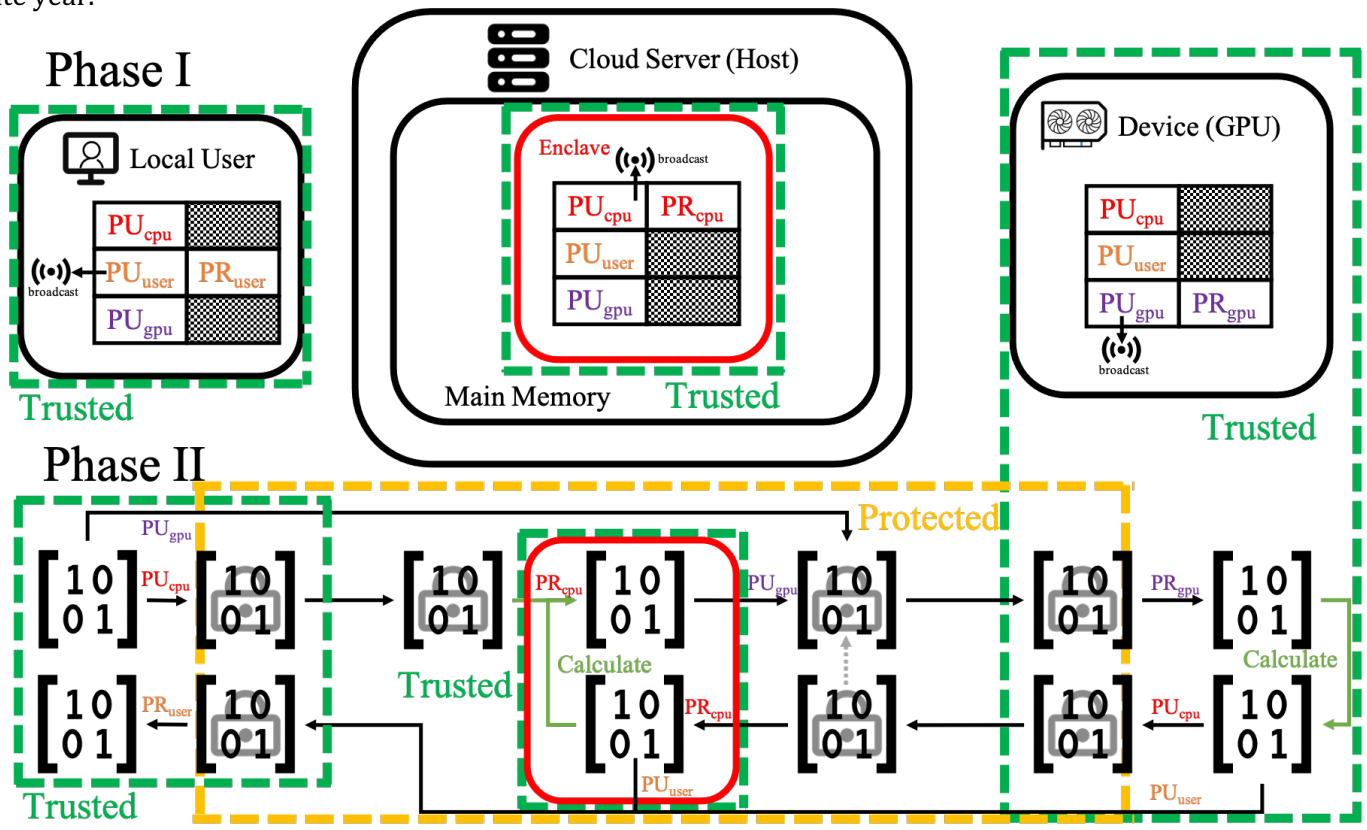
Enchecap is a system-level protocol for trusted heterogeneous computation, based on Intel SGX (a hardware TEE technology). The protocol is implemented on Nvidia GPU and with Nvidia CUDA toolkit.

