

A Hamiltonian Monte Carlo Method for Probabilistic Adversarial Attack and Learning

Hongjun Wang, Guanbin Li, Xiaobai Liu and Liang Lin

Abstract—Although deep convolutional neural networks (CNNs) have demonstrated remarkable performance on multiple computer vision tasks, researches on adversarial learning have shown that deep models are vulnerable to adversarial examples, which are crafted by adding visually imperceptible perturbations to the input images. Most of the existing adversarial attack methods only create a single adversarial example for the input, which just gives a glimpse of the underlying data manifold of adversarial examples. An attractive solution is to explore the solution space of the adversarial examples and generate a diverse bunch of them, which could potentially improve the robustness of real-world systems and help prevent severe security threats and vulnerabilities. In this paper, we present an effective method, called Hamiltonian Monte Carlo with Accumulated Momentum (HMCAM), aiming to generate a sequence of adversarial examples. To improve the efficiency of HMC, we propose a new regime to automatically control the length of trajectories, which allows the algorithm to move with adaptive step sizes along the search direction at different positions. Moreover, we revisit the reason for high computational cost of adversarial training under the view of MCMC and design a new generative method called Contrastive Adversarial Training (CAT), which approaches equilibrium distribution of adversarial examples with only few iterations by building from small modifications of the standard Contrastive Divergence (CD) and achieve a trade-off between efficiency and accuracy. Both quantitative and qualitative analysis on several natural image datasets and practical systems have confirmed the superiority of the proposed algorithm.

Index Terms—Adversarial Example, Adversarial Training, Robustness and Safety of Machine Learning.



1 INTRODUCTION

WITH the rapid development and superior performance achieved in various vision tasks, deep convolutional neural networks (CNNs) have eventually led to pervasive and dominant applications in many industries. However, most deep CNN models could be easily misled by natural images with imperceptible but deceptive perturbations. These crafted images are known as adversarial examples, which have become one of the biggest threats in real-world applications with security-sensitive purposes [1], [2], [3]. Devising an effective algorithm to generate such deceptive examples can not only help to evaluate the robustness of deep models, but also promote better understanding about deep learning for the future community development.

In the past literature, most state-of-the-art methods are well-designed for generating a *single* adversarial example only, for example, by maximizing the empirical risk minimization (ERM) over the target model, and might not be able to exhaustively explore the solution space of adversarial examples. In our opinion, adversarial examples of a deep model might form an underlying data manifold [4], [5], [6], [7] rather than scattered outliers of the classification surface. Therefore, we argue that it is desirable and critical for

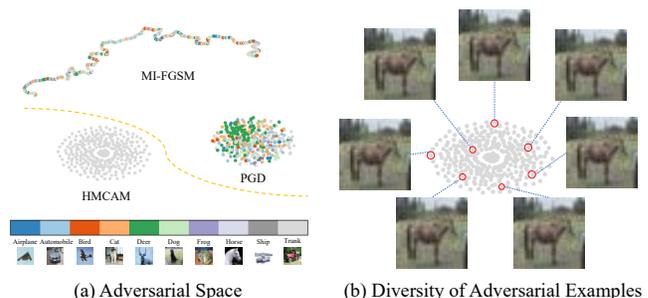


Fig. 1. Iterative Deterministic Generator vs. Stochastic MCMC-based Generator. We choose a natural image to generate 500 adversarial examples and visualize these samples by t-SNE [8]. In contrast to two typical iterative deterministic methods (PGD [9] with 500 random restarts and MI-FGSM [10] selecting samples at the final 500 iterations), MCMC-based method explores the solution space of adversarial examples and finds out the decision boundary of target classifier which is easily misled to erratic discrimination, then generates multiple diverse adversarial examples to attack. It is clear that our method automatically generates all of the 500 samples for a certain category in the untargeted attack scenario. When gradually increasing the number of MCMC sampling, the generated sequence of adversarial examples and their corresponding frequencies collectively depict the true underlying distribution of adversarial examples.

This work was supported in part by the National Key Research and Development Program of China under Grant No.2018YFC0830103, in part by the National Natural Science Foundation of China under Grant No.61976250, No.61702565 and No.U1811463, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant No.2020B1515020048, in part by National High Level Talents Special Support Plan (Ten Thousand Talents Program). This work was also sponsored by CCF-Tencent Open Research Fund. (Corresponding author: Guanbin Li)

H. Wang, G. Li and L. Lin are with the school of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (e-mail: wanghq8@mail2.sysu.edu.cn; liguanbin@mail.sysu.edu.cn; linliang@ieee.org). X. Liu is with the Department of Computer Science, San Diego State University, San Diego, CA, 92182, USA (e-mail: xiaobai.liu@sdsu.edu).

adversarial attack and learning methods to have the ability of generating multiple diverse adversarial examples in one run for the following reasons. First, the diversity of adversarial examples can fully verify the robustness of an unknown system. Second, developing an attack with multiple distinct adversarial examples would enable adversarial training with such examples, which could make the model more robustness against white-box attacks. Third, it is necessary to preserve multiple adversarial examples since the solution space of

adversarial examples only depends on the targeted model and its input image even if the objective energy function of adversarial examples is constantly being improved [11], [12], [13], [14], e.g. mapping the clipped gradient descent into tanh space or adding KL-divergence term. A series of adversarial samples can better depict the manifold of the solution space than a single global optimal, which can also bring more stable and superior performance on attacking. In fact, training these representative generative models also suffers from instability due to the difficulty of finding the exact Nash equilibrium [15], [16] or tackling memorization [17], [18], [19].

Motivated by the aforementioned observations, we rethink the generation of adversarial examples from the view of probabilistic distribution and develop an innovative paradigm called Hamiltonian Monte Carlo with Accumulated Momentum (HMCAM) for generating a sequence of adversarial examples in one run. Given the attack objective energy function, the HMCAM method first constructs a joint distribution by Hamiltonian equations and the Metropolis-Hastings algorithm is used to determine whether to transition to the candidate sample via the acceptance function based upon the proposal distribution and the candidate-generating density. To improve the efficiency of HMC, we further propose a new regime called accumulated momentum to adaptively control the step sizes, which allows the algorithm to move with different step sizes along the search direction at different positions. Conceptually, our HMCAM paradigm also reveals the roles of the well-known FSGM family algorithms, including FSGM [20], I-FGSM [21], PGD [9] and MI-FGSM [10]. These methods can be considered as special cases of HMC with minor modifications. Inspired by our new paradigm, we further design a new generative method, called Contrastive Adversarial Training (CAT), which approaches equilibrium distribution of adversarial examples with only few iterations by building from small modifications of the standard Contrastive Divergence [22]. We verify the effectiveness of both the adversarial attack and the training algorithms in multiple scenarios. For the investigation of adversarial attack, we test our algorithm on single and ensemble models in both white-box and black-box manners. Extensive experiments conducted on the CIFAR10 dataset show that our method achieves much higher success rates with fewer iterations for black-box models and maintains similar success rates for white-box models. We also evaluate the proposed HMCAM on the CAAD 2018 defense champion solution [23]. It outperforms the official baseline attack and M-PGD (PGD with momentum) by a large margin, which clearly demonstrates the effectiveness of the proposed adversarial method. To further show the practical applicability of our proposed method, we launch our attack on the real-world celebrity recognition system such as Clarifai, AWS and Azure. Compared with traditional iterative attack methods, HMCAM is able to generate more successful malicious examples to fool the systems through sampling from the likelihood models. For adversarial training, our CAT algorithm achieves much higher robustness than any other state-of-the-art adversarial training methods on both the CIFAR-10 and MNIST datasets and reaches a balance of performance and efficiency. In summary, this paper has the following contributions:

- We formulate the problem of generating adversarial examples in a HMC framework, which can produce multiple fair samples and better represent the underlying distribution of the adversarial examples. These fair samples can well reflect the typical state of the underlying system.
- We design a new regime called accumulated momentum to adaptively control the step sizes, which allows the algorithm to move with different step sizes along the search direction at different positions, and thus improves the efficiency of HMC.
- We thoroughly compare the effectiveness of our algorithms in various settings against several iterative attack methods on both CIFAR10 and ImageNet, including the champion solution in the defense track of CAAD 2018 competitions. We also investigate the high efficiency of HMC framework in adversarial training and show the practical applicability of our HMCAM by successfully attacking the real-world celebrity recognition system.

