Detecting Adversarial Examples via Neural Fingerprinting

Sumanth Dathathri [*1] **Stephan Zheng** [*1] **Richard M. Murray** [1] **Yisong Yue** [1]

Abstract

Deep neural networks are vulnerable to adversarial examples, which dramatically alter model output using small input changes. We propose *NeuralFingerprinting*, a simple, yet effective method to detect adversarial examples by verifying whether model behavior is consistent with a set of *secret fingerprints*, inspired by the use of biometric and cryptographic signatures. The benefits of our method are that 1) it is fast, 2) it is prohibitively expensive for an attacker to reverse-engineer which fingerprints were used, and 3) it does not assume knowledge of the adversary. In this work, we pose a formal framework to analyze fingerprints under various threat models, and characterize *NeuralFingerprinting* for linear models. For complex neural networks, we empirically demonstrate that *NeuralFingerprinting* significantly improves on state-of-the-art detection mechanisms by detecting the strongest known adversarial attacks with *98-100%* AUC-ROC scores on the MNIST, CIFAR-10 and MiniImagenet (20 classes) datasets. In particular, the detection accuracy of *NeuralFingerprinting* generalizes well to unseen test-data under various black- and white-box threat models, and is robust over a wide range of hyperparameters and choices of fingerprints.

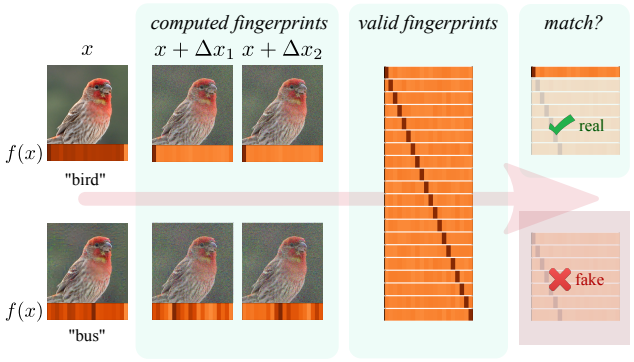


Figure 1. Detecting adversarial examples using *NeuralFP* with $N = 2$ fingerprints, for K-class classification. *NeuralFP* separates real data x (top) from adversarial examples $x' = x + \eta$ (bottom) by 1) adding N perturbations Δx^i, $(i = 1, \ldots, N)$ to each input, and 2) comparing model features $\varphi(x + \Delta x^i)$ with K sets of *valid fingerprints* $\varphi(x) + \Delta y^{i,j}$, $(j = 1 \ldots K)$. If a match is found, it classifies the input as "real" (x), and if not, flags it "fake" (x').

1. Introduction

Deep neural networks are highly effective pattern-recognition models for many applications, such as computer vision, speech recognition and sequential decision-making. However, neural networks are vulnerable to adversarial examples: an attacker can add small perturbations to input data, that maximally change the model's output (Szegedy et al., 2013; Goodfellow et al., 2014). Hence, a key challenge is how to make neural networks reliable and robust for large-scale applications in noisy environments or mission-critical applications, such as autonomous vehicles. To make neural networks robust against adversarial examples, we propose *NeuralFingerprinting* (*NeuralFP*): a fast, secure and effective method to detect adversarial examples.

The key intuition for *NeuralFP* is that we can encode *secret* fingerprint patterns into the behavior of a neural network around the input data. This pattern characterizes the network's expected behavior around real data and can thus be used to reject fake data, where the model outputs are not consistent with the expected fingerprint outputs. This process is shown in Figure 1.

This approach is highly effective as encoding fingerprints is feasible and simple to implement during training, and evaluating fingerprints is computationally cheap. *NeuralFingerprinting* is secure: to craft a successful adversarial example, an attacker would have to find input perturbations that significantly change the network's output, *and do not violate the secret fingerprints*. However, it is computationally and statistically expensive for an attacker to reverse-engineer the secret fingerprints used, and random attacks have low probability of success. Furthermore, *NeuralFingerprinting* does not require knowledge of the adversary's attack method, and differs from state-of-the-art methods (Meng & Chen, 2017; Xingjun Ma, 2018) that detect adversarial examples using

[*]Equal contribution [1]Caltech, Pasadena, CA. Correspondence to: Sumanth Dathathri <sdathath@caltech.edu>, Stephan Zheng <stephan@caltech.edu>.

auxiliary classifiers.

In this work, we theoretically characterize the feasibility and security of *NeuralFP*, and experimentally validate that *NeuralFP* achieves almost perfect detection AUC scores against state-of-the-art adversarial attacks on various datasets. To summarize, our key contributions are:

- We present *NeuralFP*: a simple and secure method to detect adversarial examples that does not rely on knowledge of the attack mechanism.
- We describe a formal framework to characterize the hardness of reverse-engineering fingerprint patterns and characterize the effectiveness of *NeuralFP* for linear classification.
- We empirically demonstrate that *NeuralFP* achieves state-of-the-art near-perfect AUC-scores against the strongest known adversarial attacks. In particular, we show that *NeuralFP* correctly distinguishes between unseen test data and adversarial examples.
- We emprically show that the performance of *NeuralFP* is robust to the choice of fingerprints and is effective for a wide range of choices of fingerprints and hyperparameters.
- We also show that *NeuralFP* can be robust even in the whitebox-attack setting, where an adaptive attacker has knowledge of the fingerprint data.

Source code is available at `https://github.com/StephanZheng/neural-fingerprinting`.

2. Fingerprinting for Adversarial Defense

We consider supervised classification, where we aim to learn a model $f(x;\theta)$ from labeled data $\{(x^i, y^{*i})\}_{i\in\mathcal{I}}$, where $x \in \mathbb{R}^d$ and y is a 1-hot label vector $y \in \{0,1\}^K$ (K classes). The model f predicts class probabilities $P(y|x;\theta)$:

$$f(x;\theta)_j = \frac{\exp h(x)_j}{\sum_l \exp h(x)_l}, \tag{1}$$

with logits $h(x) \in \mathbb{R}^K$ and can be learned via a loss function $L(x, y; \theta)$, e.g. cross-entropy loss. Formally, this data is generated by sampling from a *data-generating distribution* $p(x, y)$, which characterizes "real" data ($p(x, y) > 0$).