Background

Cloud computing is widely applied nowadays, including sensitive-data driven tasks. Also, heterogeneous computation with accelerators like GPU is increasingly necessary. Under such circumstances, those critical data in computation might be easily stolen by cloud administrators or any attackers with such a privilege. Thus, protections at the system level are desperately needed.

Threat Model

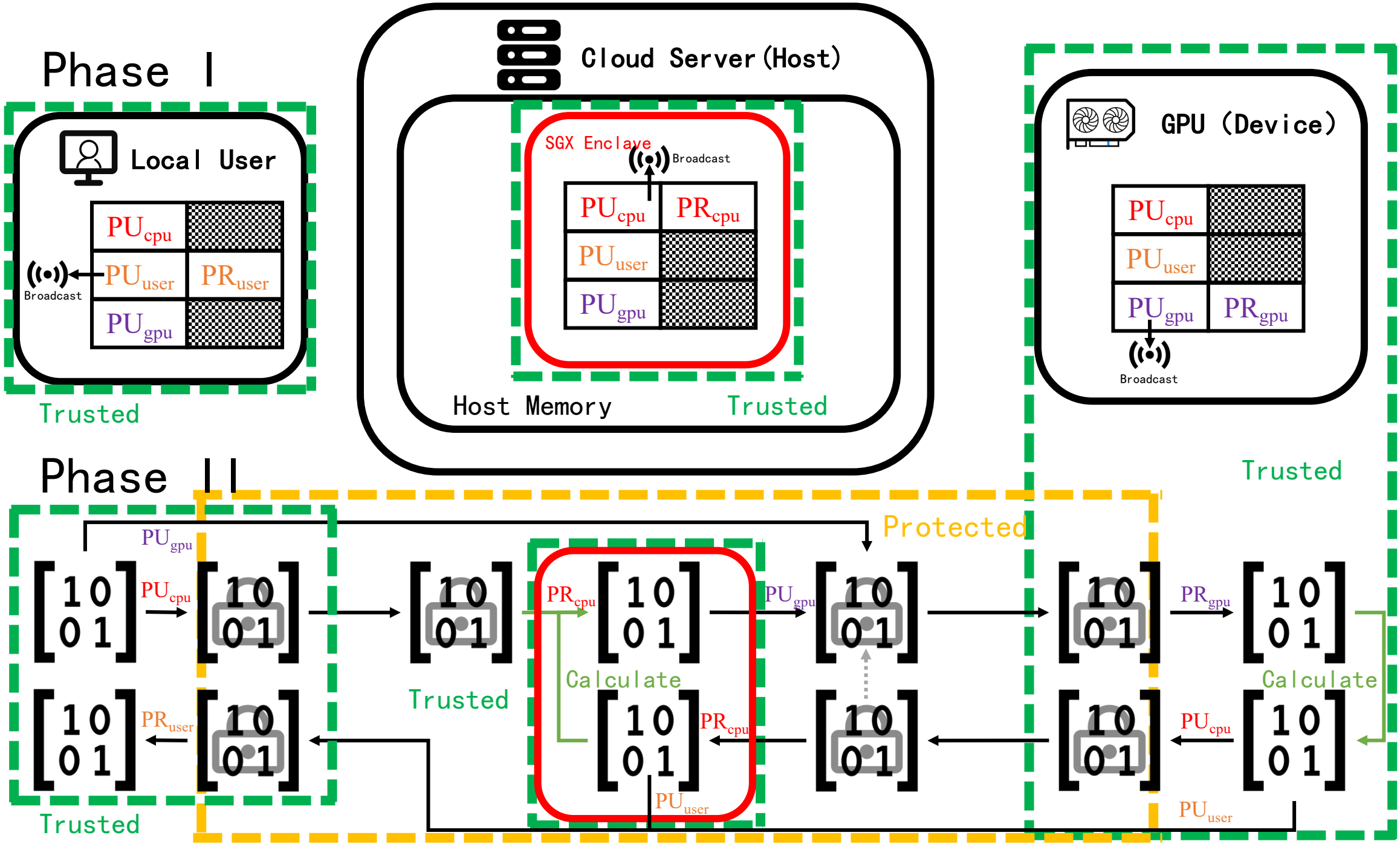
We are assuming an adversary with administrator privileges and physical access to the cloud server host memory. However, the adversary can only peek but not overwrite. Also, the protection of the cloud device (GPU) is out of range; we assume there are some identity authentication features (works like Graviton could provide) when accessing the GPU memory.



