IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received December 23, 2021, accepted February 14, 2022, date of publication February 21, 2022, date of current version March 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3153036

RAILS: A Robust Adversarial Immune-Inspired Learning System

REN WANG[1], TIANQI CHEN[2], STEPHEN M. LINDSLY[3], COOPER M. STANSBURY[3], ALNAWAZ REHEMTULLA[4], INDIKA RAJAPAKSE[3], AND ALFRED O. HERO III[1], (Life Fellow, IEEE)

[1]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA
[2]Department of Statistics, University of Michigan, Ann Arbor, MI 48109, USA
[3]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA
[4]Department of Radiation Oncology, University of Michigan, Ann Arbor, MI 48109, USA

Corresponding authors: Alfred O. Hero III (hero@eecs.umich.edu) and Indika Rajapakse (indikar@umich.edu)

This work was supported in part by the United States Department of Defense, Defense Advanced Research Projects Agency (DARPA) under Grant HR00112020011 and by the Army Research Office (ARO) under grants W911NF-15-1-0479 and W911NF-19-1-0269.

This work involved human subjects or animals in its research. All ethical and experimental procedures and protocols using mouse models were approved by the University of Michigan Institutional Animal Care and Use Committee (IACUC), as described in Protocol PRO00010270.

ABSTRACT Adversarial attacks against deep neural networks (DNNs) are continuously evolving, requiring increasingly powerful defense strategies. We develop a novel adversarial defense framework inspired by the adaptive immune system: the Robust Adversarial Immune-inspired Learning System (RAILS). Initializing a population of exemplars that is balanced across classes, RAILS starts from a uniform label distribution that encourages diversity and uses an evolutionary optimization process to adaptively adjust the predictive label distribution in a manner that emulates the way the natural immune system recognizes novel pathogens. RAILS' evolutionary optimization process explicitly captures the tradeoff between robustness (diversity) and accuracy (specificity) of the network, and represents a new immune-inspired perspective on adversarial learning. The benefits of RAILS are empirically demonstrated under eight types of adversarial attacks on a DNN adversarial image classifier for several benchmark datasets, including: MNIST; SVHN; CIFAR-10; and CIFAR-10. We find that PGD is the most damaging attack strategy and that for this attack RAILS is significantly more robust than other methods, achieving improvements in adversarial robustness by $\geq$ 5.62%, 12.5%, 10.32%, and 8.39%, on these respective datasets, without appreciable loss of classification accuracy. Codes for the results in this paper are available at https://github.com/wangren09/RAILS.

INDEX TERMS Bio-inspired deep learning, adversarial robustness, deep network defense strategies.

I. INTRODUCTION

The state of the art in supervised deep learning has dramatically improved over the past decade [1]. Deep learning techniques have led to significant advances in applications such as: face recognition [2]; object detection [3]; and natural language processing [4]. Despite these successes, deep learning techniques are not resilient to evasion attacks (a.k.a. adversarial attacks) on test inputs and poisoning attacks on training data [5]–[7]. The adversarial vulnerability of deep neural networks (DNN) have restricted their application, motivating researchers to develop effective defense methods. The focus of this paper is to develop a novel deep defense framework inspired by the mammalian immune system.

The associate editor coordinating the review of this manuscript and approving it for publication was Kostas Kolomvatsos.

Current adversarial defense strategies can be divided into four classes: (1) detection of adversarial samples [8]–[10]; (2) Robust training [11]–[14]; (3) data denoising and reconstruction [15], [16]; and (4) deep adversarial learning architectures [17], [18]. The first class of methods defends a DNN using simple outlier detection models for detecting adversarial examples. However, it has been shown that such adversarial detection methods can be easily defeated [19]. Robust training aims to harden the model to deactivate attacks such as evasion attacks. Known robust training methods are often tailored to a certain level of attack strength in the context of ℓ_p-perturbation. Moreover, the trade-off between accuracy and robustness presents design challenges [12]. The data denoising and reconstruction class of methods is driven by an intuitive idea that adversarial examples can be mapped to the manifold of clean examples through data reconstruction by denoising. However, while denoising can reduce adversarial perturbations it can also distort the inputs [20], providing new opportunities for an attacker to exploit the defense mechanism [21]. Deep adversarial learning architectures directly design the defense into the layers of the neural network, e.g., by robustifying them with k-NN's [17] or modifying a generative adversarial network (GAN) [18]. Despite these advances, current methods still have difficulty providing an acceptable level of robustness to novel attacks [22].

To design an effective defense, it is natural to consider a learning strategy that emulates mechanisms of the naturally robust biological immune system. The power of artificial immune system (AIS) models have been established in many other applications [23]–[25]. While AIS approaches to enhancing DNN adversarial robustness have been previously developed [26], [27], they have been restricted to simple emulations of the innate immune system. In this paper, we propose a new framework, Robust Adversarial Immune-inspired Learning System (RAILS), that can effectively defend deep learning architectures against aggressive attacks based on a refined biology model of the adaptive immune system. Built upon a class-wise k-Nearest Neighbor (kNN) structure, given a test sample RAILS finds an initial small population of proximal samples, balanced across different classes, with uniform label distribution. RAILS then promotes the specificity of the label distribution towards the ground truth label through an evolutionary optimization. RAILS can efficiently correct adversarial label flipping by balancing label diversity against specificity. While RAILS can be applied to defending against many different types of attacks, in this paper we restrict attention to evasion attacks on the input. Figure 1 shows that RAILS outperforms existing methods on various types of evasion attacks.

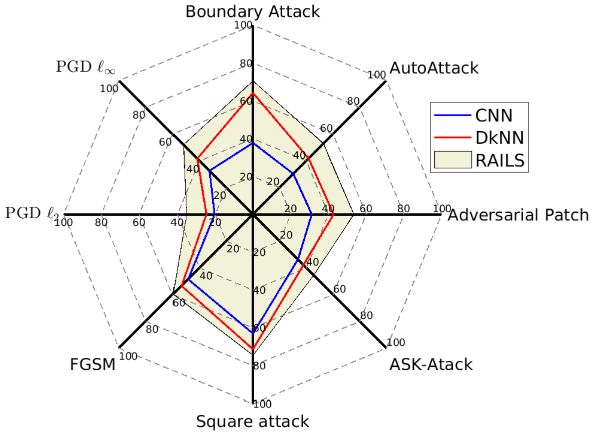


FIGURE 1. RAILS launches the best defense against different types of attacks. Radar plot showing that RAILS has higher robust accuracy than the adversarially trained CNN [11] and Deep k-Nearest Neighbor (DkNN) [17] in defending against eight types of attacks: ℓ_∞-PGD attack/ ℓ_2-PGD attack [11], Fast Gradient Sign Method (FGSM) [5], Square Attack [28], Adversarial Patch [29], AutoAttack [30], Boundary Attack [31], and a (customized) ASK-Attack [32]. The benign accuracy for CNN, DkNN, and RAILS are 87.26%, 86.63%, and 82%. We refer readers to Section IV for more results.