2 RELATED WORK

Adversarial Attacks. Since Szegedy *et al.* [20] first revealed that deep learning models were vulnerable to adversarial attacks, learning how to generate adversarial examples has quickly attracted wide research interest. Goodfellow *et al.* [24] developed a single gradient step method to generate adversarial examples, which was known as the fast gradient sign method (FGSM). Kurakin *et al.* [21] extended FGSM to an iterative version and obtained much stronger adversarial examples. Based on their works, Madry *et al.* [9] started projected gradient descent (PGD) from several random points in the L_∞ -ball around the natural example and iterate PGD. Dong *et al.* [10] proposed to add the momentum term into iterative process to boost adversarial attacks, which won the first places in the NIPS 2017 Adversarial Attacks and Defenses Competition. Due to the high efficiency and high success rates, the last two methods have been widely used as baseline attack models in many competitions. Our method also belongs to the iterative attack family but has much faster convergence and better transferability than alternative methods. When compared with recent similar works on distributional attack [13], [25], our HMC-based methods can better explore the distribution space of adversarial samples and reveal the reason for the high computational cost of adversarial training from the perspective of MCMC.

Adversarial Defense. To deal with the threat of adversarial examples, different strategies have been studied with the aim of finding countermeasures to protect ML models. These approaches can be roughly categorized into two main types: (a) detection only and (b) complete defense. The goal of the former approaches [26], [27], [28], [29], [30], [31], [32] is to reject the potential malignant samples before feeding them to the ML models. However, it is meaningless to pinpoint the defects for developing more robust ML models. Complimentary to the previous defending techniques, the latter defense methods often involve modifications in the training process. For example, gradient masking [33], [34], [35] or randomized models [36], [37], [38], [39], [40] obfuscate the gradient information of the classifiers to confuse the

attack mechanisms. There are also some add-on modules [23], [41], [42], [43], [44], [45] being appended to the targeted network to protect deep networks against the adversarial attacks. Besides all the above methods, adversarial training [9], [21], [24], [46], [47] is the most effective way, which has been widely verified in many works and competitions. However, limited works [48], [49] focus on boosting robust accuracy with reasonable training time consumption.

Markov Chain Monte Carlo Methods. Markov chain Monte Carlo (MCMC) [50] established a powerful framework for drawing a series of fair samples from the target distribution. But MCMC is known for its slow convergence rate which prevents its wide use in time critical fields. To address this issue, Hamiltonian (or Hybrid) Monte Carlo method (HMC) [51], [52] was introduced to take advantages of the gradient information in the target solution space and accelerate the convergence to the target distribution. Multiple variants of HMC [53], [54], [55] were also developed to integrate adaptive strategies for tuning step size or iterations of leapfrog integrator. Recently, the fusion of MCMC and machine learning hastens wide range of applications, including data-driven MCMC [56], [57], adversarial training [58], cooperative learning [59], which shows great potential of MCMC in deep learning.

3 METHODOLOGY

In this section, we briefly review the Markov chain Monte Carlo (MCMC) method [50] and Hamiltonian Monte Carlo (HMC) methods [51], [52]. Then we will explain that most of the existing methods for generating adversarial examples are the specializations of HMC. Finally, we illustrate how to modify the update policy of the momentum item in HMC to obtain a better trajectory.

3.1 Review: MCMC and Hamiltonian Monte Carlo

We now give the overall description of Metropolis-Hasting based MCMC algorithm. Suppose p is our target distribution over a space \mathcal{D} , MCMC methods construct a Markov Chain that has the desired distribution p as its stationary distribution. At the first step, MCMC chooses an arbitrary point x_0 as the initial state. Then it repeatedly performs the dynamic process consisting of the following steps: (1) Generate a candidate sample \tilde{x} as a ‘‘proposed’’ value for x_{t+1} from the candidate-generating density $Q(x_t|\tilde{x})$, which generates a value \tilde{x} from $Q(x_t|\tilde{x})$ when a process is at the state x_t . (2) Compute the acceptance probability $\xi = \min(1, \frac{p(\tilde{x})Q(x_t|\tilde{x})}{p(x_t)Q(\tilde{x}|x_t)})$, which is used to decide whether to accept or reject the candidate. (3) Accept the candidate sample as the next state with probability ξ by setting $x_{t+1} = \tilde{x}$. Otherwise reject the proposal and remain $x_{t+1} = x_t$. Although MCMC makes it possible to sample from any desired distributions, its random-walk nature makes the Markov chain converge slowly to the stationary distribution $p(x)$.

In contrast, HMC employs physics-driven dynamics to explore the target distribution, which is much more efficient than the alternative MCMC methods. Before introducing HMC, we start out from an analogy of Hamiltonian systems in [52] as follows. Suppose a hockey puck sliding over a surface of varying height and both the puck and the surface

Algorithm 1 Hamiltonian Monte Carlo

Inputs: Target distribution $p(\theta)$, initial position $\theta^{(1)}$ and step size α

```

1: /*Hamiltonian system construction*/
2:  $U(\theta) = -\log p(\theta)$ ,  $K(v) = v^T \mathbf{I}^{-1} v / 2$ 
3: for  $s = 1, 2, \dots$  do
4:    $v_0 \sim \mathcal{N}(0, \mathbf{I})$ ,  $\theta_0 = \theta^{(s)}$ 
5:   /*Leapfrog integration*/
6:    $v_0 \leftarrow v_0 - \frac{\alpha}{2} \nabla U(\theta_0)$ 
7:   for  $t = 1$  to  $T$  do
8:      $\theta_t \leftarrow \theta_{t-1} + \alpha \nabla K(v_{t-1})$ 
9:      $v_t \leftarrow v_{t-1} - \alpha \nabla U(\theta_t)$ 
10:  end for
11:   $v_T \leftarrow v_T - \frac{\alpha}{2} \nabla U(\theta_T)$ 
12:  /*Metropolis-Hastings correction*/
13:   $u \sim \text{Uniform}(0,1)$ 
14:  if  $u < \min(1, e^{H(\theta_T, v_T) - H(\theta_s, v_s)})$  then
15:     $\theta^{(s+1)} \leftarrow \theta_T$ 
16:  else
17:     $\theta^{(s+1)} \leftarrow \theta^{(s)}$ 
18:  end if
19: end for

```

are frictionless. The state of the puck is determined by *potential energy* $U(\theta)$ and *kinetic energy* $K(v)$, where θ and v are the position and the momentum of the puck. The evolution equation is given by the Hamilton’s equations:

$$\begin{cases} \frac{\partial \theta}{\partial t} = \frac{\partial H}{\partial v} = \nabla_v K(v) \\ \frac{\partial v}{\partial t} = \frac{\partial H}{\partial \theta} = -\nabla_\theta U(\theta). \end{cases} \quad (1)$$

Due to the reversibility of Hamiltonian dynamics, the total energy of the system remains constant:

$$H(\theta, v) = U(\theta) + K(v). \quad (2)$$

As for HMC, it contains three major parts: (1) Hamiltonian system construction; (2) Leapfrog integration; (3) Metropolis-Hastings correction. Firstly, the Hamiltonian is an energy function for the joint density of the variables of interest θ and auxiliary momentum variable v , so HMC defines a joint distribution via the concept of a canonical distribution:

$$p(\theta, v) \propto \exp\left(\frac{-H(\theta, v)}{\tau}\right), \quad (3)$$

where $\tau = 1$ for the common setting. Then, HMC discretizes the system and approximately simulates Eq. (1) over time via the leapfrog integrator. Finally, because of inaccuracies caused by the discretization, HMC performs Metropolis-Hastings [60] correction without reducing the acceptance rate. A full procedure of HMC is described in Algorithm 1.

According to Eq. (2) and (3), the joint distribution can be divided into two parts:

$$p(\theta, v) \propto \exp\left(\frac{-U(\theta)}{\tau}\right) \exp\left(\frac{-K(v)}{\tau}\right). \quad (4)$$

Since $K(v)$ is an auxiliary term and always setting $K(v) = v^T \mathbf{I}^{-1} v / 2$ with identity matrix \mathbf{I} for standard HMC, our aim is that the potential energy $U(\theta)$ can be defined as $U(\theta) = -\log p(\theta)$ to explore the target density p more efficiently than

using a proposal probability distribution. If we can calculate $\nabla_{\theta}U(\theta) = -\frac{\partial \log(p(\theta))}{\partial \theta}$, then we can simulate Hamiltonian dynamics that can be used in an MCMC technique.