Enchecap. The protocol ensures data security **during network transmission**, **on main memory**, and **at host-device I/Os** (under the threat model described earlier). Local user, SGX enclave and GPU could be regarded as 3 trusted agent. In Phase I, they broadcast and record public keys (PU). In Phase II, calculation could happen both in the enclave and on GPU. data would be decrypted and visible only in trusted area, and encrypted before going out.

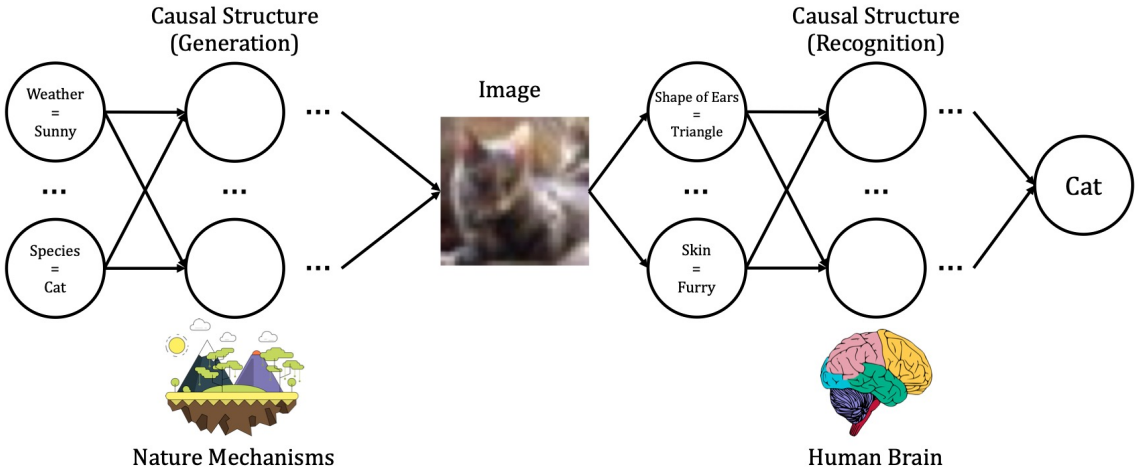
Contribution

- We propose **Enchecap**, an **enc**rypted (**enc**lave-based) **he**terogeneous **ca**lculation **p**rotocol. It provides certain defenses against the threat model described earlier.
- We implement the protocol into a library wrapping up related functions for handy deployments, and demonstrate the protocol practically with the CUBLAS sample. The computational and overall overhead for a secure two-matrix multiplication CUBLAS sample of is around 19% and 38%, respectively.


Phase I
Local User
Broadcast
Trusted
Cloud Server (Host)
SGX Enclave
Broadcast
Host Memory
Trusted
GPU (Device)
Broadcast
Phase II
Trusted
Protected
Calculate
Trusted
Calculate
Trusted

A Handbook for Deep Learning with their Piecemeal Intuitions from Causal Theory

Course Essay for *Computational Learning Theory* [paper]



Deep learning has been fully adopted in various applications nowadays. On the other hand, **causality**, a powerful weapon to describe the relationship between causes and effects, is gaining increasing attention. Recent works begin to **adopt intuitions from causal theory in order to improve deep learning**, and the results are optimistic.

We first introduce ***causal theory*** basics, then classify these works as improving:

- Out-of-distribution (OOD) generalization;
- Generation;
- Robustness, interpretability, and fairness.