Compared to existing defense methods, we make the **following contributions**:

- RAILS achieves better adversarial robustness by assigning a uniform label distribution to each input and evolving it to a distribution that is concentrated about the input's true label class. (see Section II and Table 2)
- RAILS incorporates a life-long robustifying process by adding synthetic "virtual data" to the training data. (see Section II and Table 5)
- RAILS evolves the distribution via mutation and crossover mechanisms and is not restricted to ℓ_p or any other specific type of attack. (see Section III and Figure 1)
- We demonstrate that RAILS improves robustness of existing methods for different types of attacks (Figure 1). In particular, RAILS improves robustness against the highly damaging PGD attack by $\geq$ 5.62%/12.5%/10.32%/8.39% for the MNIST, SVHN, CIFAR-10 and CIFAR-100 datasets (Table 2). Furthermore, we show that the RAILS implementation of life-long learning with training data augmentation yields a 2.3% robustness improvement with only 5% augmentation of the training data (Table 5).
- RAILS is the first adversarial defense framework to be based on the biology of the adaptive immune system. In particular: (a) RAILS computationally emulates the principal mechanisms of the immune response (Figure 2); and (b) our computational and biological experiments demonstrate the fidelity of the emulation - the learning patterns of RAILS and the immune system are closely aligned (Figure 7).

A. RELATED WORK

After it was established that DNNs were vulnerable to evasion attacks [6], different types of defense mechanisms have been proposed. An intuitive idea is to eliminate the adversarial examples through outlier detection, including training an additional discrimination sub-network [8], [10] and using kernel density estimation [9]. The above approaches rely on the fundamental assumption that the distributions of benign and adversarial examples are easily distinguishable, an assumption that has been challenged in [19].

In addition to adversarial attack detection, other methods have been proposed that focus on robustifying the deep architecture during the learning phase [11]–[13]. One recent approach combines training with perturbed inputs and hierarchical feature alignment between the adversarial and clean domains to robustify the feature learning process [14]. Though such defenses are effective against adversarial examples with moderate levels of ℓ_p attack strength, they have limited power to defend against stronger attacks, and there is often a sacrifice in overall classification accuracy. In contrast, RAILS is developed to defend against diverse powerful attacks with less sacrifice in accuracy, and can improve any model's robustness, including robust models.

A different family of defenses models adversarial inputs as deviating from the manifold of clean data. This motivates the use of projection methods for denoising the inputs, where the inputs are mapped to the manifold [15], [16]. Examples include mapping adversarial examples to a high-resolution manifold using wavelet denoising and super resolution tech-

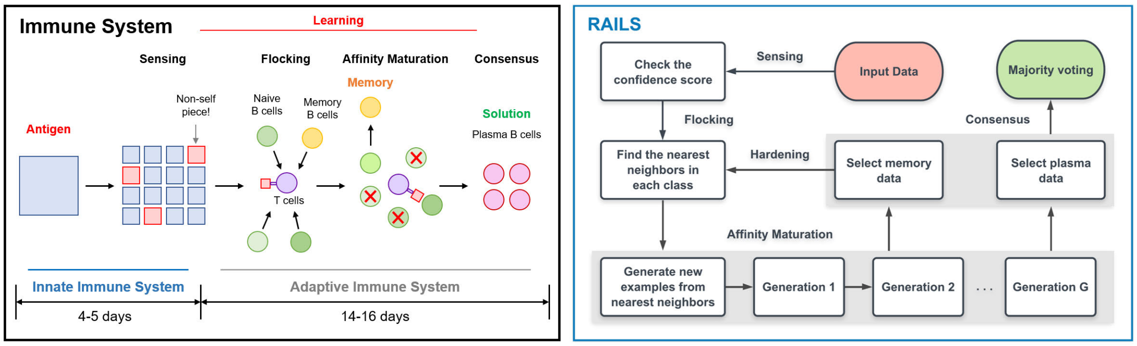


FIGURE 2. **Simplified immune system (left) and RAILS computational workflow (right): Both systems are composed of a four-step process, which includes initial detection (sensing), recruiting candidates for diversity (flocking), enlarging population size and promoting specificity (affinity maturation) [33], and obtaining the final solution (consensus).**

niques [15] and to a low dimensional quasi-natural space with a sparse transformation layer [16]. Despite the simple and clear motivation, denoising methods have their own limitations. For example, they can introduce distortions into clean inputs [20] and have shown to fail to be robust to many adversarial attacks [21], [34]. Instead of attempting to modify inputs, RAILS evolves a statistical population of clones of the input, resulting in enhancing resilience to attacks.

Another approach is to incorporate different architectures to robustify deep classifiers [17], [18]. An example is the deep k-Nearest Neighbor (DkNN) classifier [17] that robustifies against instance perturbations by applying kNN's to features extracted from each layer. However, a single kNN classifier applied on the whole dataset is easily to be fooled by strong attacks (Figure 4). Conversely, RAILS incorporates an evolutionary *diversity* to *specificity* defense mechanism which can provide additional robustness to existing DNNs.

Network defense mechanisms inspired by the natural immune system have been proposed for other applications, different from the deep learning application considered in this paper. Among these, artificial immune system (AIS) approaches [35] have been used to defend against wormhole attacks on mobile ad hoc networks [23], flooding attacks on software-defined networks [24], and denial of service attacks on the internet [25]. However, the closest point of tangency to our RAILS approach is recent work that borrows concepts from the innate immune system to detect adversarial examples in DNN's. The innate immune system, also known as the non-specific immune system, is nature's first line of defense that launches an immediate non-specific response to contain the pathogen using chemical cellular, and extracellular mechanisms to prevent pathogen mobility and spread. Mechanisms of innate immunity that have been emulated in machine learning include: negative selection algorithm approaches [26]; and cellular machanisms for early pathogen recognition [27]. Different from the innate immune system, the response of the adaptive immune system is longlasting, specific and sustained, using clonal expansion to produce B-cell and T-cell lymphocytes having antigen receptors specific to the pathogen. To the best of our knowledge, the proposed RAILS adversarial defense framework is the first to be based on the complex biology of the adaptive immune system.

Another line of research relevant to ours is adversarial transfer learning [36], [37], which aims to maintain robustness when there is covariate shift between training data and test data. We remark that covariate shift is naturally handled by the mutation mechanism in our adaptive immune system emulation of RAILS that adapts the defense to novel attacks.

B. LEARNING STRATEGIES OF IMMUNE SYSTEM

Systems robustness is a property that must be intentionally designed into the architecture, and one of the greatest examples of this is the mammalian adaptive immune system [38]. The adaptive immune system is incredibly complex and not something that we can hope to replicate at this time. However, we can simplify its robust learning process into these four steps: sensing, flocking, affinity maturation, and consensus (Figure 2 left) [39], [40]. The architecture of the adaptive immune system ensures a robust response to foreign antigens, splitting the work between active sensing and competitive growth to produce an effective antibody. *Sensing* of a foreign attack leads to antigen-specific B-cells *flocking* to some temporary structures for *affinity maturation* [33]. In the affinity maturation phase, a diverse initial set of antigen-specific B-cells divide to populate the temporary structures. Then the genetic identity of each B-cell is encoded by the shape and sequence of its protein, which can change from generation to generation. The degree to which the encoding of the B-cell matches the antigen is called the affinity. The B-cells with the highest affinity to the antigen are selected to divide and mutate, which leads to new B-cells with higher affinity to the antigen [41]. B-cells that reach *consensus*, or achieve a threshold affinity against the foreign antigen, undergo terminal differentiation into plasma B-cells. Plasma B-cells represent the antigen-specific solutions. Memory B-cells are selected and stored to defend against similar attacks in the future.

C. NOTATION AND PRELIMINARIES

Given a mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ and $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, we first define the affinity score $A(f; \mathbf{x}_1, \mathbf{x}_2)$ between $\mathbf{x}_1$

and $\mathbf{x}_2$. This affinity score can be defined in many ways, e.g, cosine similarity, inner product, or inverse ℓ_p distance, but here we use $A(f; \mathbf{x}_1, \mathbf{x}_2) = -\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\|_2$, the negative Euclidean distance. In the context of DNN, f denotes the feature mapping from input to feature representation, and A measures the similarity between two inputs. Higher affinity score indicates higher similarity.