Adversarial attacks produce perturbations that exploit the behavior of a neural network at an input point x, in particular when x is a high-dimensional vector. An adversarial perturbation Δx causes a large change in model output, i.e. for $\delta, \epsilon > 0$:

$$\|\Delta x\| < \delta, \quad \|f(x + \Delta x) - f(x)\| > \epsilon, \tag{2}$$

Algorithm 1 *NeuralFP*

Input: example x, comparison function D (see Eqn 9).
Input: threshold $\tau > 0$.
Input: (secret) $\{(\Delta x^i, \Delta y^{i,j})\}_{i=1\dots N, j=1\dots K}$.
Output: `accept` / `reject`.
if $\exists j : D(x, f, \xi^{i,j}) \leq \tau$ **then**
  Return: `accept` # *x is real*
else
  Return: `reject` # *x is fake*
end if

such that e.g. the class predicted by the model changes:

$$k' = \underset{j}{\text{argmax}} f(x + \Delta x)_j \neq \underset{j}{\text{argmax}} f(x)_j = k. \tag{3}$$

We define such data (x') to be "fake", i.e. formally

$$p(x', \underset{j}{\text{argmax}} f(x';\theta)_j) = 0. \tag{4}$$

For instance, the Fast-Gradient Sign Method (Goodfellow et al., 2014) perturbs x to $x + \Delta x$ along the gradient

$$\Delta x = \epsilon \cdot \text{sign}\, \frac{\partial L(x, y; \theta)}{\partial x}, \quad \epsilon > 0. \tag{5}$$

Neural Fingerprinting. Our goal is to defend neural networks by robustly detecting adversarial examples. Specifically, we introduce *NeuralFP*: a method that detects whether an input-output pair $(x, \hat{y})$ is consistent with the data distribution ("real"), or is adversarial ("fake"). This algorithm is summarized in Algorithm 1 and Figure 1.

The key idea of *NeuralFP* is to detect adversarial inputs by checking if the network output in *specific points* around x closely resembles a set of *fingerprints* that can be *chosen by the defender*. These chosen outputs are embedded into the network during training. Formally, a fingerprint ξ is:

$$\xi = (\Delta x, \Delta y). \tag{6}$$

For K-class classifcation, we define a set of fingerprints:

$$\xi^{i,j} = (\Delta x^i, \Delta y^{i,j}), \quad i = 1, \dots, N, j = 1, \dots, K, \tag{7}$$

where $\xi^{i,j}$ is the i^{th} fingerprint for class j. Here, the Δx^i ($\Delta y^{i,j}$) are input (output) perturbations that are chosen by the defender. Note that across classes $j = 1, 2 \dots, K$, we use the same input directions Δx^i.

NeuralFP acts as follows: it classifies a new input x' as *real* if the change in model output $F(x' + \Delta x^i)$ is close to the $\Delta y^{i,j}$ *for some class* j, for all i. Here, we use a comparison function D and threshold $\tau > 0$ to define the level of agreement required, i.e. we declare x' *real* when

$$x' \text{ is real} \Leftrightarrow \exists j : D(x, f, \xi^{\cdot,j}) \leq \tau. \tag{8}$$

As the comparison function D, we take:

$$D(x, f, \xi^{\cdot,j}) = \frac{1}{N}\sum_{i=1}^{N} \|F(x, \Delta x^i) - \Delta y^{i,j}\|_2 \quad (9)$$

$$F(x, \Delta x^i) = \varphi(x + \Delta x^i) - \varphi(x), \quad (10)$$

$$\varphi(x) = \frac{h(x)}{\|h(x)\|}, \quad (11)$$

where $\varphi(x)$ are normalized logits. Hence, *NeuralFP* is defined by the data:

$$\texttt{NFP} = (\xi, D, \tau)\,. \quad (12)$$

Encoding Fingerprints. Once a defender has constructed a set of desired fingerprints (7), the chosen fingerprints can be embedded in the network's response by adding a fingerprint regression loss during training. We elaborate on how to choose fingerprints hereafter. Given a classification model (1) with logits $h(x) \in \mathbb{R}^K$, the fingerprint loss is:

$$L_{\text{fp}}(x, y, \xi; \theta) = \frac{1}{N}\sum_{i=1}^{N} \|F(x, \Delta x^i) - \Delta y^{i,k}\|_2^2, \quad (13)$$

where k is the ground truth class for example x and $\Delta y^{i,k}$ are the fingerprint outputs. Note that we *only train on the fingerprints for the ground truth class*. The total training objective then is:

$$\min_{\theta} \sum_{(x,y)} L_0(x, y; \theta) + \alpha L_{\text{fp}}(x, y, \xi; \theta), \quad (14)$$

where L_0 is a loss function for the task (e.g. cross-entropy loss for classification) and α a positive scalar. In the hereafter we will use $\alpha = 1$, but in practice, we choose α such that it balances the task and fingerprint losses.

Computational complexity. Evaluating fingerprints requires $O(NK)$ extra computation, as Algorithm 1 requires $O(NK)$ extra forward passes to compute the differences $F(x, \Delta x^i)$. A straightforward implementation is to check (8) iteratively for all classes, and stop whenever an agreement is seen or all classes have been exhausted. However, this operation can in principle be parallelized and performed in minibatches for real-time applications.

2.1. Threat Model Analysis

There are several levels of threat models with increasing levels of attacker knowledge of $\texttt{NFP} = (\eta, D, \tau)$ and the model $f(x; \theta)$ for which we can analyze the security and effectiveness of *NeuralFP*. First, we will assume that the attacker has access to the parameters θ and can query the model $f(x; \theta)$ and its derivatives without limitation.