3.2 Simulating Adversarial Examples Generating by HMC

Considering a common classification task, we have a dataset \mathcal{D} that contains normalized data $x \in [0, 1]^d$ and their one-hot labels y . We identify a target DNN model with an hypothesis $f(\cdot)$ from a space \mathcal{F} . The cross entropy loss J function is used to train the model. Assume that the adversarial examples for x with label y are distributed over the solution space Ω . Given any input pair (x, y) , for a specified model $f(\cdot) \in \mathcal{F}$ with fixed parameters, the adversary wants to find such examples \tilde{x} that can mislead the model:

$$\Omega = \arg \max_{N(x) \subset \mathcal{N}(x)} \int J(\tilde{x}, y) p(\tilde{x}|x, y) d\tilde{x}, \quad (5)$$

where $\mathcal{N}(x)$ is the neighboring regions of x and defined as $x' \in \mathcal{N}(x) := \{\|x' - x\|_{1,2,\text{or}\infty} \leq \epsilon\}$. From the perspective of Bayesian statistics, we can make inference about adversarial examples over a solution space Ω from the posterior distribution of \tilde{x} given the natural inputs x and labels y .

$$\tilde{x} \sim p(\tilde{x}|x, y) \propto p(y|\tilde{x})p(\tilde{x}|x), \quad \tilde{x} \in \Omega. \quad (6)$$

In Hamiltonian system, it becomes to generate samples from the joint distribution $p(\theta, v)$. Let $\theta = \tilde{x}$, according to Eq. (6) and (4), we can express the posterior distribution as a canonical distribution (with $\tau = 1$) using a potential energy function defined as:

$$\begin{aligned} U &= \frac{1}{N} \sum_{i=1}^N -\log p(y^{(i)}|\tilde{x}^{(i)}) - \log p(\tilde{x}|x) \\ &= J(\tilde{x}, y) - \log p(\tilde{x}|x). \end{aligned} \quad (7)$$

Since $J(\tilde{x}, y)$ is the usual classification likelihood measure, the question remains how to define $p(\tilde{x}|x)$. A sensible choice is a uniform distribution over the L_p ball around x , which means we can directly use a DNN classifier to construct a Hamiltonian system for adversarial examples generating as the base step of HMC.

Recall that the development of adversarial attacks is mainly based on the improvement of the vanilla fast gradient sign method, which derives I-FGSM, PGD and MI-FGSM. For clarity, we omit some details about the correction due to the constraint of adversarial examples. The core policy of the family of fast gradient sign methods is:

$$\tilde{x}_t = \tilde{x}_{t-1} + \alpha \cdot \text{sign}(g_t), \quad (8)$$

where g_t is the gradient of J at the t -th iteration, i.e., $\nabla_x J(\tilde{x}_{t-1}, y)$. It is clear that the above methods are the specialization of HMC by setting:

$$\begin{aligned} \theta_t &= \tilde{x}_t, \quad v_t = g_t \\ H(\theta, v) &= J(\theta) + |v|. \end{aligned} \quad (9)$$

More specifically, I-FGSM can be considered as the de-generation of HMC, which explicitly updates the position item θ but implicitly changes the momentum item v at every iteration. One of the derivation of I-FGSM, MI-FGSM, has explicitly updated both θ and v by introducing $g_t =$

Algorithm 2 HMCAM

Inputs: Target DNN model $f(\cdot)$ with loss function J , initial position $\theta^{(0)} = x$, step size α , sampling transition S , updating iteration T , magnitude of perturbation ε and small constant δ

Inputs: exponential decay rates for the moment estimates $\beta_1 = 0.95, \beta_2 = 0.999$

```

1: /*Hamiltonian system construction*/
2:  $U(\theta) = J, K(v) = |v|$ 
3: for  $s = 1$  to  $S$  do
4:   Initialize  $v_0 \leftarrow 0; e_0 \leftarrow 0; \hat{e}_0 \leftarrow 0$ 
5:   /*Accumulated Momentum*/
6:   for  $t = 1$  to  $T$  do
7:      $b_1 \leftarrow 1 - \beta_1^t, \quad b_2 \leftarrow 1 - \beta_2^t$ 
8:      $v_t \leftarrow \beta_1 \cdot v_{t-1} - (\beta_1 - 1) \cdot \nabla_{\theta} J(\theta_{t-1}, y)$ 
9:      $e_t \leftarrow \beta_2 \cdot e_{t-1} - (\beta_2 - 1) \cdot \nabla_{\theta}^2 J(\theta_{t-1}, y)$ 
10:     $\hat{e}_t \leftarrow \max(e_t, \hat{e}_t)$ 
11:     $\theta_t \leftarrow \theta_{t-1} + \min(\varepsilon, \frac{\alpha}{b_1(\sqrt{\frac{\hat{e}_t}{b_2} + \delta}})) \nabla_v K(v_t)$ 
12:     $\theta_t \leftarrow \mathbb{P}_{\varepsilon\text{-ball}}(\theta_t)$ 
13:  end for
14:  /*Metropolis-Hastings correction*/
15:   $u \sim \text{Uniform}(0,1)$ 
16:  if  $u < \min(1, e^{H(\theta_T, v_T) - H(\theta_s, v_s)})$  then
17:     $\theta^{(s+1)} \leftarrow \theta_T$ 
18:  else
19:     $\theta^{(s+1)} \leftarrow \theta^{(s)}$ 
20:  end if
21: end for
22: Return A sequence of adversarial examples  $\{\theta\}$ 

```

$\mu g_{t-1} + \frac{1}{\|\nabla J(\tilde{x}_{t-1}, y)\|_1} \nabla J(\tilde{x}_{t-1}, y)$ after Eq. (8) at each step with the decay factor $\mu = 1$. The other derivative PGD runs Eq. (8) on a set of initial points $\tilde{x}_0 \in \{\tilde{x}_0^{(1)}, \tilde{x}_0^{(2)}, \dots, \tilde{x}_0^{(S)}\}$ adding different noises, which can be treated as a parallel HMC but the results are mutually independent.

3.3 Adaptively Exploring the Solution Space with Accumulated Momentum

Although the above formulation has proved that HMC can be used to simulate adversarial examples generating, one major problem of these methods is that θ and v are not independent because of $v_t = \nabla J(\theta_{t-1})$ as discussed in Eq. (9). The other disadvantage is in optimization: SGD scales the gradient uniformly in all directions, which can be particularly detrimental for ill-scaled problems. Like the need to choose step size in HMC, the laborious learning rate tuning is also troublesome.

To overcome the above two problems, we present a Hamiltonian Monte Carlo with Accumulated Momentum (HMCAM) for adversarial examples generating. The resulting HMCAM algorithm is shown in Algorithm 2. The core of our accumulated momentum strategy is using exponential moving average (EMA) to approximate the first and second moment of the stochastic gradient by weighted accumulating the history moment information. Let us initialize the exponential moving average as $v_0 = e_0 = 0$. After t inner-loop

Methods	Hamiltonian system construction			Iteration		Metropolis-Hastings correction
	potential energy?	kinetic energy?	sampling?	θ update?	v update?	
FGSM [24]	✓, but implicit	✓, but implicit	x	x	x	x
I-FGSM [21]	✓, but implicit	✓, but implicit	x	✓	x	x
PGD [9]	✓, but implicit	✓, but implicit	✓, but independent	✓	x	x
MI-FGSM [10]	✓, but implicit	✓, but implicit	x	✓	✓	x

TABLE 1
Relationship between HMC and the family of fast gradient sign methods.

steps, the accumulated momentum v_t is:

$$\begin{aligned}
v_t &= (1 - \beta_1) \sum_{i=1}^t \beta_1^{t-i} \nabla_{\theta} J(\theta_{i-1}) \\
&= (1 - \beta_1) \underbrace{\nabla_{\theta} J(\theta_{t-1})}_{\text{Current term}} \\
&\quad + \beta_1 \underbrace{[\nabla_{\theta} J(\theta_{t-2}) + \beta_1 \nabla_{\theta} J(\theta_{t-3}) + \dots + \beta_1^{t-2} \nabla_{\theta} J(\theta_0)]}_{\text{History term}}.
\end{aligned} \tag{10}$$

The derivation for the second moment estimate e_t is completely analogous. Owing to the fact that the decay rates β_1 close to 1 is typically recommended in practice, the contribution of older gradients decreases exponentially. But meanwhile, we can observe in Eq. (10) that the current gradient only accounts for $1 - \beta_1 \rightarrow 0$, which is much smaller than β_1 . This indicates that performing exponential moving averages for the step in lieu of the gradient greatly reduces the relevance between v_t and the current position θ_{t-1} . That makes the sequence of samples into an approximate Markov chain.

As for step size, there always be a tradeoff between using long trajectories to make HMC more efficient or using shorter trajectories to update more frequently. Ignoring small constant δ , our accumulated momentum is to update the position by:

$$\theta_t = \theta_{t-1} + \alpha \Delta \theta = \theta_{t-1} + \frac{\sqrt{1 - \beta_2^t}}{(1 - \beta_1^t)} \cdot \frac{\alpha}{|v_t|} \cdot \frac{v_t}{\sqrt{e_t}}, \tag{11}$$

where $\sqrt{1 - \beta_2^t}/(1 - \beta_1^t)$ corrects the biased estimation of moments towards initial values at early stages due to the property of EMA. When approaching to the minima, $\alpha/|v_t|$ automatically decreases the size of the gradient steps along different coordinates. Because $v_t/\sqrt{e_t}$ leads to smaller effective steps in solution space when closer to zero, this anisotropic scale of step size helps θ to escape sharp local minimal at the later period of the learning process at some coordinates, which leads to better generalization. We apply similar idea as [61] by replacing e to \hat{e} that maintains the maximum of all history e to keep a non-increasing step size $(\sqrt{e_{t+1}} - \sqrt{e_t})/\alpha_t \geq 0$. To guarantee the step size does not exceed the magnitude of adversarial perturbations, we confine the α to a predefined maximum ε by applying element-wise min.