II. RAILS: OVERVIEW

In this section, we give an overview of RAILS, and provide a comparison to the natural immune system. The architecture of RAILS is illustrated in Figure 3. For each selected hidden layer l, RAILS builds class-wise kNN architectures on training samples $\mathcal{D}_{tr}$. Then for each test input $\mathbf{x}$, a population of candidates is selected and goes through an evolutionary optimization process to obtain the optimal solution. In RAILS, two types of data are obtained after the evolutionary optimization: 'plasma data' for optimal predictions of the present inputs, and 'memory data' for the defense against future attacks. These two types of data correspond to plasma B cells and memory B cells in the biological system, and play important roles in *static learning* and *adaptive learning*, respectively.

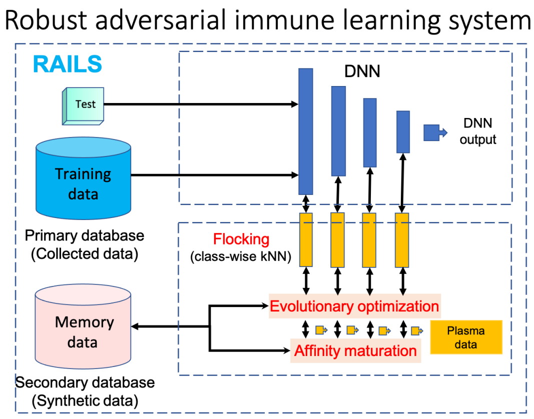

Robust adversarial immune learning system
RAILS
Test
Training data
Primary database (Collected data)
Memory data
Secondary database (Synthetic data)
DNN
DNN output
Flocking (class-wise kNN)
Evolutionary optimization
Affinity maturation
Plasma data

FIGURE 3. **The architecture of RAILS. For each test input, a special type of data - plasma data - is generated by evolutionary optimization, that contributes to predicting the class of the input (test) sample. Another type of data - memory data - is generated and stored to help defend against similar attacks in the future. Plasma data and memory data are analogous to plasma B cells and memory B cells in the immune system.**

A. DEFENSE WITH STATIC LEARNING

Adversarial perturbations can severely affect deep classifiers, forcing the predictions to be dominated by adversarial classes rather than the ground truth. For example, a single kNN classifier is vulnerable to adversarial inputs, as shown in Figure 4. The purpose of static learning is to address this issue, i.e., maintaining or increasing the prediction probability of the ground truth $p_l^{y_{\text{true}}(\mathbf{x})}(\mathbf{x})$ when the input $\mathbf{x}$ is manipulated by an adversary. The key components include **(i)** a label initialization via class-wise kNN search on $\mathcal{D}_{tr}$ that guarantees labels across different classes are uniformly distributed for each input; and **(ii)** an evolutionary data-label optimization that promotes label distribution specificity towards the input's true label class. Our hypothesis is that the covariate shift of the adversarial examples from the distribution of the ground truth class is small in the input space, and therefore new examples inherited from parents of ground truth class y_{true} have higher chance of reaching high-affinity. The evolutionary optimization thus promotes the label *specificity* towards the ground truth. The solution denotes the data-label pairs of examples with high-affinity to the input, which we call plasma data. After the process, a majority vote of plasma data is used to make the class prediction. We refer readers to Section III for more implementation details and Section IV-A for visualization. In short, static learning defenses seek to correct the predictions of current adversarial inputs and do not plan ahead for future attacks.

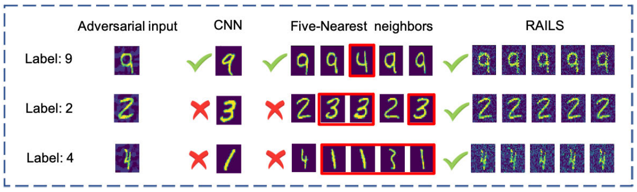


FIGURE 4. **Representation examples showing RAILS correcting (improving) the wrong (unconfident) predictions by CNN and kNN. kNN predict all three examples with ≤ 100% confidence of the ground truth class, and CNN gets wrong predictions for images 2 and 4. RAILS provides correct predictions with high confidence scores.**

B. DEFENSE WITH ADAPTIVE LEARNING

Different from static learning, RAILS adaptive learning tries to use information from past attacks to harden the classifier to defend against future attacks. Hardening is done by leveraging another set of data - memory data generated during evolutionary optimization. Unlike plasma data, memory data is selected from examples with moderate-affinity to the input, which can rapidly adapt to new variants of the adversarial examples. This selection logic is based on maximizing coverage over future attack-space rather than optimizing for the current attack. Adaptive learning is a life-long learning process and can use hindsight to greatly enhance resilience of $p_l^{y_{\text{true}}(\mathbf{x})}(\mathbf{x})$ to attacks. This paper will focus on static learning and single-stage adaptive learning that implements a single cycle of classifier hardening.

C. A BIOLOGICAL PERSPECTIVE

RAILS is inspired by and closely associated with the biological immune system. The architecture of the adaptive immune system ensures a robust response to foreign antigens to produce an effective antibody. Figure 2 displays a comparison between the immune system workflow and the RAILS workflow. Both systems are composed of a four-step process. For example, RAILS emulates flocking from the immune system by initializing a population of candidates that provide *diversity*, and emulates affinity maturation via an evolutionary optimization process to promote *specificity*.

Similar to the functions of plasma B cells and memory B cells generated in the immune system, RAILS generates plasma data for predictions of the present inputs (immune system defends antigens though generating plasma B cells) and generates memory data for the defense against future attacks (immune system continuously increases its degree of robustness through generating memory B cells). We refer readers to Appendix A for a table of correspondences between RAILS operands and mechanisms in the immune system. In addition, the learning patterns of RAILS and the immune system are closely aligned, as shown in Figure 7.

III. DETAILS ON RAILS WORKFLOW

Algorithm 1 shows the four-step workflow of RAILS. We explain each step below.

A. SENSING

This step performs an initial discrimination between adversarial and benign inputs to prevent the RAILS computation from becoming overwhelmed by false positives, i.e., only implementing the main steps of RAILS on suspicious inputs. While there are many outlier detection procedures that could be used for this step [9], [42], we can exploit the fact that the DNN and kNN applied on hidden layers will tend to make similar class predictions for benign inputs. Thus we propose using an cross-entropy measure to generate an *adversarial threat score* for each input $\mathbf{x}$. In the main text results, we skip the sensing stage since the major benefit from sensing is providing an initial detection. We refer readers to Appendix E for more details.

B. FLOCKING

The initial population from each class needs to be selected with a certain degree of affinity measured using the hidden representations in order to satisfy our hypothesis, as illustrated in Figure 7. By constructing class-wise k-Nearest Neighbor (kNN) architecture, we find the kNN that have the highest initial affinity score to the input data from each class and each selected layer. Mathematically, we select

$$\begin{aligned}&\mathcal{N}_l^c = \{(\hat{\mathbf{x}}, y_c) | R_c(\hat{\mathbf{x}}) \le k, (\hat{\mathbf{x}}, y_c) \in \mathcal{D}_c\}\\ &\text{Given}\\ &A(f_l; \mathbf{x}_i^c, \mathbf{x}) \le A(f_l; \mathbf{x}_j^c, \mathbf{x}) \Leftarrow R_c(i) > R_c(j)\\ &\qquad \forall c \in [C], l \in \mathcal{L}, \quad \forall i, j \in [n_c], \end{aligned} \tag{1}$$

where x is the input. $\mathcal{L}$ is the set of the selected layers. $\mathcal{D}_c$ is the training data from class c and the size $|\mathcal{D}_c| = n_c$. $R_c : [n_c] \to [n_c]$ is a ranking function that sorts the indices based on the affinity score. In the adaptive learning context, if the memory database has been previously populated, flocking will select the nearest neighbors using both the training data and the memory data. The immune system leverages flocking step to find initial B cells and form temporary structures for affinity maturation [33]. Note that in RAILS, the kNN sets $\mathcal{N}_l^c$ are *constructed independently* for each class, thereby ensuring that every class is *fairly represented* in the initial population.