We can then characterize the security and feasibility of *NeuralFP* in both the **whitebox-attack** (attacker has perfect knowledge of NFP) and **blackbox-attack** setting (attacker knows nothing about NFP but **is aware** of the model weights). For example, a physical instance of a whitebox-attack occurs when the attacker has access to the compute instance where the forward passes $f(x; \theta)$ are executed (e.g. the model $f(x; \theta)$ is queried on a local device). In this case, it should be assumed that the attacker can get access to the precise fingerprints that are used by the defender, e.g. by reading the raw memory state. A blackbox-attack might occur when the internal state of the compute instance is shielded, e.g. when $f(x; \theta)$ is queried in the cloud.

Additionally, we define the notion of **whitebox-defense** (the defender has knowledge of the attacker's strategy) and **blackbox-defense** (defender has no knowledge about attacker).

Whitebox-attack. If the attacker has full knowledge of the fingerprint data NFP, a key question is how vulnerable the model is, i.e. how many example x'-s can an attacker find that will be falsely flagged as "real"?

Firstly, we will demonstrate in Theorem 1 that for Support Vector Machines (binary classification with linear models), we can characterize the region of inputs that will be classified as "real" for a given set of fingerprints $\xi^{i.j}$ and to what extent this corresponds with the support of the data distribution $p(x, y)$.

Secondly, a whitebox-attacker could use knowledge of NFP to adaptively construct adversarial examples using gradient-based methods. However, we will show empirically in Section 3.4 that such methods have low odds of success.

Blackbox-attack. In the blackbox-attack setting, the attacker has no knowledge of NFP, i.e. does not know the metric D, threshold τ, and (the number NK of) fingerprints $\xi^{i.j} = \left(\Delta x^i, \Delta y^{i.j}\right)$. An attacker would 1) have to find the fingerprints used by the defender and then 2) construct adversarial examples that are compatible with the fingerprints.

First, to reverse-engineer the fingerprints ξ, it is intuitive to see this is combinatorially and statistically hard. Consider a simple case where only the $\Delta y^{i.j}$ are unknown, and that the attacker knows that each fingerprint is discrete, i.e. each component $\Delta y_k^{i.j} = \pm 1$. Then the attacker would have to search over combinatorially ($\mathcal{O}(2^{NK})$) many Δy to find the subset of Δy that satisfy the detection criterion in equation (8). A defender can shrink this space of feasible fingerprints by setting the threshold level of τ. Hence, we conjecture that reverse-engineering the set of feasible fingerprints for a general model class $f(x; \theta)$ is exponentially hard, but leave a full proof for future research.

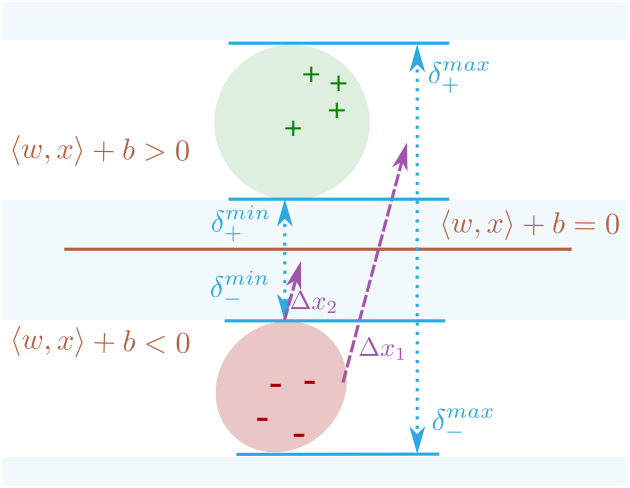


Figure 2. Geometry of fingerprints for support vector machines with binary labeled data that are separable by the hyperplane $\langle w, x\rangle + b = 0$. The distances $\delta_\pm^{max}$ ($\delta_\pm^{min}$) denote the maximal (minimal) distances of the positive (x_+) and negative (x_-) examples to the hyperplane. The fingerprint Δx^1 with $\langle \Delta x^1, e\rangle = \delta_-^{min}$ will have $f(x_- + \Delta x) < 0$ and $f(x_-) < 0$ for all x_- in the data distribution (red region), i.e. always never a change in prediction (except for support vectors). Hence, Δx^1 will flag all x' in the shaded blue regions $-\delta_-^{min} < \langle x', e\rangle < 0$ as "fake", since for those x' it will *always* see a change in predicted class. Similarly, Δx^2 with $\langle \Delta x^2, e\rangle = \delta_-^{max}$ always sees a class change for real x_-, and thus flags all x' with $\langle x', e\rangle < -\delta_-^{max}$ as "fake".

Secondly, a successful attacker needs to construct $x' = x + \Delta x$ that maximizes the loss:

$$\max_{x'} L(x', y; \theta), \quad \text{such that } \exists j : D(x', f, \xi^{\cdot,j}) < \tau, \quad (15)$$

ensuring that the model outputs $\{f(x' + \Delta x^i)\}$ are τ-close to the fingerprints $\{\Delta y^{i,j}\}$ for some class j. This poses a difficult constrained optimization problem, where the feasible set of x' is non-convex. Hence, in general it is hard for a random attacker to succeed, i.e. find a feasible solution.

In particular, the constrained optimization (15) shows intuitively that a random attack would likely fail, as the feasible set of x' can be made reasonably small for small thresholds τ (we show in 3.1 (e.g. Figure 5) small thresholds do not significantly impact the robustness of *NeuralFP*).

2.2. Characterizing Fingerprints

We now investigate the effect of fingerprints and to what extent fingerprints characterize data distributions. In particular, given fingerprint data `NFP` as in (12), can we characterize which inputs x will be classified as "real", how that depends on the number N of fingerprints and how efficient fingerprints are?