After every full inner iteration, we calculate the acceptance rate of the candidate sample by M-H sampling and reinitialize the first/second moment as well as the maximum of second moment to zero and then perform the next generation. M-H algorithm distributes the generating samples to staying in high-density regions of the

Algorithm 3 Contrastive Adversarial Training

Input: A DNN classifier $f_{\omega}(\cdot)$ with initial learnable parameters ω_0 ; training data x with visible label y ; number of epochs N ; length of trajectory K ; repeat time T ; magnitude of perturbation ε ; learning rate κ ; step size α .
/*Stage-0: Construct Hamiltonian system*/
 $U(\theta, \omega, \tilde{\omega}, y, k) = -J_{cd}(f_{\omega}(\theta^{k-1}), f_{\tilde{\omega}}(\theta^K), y)$, $\mathcal{K}(v) = |v|$

Initialize $\omega = \tilde{\omega} = \omega_0$, $\theta^K = \theta^0$.

for epoch = $1 \dots N/(TK)$ **do**
 $\theta^0 \leftarrow x + v_0$, $v_0 \sim \text{Uniform}(-\varepsilon, \varepsilon)$.

for $t = 1$ to T **do**

/*Stage-1: Generate adversarial examples by K -step contrastive divergence*/

for $k = 1$ to K **do**

$\theta^k \leftarrow \theta^{k-1} + \varepsilon \cdot \nabla \mathcal{K}(v_{t-1})$

$v_t \leftarrow v_{t-1} - \alpha \nabla U(\theta, \omega, \tilde{\omega}, y, k)$

$v_t \leftarrow \text{clip}(v_t, -\varepsilon, \varepsilon)$

end for

/*Stage-2: Update parameters of DNN by generated adversarial examples*/

$\mathbf{g}_{\omega} \leftarrow \mathbb{E}_{(\theta, y)} [\nabla_{\omega} J_{ce}(f_{\omega}(\theta^K), y)]$

$\tilde{\omega} \leftarrow \omega$

$\omega \leftarrow \omega - \kappa \mathbf{g}_{\omega}$

end for

end for

candidate distribution or only occasionally visiting low-density regions through the acceptance probability. As more and more sample are produced, the distribution of samples more closely approximates the desired distribution and its returning samples are more in line with such distribution than other works like PGD with random starts.

4 CONTRASTIVE ADVERSARIAL TRAINING

Assume softmax is employed for the output layer of the model $f(\cdot)$ and let $f(x)$ denote the softmax output of a given input $x \in \mathbb{R}^d$, i.e., $f(x) : \mathbb{R}^d \rightarrow \mathbb{R}^C$, where C is the number of categories. We also assume that there exists an oracle mapping function $f^* \in \mathcal{F} : x \mapsto y^*$, which pinpoints the belonging of the input x to all the categories by accurate confidence scores $y^* \in \mathbb{R}^C$. The common training is to minimize the cross-entropy (CE) loss, which is defined as:

$$f = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(x, y) \sim \mathcal{D}} [L_{ce}(f(x), y)], \tag{12}$$

where y is the manual one-hot annotation of the input x since y^* is invisible. The goal of Eq. (12) is to update the parameters of f for better approaching f^* , which leads to:

$$f(x) \approx y \approx y^* = f^*(x). \tag{13}$$

Suppose the target DNN model correctly classifies most of the input after hundreds of iterations, it will still be badly misclassified by adversarial examples (i.e., $\arg \max_{c \in \{1, \dots, C\}} f(\tilde{x})_c \neq y[c]$). In adversarial training, these constructed adversarial examples are used to update the model using minibatch SGD. The objective of this minmax game can be formulated as a robust optimization following [9]:

$$f' = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\max_{\tilde{x} \in \mathcal{N}(x)} L_{ce}(f(\tilde{x}), y) \right]. \quad (14)$$

As mentioned in Section 3.2, the inner maximization problem can be reformulated as the process of HMC. It is obvious that the high time consumption of adversarial training is caused by the long trajectory of HMC. But running a full trajectory for many steps is too inefficient since the model changes very slightly between parameter updates. Thus, we take advantage of that by initializing a HMC at the state in which it ended for the previous model. This initialization is often fairly close to the model distribution, even though the model has changed a bit in the parameter update. Besides, the high acceptance rate of HMC indicates that it is not necessary to run a long Markov Chain from the initial point. Therefore, we can simply run the chain for one full step and then update the parameters to reduce the tendency of the chain to wander away from the initial distribution on the first step instead of running the full trajectory to equilibrium. We take small number K of transitions from the data sample $\{x_i\}_i^n = 1$ as the initial values of the MCMC chains and then use these K -step MCMC samples to approximate the gradient for updating the parameters of the model. Algorithm 3 summarizes the full algorithm.

Moreover, we also present a new training objective function J_{cd} , which minimizes the difference of KL divergence between two adjacent sampling steps to substitute the common KL loss:

$$J_{cd} = \rho(Q^0 \parallel Q^\infty) - \lambda(Q^1 \parallel Q^\infty), \quad (15)$$

where \parallel denotes a Kullback-Leibler divergence and ρ and λ are the balanced factors. The intuitive motivation for using this J_{cd} is that we would like every state in HMC exploring to leave the initial distribution Q_0 and $Q^0 \parallel Q^\infty$ would never exceed $Q^1 \parallel Q^\infty$ until Q_1 achieves the equilibrium distribution. We set $\lambda = 2, \rho = 1$ and analyze how this objective function influences the partial derivative of the output probability vector with respect to the input. Due to the fact that the equilibrium distribution Q^∞ is considered as a fixed distribution and the chain rule, we only need to focus on the derivative of the softmax output vector with respect to its input vector in the last layer as follows:

$$\begin{aligned} \nabla U_{\text{last}} &= 2 \sum_c y_c \frac{\partial \log f_\omega(\tilde{x}^K)_c}{\partial \tilde{x}'} - \sum_c y_c \frac{\partial \log f_{\tilde{\omega}}(\tilde{x})_c}{\partial \tilde{x}'} \\ &= 2 f_\omega(\tilde{x}^K)_c \sum_c y_c - f_{\tilde{\omega}}(\tilde{x})_c \sum_c y_c - y \\ &= f_\omega(x^K) - (y - \Delta f), \end{aligned} \quad (16)$$

where $\Delta f = f_\omega(x^K) - f_{\tilde{\omega}}(\tilde{x})$. Based on this abbreviation, we can easily get the relationship between Eq. (16) and $\frac{\partial J_{ce}}{\partial \tilde{x}'} = f_\omega(x^K) - y$. For each adversarial example generation,

Eq. (16) makes an amendment of y which is determined by the difference of current and the last K -step HMC samples output probability. Since f_ω and $f_\omega(x)$ are more closer to f^* and y^* than $f_{\tilde{\omega}}$ and $f_{\tilde{\omega}}(x)$, each update of \tilde{x} would be better corrected.

5 EXPERIMENT

In this section, we conduct extensive experimental evaluations of our proposed methods on three benchmarks: CIFAR10 [62], ImageNet [63] and MNIST [64]. Firstly, we briefly introduce the major implementation settings in Section. 5.1, and perform comprehensive comparisons to verify the superiority of our HMCAM method on single and ensemble models in both white-box and black-box manners in Section. 5.2 and Section. 5.3. Then, we perform detailed ablation studies to demonstrate the influence of different aspects in HMCAM and explore the possibility of few sample learning for competitive results in adversarial training in Section. 5.4. To further test the efficiency of CAT method in adversarial training, we provide detailed quantitative comparison results of our proposed models in Section. 5.5. Finally, to investigate the generalization of our approach, we also perform experiments on ImageNet against the champion solution in the defense track of CAAD 2018 competitions in Section. 5.6.1 and attempt to launch attack on public face recognition systems in Section. 5.6.2.

5.1 Datasets and Implementation Details

Datasets. We employ the following four benchmark datasets for a comprehensive evaluation to validate the effectiveness of our HMCAM and CAT methods.