C. AFFINITY MATURATION (EVOLUTIONARY OPTIMIZATION)

As flocking brings diversity to the label distribution of the initial population, the affinity maturation step, in contrast, promotes specificity towards the ground truth class. Here we use evolutionary optimization to generate new examples (offspring) from the existing examples (parents) in the population. The evolution happens within each class *independently*, and new generated examples from different classes are not affected by one another before the consensus stage. The first generation parents in each class are the K nearest neighbors found by (1) in the flocking step, where K is the number of nearest neighbors. Given a total population size TC, the 0-th generation is obtained by copying each nearest neighbor T/K times with random mutations. Given the population of class c in the $(g-1)$-st generation $\mathcal{P}_c^{(g-1)} = [\mathbf{x}_c(1), \mathbf{x}_c(2), \cdots, \mathbf{x}_c(T)] \in \mathbb{R}^{d\times T}$, the candidates for the g-th generation are selected as

$$\hat{\mathcal{P}}_c^{(g-1)} = \mathcal{P}_c^{(g-1)} Z_c^{(g-1)}, \tag{2}$$

where $\hat{\mathcal{P}}_c^{(g-1)}$ denotes the set of randomly selected reproductions of the population at the previous generation $g-1$. These are computed before applying mutation and crossover operations to populate the g-th generation $\mathcal{P}_c^{(g)}$. $Z_c^{(g-1)} \in \mathbb{R}^{T\times T}$ is a binary selection matrix whose columns are independent and identically distributed draws from Mult$(1, \mathbf{P}_c)$, the multinomial distribution with probability vector $\mathbf{P}_c \in [0, 1]^T$. The process can also be viewed as creating new nodes from existing nodes in a Preferential Attachment (PA) evolutionary graph [43], where the details can be viewed in Appendix D. After *selection*, RAILS generates new examples through the operations of *mutation* and *cross-over*, which will be discussed in more detail later. After new examples are generated, RAILS calculates each example's affinity relative to the input. The new examples are associated with labels that are inherited from their parents, which always come from the same class. According to our hypothesis in Section II, examples inherited from parents of the ground truth class y_{true} have a higher chance of reaching high-affinity, and thereby the population members with high-affinity are concentrating about the input's true class.

D. CONSENSUS

Consensus is responsible for the final selection and predictions. In this step, RAILS selects generated examples with high-affinity scores to be plasma data, and examples with moderate-affinity scores are saved as memory data. The selection is based on a ranking function.

$$S_{\text{opt}} = \{(\tilde{\mathbf{x}}, \tilde{y}) | R_g(\tilde{\mathbf{x}}) \le \gamma |\mathcal{P}^{(G)}|, (\tilde{\mathbf{x}}, \tilde{y}) \in \mathcal{P}^{(G)}\}, \tag{3}$$

where $R_g : [|\mathcal{P}^{(G)}|] \to [|\mathcal{P}^{(G)}|]$ is the same ranking function as R_c except that the domain is a set having cardinality equal to that of the final population $\mathcal{P}^{(G)}$. γ is a proportionality parameter and is selected as 0.05 and 0.25 for plasma data and memory data, respectively. Note that the memory data can be selected in each generation. For simplicity, we select memory data only in last generation. Memory data will be saved in the secondary database and used for model hardening.

Given that all examples in the population are associated with a label inherited from their parents, RAILS uses majority voting of the plasma data for prediction of the class label of $\mathbf{x}$.

Algorithm 1 Robust Adversarial Immune-Inspired Learning System (RAILS)

```
Require: Test data point $\mathbf{x}$; Training dataset $\mathcal{D}_{tr} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_C\}$; Number of Classes $C$; Model $M$ with
    feature mapping $f_l(\cdot)$, $l \in \mathcal{L}$; Affinity function $A$.
    First Step: Sensing
 1: Check the threat score given by an outlier detection strategy to detect the threat of $x$.
    Second Step: Flocking
 2: for $c = 1, 2, \ldots, C$ do
 3:   In each layer $l \in \mathcal{L}$, find the k-nearest neighbors $\mathcal{N}_l^c$ of $\mathbf{x}$ in $\mathcal{D}_c$ by ranking the affinity score.
 4: end for
    Third Step: Affinity Maturation
 5: For each layer $l \in \mathcal{L}$, do
 6: Generate $\mathcal{P}_c^{(0)}$ through mutating each of $\mathbf{x}' \in \mathcal{N}_l^c$ $T/K$ times, $\forall c \in [C]$.
 7: for $g = 1, 2, \ldots, G$ do
 8:   for $t = 1, 2, \ldots, T$ do
 9:     Select data-label pairs $(\mathbf{x}_c, y_c)$, $(\mathbf{x}'_c, y_c)$ from $\mathcal{P}_c^{(g-1)}$ based on $\mathbf{P}_c^{(g-1)}$.
10:     $\mathbf{x}_{\text{os}} = Mutation\big(Crossover(\mathbf{x}_c, \mathbf{x}'_c)\big)$; $\mathcal{P}_c^{(g)} \longleftarrow (\mathbf{x}_{\text{os}}, y_c)$.
11:   end for
12: end for
13: Calculate the affinity score $A(f_l; \mathcal{P}^{(G)}, \mathbf{x})$, $\forall c \in [C]$ given $\mathcal{P}^{(G)} = \mathcal{P}_1^{(G)} \bigcup \cdots \bigcup \mathcal{P}_C^{(G)}$.
14: end For
    Fourth Step: Consensus
15: Select the top 5% as plasma data $S_p^l$ and the top 25% as memory data $S_m^l$ based on the affinity scores, $\forall l \in \mathcal{L}$;
    Obtain the prediction $y$ of $\mathbf{x}$ using the majority vote of the plasma data.
16: Output: $y$, the memory data $S_m = \{S_m^1, S_m^2, \cdots, S_m^{|\mathcal{L}|}\}$
```

E. COMPUTATIONAL COST

The computational cost of RAILS is dominated by the flocking and affinity maturation stage. kNN structure construction in flocking is a fixed setup cost that can be handled off-line with fast approximate kNN approximation [44], [45]. There are three strategies for reducing the computational cost of the affinity maturation stage. First, the evolutionary optimization can be replaced by a mean field approximation. Second, parallelization can be used to accelerate the computations since each sample is generated and utilized separately. Third, one can use a more stringent false positive threshold in the sensing step, thereby reducing the number of false positives resulting in a reduction in the downstream computational burden. More discussion can be viewed in Appendix D.