Support Vector Machines. To do so, we first consider fingerprints for Support Vector Machines, i.e. binary classification with linear models f:

$$f(x) = \langle w, x\rangle + b, \quad \hat{y} = \text{sign } f(x), \quad (16)$$

on inputs $x \in \mathbb{R}^n$, where $n \gg 1$ (e.g. $n = 900$ for MNIST): The binary classifier defines a hyperplane $f(x) = 0$, which aims to separate positive ($\hat{y} = +1$) from negative examples ($\hat{y} = -1$). We will assume that the data-generating distribution $p(x)$ for the positive and negative examples is perfectly separated by a hyperplane, defined by the normal $e = \frac{w}{\|w\|}$, $\|e\| = 1$. We define the minimal and maximal distance from the examples to the hyperplane along e as:

$$\delta_\pm^{min} = \min_{a:y^a=\pm 1} |\langle x^a, e\rangle|, \; \delta_\pm^{max} = \max_{a:y^a=\pm 1} |\langle x^a, e\rangle| \quad (17)$$

In this setting, the set of x classified as "real" by fingerprints is determined by the geometry of $f(x)$, where for detection we measure the *exact change* in predicted class (e.g. we use $h(x) = \text{sign } (\langle w, x\rangle + b)$ and $\tau = 0$). Theorem 1 below then characterizes fingerprints for SVMs:

Theorem 1 (Decision Boundaries of Fingerprints for Support Vector Machines)**.** *Assume the metric D is as in equations (9-11), e.g. measure first-order differences, $\tau = 0$, and use $h(x) = sign\ (\langle w, x\rangle + b)$. Let $e = \frac{w}{\|w\|}$. For support vector machines with separable data, the set of fingerprints*

$$(\Delta x^1 = \lambda_1 e, \Delta y^{1,-} = 0), \; \lambda_1 = \delta_-^{min} \quad (18)$$
$$(\Delta x^2 = \lambda_2 e, \Delta y^{2,-} = +2), \; \lambda_2 = \delta_-^{max} \quad (19)$$
$$(\Delta x^3 = \lambda_3 e, \Delta y^{3,+} = -2), \; \lambda_3 = -\delta_+^{max} \quad (20)$$
$$(\Delta x^4 = \lambda_4 e, \Delta y^{4,+} = 0), \; \lambda_4 = -\delta_+^{min} \quad (21)$$

will detect adversarial perturbations $x' = x_\pm + \eta$ as "fake" for which one the following hold:

$$\begin{cases} \langle x', e\rangle > \delta_+^{max}, & 0 < \langle x', e\rangle < \delta_+^{min}, \\ \langle x', e\rangle < -\delta_-^{max}, & -\delta_-^{min} < \langle x', e\rangle < 0, \end{cases} \quad (22)$$

This choice of λs is optimal: these fingerprints minimize the set of inputs x that can be misclassified as "real" using a first-order metric D.

Proof. We illustrate this proof for two fingerprints in Figure 2. Consider any perturbation $\eta = \lambda e$ that is positively aligned with w, and has $\langle \eta, e\rangle = \delta_-^{min}$. Then for any negative example $(x_-, -1)$ (except for the support vectors that lie exactly δ_-^{min} from the hyperplane), adding the perturbation η does not change the class prediction:

$$\text{sign } f(x_-) = -1, \quad \text{sign } f(x_- - \eta) = -1. \quad (23)$$

The fingerprint in (18) is an example of such an η. However, if λ is large enough, that is:

$$\langle \eta, e \rangle = \delta_{-}^{max}, \tag{24}$$

(e.g. the fingerprint in (19)), for *all* negative examples $(x_-, -1)$ the class prediction will *always* change (except for the x_- that lie exactly δ_{-}^{max} from the hyperplane):

$$\text{sign } f(x_-) = -1, \quad \text{sign } f(x_- + \eta) = +1, \tag{25}$$

Note that if η has a component smaller (or larger) than $\delta_{\pm}^{min}$, it will exclude *fewer* (more) examples, e.g. those that lie closer to (farther from) the hyperplane. Similar observations hold for fingerprints (20) and (21) and the positive examples x_+. Hence, it follows that for any x that lies too close to the hyperplane (closer than $\delta_{\pm}^{min}$), or too far (farther than $\delta_{\pm}^{max}$), the model output after adding the four fingerprints will never perfectly correspond to their behavior on examples x from the data distribution. For instance, for any x that is closer than δ_{+}^{min} to the hyperplane, (21) will always cause a change in class, while none was expected. Similar observations hold for the other regions in (22). Since the SVM is translation invariant parallel to the hyperplane, the fingerprints can only distinguish examples based on their distance perpendicular to the hyperplane. Hence, this choice of λs is optimal. □

Note that Theorem 1 by itself *does not* prevent attacks parallel to the decision boundary; an adversary could in principle add a (large) perturbation η that pushes a negative example x_- across the SVM decision boundary to a region where the data distribution $p(x_- + \eta, y) = 0$, but which is classified as positive by the SVM, i.e. sign $f(x_- + \eta) = +1$. However, we can further prevent attacks that stray far from the data distribution, e.g. that go too far in a direction parallel to the hyperplane, by checking the distance to the nearest example in the dataset. This essentially would restrict the adversary to perturbations of limited magnitude, i.e. $\|\eta\| < \epsilon$, although could be computationally expensive.

Imperfect Classifiers. In addition, Theorem 1 assumed that the data was perfectly separable. However, if this is not the case and there are misclassified examples $\tilde{x}$, when using exact matching, the fingerprints would not detect adversarial perturbations correctly *for misclassified examples* and flag $\tilde{x}$ itself as fake. However, we observed empirically that this problem is ameliorated when using *soft matching* and nonlinear models, although theoretically characterizing this is an interesting question for future research.

Nonlinear Models. For the general setting, e.g. for nonlinear models f, Theorem 1 can be extended if the data is (locally) separable in some feature space Φ and we can write a general (local) model as

$$y = \langle w, \phi(x) \rangle + b. \tag{26}$$

In this case, the fingerprints can be defined analogous to (20-19). As such, it is straightforward to lift the analysis of Theorem 1 to this setting as well. However, depending on the feature space Φ chosen, such as analysis might only be applicable to a *local* region of the input space and require more complex fingerprints.

2.3. Choosing and Encoding Fingerprints.

When applying *NeuralFP* to complex models such as deep neural networks, a key challenge is how to choose the fingerprints ξ such that the region of inputs x that are classified as "real" corresponds as much as possible with the support of the data distribution $p(x)$.

