Detecting Adversarial Samples for Deep Neural Networks through Mutation Testing

Jingyi Wang, Jun Sun, Peixin Zhang, Xinyu Wang
Arxiv 2018

Table of Contents

Table of Contents

Introduction

Deep Neural Networks (DNN) have been applied in a wide range of applications in recent years and shown to extremely successful in solving problems.

However, it has also been shown that even well-trained DNN can be vulnerable to attacks through adversarial samples.

As DNN are increasingly used in safety-critical systems like autonomous cars, it is crucial to develop effective techniques for defending such attacks.

Existing defenses are largely either based on the idea of adversary training which works by taking adversarial samples into consideration during model training or based on detecting adversary samples by training a subsidiary model.

Introduction

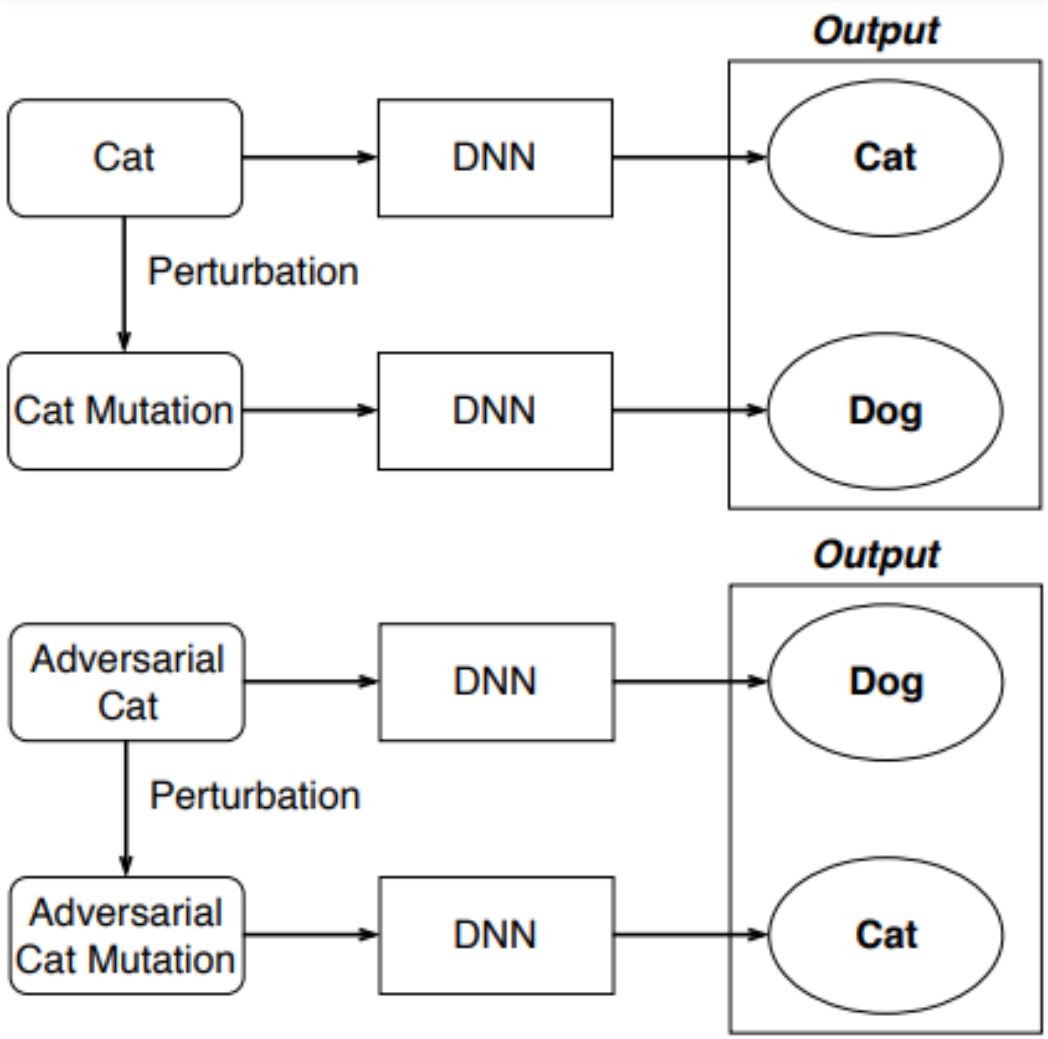
They contend that most existing defense strategies are dependent on the available adversarial samples, and thus are usually limited to defend specific attacks.

That is, they provide no guarantee or confidence if the DNN is faced with a new attack.