- **CIFAR10** [62] is a widely used dataset consisting of 60,000 colour images of 10 categories. Each category has 6,000 images. Due to the resource limitation, we mainly focus on the CIFAR10 [62] dataset with extensive experiments to validate the effectiveness of the proposed methods on both adversarial attack and training.
- **ImageNet** [63] a large dataset with 1,283,166 images in the training set and 50,000 images in the validation set images collected from the Web. It has 1,000 synsets used to label the images. As it is extremely time-consuming to train a model from scratch on ImageNet, we only use it to test the generalization of our approach, which fights against the champion solution in the defense track of CAAD 2018 competitions.
- **MNIST** [64] is a database for handwritten digit classification. It consists of 60,000 training images and 10,000 test images, which are all 28×28 greyscale images, representing the digits 0~9. In this experiment, we only perform different adversarial training methods on MNIST.

Implementation details. For adversarial attack, we pick six models, including four normally trained single models (ResNet32 [65], VGG16 (without BN) [66], ResNetXt29-8-64 [67] and DenseNet121 [68]) and one adversarially trained ensemble models (Resnet32_A). The hyper-parameters of different attack methods follow the default settings in [69] and the total iteration number is set to $N = 100$ (in most

Model	Attack	ResNet32	VGG16	ResNetXt	Densenet121	ResNet32_A
ResNet32	FGSM	38.31%	29.30%	19.89%	22.64%	3.79%
	PGD	98.12%	34.92%	49.44%	56.50%	4.60%
	M-PGD	98.93%	37.89%	55.48%	61.01%	7.44%
	AI-FGSM (Ours)	98.76%	42.23%	58.12%	64.12%	9.56%
	HMCAM (Ours)	98.76%	42.69%	58.76%	65.20%	10.01%
VGG16	FGSM	37.86%	56.34%	27.34%	31.54%	4.22%
	PGD	59.39%	80.55%	50.50%	55.72%	5.52%
	M-PGD	64.02%	83.64%	54.95%	60.48%	7.75%
	AI-FGSM (Ours)	64.77%	86.76%	53.38%	59.45%	9.83%
	HMCAM (Ours)	68.60%	93.29%	55.39%	62.70%	10.26%
ResNetXt	FGSM	27.44%	28.52%	31.74%	24.03%	4.50%
	PGD	65.48%	35.19%	96.60%	69.13%	6.55%
	M-PGD	72.81%	38.50%	98.02%	76.55%	10.11%
	AI-FGSM (Ours)	74.42%	42.73%	97.65%	77.09%	13.48%
	HMCAM (Ours)	74.92%	42.53%	97.75%	78.37%	14.11%
Densenet121	FGSM	26.87%	29.40%	20.42%	30.96%	4.42%
	PGD	63.38%	35.70%	57.22%	95.34%	5.67%
	M-PGD	66.07%	39.16%	59.48%	97.83%	8.33%
	AI-FGSM (Ours)	69.64%	41.41%	63.35%	96.49%	9.77%
	HMCAM (Ours)	69.82%	42.45%	63.87%	96.39%	10.36%

TABLE 2

The success rates of several of non-targeted attacks against a **single network** on CIFAR10. The maximum perturbation is $\epsilon = 2/255$. The *italic* columns in each block indicate white-box attacks while the rest are all black-box attacks which are more practical but challenging. Results have shown that our proposed methods (AI-FGSM and HMCAM) greatly improve the transferability of generated adversarial examples. We compare our AI-FGSM and HMCAM with FGSM, PGD and M-PGD (MI-FGSM+PGD), respectively.

cases $T = N$ except HMCAM). We fix $T = 50$ and $S = 2$ for HMCAM, and the decay rate μ is set to 1.0 for M-PGD (MI-FGSM+PGD). The magnitude of maximum perturbation at each pixel is $\epsilon = 2/255$. For simplicity, we only report the results based on L_∞ norm for the non-targeted attack.

For adversarial training, we follow the training scheme used in Free [48] and YOPO [49] on CIFAR10. We choose the standard Wide ResNet-34 and Preact-ResNet18 following previous works [9], [49]. For PGD adversarial training, we set the total epoch number $N = 105$ as a common practice. The initial learning rate is set to $5e-2$, reduced by 10 times at epoch 79, 90 and 100. We use a batch size of 256, a weight decay of $5e-4$ and a momentum of 0.9 for both algorithms. During evaluating, we test the robustness of the model under CW [12], M-PGD and 20 steps of PGD with step size $\epsilon = 2/255$ and magnitude of perturbation $\epsilon = 8/255$ based on L_∞ norm. When performing YOPO and Free, we train the models for 40 epochs and the initial learning rate is set to 0.2, reduced by 10 times at epoch 30 and 36. As for ImageNet, we fix the total loop times $T * K = 4$ same as Free-4 [48] for fair comparison. For all methods, we use a batch size of 256, and SGD optimizer with momentum 0.9 and a weight decay of $1e-4$. The initial learning rate is 0.1 and the learning rate is decayed by 10 every $30/TK$ epochs. We also set step size $\epsilon = 4/255$ and magnitude of perturbation $\epsilon = 4/255$ based on L_∞ norm.

5.2 Attacking a Single Model

We compare the attack success rates of HMCAM with the family of FGSM on a single network in Table 2. The adversarial examples are created by one of the six networks in turns and test on all of them. The *italic* columns in each block indicate white-box attacks and others refer to black-box attacks. From the Table 2, we can observe that HMCAM outperforms all other FGSM family attacks by a large margin in black-box scenario, and maintains comparable results on all white-box attacks with M-PGD. For example, HMCAM obtains success rates of 74.92% on ResNetXt29-8-64 (white-

box attack), 78.37% on DenseNet121 (black-box attack on normally trained model) and 14.11% on Resnet32_A (black-box attack on adversarially trained model) if adversarial examples are crafted on ResNetXt29-8-64, while M-PGD only reaches the corresponding success rates of 72.81%, 42.53% and 10.11%, respectively. Considering that the white-box attack is usually used as a launch pad for the black-box attack, this demonstrates the practicality and effectiveness of our HMCAM for improving the transferability of adversarial examples.

Note that AI-FGSM is a special case of HMCAM ($T = N$, $S = 1$), which means AI-FGSM only carries out the inner loop in Algorithm 2 for position and momentum updating. But AI-FGSM also reaches much higher success rates than FSGM family. This shows the superiority of our accumulated momentum strategy.

5.3 Attacking an Ensemble of Models

Although our AI-FGSM and HMCAM better improve the success rates for attacking model in black-box scenario, the results of all the attack methods on adversarially trained model, e.g., Resnet32_A, are far from satisfactory. To solve this problem, generating adversarial examples on the ensemble models [10], [70], [71] rather than a single model have been broadly adopted in the black-box scenario for enhancing the transferability and shown its effectiveness.

For the ensemble-based strategy, each one of the six models introduced above will be selected as the hold-out model while the rest build up an ensemble model. The ensemble weights are set equally for all the six models. The results are shown in Table 3. The *ensemble* block consists of the *white-box* attack which uses the ensemble model to attack itself, and the *hold-out* block is composed of the *black-box* attack that utilizes the ensemble model to generate adversarial examples for its corresponding hold-out model.

We can observe from Table 3 that our AI-FGSM and HMCAM always show much better transferability than other methods no matter which target model is selected. For

example, the adversarial examples generated by an ensemble of ResNet32, VGG16 and DenseNet121 (ResNetXt29-8-64 hold-out) can fool ResNetXt29-8-64 with a 83.07% success rate. Moreover, our proposed methods can remarkably boost the transferability of adversarial examples on adversarially trained model.

5.4 Ablation Study on Adversarial Attack

In the following sections, we perform several ablation experiments to investigate how different aspects of HMCAM influence its effectiveness. For simplicity, we only attack five single models introduced in the previous section, and focus on comparing our HMCAM with M-PGD since M-PGD is one of the most effective iterative attack method so far. We report the results in both white-box and black-box scenarios.

5.4.1 Influence of Iteration Number

To further demonstrate how fast our proposed method converges, we first study the influence of the total iteration number N on the success rates. We clip a snippet over a time span of 10 iterations from the very beginning. Results are shown in Fig. 2.

These results indicate that (1) the success rate of HMCAM against both white-box and black-box models are higher than M-PGD at all stages when combining with the extensive comparisons in Table 2, which shows the strength of our HMCAM. (2) Even when the number of iterations is one order lower than that in Table 2, the success rate of both HMCAM and M-PGD are still higher than PGD on the black-box scenario. Moreover, HMCAM ($N = 10$) reaches higher values than PGD ($N = 100$), demonstrating that HMCAM has strong attack ability and fast converges on both the white-box and black-box scenarios.