Theorem 1 characterizes fingerprints for SVMs, where for detection we used the exact change in predicted class. However, in practice using exact class changes is challenging. First, the feature space Φ can be highly complex (e.g. for deep neural networks) and hence intractable to describe the geometry of the decision boundaries of $f(x; \theta)$. However, *NeuralFP* utilizes a *softer* notion of fingerprint matching by checking whether the model outputs match (a pattern of) *changes in logits* as in equation (10), which can be *learned* by the model during training.

In this work, we focus on deep neural networks that have high model capacity, such that fitting almost arbitrary fingerprint patterns accurately is a feasible goal[1].

Fingerprints Sampling Strategies. In particular, this motivates a general and straightforward method to construct the N fingerprints $\xi^{i,j} = \left(\Delta x^i, \Delta y^{i,j}\right)$, e.g. by randomly sampling directions Δx^i, and encouraging constancy and/or changes to another class (e.g. (20-19) for positive examples in the SVM case). For example, a simple choice of fingerprints that increases class probabilities along Δx^i uses 1-hot vectors:

$$\Delta x^i \sim \mathcal{N}(0, \epsilon), \;\; \Delta y^{i,j} = \mathbf{1}\left[j' = j\right], \;\; \epsilon > 0, \tag{27}$$

although more complex choices are feasible as well. We will show in Section 3.3 that the performance of *NeuralFP* is robust over a wide range of fingerprints ξ.

3. Evaluating the detection of adversarial attacks

We now empirically validate the effectiveness of *NeuralFP*, as well as analyze the behavior and robustness of *NeuralFP*[2]. Our goal is to answer the following questions:

[1] In fact, (Zhang et al., 2016) has shown that neural networks can fit arbitrary random or complex patterns of prediction labels.

[2] Code for experiments: https://github.com/StephanZheng/neural-fingerprinting

Data	Method	Defense type	FGM	JSMA	BIM-a	BIM-b	CW-L_2
MNIST	LID	Blackbox	99.68	96.36	99.05	99.72	98.66
	LID	Whitebox	99.68	98.67	99.61	99.90	99.55
	NeuralFP	Blackbox	**100.0**	**99.97**	**99.94**	**99.98**	**99.74**
CIFAR-10	LID	Blackbox	82.38	89.93	82.51	91.61	93.32
	LID	Whitebox	82.38	95.87	82.30	99.78	**98.94**
	NeuralFP	Blackbox	**99.96**	**99.91**	**99.91**	**99.95**	98.87
MiniImagenet-20	*NeuralFP*	Blackbox	99.96	-	-	99.68	-

Table 1. Detection AUC-ROC against *blackbox-attackers* (know model $f(x;\theta)$, but not fingerprints; see Section 2.1) for MNIST, CIFAR-10 and MiniImagenet-20 tasks on test-set ("negatives") and corresponding adversarial ("positives") samples (1328 *pre-test* samples each). *NeuralFP* outperforms LID on MNIST and CIFAR-10 on almost all attacks, except for the CW-L_2 attack. However, LID only outperforms *NeuralFP* slightly in the whitebox setting, i.e. with data augmentation with CW-L_2 adversarial examples, while *NeuralFP* is always attack agnostic. On MiniImagenet-20, *NeuralFP* achieves near-perfect AUC-ROC versus FGM and BIM-b. Not all scores for MiniImagenet-20 are reported: 1) attacks are prohibitively slow (JSMA, CW-L_2) and 2) due to time constraints, LID scores were not obtainable.

- How well does *NeuralFP* distinguish between normal and adversarial examples?
- How sensitive and robust is *NeuralFP* to changes in hyperparameters?
- How do the fingerprints of normal and adversarial examples differ?
- How robust is *NeuralFP* to a attacker that has full knowledge of fingerprints?
- Does *NeuralFP* scale to high-dimensional inputs?

To do so, we report the AUC-ROC performance of *NeuralFP* on MNIST, CIFAR-10 and MiniImagenet-20 datasets against four state-of-the art adversarial attacks, using (Xingjun Ma, 2018) (LID), the strongest-known detection based defense as the baseline. We compare against LID in both the blackbox-defense and whitebox-defense setting (LID has knowledge of the attack mechanism and the LID classifier has been trained on FGM examples).

Adversarial Attacks. We test on the following attacks:

- *Fast Gradient Method* (FGM) (Goodfellow et al., 2014) and *Basic Iterative Method* (BIM) (Kurakin et al., 2016) are both gradient based attacks with BIM being an iterative variant of FGM. We consider both BIM-a (iterates until misclassification has been achieved) and BIM-b (iterates a fixed number of times (50)).
- *Jacobian-based Saliency Map Attack* (JSMA) (Papernot et al., 2015) iteratively perturbs two pixels at a time, based on a saliency map.
- *Carlini-Wagner Attack* (CW-L_2): an optimization-based attack, is one of the strongest known attacks (Carlini & Wagner, 2016), which has broken many state-of-the-art defenses. (Carlini & Wagner, 2017a).