F. OPERATIONS IN THE EVOLUTIONARY OPTIMIZATION

Three operations support the creation of new examples: selection, cross-over, and mutation. The *selection* operation is shown in (2). We compute the selection probability for each candidate through a softmax function.

$$\begin{aligned}\mathbf{P}(\mathbf{x}_i) &= Softmax(A(f_l; \mathbf{x}_i, \mathbf{x})/\tau) \\ &= \frac{\exp\left(A(f_l; \mathbf{x}_i, \mathbf{x})/\tau\right)}{\sum_{\mathbf{x}_j \in S} \exp\left(A(f_l; \mathbf{x}_j, \mathbf{x})/\tau\right)},\end{aligned} \tag{4}$$

where S is the set of data points and $x_i \in S$. $\tau > 0$ is the sampling temperature that controls sharpness of the softmax operation. Given the selection probability $\mathbf{P}$, defined on the current generation in (4), the candidate set $\{(\mathbf{x}_i, y_i)\}_{i=1}^T$ for the next generation is randomly drawn (with replacement).

The *cross-over* operator combines two parents $\mathbf{x}_c$ and $\mathbf{x}'_c$ from the same class, and generates new offspring by randomly selecting each of its elements (pixels) from the corresponding element of either parent. Mathematically,

$$\begin{aligned}\mathbf{x}'_{\text{os}} &= Crossover(\mathbf{x}_c, \mathbf{x}'_c) \\ &= \begin{cases} \mathbf{x}_c^{(i)} & \text{with prob } \frac{A(f_l; \mathbf{x}_c, \mathbf{x})}{A(f_l; \mathbf{x}_c, \mathbf{x}) + A(f_l; \mathbf{x}'_c, \mathbf{x})}, \\ \mathbf{x}_c'^{(i)} & \text{with prob } \frac{A(f_l; \mathbf{x}'_c, \mathbf{x})}{A(f_l; \mathbf{x}_c, \mathbf{x}) + A(f_l; \mathbf{x}'_c, \mathbf{x})} \end{cases} \forall i \in [d],\end{aligned} \tag{5}$$

where i represents the i-th entry of the example and d is the dimension of the example. The *mutation* operation randomly and independently mutates an offspring with probability ρ, adding uniformly distributed noise in the range $[-\delta_{\max}, -\delta_{\min}] \cup [\delta_{\min}, \delta_{\max}]$. The resulting perturbation vector is subsequently clipped to satisfy the domain constraint that examples lie in $[0, 1]^d$.

$$\begin{aligned}\mathbf{x}_{\text{os}} = Mutation(\mathbf{x}'_{\text{os}}) = \text{Clip}_{[0,1]}\big(&\mathbf{x}'_{\text{os}} \\ &+ \mathbf{1}_{[Bernoulli(\rho)]}\mathbf{u}([-\delta_{\max}, -\delta_{\min}] \cup [\delta_{\min}, \delta_{\max}])\big),\end{aligned} \tag{6}$$

where $\mathbf{1}_{[Bernoulli(\rho)]}$ takes value 1 with probability ρ and value 0 with probability $1 - \rho$. $\mathbf{u}([-\delta_{\max}, -\delta_{\min}] \cup [\delta_{\min}, \delta_{\max}])$ is the vector in $\mathbb{R}^d$ having i.i.d. entries drawn from the punctured uniform distribution $\mathcal{U}([-\delta_{\max}, -\delta_{\min}] \cup [\delta_{\min}, \delta_{\max}])$. $\text{Clip}_{[0,1]}(\mathbf{x})$ is equivalent to $\max(0, \min(\mathbf{x}, 1))$.

IV. EXPERIMENTAL RESULTS

We conduct experiments in the context of image classification using several benchmark image classification datasets. We compare RAILS with Convolutional Neural Network (CNN) Classification and Deep k-Nearest Neighbors (DkNN) Classification [17] on the MNIST [46], SVHN [47], CIFAR-10 and CIFAR-100 [48] datasets. We test our framework using a four-convolutional-layer neural network for MNIST, VGG16 [49] for the SVHN dataset, and adversarially trained VGG16 for the CIFAR-10 and

CIFAR-100 datasets. We refer readers to Appendix B for more details of the datasets, models, and parameter selection. In addition to the benign test examples, we also generate an equal number of adversarial examples generated from adversarial attacks using MNIST, SVHN, CIFAR-10, and CIFAR-100 data. The attack strength is $\epsilon = 40/60$ for MNIST, and $\epsilon = 8$ for SVHN, CIFAR-10, CIFAR-100 by default. Performance comparisons are based on standard accuracy (SA) evaluated using benign (unperturbed) test examples and robust accuracy (RA) evaluated using the adversarial (perturbed) test examples. Besides the default ℓ_∞ norm PGD attack [11], we also implement seven other attacks (Figure 1 and Table 3).

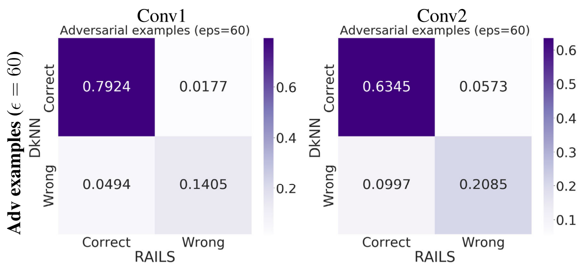


FIGURE 5. RAILS has fewer incorrect predictions for those data that DkNN gets wrong. Confusion Matrices of adversarial examples classification in Conv1 and Conv2 (RAILS vs. DkNN).

A. PERFORMANCE IN SINGLE LAYERS

We first test RAILS in a single layer of the CNN model and compare the obtained accuracy with the results from the DkNN. Table 1 shows the comparisons in the input layer, the first convolutional layer (Conv1), and the second convolutional layer (Conv2) on MNIST. One can see that for both standard accuracy and robust accuracy, RAILS performs better than the DkNN in the hidden layers and achieve better results in the input layer. The input layer results indicate that RAILS can also outperform supervised learning methods like kNN. The confusion matrices in Figure 5 show that RAILS has fewer incorrect predictions for those data that DkNN gets wrong. Each value in Figure 5 represents the percentage of intersections of RAILS (correct or wrong) and DkNN (correct or wrong).

B. RAILS LEARNING PROCESS

Flocking provides a balanced initial population while affinity maturation within RAILS creates new examples in each generation. To better understand the capability of RAILS, we can visualize the changes of some key indices during runtime.

1) VISUALIZATION OF RAILS EVOLUTIONARY PROCESS

Picking the top 5% data points with the highest affinity in each generation, Figure 6 shows the evolution over ten generations of RAILS samples of the population (B-cells) proportion and (exponentiated) affinity relative to two clean (non-adversarial) input examples taken from CIFAR-10. RAILS makes the correct "bird" predictions while the DkNN makes incorrect predictions for both examples. The second column depicts the proportion of the true class in the selected population of each generation. Data from the true class occupies the majority of the population when the generation number increases, which indicates that RAILS can obtain a correct prediction and a high confidence score simultaneously. Meanwhile, affinity maturation over multiple generations yields increasing affinity within the true class, as shown in the third column. To visualize changes in feature distribution during the affinity maturation stage, we show in Figure 8 the two-dimensional t-distributed stochastic neighbor embedding (t-SNE) of the feature representations of adversarial CIFAR-10 inputs (antigens) and the associated populations (B-cells). The features shown in the figure are those of convolutional layer three, and are representative of the feature behavior at other layers. As shown in Figure 8, the antigen is misclassified and B-cells are uniformly spread over the feature space at the beginning of the affinity maturation. As the affinity maturation process progresses, the antigen's ground truth class B-cell population (colored in blue) converges to a cluster that covers the antigen.