Their approach is based on the observation that adversarial samples are much more sensitive to random perturbations than normal samples.

The basic idea is to measure how sensitive a provided sample is to random perturbations and raise an alarm if the sensitivity is above certain threshold

Introduction



Problem Definition

Denote the target DNN by $f(X) : X \rightarrow C$, where X is the set of input samples and C is the set of output labels.

Given a sample x, denote its true label (obtained by human observer) by c_x.
Say that a sample x is a normal sample if $f(x) = c_x$ and is an adversarial sample if $f(x) \neq c$.

Notice that according to our definition, a sample in the training data which is wrongly-labeled is also an adversarial sample.

Given an arbitrary sample x and the DNN f, how can we effectively detect whether x is a normal sample or an adversarial sample?
Can we provide some confidence for the detection result as well?

Mutation Testing

Given an arbitrary sample x for a DNN, we obtain a set of mutations $X_m(x)$, each of which $x_m = x + \epsilon$ is obtained by imposing a minor random perturbation ϵ on x that is label-preserving.

We remark that we restrict the perturbations in domain specific ways to guarantee that the mutations are realistic.

For every mutation $x_m \in X_m(x)$ we obtain its output label $f(x_m)$ by feeding x_m into the DNN.

For most of the mutations in $Xm(x), f(xm)$ should be $f(x)$ because we are imposing a small, realistic and thus hopefully label-preserving perturbation on x.

Mutation Testing

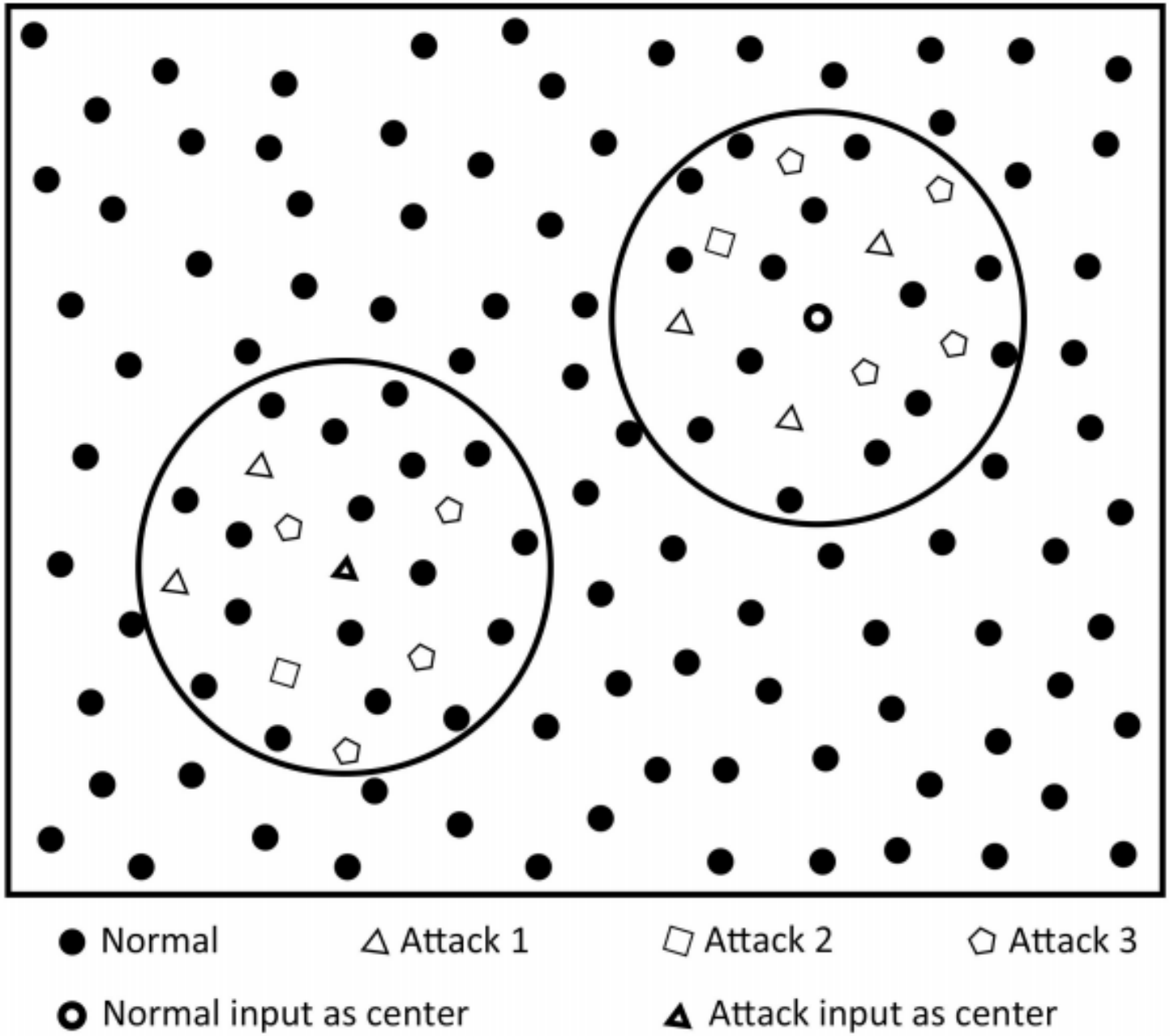
Calculate the sensitivity of a sample x to perturbations as follows:

$$\kappa(x) = \frac{|\{x_m | x_m \in X_m(x) \wedge f(x_m) \neq f(x)\}|}{|X_m(x)|}$$

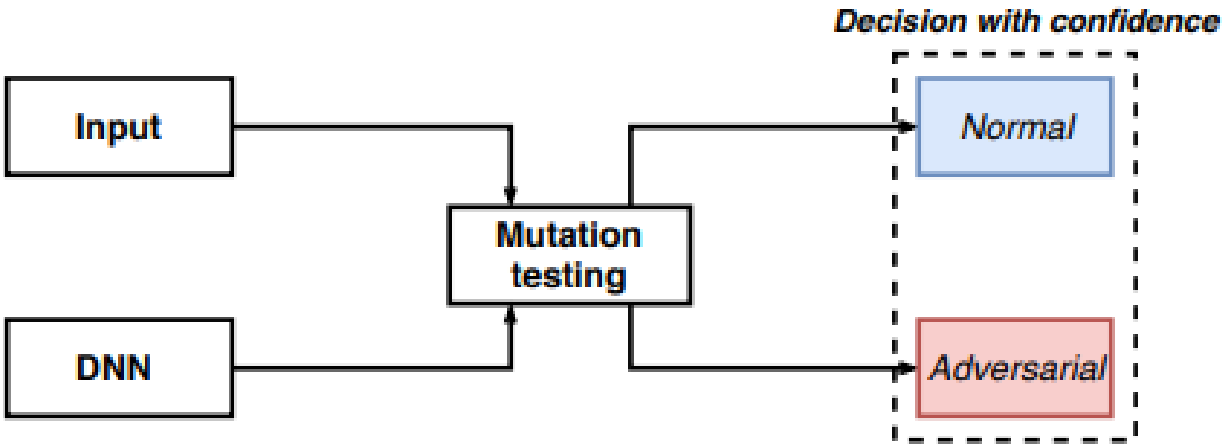
Their observation is that κ_{adv} is significantly larger than κ_{nor}.

Table 1: The confidence interval (99% significance level) of κ_{nor} and κ_{adv} of 500 images randomly drawn from MNIST and CIFAR10 dataset with 1000 mutations each under different attacks.

Dataset	*StepSize*	κ_{nor}	κ_{adv}				
			FGSM	C&W	JSMA	Black-box	wrongly-labeled
MNIST	1	0.13 % ± 0.04 %	4.86 % ± 0.61 %	10.41 % ± 1.39 %	6.85 % ± 0.58 %	5.98 % ± 2.78 %	3.68 % ± 0.65 %
	5	3.00 % ± 0.16 %	14.82 % ± 0.86 %	20.60 % ± 1.41 %	15.91 % ± 0.93 %	13.2 % ± 2.4 %	10.32 % ± 0.98 %
	10	6.31 % ± 0.21 %	19.34 % ± 1.07 %	27.07 % ± 1.51 %	21.55 % ± 1.11 %	17.05 % ± 2.36 %	14.47 % ± 1.15 %
CIFAR10	1	20.91 % ± 3.27 %	61.95 % ± 5.87 %	54.06 % ± 3.45 %	69.27 % ± 2.64 %	62.00 % ± 7.03 %	42.27 % ± 3.12 %
	5	24.30 % ± 3.04 %	63.38 % ± 5.24 %	56.91 % ± 3.13 %	67.84 % ± 2.86 %	63.55 % ± 6.29 %	47.60 % ± 2.94 %
	10	28.30 % ± 2.87 %	64.50 % ± 4.86 %	59.30 % ± 2.87 %	66.15 % ± 3.16 %	64.92 % ± 5.72 %	48.23 % ± 2.56 %