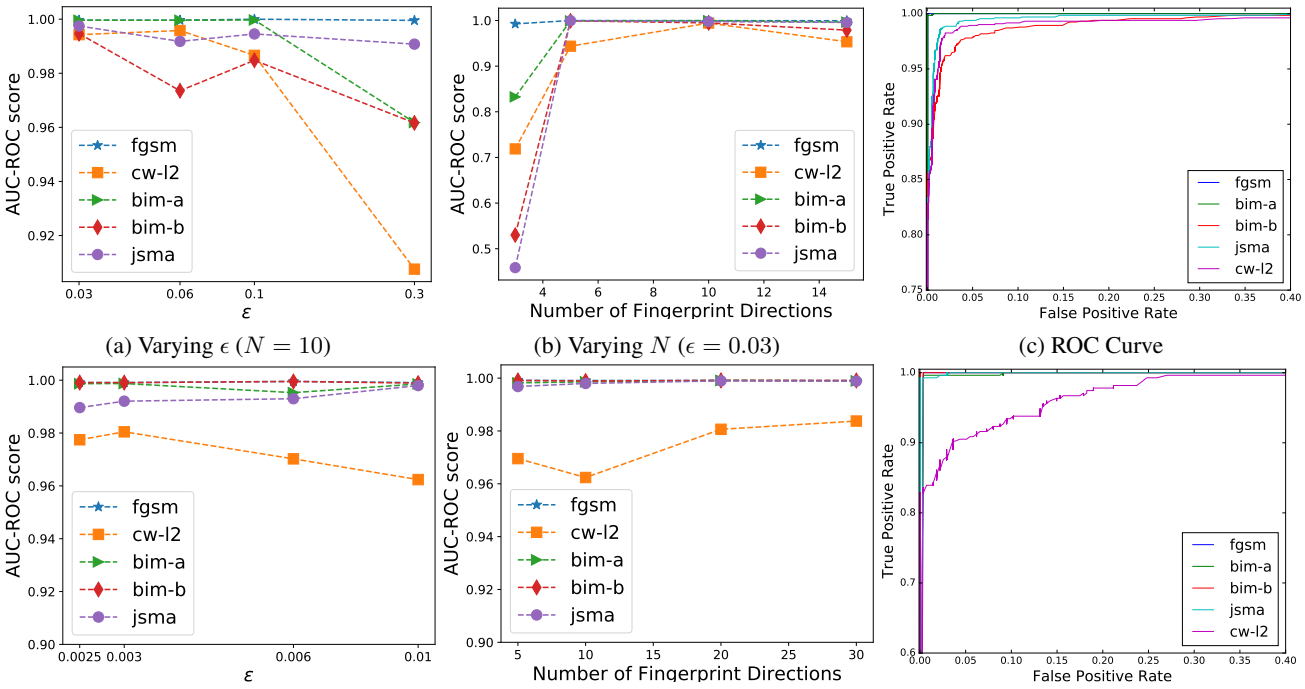
For each dataset, we consider a randomly sampled *pre-test*-set of 1328 test-set images, and discard misclassified pre-test images. For the *test-set* of remaining images, we generate adversarial perturbations by applying each of the above mentioned attacks. We report AUC-ROC on sets composed in equal measures of the *test-set* and *test-set* adversarial examples with varying threshold τ. See the appendix for details regarding all model architectures used and construction of MiniImagenet-20.

3.1. Effectiveness

MNIST We trained a 5-layer ConvNet to $99.3 \pm 0.1\%$ test-accuracy. The set of $\Delta x^i \in \mathbb{R}^{28\times 28}$ are chosen at random, with each pixel perturbation chosen uniformly in $[-\epsilon, \epsilon]$. For each Δx^i, if x is of label-class k, $\Delta y^{i,k} \in \mathbb{R}^{10}$ is chosen to be such that $\Delta y^{i,k}_{l\neq k} = -0.235$ and $\Delta y^{i,k}_{l=k} = 0.7$, with $\|\Delta y\|_2 = 1$. The AUC-ROC for the best-performing fingerprints (best N and ϵ) using grid-search is reported in Table 1. We see that *NeuralFP* achieves near-perfect detection with AUC-ROC of $99 - 100\%$ across all attacks.

CIFAR-10. For CIFAR-10, we trained a 7-layer ConvNet (similar to (Carlini & Wagner, 2016)) to $85 \pm 1\%$ accuracy. The Δx^i and $\Delta y^{i,j}$ are chosen similarly as for MNIST. Table 1 shows that in average, across attacks, *NeuralFP* outperforms LID-blackbox on average by **11.77%** and LID-whitebox defense by **8%**. Even in comparison with LID-whitebox, *NeuralFP* is competitive (CW-L_2) or outperforms LID (other attacks).

MiniImagenet-20. To illustrate the scalability of *NeuralFP*, we also evaluated on *MiniImagenet* (Vinyals et al., 2016) with 20 classes randomly chosen from the 100 MiniImageNet classes. For this, we trained an AlexNet network on 10,600 images (not downsampled) with 91.1% top-1 accuracy. We generated test-set adversarial examples using BIM-b with 50 steps (NIP) and FGM. Here,



(a) Varying ϵ ($N = 10$) (b) Varying N ($\epsilon = 0.03$) (c) ROC Curve

(d) Varying ϵ ($N = 10$) (e) Varying N ($\epsilon = 0.01$) (f) ROC Curve

Figure 3. AUC-ROC performance for different hyperparameter settings (left, middle) and ROC curves (right). Top row: MNIST, bottom row: CIFAR-10. We see that the performance of *NeuralFP* is robust across attacks and hyperparameters, with the AUC-ROC between $90 - 100\%$ for most settings. The AUC-ROC is lowest versus CW-L_2, which is the strongest known attack. Increasing N generally improves performance, indicating more fingerprints frustrate attackers more. Increasing ϵ decreases the AUC on CW-L_2 only, suggesting that as adversarial examples become of smaller magnitude, *NeuralFP* requires smaller detection ranges to be effective.

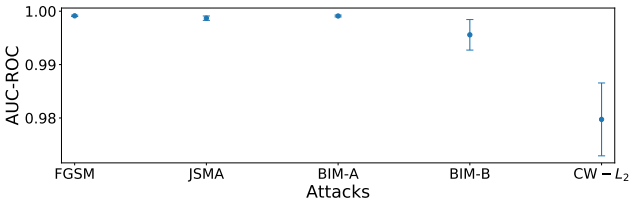


Figure 4. AUC-ROC mean μ and standard-deviation σ for 32 randomly sampled fingerprints (including randomizing N) for CIFAR-10. The AUC-ROC across all attacks varies little ($\sigma < 1\%$), with σ highest for CW-L_2, suggesting *NeuralFP* is robust across hyperparameters.

NeuralFP achieves an AUC-ROC score of $> 99.5\%$ with $(N, \epsilon) = (20, 0.05)$ on both attacks, similar to the near-perfect AUC-ROC for MNIST and CIFAR-10.

Visualization. In Figure 5 we visualize the fingerprint-loss for test and adversarial examples. The fingerprint-loss differs significantly for most test and adversarial examples, resulting in the AUC-ROC scores being close to 100%.