2) IN-VITRO B-CELL EXPERIMENT CONFIRMS RAILS EMULATION

To demonstrate that the proposed RAILS computational system captures important properties of the actual (in-vitro) immune system we compare the learning curve of RAILS to the learning curve of B-cell antigen recognition (see Appendix C for a description of the biological experiment we performed). For the biological experiment the measured affinity between a population of actual B-cells and an antigen is obtained experimentally over time (several hours). For RAILS each test input (potentially the adversarial example) is treated as an antigen and the affinity is computed as RAILS iterates over time. Figure 7 shows that both the in-vitro immune system and RAILS have similar learning patterns. One can also see that the affinity increases again after the decrease, indicating both the immune system and RAILS can escape from a local optimal under strong attacks. The difference between the green and red curves is that the initial population for the red curve is found based on another test input (antigen), which has lower correlation to the current input (antigen). The non-convergence of the red curve indicates that the initial population should be selected close to the input, and the flocking using kNN search emulated the

TABLE 1. RAILS outperforms DkNN on single layers. Standard Accuracy (SA)/Robust Accuracy (RA) performance of RAILS versus DkNN in single layer (MNIST).

		Input	Conv1	Conv2
SA	**RAILS**	**97.53%**	**97.77%**	**97.78%**
	DkNN	96.88%	97.4%	97.42%
RA	**RAILS**	**93.78%**	**92.56%**	**89.29%**
($\epsilon = 40$)	DkNN	91.81%	90.84%	88.26%
RA	**RAILS**	**88.83%**	**84.18%**	**73.42%**
($\epsilon = 60$)	DkNN	85.54%	81.01%	69.18%

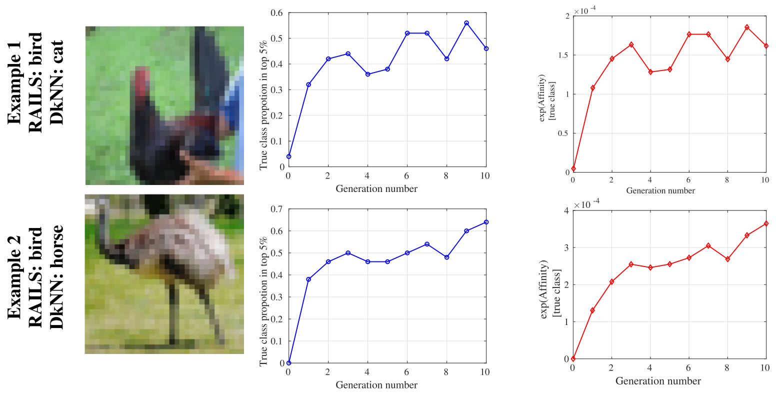


FIGURE 6. Proportion and affinity of the population from the ground truth class of input with respect to the generation number (RAILS on CIFAR-10). We plot all curves by selecting data points with affinity in the top 5% of all classes' data points in each generation. (1) Second column: Data from the true class occupies the majority of the population when the generation number increases (2) Third column: Affinity maturation over multiple generations produces increased affinity (after a temporary decrease in the searching phase) within the true class.

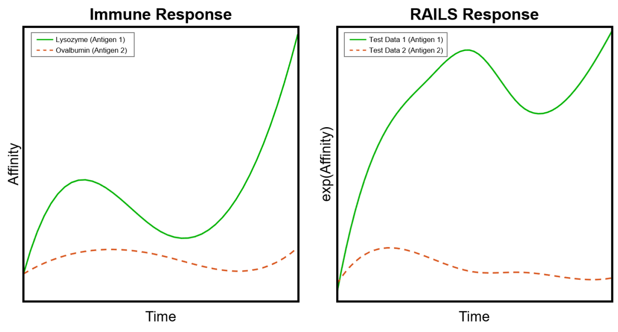


FIGURE 7. RAILS emulates the learning patterns of the biological adaptive immune response. Correspondence between learning curves of a natural immune system *in vitro* experiment (left) and the RAILS computational experiment (right). Adversarial input in RAILS corresponds to antigen in the immune system. When the initial candidates are selected based on the input (the green lines), RAILS and the immune system can both jump out of local optimal and find the correct solution. When the candidates are selected based on a different input (the red dashed lines), neither responses converge.

natural flocking process. We refer readers to Appendix C for more details.

C. OVERALL PERFORMANCES IN DIFFERENT SCENARIOS

1) DEFENSE AGAINST PGD ATTACK FOR CIFAR-10 AND SVHN

We compare RAILS with CNN and DkNN in terms of standard accuracy (SA) and robust accuracy (RA). The results are shown in Table 2. On MNIST with $\epsilon = 60$, one can see that RAILS delivers a 5.62% improvement in RA over DkNN without appreciable loss of SA. On CIFAR-10 (SVHN), RAILS leads to 10.32% (12.5%) and 19.4% (46%) robust accuracy improvements compared to DkNN and CNN, respectively. We refer readers to Appendix F for more results under different strengths and types of attacks. Note that there is no competitive relationship between RAILS and robust

TABLE 2. RAILS achieves higher robust accuracy (RA) at small cost of standard accuracy (SA) for all three datasets (MNIST, SVHN and CIFAR-10) as compared to CNN and DkNN.

		SA	RA
MNIST	**RAILS (ours)**	97.95%	**76.67%**
($\epsilon = 60$)	CNN	**99.16%**	1.01%
	DkNN	97.99%	71.05%
SVHN	**RAILS (ours)**	90.62%	**48.26%**
($\epsilon = 8$)	CNN	**94.55%**	1.66%
	DkNN	93.18%	35.7%
CIFAR-10	**RAILS (ours)**	82%	**52.01%**
($\epsilon = 8$)	CNN	**87.26%**	32.57%
	DkNN	86.63%	41.69%

training since RAILS is a general method that can improve any models' robustness, even a robust trained model.

2) DEFENSE AGAINST EIGHT DIFFERENT ATTACK TYPES

Here we show the results of RAILS defending against eight types of attacks: (1,2) ℓ_∞-PGD attack and ℓ_2-PGD attack [11] (3) Fast Gradient Sign Method (FGSM) [5], a fast alternative version of PGD (4) Square Attack (Sq-Attack) [28], a type of score-based black-box attack (5) Boundary Attack [31], a type of decision-based black-box attack (6) AutoAttack [30], a multi-level white-box attack (7) Adversarial Patch (Adv-P) [29], an attack with unified perturbations across different inputs (8) a (customized) ASK-Attack that is directly applied on the flocking step [32]. We refer readers to Appendix B for details of the threat models. The results of RAILS defending against these attacks can be viewed in Figure 1 and Table 3. On CIFAR-10, RAILS improves the robust accuracy of CNN (DkNN) on ℓ_∞-PGD/ℓ_2-PGD/FGSM/Sq-Attack/Boundary Attack/AutoAttack/Adv-P/ASK-Attack by 19.43%/14.8%/11.18%/11.5%/32.79%/ (22.58%/22.36%−/11.81%10.31%/10.46%/6.24%/3.2%/