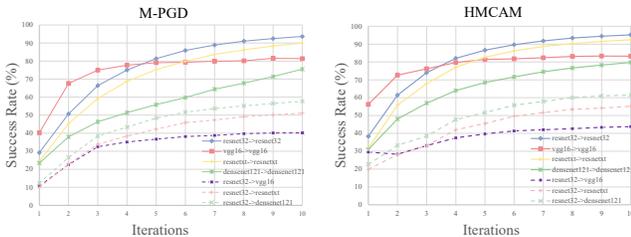


Fig. 2. The success rates of M-PGD (left) and HMCAM (right) on CIFAR10 over the first 10 iterations, with $\epsilon = 2/255$. Solid lines represent the white-box attacks and dashed lines represent the black-box attacks. “A \rightarrow B” means that model B is attacked by adversarial examples generated by model A.

5.4.2 Influence of Step Size

We also study the influence of the step size α on the success rates under both white-box and black-box settings. For simplicity, we fix the total iteration $N = 100$ and set $S = 1$ for HMCAM. We control the step size α in the range of $\{0.001, 0.01, 0.03, 0.1\} \times e^{-2}$. The results are plotted in Fig. 3. It can be observed that HMCAM outperforms M-PGD on both small and large step size. Under both the white-box and the black-box settings, our HMCAM is insensitive to the step size attributing to the accumulated momentum strategy.

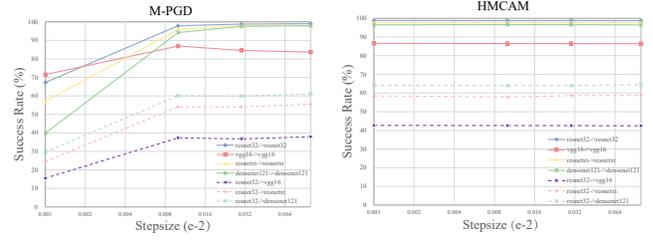


Fig. 3. The success rates of M-PGD (left) and HMCAM (right) on CIFAR10 after 100 iterations, with $\epsilon = 2/255$. Solid lines represent the white-box attacks and dashed lines represent the black-box attacks. “A \rightarrow B” means that model B is attacked by adversarial examples generating by model A.

5.4.3 Fewer samples for competitive results

Since HMCAM is able to explore the distribution of adversarial examples, we finally investigate what aspects of systems are strengthened by our method. We also investigate whether the competitive result can be achieved with fewer samples when compared to the regular adversarial training. We generate adversarial images using FGSM, BIM and PGD to adversarially retrain the model and remain M-PGD to attack. We fix the total iteration $N = S * T = 100$. To test the diversity of our generated samples, we select only $d = 50$ samples from the whole training set for generating adversarial samples, then mixed into the training set for adversarial training. For fair comparison, we allow other methods except HMCAM to select more samples satisfying $d' = d * S$. We sweep the sampling number S among $\{1, 2, 5, 10, 20, 50, 100\}$. The results are plotted in Fig. 4. It is clear to see that the system trained by our HMCAM, only using two orders of magnitude fewer natural samples than any other method, can achieve comparable robustness. Considering the compared methods utilize the extra samples truly on the adversarial manifold, this indicates that our HMCAM draws the distribution of adversarial examples with few samples indeed.

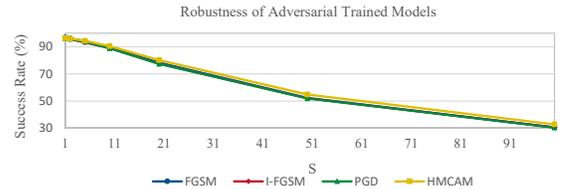


Fig. 4. Comparison with different adversarial training methods on CIFAR10. We use M-PGD as the attacker and report its success rate, with $\epsilon = 2/255$. Our HMCAM can use two orders of magnitude fewer samples than other methods to simulate the target distribution.

5.5 Efficiency for Adversarial Training

In this subsection, we investigate whether the training time of adversarial training can benefit from the view of HMC since the high computational cost of adversarial training can be easily attributed to the long trajectory of MCMC finding the stationary distribution of adversarial examples. We take fixed but small number k of transitions from the data sample as the initial values of the MCMC chains and then use these k -step MCMC samples to approximate the gradient for updating the

Attack	-ResNet32		-VGG16		-ResNetXt29-8-64		-DenseNet121		-ResNet32_A	
	Ensemble	Hold-out	Ensemble	Hold-out	Ensemble	Hold-out	Ensemble	Hold-out	Ensemble	Hold-out
FGSM	28.27%	30.74%	31.08%	31.43%	29.47%	25.34%	29.64%	27.70%	28.45%	7.53%
PGD	90.97%	80.73%	94.79%	50.30%	91.05%	80.14%	92.13%	82.11%	92.11%	13.88%
M-PGD	92.13%	81.92%	96.47%	52.48%	92.60%	80.83%	93.60%	83.85%	92.48%	21.23%
AI-FGSM (Ours)	92.62%	83.12%	96.06%	56.54%	92.94%	82.29%	93.47%	84.46%	93.04%	28.31%
HMCAM (Ours)	92.81%	83.76%	96.29%	57.43%	92.99%	83.07%	93.88%	85.38%	94.18%	30.11%

TABLE 3

The success rates of several of non-targeted attacks against an **ensemble of networks** on CIFAR10. The maximum perturbation is $\varepsilon = 2/255$. We report the results on the ensemble network itself (white-box scenario) and its corresponding hold-out network (black-box scenario). Model with “-” indicates it is the hold-out network. We compare our AI-FGSM and HMCAM with FGSM, PGD and M-PGD (MI-FGSM+PGD), respectively.

Methods	Natural	PGD-20 Attack	M-PGD-20 Attack	CW Attack	Speed (mins)
Natural train	93.78%	0.00%	0.00%	0.00%	47
PGD-10 [9]	84.96%±0.12%	41.58%±0.11%	39.47%±0.27%	58.88%±0.33%	132
Free-8 [48]	82.44%±0.37%	42.07%±0.44%	41.88%±0.53%	57.02%±0.22%	110
YOPO-5-3 [49]	82.65%±0.75%	42.56%±0.83%	41.85%±0.44%	56.93%±0.71%	66
CAT (Ours)	81.54%±0.31%	49.37%±0.27%	48.56%±0.09%	61.28%±0.29%	114

TABLE 4

Validation accuracy and robustness of Preact-ResNet18 on CIFAR10. The maximum perturbation of all the attackers is $\varepsilon = 8/255$. We report average over 5 runs on a single NVIDIA GeForce GTX XP GPU. The best result under different attack methods is in bold.

Methods	Natural	PGD-20 Attack	M-PGD-20 Attack	CW Attack	Speed (mins)
Natural train	94.58%	0.00%	0.00%	0.00%	212
PGD-10 [9]	87.11%±0.37%	48.4%±0.22%	44.37%±0.11%	45.91%±0.14%	2602
Free-8 [48]	84.29%±1.44%	47.8%±1.32%	47.01%±0.19%	46.71%±0.22%	646
YOPO-5-3 [49]	84.72%±1.23%	46.4%±1.49%	47.24%±0.25%	47.5%±0.37%	457
CAT (Ours)	85.39%±0.33%	53.3%±0.64%	52.41%±0.18%	52.55%±0.2%	672

TABLE 5

Validation accuracy and robustness of Wide ResNet34 on CIFAR10. The maximum perturbation of all the attackers is $\varepsilon = 8/255$. We report average over 5 runs on a single NVIDIA GeForce GTX XP GPU. The best result under different attack methods is in bold.

parameters of model. We calculate the deviation value of the last 5 evaluations and report the average over 5 runs. Results about Preact-ResNet18 and Wide ResNet34 on CIFAR10 are shown in Table 4 and Table 5, respectively. Our CAT method greatly boost the robust accuracy in a reasonable training speed.

We also present a comparison in terms of both clean accuracy and robust accuracy per iteration on all methods evaluated during training in Figure. 5. When compared with YOPO, the robust accuracy of our CAT method rises steadily and quickly while YOPO vibrates greatly and frequently.

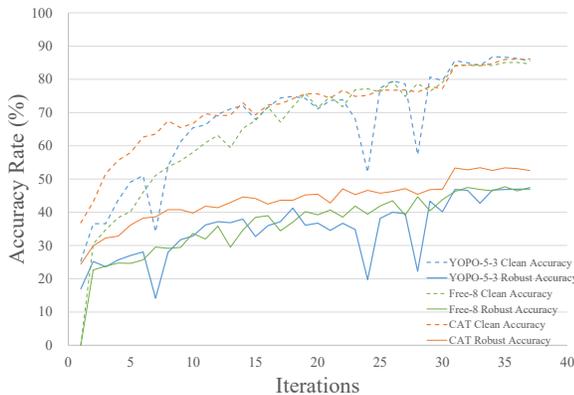


Fig. 5. Comparison with different adversarial training methods on both clean accuracy and robust accuracy (against PGD-10 with $\varepsilon = 8/255$) of Wide ResNet34 on CIFAR10 at every iteration.