3.2. Sensitivity and Efficiency Analysis

Next, we study the effect of the hyper-parameters N (number of fingerprint directions) and ϵ on the AUC-ROC for MNIST and CIFAR-10. Figure 3 shows that *NeuralFP* performs well across a wide range of hyperparameters and is robust to variation in the hyperparameters. With increasing ϵ, the AUC-ROC for CW-L_2 decreases. A possible explanation is that CW-L_2 produces smaller adversarial perturbations than other attacks, and for larger ϵ, fingerprints are less sensitive to those small adversarial perturbations. However, the degradation in performance is not substantial ($\sim 4 - 8\%$) as we increase ϵ over an order of magnitude. With increasing N, we see that the AUC-ROC generally increases across attacks. We conjecture that a larger number of fingerprints are sensitive to perturbations in a larger number of directions, and hence result in better detection.

Figure 4 depicts the mean and standard deviation of AUC-ROC scores for 32 sets of randomly chosen fingerprints ξ and varying hyperparameters for CIFAR-10. The mean AUC-ROC for all attacks is $98\% - 100\%$, with the standard deviation being less than 1%. This indicates that the performance of *NeuralFP* is not very sensitive to the chosen fingerprint directions and shows that neural networks can

learn complex patterns along arbitrary directions.

Furthermore, the test accuracy with *NeuralFP* for CIFAR-10 is $85 \pm 1\%$ and for MNIST is in the range $99.3 \pm 0.1\%$. This is similar to the accuracy obtained when training the same models without fingerprints, illustrating that fingerprinting does not degrade prediction performance on the test-set, while the high AUC-ROC indicates that the fingerprints generalize well to the test-set, but not to adversarials.

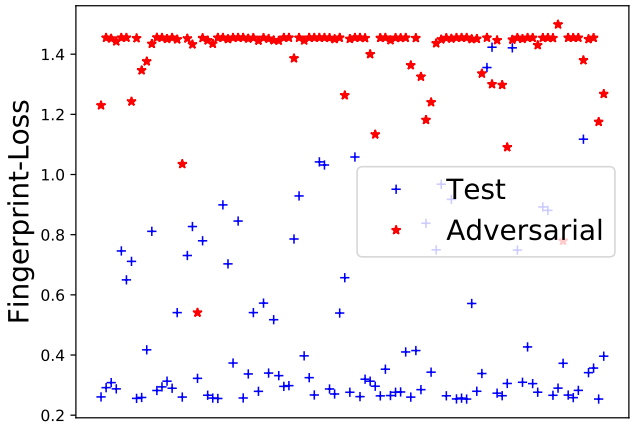


Figure 5. Fingerprint losses on 100 random test (blue) and adversarial (red) CIFAR-10 images. We see a clear separation in loss, illustrating that *NeuralFP* is effective across many thresholds τ.

3.3. Robustness to Choice of Fingerprints

Instead of simple $\Delta y^{i,j}$, we can encode more complex fingerprints. For instance, we trained a network on CIFAR-10 using random $\Delta y^{i,j}$: for each Δx^i, if x is in class k:

$$\Delta y^{i,k}_{l \neq k} = -0.235p, \Delta y^{i,k}_{l=k} = 0.7p. \quad (28)$$

Here p is a random variable with $Pr(p = 1) = 0.5$ and $Pr(p = -1) = 0.5$ that is resampled for each Δx^i, making it prohibitively hard for a brute-force attacker to guess. For this `NFP`, we achieve AUC-ROC of $> 95\%$ across attacks with $N = 30, \epsilon = 0.05$, without extensive tuning.

3.4. Robustness to Adaptive Whitebox-Attackers

We further considered an adaptive attacker that has knowledge of the predetermined fingerprints and model weights, similar to (Carlini & Wagner, 2017a). Here, the adaptive attacker (Adaptive-CW-L_2) tries to find an adversarial example x' that also minimizes the fingerprint-loss, attacking a CIFAR-10 model trained with *NeuralFP*. To this end, the CW-L_2 objective is modified as:

$$\min_{x'} \|x - x'\|_2 + \gamma \left(L_{\text{CW}}(x') + L_{\text{fp}}(x', y^*, \xi; \theta) \right) \quad (29)$$

Here, y^* is the label-vector, $\gamma \in [10^{-3}, 10^6]$ is a scalar found through a bisection search, L_{fp} is the fingerprint-loss we trained on and L_{CW} is an objective encouraging misclassification. Under this threat model, *NeuralFP* achieves an AUC-ROC of **98.79%** against Adaptive-CW-L_2, with $N = 30$ and $\epsilon = 0.006$ for a set of unseen test-samples (1024 *pre-test*) and the corresponding adversarial examples. In contrast to other defenses that are vulnerable to Adaptive-CW-L_2 (Carlini & Wagner, 2017a), we find that *NeuralFP* is robust even under this whitebox-attack threat model.

4. Related Work

Several forms of defense to adversarial examples have been proposed, including adversarial training, detection and reconstructing images using adversarial networks (Meng & Chen, 2017). However, (Carlini & Wagner, 2017a;b) showed many defenses are still vulnerable. (Madry et al., 2017) employs robust-optimization techniques to minimize the maximal loss the adversary can achieve through first-order attacks. (Raghunathan et al., 2018; Kolter & Wong, 2017) train on convex relaxations of the network to maximize robustness. Although these works are complementary to *NeuralFP*, they do not scale very well. Several other recent defenses attempt to make robust predictions based by relying on randomization (Cihang Xie, 2018), introducing non-linearity that is not differentiable (Jacob Buckman, 2018) and by relying on Generative Adversarial Networks (Yang Song, 2018; Pouya Samangouei, 2018) for denoising images. Instead, we focus on *detecting* adversarial attacks.

Detection. (Xingjun Ma, 2018) detect adversarial samples using an auxiliary logistic regression classifier, which is trained to use an expansion-based measure, *local intrinsic dimensionality (LID)*. A similar approach to detection is based on Kernel Density (KD) and Bayesian-Uncertainty (BU) using artifacts from pre-trained networks (Feinman et al., 2017). In contrast with these methods, *NeuralFP* encodes information into the network response during training, and does not depend on auxiliary detectors.