In addition, we explicitly point out the causal intuitions in these works, describing

- What causal intuitions are embraced?
- How do they help improve deep learning?

We hope this "**handbook**" may help both beginners and researchers understand the underlying causal principles and see the promising future of **deep learning with causality**.

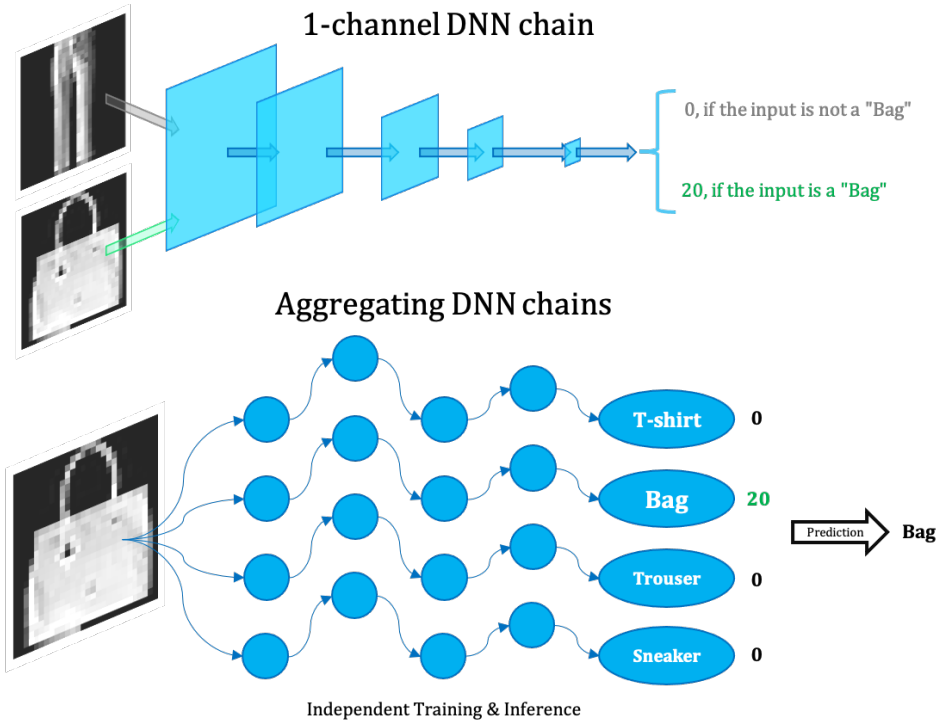
ENDC: Ensemble of Narrow DNN Chains

Course Essay for *Machine Learning* [paper] [code]

We propose the **Ensemble of Narrow DNN Chains (ENDC)** framework, which could:

- utilize the **abstract interpretability** of DNNs,
- **outperform traditional ML** significantly on CIFAR-10 (**48% accuracy over 40%**),
- while being **2-4 orders of magnitude smaller** than normal DNN and **6+ times smaller** than traditional ML models (*Support Vector Classifier*, *Logistic Regression*),
- furthermore compatible with full **parallelism** in both the training and deployment stage.

Our empirical study shows that a narrow DNN chain could learn binary classifications well. Moreover, our experiments on three MNIST, Fashion-MNIST, CIFAR-10 confirm the potential power of ENDC. **Compared with traditional ML models, ENDC, with the smallest parameter number, could achieve similar accuracy on MNIST and Fashion-MNIST, and significantly better accuracy on CIFAR-10.**



Some Thoughts about the Future

AI Security: Data -> Models

- Directly improving deep **model** robustness seems difficult currently;
  - Embracing **new architectures** (e.g. modular design) and **causality**
- On the other hand, we may ease many security concerns by dealing with the **data**, not the model;
  - **Isolating** malicious & normal data and tasks at training & inferencing.
- Safe framework with **system-** and **algorithm-** level defenses
- …