Normal
Attack 1
Attack 2
Attack 3
Normal input as center
Attack input as center

Adversary detection



Adversary detection

Algorithm 1: $DetectAdv(x, f, \kappa_1, \mu, \alpha, \beta, \sigma, StepSize)$

Let $stop = false$;
Let $c = 0$ be the count of mutations that satisfy $f(x_m) \neq f(x)$;
Let $n = 0$ be the count of total mutations generated so far;
while *!stop* **do**
    Randomly generate a mutation x_m of x with step size $StepSize$;
    $n = n + 1$;
    if $f(x_m) \neq f(x)$ **then**
        $c = c + 1$;
        Calculate the SPRT probability ratio as pr;
        if $pr \geq \frac{1-\beta}{\alpha}$ **then**
            Accept the hypothesis that $\kappa(x) > \mu \cdot \kappa_1$ and report the input as an adversarial input with error bounded by β;
            return;
        if $pr \leq \frac{\beta}{1-\alpha}$ **then**
            Accept the hypothesis that $\kappa(x) \leq \mu \cdot \kappa_1$ and report the input as a normal input with error bounded by α;
            return;

Adversary detection

If the hypothesis is accepted, it means that the sample has a higher sensitivity than a normal sample. Thus, they report that the input x is an adversarial sample with error bounded by β.

If the hypothesis is rejected, they report that the input is a normal sample with error bounded by α.

They apply the Sequential Probability Ratio Test (SPRT) to control the sampling procedure:

$$pr = \frac{p_1^c(1-p_1)^{n-c}}{p_0^c(1-p_0)^{n-c}}$$

where with $p_1 = \mu.\kappa_1 + \sigma$ and $p_0 = \mu.\kappa_1 - \sigma$

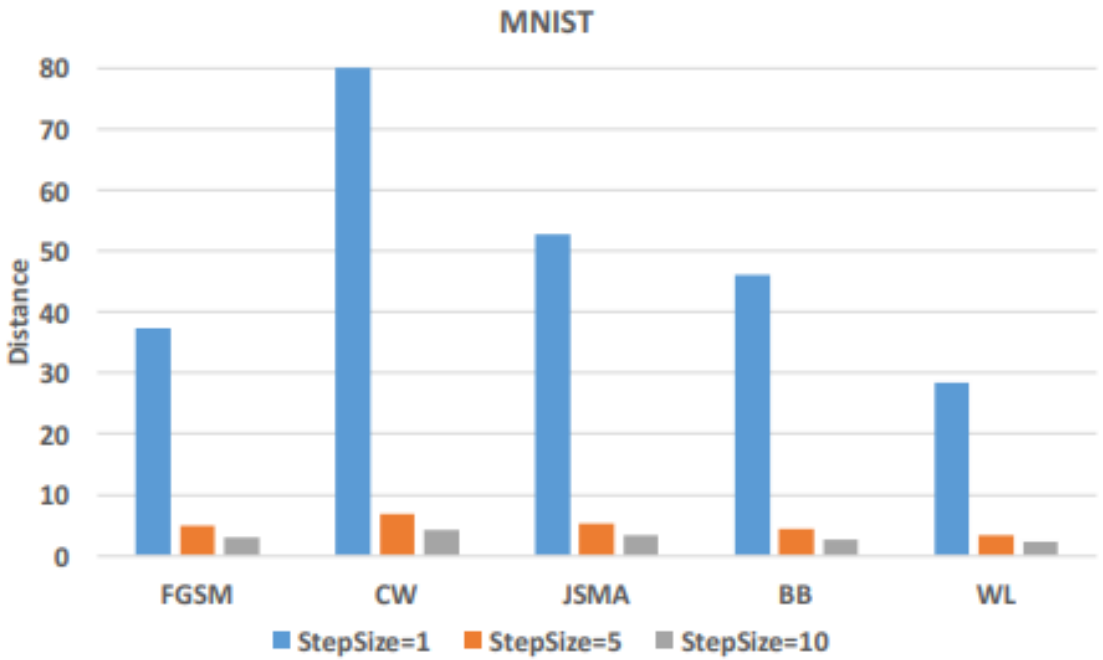
Experiments

There are several goals they want to achieve through the experiments.

- We aim to evaluate our hypothesis, i.e., there is a significant difference between κ_{nor} and κ_{adv} (for different attacks).
- we aim to show that our detection algorithm is able to detect adversarial samples effectively and efficiently.
- We also aim to provide some practical guidance on the choice of parameters.
- we aim to show that our algorithm improves sample labeling in general.

They adopt the MNIST and CIFAR10 datasets in experiments.

Experiments



Experiments