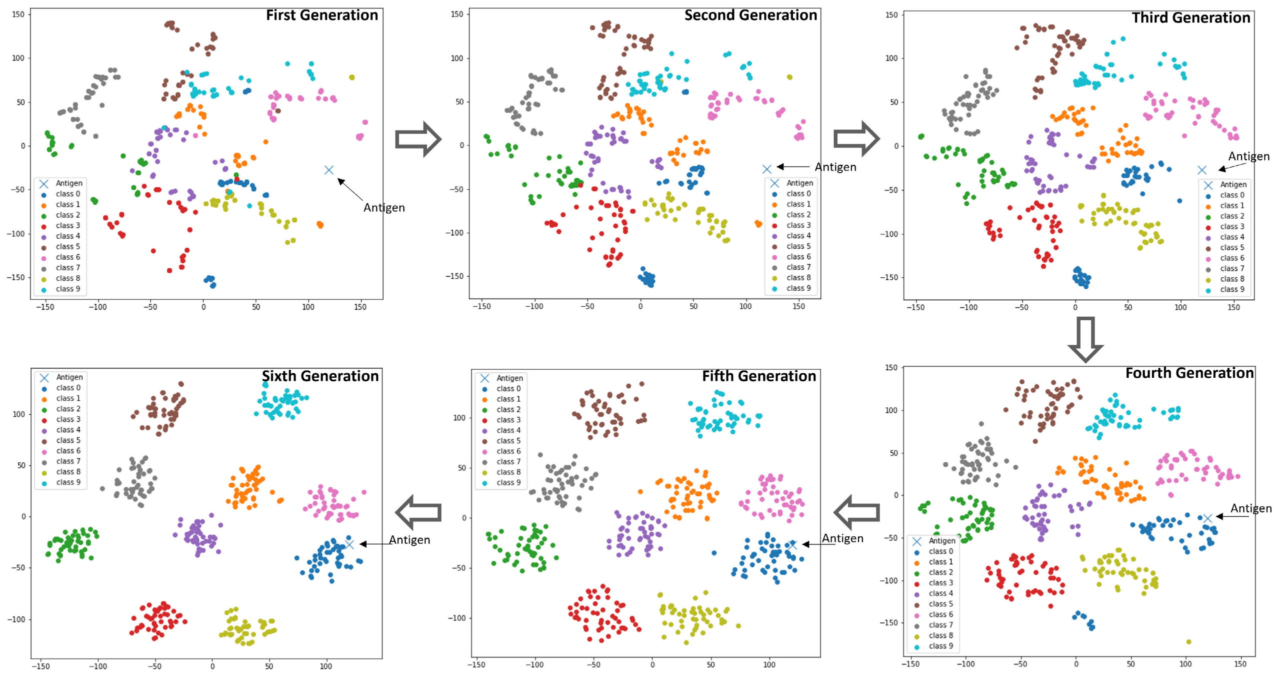


FIGURE 8. Visualization of the evolution of B-cell feature samples over several generations of the the RAILS affinity maturation process. Feature representations of an adversarial CIFAR-10 input (antigen) and the RAILS selected feature populations of B-cells are mapped to a two-dimensional space (t-SNE). The random initialization of the population displays the B-cells as uniformly distributed over feature space. After six generations the affinity maturation process produces B-cells that cluster around the antigen and correctly identify its true class.

6.4%/11.07% − /10.8%/7.7%). More experimental results can be found in Appendix F.

TABLE 3. RAILS achieves higher robust accuracy (RA) under eight types of attacks as compared to CNN and DkNN.

	RAILS	DkNN	CNN
ℓ_∞-PGD ($\epsilon = 8$)	**52.01%**	41.69%	32.57%
ℓ_2-PGD ($\epsilon = 127.5$)	**35.1%**	24.64%	20.3%
FGSM ($\epsilon = 8$)	**59.7%**	53.46%	48.52%
Sq-Attack ($\epsilon = 20$)	**74.5%**	71.3%	53.7%
Boundary Attack (ℓ_2)	**70.6%**	64.2%	37.81%
AutoAttack ($\epsilon = 8$)	**52.84%**	41.77%	30.26%
Adv-P (ratio= 0.1)	**53.5%**	42.7%	31.14%
ASK-Attack ($\epsilon = 8$)	**45.5%**	37.8%	34.21%

We further evaluate RAILS on CIFAR-100 under the three most powerful attacks taken from Table 3: ℓ_∞-PGD; AutoAttack; and Boundary Attack. The standard accuracy of RAILS, DkNN, and CNN are 61.03%, 62.92%, and 65.57%. The results of defending against the three types of attacks on CIFAR-100 are shown in Table 4. One can see that RAILS outperforms adversarially trained CNN and DkNN.

3) DEFENSE AGAINST HUMAN-PERCEPTIBLE DISTURBANCES

We test RAILS against disturbances visible to the naked eye using CIFAR-10 data. We consider the ℓ_∞-PGD attack with $\epsilon = 28$. Figure 9 shows the benign examples and their adversarial counterparts with large disturbances. The differences can be clearly observed. Under the human perceptible attack, the accuracy for RAILS, DkNN, and the adversarially trained CNN are 33.26%, 19.53%, and 0%. The results demonstrate that RAILS can effectively defend against human perceptible perturbations as compared with DkNN and CNN.

TABLE 4. RAILS achieves higher robust accuracy (RA) than DkNN and CNN on CIFAR-100 under the three attacks.

	RAILS	DkNN	CNN
ℓ_∞-PGD ($\epsilon = 8$)	**41.35%**	32.96%	23.7%
AutoAttack ($\epsilon = 8$)	**42.84%**	32.86%	25.63%
Boundary Attack (ℓ_2)	**53.6%**	49.51%	29.1%

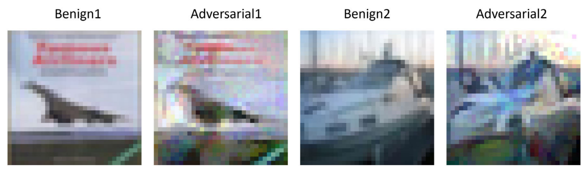


FIGURE 9. Two CIFAR-10 examples of human perceptible perturbations. The adversarial examples are generated by ℓ_∞-PGD attack with $\epsilon = 28$.

4) DIVERSITY VERSUS SPECIFICITY

The DkNN finds a group of feature space k-nearest neighbors that at each layer classify an input sample in a single shot. In contrast, starting from a initial uniform label distribution at each layer, RAILS constructs a classifier after maturation of several generations of feature representing B-cells using an evolutionary optimization process. Results in Table 2 show that evolving a population of features from highly diverse to highly specific provides additional robustness with little sacrifice on benign accuracy.

5) ABLATION STUDY

Using the CIFAR-10 dataset and the third convolutional layer of a VGG16 model, we perform an ablation study to clarify the relative influence of different RAILS components on performance. Our findings are summarized as follows: **(i)** increasing the number of nearest neighbors in a certain range improves performance; **(ii)**; higher mutation probability increases robust accuracy **(iii)**; the magnitude of mutation is sensitive to the input data, but may be optimized to increase robust accuracy. We refer readers to Appendix F for more details on the ablation study. We also show that when we turn off the affinity maturation stage, the robust accuracy drops from 59.2% to 55.65% (on 2000 test examples), indicating the importance of including the affinity maturation step in RAILS.

6) SINGLE-STAGE ADAPTIVE LEARNING

In the previous sections we demonstrated that static learning is effective in predicting the class of current adversarial inputs. Here we show that RAILS can be implemented with single-stage adaptive learning (SSAL) to further improve accuracy and robustness. While the idea is not pursued in this paper, our SSAL results suggest that RAILS may be gainfully extended to the on-line learning setting. SSAL is implemented as follows. We first train a RAILS classifier on the training data as described in previous sections. Then we used RAILS to generate 3000 memory data (B-cells) when a subset of test data taken from MNIST was used as input to the initially trained RAILS. We then merged this new data with the population of training data, creating an augmented training set. Finally, we randomly selected and adversarially modified another 1000 test samples of MNIST, and, using RAILS with its expanded training data, evaluated its adversarial classification accuracy. Table 5 shows that the SSAL improves RA of DkNN by 2.3% with no SA loss using by augmenting the training data with only 3000 memory data samples (a total of 5% increase of the training data).