For ImageNet, we report the average results over last

three runs. Comparison between free adversarial training and ours are shown in Table 6. Although the 2-PGD trained ResNet-50 model still maintains its leading role in the best robust accuracy, it takes three times longer than our CAT method. Actually, when compared with its high computational cost of ImageNet training, this performance gain can be considered inefficient or even impractical for resource limited entities. We also compare ResNet-50 model trained by our CAT method with the Free-4 trained, model trained by CAT produces much more robust models than Free-4 against different attacks in almost the same order of time.

We also investigate our CAT method on MNIST. We choose a simple ConvNet with four convolutional layers followed by three fully connected layers, which is of the same as [49]. For PGD adversarial training, we train the models for 55 epochs. The initial learning rate is set to 0.1, reduced by 10 times at epoch 45. We use a batch size of 256, a weight decay of $5e-4$ and a momentum of 0.9. For evaluating, we perform a PGD-40 and CW attack against our model and set the size of perturbation as $\varepsilon = 0.3$ based on L_∞ norm as a common practice [9], [49], [72]. Results are shown in Table 7.

5.6 Competitions and Real World Systems Attack

5.6.1 Attack CAAD 2018 Defense Champion

Adversarial Attacks and Defenses (CAAD) 2018 is an open competition involving an exciting security challenge which stimulate the interest of a wide range of talents from industry and academia on adversarial learning. In the defense track

Methods	Clean Data	PGD-10 Attack	PGD-20 Attack	PGD-50 Attack	MI-FGSM-20 Attack	Speed (mins)
Natural train	75.34%	0.14%	0.06%	0.03%	0.03%	1437
PGD	63.95%	36.89%	36.44%	36.17%	35.29%	8928
Free-4	60.26%	31.12%	30.29%	30.07%	29.43%	2745
CAT (Ours)	59.23%	35.91%	35.72%	35.76%	34.67%	2992

TABLE 6

Validation accuracy and robustness of ResNet50 on ImageNet. The maximum perturbation of all the attackers is $\varepsilon = 4/255$. We report average over the final 3 runs. The best (or almost best) results under different attack methods are in bold. Our CAT achieves a trade-off between efficiency and accuracy.

	Clean Data	PGD-40 Attack	CW Attack
PGD-40	99.50%	97.17%	93.27%
Free-10	98.29%	95.33%	92.66%
YOFO-5-10	99.98%	94.79%	92.58%
CAT (Ours)	99.36%	97.48%	94.77%

TABLE 7

Validation accuracy and robustness of a small CNN on MNIST. The maximum perturbation of all the attackers is $\varepsilon = 0.3$. The best result under different attack methods is in bold.

of CAAD 2018, the champion solution [23] devised new network architectures with novel non-local means blocks and better adversarial training scheme, which greatly surpassed the runner-up approach under a strict criterion. We download the meticulously pretrained models¹ and apply our proposed method to attack the approach with default settings. We compare three attack methods (baseline, M-PGD and our HMCAM) on ResNet152_A, ResNet152_{AD} and ResNeXt101_{AD} with 10/100 attack iterations, where A is denoted as using adversarial training and D presents being equipped with denoising blocks. Results are shown in Table 8. Note that the baseline attack method is one of the strongest white-box attacker as recent works [23], [46]. From the Table 8, we can see that M-PGD is ineffective for attacking adversarially trained models with denoising blocks. Our proposed method outperforms both official baseline method and M-PGD. It is worth mentioning that our proposed method also outperforms one of the recent distributionally adversarial attack method DAA [13], which proposes a specific energy functionals combined the cross-entropy loss with the KL-divergence term for better adversarial-sample generation. Actually, DAA can be considered as a special case of the potential energy U .

5.6.2 Attack on Public Face Recognition Systems

To further show the practical applicability of attack, we apply our HMCAM to the real-world celebrity recognition APIs in Clarifai², AWS³ and Azure⁴. These celebrity recognition APIs allow users to upload any face images and recognize the identity of them with confidence score. The users have no knowledge about the dataset and types of models used behind these online systems. We choose 10 pairs of images from the LFW dataset and learn perturbations from local facenet model to launch targeted attack, whose goal is to mislead the API to recognize the adversarial images as our

selected identity. We randomly pick up 10 celebrities as victims from Google and 10 existing celebrities as targets from LFW, ensuring that all colors and genders are taken into account. Then we apply the same strategy as Geekpwn CAAD 2018 method that pulls victims towards their corresponding targets by the inner product of their feature vectors and generates noise to them. Finally, we examine their categories and confidence scores by uploading these adversarial examples to the online systems API.

We fix $\varepsilon = 16/255$ and total iteration number $N = 100$. Besides, we also set $S = 5$ to generate a sequence of adversarial examples to test the robustness of these online systems. Here we propose a *strict evaluation criterion* derived from [23] for our HMCAM attacker, which we also call “*all-or-nothing*”: an attack is considered successful only if **all** the adversarial examples in our generated sequence can deceive the system. This is a challenging evaluation scenario. As shown in Table 9, quite a part of them pass the recognition of the online systems and output the results we want. The qualitative results are given in the supplementary document. Note that we also compare our HMCAM method with one of state-of-the-art black-box attack method \mathcal{N} Attack [25], which aims at finding a probability density distribution around the input and estimates the gradient by a modified NES [73] method. Comparisons between \mathcal{N} Attack and HMCAM show that the samples generated by our proposed method have the stronger transferability since HMCAM is just a white-box attack method.

6 CONCLUSION

In this paper, we formulate the generation of adversarial examples as a MCMC process and present an efficient paradigm called Hamiltonian Monte Carlo with Accumulated Momentum (HMCAM). In contrast to traditional iterative attack methods that aim to generate a single optimal adversarial example in one run, HMCAM can efficiently explore the distribution space to search multiple solutions and generate a sequence of adversarial examples. We also develop a new generative method called Contrastive Adversarial Training (CAT), which approaches equilibrium distribution of adversarial examples with only few iterations by building from

1. <https://github.com/facebookresearch/ImageNet-Adversarial-Training/blob/master/INSTRUCTIONS.md>

2. <https://clarifai.com/models/celebrity-image-recognition-model-e466caa0619f444ab97497640cefc4dc>

3. <https://aws.amazon.com/blogs/aws/amazon-rekognition-update-celebrity-recognition/>

4. <https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/>

Methods	10/100-step Success Rate (%)		
	ResNet152 _A	ResNet152 _{AD}	ResNeXt101 _{AD}
PGD [9]	5.48/31.04	4.93/27.65	5.00/31.56
M-PGD [10]	4.01/24.63	3.51/22.07	3.44/23.78
DAA [13]	4.23/27.31	4.17/24.69	4.55/28.07
HMCAM (Ours)	17.29/35.07	14.69/31.52	17.54/36.36

TABLE 8

The success rates of targeted white-box attacks on ImageNet. The maximum perturbation is $\epsilon = 16/255$. We report three advanced adversarial attacks and our HMCAM on adversarially trained models with (ResNet152_{AD}/ResNeXt101_{AD}) and without (ResNet152_A) feature denoising module.

Methods	Success Cases		
	Clarifai	AWS	Azure
Geekpwn CAAD 2018	3	0	0
\mathcal{N} Attack [25]	2	0	0
HMCAM (Ours)	8	2	1

TABLE 9

The results of our targeted attack on the real-world celebrity recognition APIs in Clarifai, AWS and Azure. We randomly selected 10 pairs of images and adopt a strict criterion called “all-or-nothing” for our HMCAM attacker, which means that the success case counts only if **all** the adversarial examples in our generated sequence can fool the systems. The maximum perturbation is $\epsilon = 16/255$.

small modifications of the standard Contrastive Divergence. Extensive results with comparisons on CIFAR10 showed that not only HMCAM attained much higher success rates than other black-box models and comparable results as other white-box models in adversarial attack, but also CAT achieved a trade-off between efficiency and accuracy in adversarial training. By further evaluating this enhanced attack against the champion solution in the defense track of CAAD 2018 competition, HMCAM outperforms the official baseline attack and M-PGD. To demonstrate its practical applicability, we apply the proposed HMCAM method to investigate the robustness of real-world celebrity recognition systems, and compare against the Geekpwn CAAD 2018 method. The result shows that the existing real-world celebrity recognition systems are extremely vulnerable to adversarial attacks in the black-box scenario since most examples generated by our approach can mislead the system with high confidence, which raises security concerns for developing more robust celebrity recognition models. The proposed attack strategy leads to a new paradigm for generating adversarial examples, which can potentially assess the robustness of networks and inspire stronger adversarial learning methods in the future.