5. Discussion and Future Work

Our experiments suggest that *NeuralFP* is an effective method for safe-guarding against the strongest known state-of-the-art adversarial attacks. However, there is room for improvement as we do not achieve 100% detection rates. An interesting line of future work is to explore if total detection can be achieved via principled approaches to choosing the `NFP`. Although empirical evidence suggests *NeuralFP* is effective, an open question is if stronger attacks can be developed that can fool *NeuralFP* or if it can be proved that *NeuralFP* is invulnerable to adversarial perturbations. Other interesting avenues include using *NeuralFP* for robust prediction and extending it to domains beyond image-processing that have been shown to be vulnerable to

adversarial attacks.

Acknowledgements This work is supported in part by NSF grants #1564330, #1637598, #1545126; STARnet, a Semiconductor Research Corporation program, sponsored by MARCO and DARPA; and gifts from Bloomberg and Northrop Grumman. The authors would like to thank Xingjun Ma for providing the relevant baseline numbers for comparison.

References

NIPS 2017: Non-targeted Adversarial Attack, howpublished = https://www.kaggle.com/c/nips-2017-non-targeted-adversarial-attack/, note = Accessed: 2017-02-07.

Carlini, N. and Wagner, D. Towards Evaluating the Robustness of Neural Networks. *ArXiv e-prints*, August 2016.

Carlini, Nicholas and Wagner, David A. Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR*, abs/1705.07263, 2017a. URL http://arxiv.org/abs/1705.07263.

Carlini, Nicholas and Wagner, David A. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *CoRR*, abs/1711.08478, 2017b. URL http://arxiv.org/abs/1711.08478.

Cihang Xie, Jianyu Wang, Zhishuai Zhang Zhou Ren Alan Yuille. Mitigating adversarial effects through randomization. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sk9yuql0Z.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting Adversarial Samples from Artifacts. *ArXiv e-prints*, March 2017.

Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

Jacob Buckman, Aurko Roy, Colin Raffel Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S18Su--CW.

Kolter, J. Zico and Wong, Eric. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017. URL http://arxiv.org/abs/1711.00851.

Kurakin, Alexey, Goodfellow, Ian J., and Bengio, Samy. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016. URL http://arxiv.org/abs/1607.02533.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv e-prints*, June 2017.

Meng, Dongyu and Chen, Hao. Magnet: a two-pronged defense against adversarial examples. *CoRR*, abs/1705.09064, 2017. URL http://arxiv.org/abs/1705.09064.

Papernot, Nicolas, McDaniel, Patrick D., Jha, Somesh, Fredrikson, Matt, Celik, Z. Berkay, and Swami, Ananthram. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015. URL http://arxiv.org/abs/1511.07528.

Pouya Samangouei, Maya Kabkab, Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkJ3ibb0-.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified Defenses against Adversarial Examples. *ArXiv e-prints*, January 2018.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Vinyals, Oriol, Blundell, Charles, Lillicrap, Tim, kavukcuoglu, koray, and Wierstra, Daan. Matching networks for one shot learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3630–3638. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning.pdf.

Xingjun Ma, Bo Li, Yisen Wang Sarah M. Erfani Sudanthi Wijewickrema Grant Schoenebeck Michael E. Houle Dawn Song James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1gJ1L2aW. accepted as oral presentation.

Yang Song, Taesup Kim, Sebastian Nowozin Stefano Ermon Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJUYGxbCW.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Layer	Parameters
Convolution + ReLU + BatchNorm	$11 \times 11 \times 64$
MaxPool	3×3
Convolution + ReLU + BatchNorm	$5 \times 5 \times 192$
MaxPool	3×3
Convolution + ReLU + BatchNorm	$3 \times 3 \times 384$
MaxPool	3×3
Convolution + ReLU + BatchNorm	$3 \times 3 \times 256$
MaxPool	3×3
Convolution + ReLU + BatchNorm	$3 \times 3 \times 156$
MaxPool	3×3
Fully Connected + ReLU + BatchNorm	3072
Dropout	-
Fully Connected + ReLU + BatchNorm	1024
Dropout	-
Softmax	20

Table 2. MiniImagenet-20 Model Used

Layer	Parameters
Convolution + ReLU + BatchNorm	$5 \times 5 \times 32$
MaxPool	2×2
Convolution + ReLU + BatchNorm	$5 \times 5 \times 64$
MaxPool	2×2
Fully Connected + ReLU + BatchNorm	200
Fully Connected + ReLU + BatchNorm	200
Softmax	10

Table 3. MNIST Model Used

Layer	Parameters
Convolution + ReLU + BatchNorm	$3 \times 3 \times 32$
Convolution + ReLU + BatchNorm	$3 \times 3 \times 64$
MaxPool	2×2
Convolution + ReLU + BatchNorm	$3 \times 3 \times 128$
Convolution + ReLU + BatchNorm	$3 \times 3 \times 128$
MaxPool	2×2
Fully Connected + ReLU + BatchNorm	256
Fully Connected + ReLU + BatchNorm	256
Softmax	10

Table 4. CIFAR Model Used

6. Appendix

Note: Code for CW-adaptive is based on code from`https://github.com/carlini/nn_robust_attacks`. Code for the other attacks was obtained from the paper (Xingjun Ma, 2018).

6.1. MNIST

For MNIST, we use the model described in Table 3.

6.2. CIFAR-10

For CIFAR-10, we use the model described in Table 4

6.3. MiniImagenet-20

MiniAlexNet Model We use a model similar to AlexNet for MiniImagenet-20. The model used is described in Table 2

MiniImagenet-20 classes We use images from the following 20 ImageNet classes for our experiments:
`n01558993, n02795169, n03062245, n03272010, n03980874, n04515003 n02110063, n02950826, n03146219, n03400231, n04146614, n04612504, n02443484, n02981792, n03207743, n03476684, n04443257, n07697537`