REFERENCES

- [1] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016.
- [2] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, “DARTS: deceiving autonomous cars with toxic signs,” *CoRR*, vol. abs/1802.06430, 2018. [Online]. Available: <http://arxiv.org/abs/1802.06430>
- [3] H. Wang, G. Wang, Y. Li, D. Zhang, and L. Lin, “Transferable, controllable, and inconspicuous adversarial attacks on person re-identification with deep mis-ranking,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [4] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. J. Goodfellow, “Adversarial spheres,” in *ICLR*, 2018.
- [5] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, “Pixeldefend: Leveraging generative models to understand and defend against adversarial examples,” in *ICLR*, 2018.
- [6] T. Tanay and L. D. Griffin, “A boundary tilting perspective on the phenomenon of adversarial examples,” *CoRR*, vol. abs/1608.07690, 2016.
- [7] D. Stutz, M. Hein, and B. Schiele, “Disentangling adversarial robustness and generalization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6976–6987.
- [8] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *ICLR*, 2018.
- [10] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, “Boosting adversarial attacks with momentum,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.
- [11] A. N. Bhagoji, W. He, B. Li, and D. Song, “Exploring the space of black-box attacks on deep neural networks,” *arXiv preprint arXiv:1712.09491*, 2017.
- [12] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.
- [13] T. Zheng, C. Chen, and K. Ren, “Distributionally adversarial attack,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 2253–2260.
- [14] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, “Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4322–4330.
- [15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Advances in neural information processing systems*, 2017, pp. 6626–6637.
- [16] F. Farnia and A. Ozdaglar, “Gans may have no nash equilibria,” *arXiv preprint arXiv:2002.09124*, 2020.
- [17] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. [Online]. Available: <http://arxiv.org/abs/1511.06434>
- [18] R. Webster, J. Rabin, L. Simon, and F. Jurie, “Detecting overfitting of deep generative networks via latent recovery,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 273–11 282.
- [19] I. Gulrajani, C. Raffel, and L. Metz, “Towards GAN benchmarks which require generalization,” in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. [Online]. Available: <https://openreview.net/forum?id=HkxKH2AcFm>
- [20] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *ICLR*, 2014.
- [21] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” in *ICLR*, 2017.
- [22] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.
- [23] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, “Feature denoising for improving adversarial robustness,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 501–509.
- [24] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *ICLR*, 2015.

- [25] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong, "Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks," *arXiv preprint arXiv:1905.00441*, 2019.
- [26] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5764–5772.
- [27] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Enhancing robustness of machine learning systems via data transformations," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2018, pp. 1–5.
- [28] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *ICLR*, 2017.
- [29] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. N. R. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- [30] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7167–7177.
- [31] G. Tao, S. Ma, Y. Liu, and X. Zhang, "Attacks meet interpretability: Attribute-steered detection of adversarial samples," in *Advances in Neural Information Processing Systems*, 2018, pp. 7717–7728.
- [32] C. Zhang, Z. Ye, Y. Wang, and Z. Yang, "Detecting adversarial perturbations with saliency," in *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*. IEEE, 2018, pp. 271–275.
- [33] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.
- [34] N. Papernot and P. McDaniel, "Extending defensive distillation," *arXiv preprint arXiv:1705.05264*, 2017.
- [35] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *ICML*, 2018, pp. 274–283.
- [36] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 369–385.
- [37] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," in *ICLR*, 2018.
- [38] H. Wang, G. Wang, G. Li, and L. Lin, "Camdrop: A new explanation of dropout and a guided regularization method for deep neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, p. 1141–1149.
- [39] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 656–672.
- [40] X. Liu, Y. Li, C. Wu, and C. Hsieh, "Adv-bnn: Improved adversarial defense through robust bayesian neural network," in *ICLR*, 2019.
- [41] N. Akhtar, J. Liu, and A. Mian, "Defense against universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3389–3398.
- [42] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *ICLR*, 2015.
- [43] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1778–1787.
- [44] H. Li, G. Li, and Y. Yu, "Rosa: Robust salient object detection against adversarial attacks," *IEEE transactions on cybernetics*, 2019.
- [45] X. He, S. Yang, G. Li, H. Li, H. Chang, and Y. Yu, "Non-local context encoder: Robust biomedical image segmentation against adversarial attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8417–8424.
- [46] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018.
- [47] X. Liu and C.-J. Hsieh, "Rob-gan: Generator, discriminator and adversarial attacker," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [48] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Advances in Neural Information Processing Systems*, 2019, pp. 3353–3364.
- [49] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," in *Advances in Neural Information Processing Systems*, 2019, pp. 227–238.
- [50] R. M. Neal, *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993.
- [51] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.
- [52] R. M. Neal et al., "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.
- [53] C. Pasarica and A. Gelman, "Adaptively scaling the metropolis algorithm using expected squared jumped distance," *Statistica Sinica*, pp. 343–364, 2010.
- [54] T. Salimans, D. Kingma, and M. Welling, "Markov chain monte carlo and variational inference: Bridging the gap," in *International Conference on Machine Learning*, 2015, pp. 1218–1226.
- [55] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1593–1623, 2014.
- [56] Z. Tu and S.-C. Zhu, "Image segmentation by data-driven markov chain monte carlo," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 657–673, 2002.
- [57] T. Chen, E. Fox, and C. Guestrin, "Stochastic gradient hamiltonian monte carlo," in *International conference on machine learning*, 2014, pp. 1683–1691.
- [58] J. Song, S. Zhao, and S. Ermon, "A-nice-mc: Adversarial training for mcmc," in *Advances in Neural Information Processing Systems*, 2017, pp. 5140–5150.
- [59] J. Xie, Y. Lu, R. Gao, and Y. N. Wu, "Cooperative learning of energy-based model and latent variable model via mcmc teaching," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [60] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
- [61] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *ICLR*, 2018.
- [62] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
- [63] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [64] Y. LeCun, "The mnist database of handwritten digits," <http://yann.lecun.com/exdb/mnist/>, 1998.
- [65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [66] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [67] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.
- [68] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [69] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.0.1," *CoRR*, vol. 1807.01069, 2018. [Online]. Available: <https://arxiv.org/pdf/1807.01069>
- [70] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *ICLR*, 2017.
- [71] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2730–2739.
- [72] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 7472–7482.
- [73] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, and J. Schmidhuber, "Natural evolution strategies," 2011.



Hongjun Wang (S'20) received his B.E. degree of information security from Sun Yat-Sen University, Guangzhou, China, in 2018. He is currently working toward the M.E. degree at Sun Yat-Sen University. His current research interests include computer vision and the security of machine learning, particularly in adversarial attacks and defenses.



Guanbin Li (M'15) is currently an associate professor in School of Data and Computer Science, Sun Yat-sen University. He received his PhD degree from the University of Hong Kong in 2016. His current research interests include computer vision, image processing, and deep learning. He is a recipient of ICCV 2019 Best Paper Nomination Award. He has authorized and co-authored on more than 60 papers in top-tier academic journals and conferences. He serves as an area chair for the conference of VISAPP.

He has been serving as a reviewer for numerous academic journals and conferences such as TPAMI, IJCV, TIP, TMM, TCyb, CVPR, ICCV, ECCV and NeurIPS.



Xiaobai Liu is currently an Associate Professor of Computer Science in the San Diego State University (SDSU), San Diego. He received his PhD from the Huazhong University of Science and Technology, China. His research interests focus on scene parsing with a variety of topics, e.g. joint inference for recognition and reconstruction, commonsense reasoning, etc. He has published 60+ peer-reviewed articles in top-tier conferences (e.g. ICCV, CVPR etc.) and leading journals (e.g. TPAMI, TIP etc.). He received a

number of awards for his academic contribution, including the 2013 outstanding thesis award by CCF(China Computer Federation). He is a member of IEEE.



Liang Lin (M'09, SM'15) is a full Professor of Sun Yat-sen University. He is an Excellent Young Scientist of the National Natural Science Foundation of China. From 2008 to 2010, he was a Post-Doctoral Fellow at the University of California, Los Angeles. From 2014 to 2015, as a senior visiting scholar, he was with The Hong Kong Polytechnic University and The Chinese University of Hong Kong. He currently leads the SenseTime R&D teams to develop cutting-edge and deliverable solutions on computer vision, data analysis and

mining, and intelligent robotic systems. He has authored and co-authored more than 100 papers in top-tier academic journals and conferences. He has been serving as an associate editor of IEEE Trans. Human-Machine Systems, The Visual Computer and Neurocomputing. He served as area/session chairs for numerous conferences, such as ICME, ACCV, ICMR. He was the recipient of the Best Paper Runners-Up Award in ACM NPAR 2010, the Google Faculty Award in 2012, the Best Paper Diamond Award in IEEE ICME 2017, and the Hong Kong Scholars Award in 2014. He is a Fellow of IET.