TABLE 5. **When implemented with memory data and single-stage adaptive learning (SSAL), RAILS hardens DkNN against future attacks.**

	Standard Accuracy (SA)	Robust Accuracy (RA) ($\epsilon = 60$)
DkNN	98.5%	68.3%
DkNN-SSAL	98.5%	**70.6%**

V. CONCLUSION

Inspired by the immune system, we proposed a new defense framework for deep learning models. The proposed Robust Adversarial Immune-inspired Learning System (RAILS) has a one-to-one mapping to a simplified architecture immune system and its learning behavior aligns with *in vitro* biological experiments. RAILS incorporates static learning and adaptive learning, contributing to a robustification of predictions and dynamic model hardening, respectively. The experimental results demonstrate the effectiveness of RAILS. We believe this work is fundamental and delivers valuable principles for designing robust deep models. In future work, we will dig deeper into the mechanisms of the immune system's adaptive learning (life-long learning) and covariate shift adjustment, which will be consolidated into our computational framework.

APPENDIX A
A ONE-TO-ONE MAPPING FROM THE IMMUNE SYSTEM TO RAILS

Table 6 provides a detailed comparison between the Immune System and RAILS. The top part shows the detailed explanations of some technical terms. The bottom part shows the four-step process of the two systems.

APPENDIX B
EXPERIMENTAL PARAMETER SETTINGS

A. DATASETS AND MODELS

We test RAILS on three public datasets: MNIST [46], SVHN [47], CIFAR-10 and CIFAR-100 [48]. The MNIST dataset is a 10-class handwritten digit database consisting of 60000 training examples and 10000 test examples. The SVHN dataset is another benchmark that is obtained from house numbers in Google Street View images. It contains 10 classes of digits with 73257 digits for training and 26032 digits for testing. CIFAR-10 (CIFAR-100) is a more complicated dataset that consists of 60000 (60000) colour images in 10 (100) classes. There are 50000 training images and 10000 test images. We use a four-convolutional-layer neural network for MNIST, and VGG16 [49] for SVHN, CIFAR-10, and CIFAR-100. For MNIST and SVHN, we conduct the affinity maturation in the inputs. Compared with MNIST and SVHN, features in the input images of CIFAR-10 and CIFAR-100 are more mixed and disordered. To reach a better performance using RAILS, we use adversarially trained models on $\epsilon = 4$ and conduct the affinity maturation in layer one instead of the input layer. We find that both ways can provide better feature representations for CIFAR-10 and CIFAR-100, and thus improve RAILS performance.

B. THREAT MODELS

Though out this paper, we consider eight different types of attacks: (1) ℓ_∞-Projected Gradient Descent (PGD) attack [11] - We implement 20-step PGD attack for MNIST, and 10-step PGD attack for SVHN and CIFAR-10. The attack strength is $\epsilon = 40/60/76.5$ for MNIST, $\epsilon = 8$ for SVHN, and $\epsilon = 8/16$ for CIFAR-10. (2) ℓ_2-PGD attack - The attack strength is $\epsilon = 127.5$ for CIFAR-10 (3) Fast Gradient Sign Method [5] - The attack strength is $\epsilon = 76.5$ for MNIST, and $\epsilon = 4/8$ for SVHN and CIFAR-10. (4) Square

TABLE 6. A one-to-one mapping from the immune system to RAILS.

	Immune System	RAILS
Antigen	A molecule or molecular structure (self/non-self)	Test example (benign/adversarial)
Affinity	The strength of a single bond or interaction between antigen and B-Cell	The negative Euclidean distance between feature maps of input and another data point
Naive B-cells	The B-cells that have been recruited to generate new B-cells	The k-nearest neighbors from each class with highest affinity to the antigen
Plasma B-cells	Newly generated B-cells with top affinity to the antigen	Newly generated examples with top affinity to the input
Memory B-cells	Generated B-cells with moderate-affinity to the antigen	Generated examples with moderate-affinity to the input
	Immune System	RAILS
Sensing	Classify between **self** and **non-self** antigens	Classify between **non-adversarial** and **adversarial** inputs using confidence scores
Flocking	Non-self antigens are presented to T cells, recruit **highest affinity naive B-cells**	Find the **nearest neighbors from each class** that have the highest initial affinity score to the input data
Affinity maturation	**Naive B-cells** *divide and mutate* to generate initial diversity. **Affinity is maximized** through selection by T cells for affinity.	Generate new examples from the **nearest neighbors** through *mutation and crossover* and calculate each example's affinity score to the input **Affinity is maximized** through selection.
Consensus	*Memory B-cells* are saved and *Plasma B-cells* are created. Antigen is recognized by **majority voting**, producing *high affinity B-cells*	Select generated examples with **high-affinity scores** to be *Plasma data*, and examples with moderate-affinity scores saved as *Memory data*. *Plasma data* use **majority voting for prediction**

Attack [28] - We implement 50-step attack for MNIST, and 30-step attack for CIFAR-10. The attack strength is $\epsilon = 76.5$ for MNIST, and $\epsilon = 20/24$ for CIFAR-10. (5) Boundary Attack [31] - We set the maximum iteration to be 500 for CIFAR-10 experiments. (6) AutoAttack [30] - The attack strength is $\epsilon = 8$ and the maximum iteration is set to be 200 for CIFAR-10 experiments. (7) Adversarial Patch [29] - We use patch ratio= 0.1. (8) ASK-Attack [32] - We implement 20-step attack for CIFAR-10.

C. PARAMETER SELECTION

By default, we set the size of the population $T = 100$ and the mutation probability $\rho = 0.15$. In Figure 6, we set $T = 100$ to obtain a better visualization. The maximum number of generations is set to $G = 50$ for MNIST, and $G = 10$ for CIFAR-10 and SVHN. When the model is large, selecting all the layers would slow down the algorithm. We use all four layers for MNIST. For CIFAR-10 and SVHN, we test on a few (20) validation examples and evaluate the kNN standard accuracy (SA) and robust accuracy (RA) on each layer. We then select layer three and layer four with SA and RA over 40%.

1) MUTATION RANGE

The mutation range selection is related to the dataset. For MINST whose features are well separated in the input, the upper bound of the mutation range could be set to a relatively large value. For the datasets with low-resolution and sensitive to small perturbations, we should set a small upper bound of the mutation range. We also expect that the mutation could bring enough diversity in the process. Therefore, we will pick a lower bound of the mutation range. We set the mutation range parameters to $\delta_{\min} = 0.05(12.75)$, $\delta_{\max} = 0.15(38.25)$ for MNIST. Considering CIFAR-10 and SVHN are more sensitive to small perturbations, we set the mutation range parameters to $\delta_{\min} = 0.005(1.275)$, $\delta_{\max} = 0.015(3.825)$.

2) SAMPLING TEMPERATURE

Note that the initial condition found for adversarial examples could be worse than benign examples. It is still possible for examples (initial B-cells found in the flocking step) of wrong classes dominating the population affinity at the beginning. To reduce the gaps between the high-affinity examples and low-affinity examples, we use the sampling temperature τ to control the sharpness of the softmax operation. The principle of selecting τ is to make sure that the high-affinity examples in one class do not dominate the affinity of the whole population at the beginning. We thus select τ to make sure that the top 5% of examples are not from the same class. We find that our method works well in a wide range of τ once the principle is reached. For MNIST, the sampling temperature τ in each layer is set to 3, 18, 18, and 72. Similarly, we set $\tau = 1/10$ and $\tau = 300$ for the selected layers for CIFAR-10 and SVHN, respectively.

3) THE HARDWARE AND OUR CODE

We apply RAILS on one Tesla V100 with 64GB memory and 2 cores. The code is written in PyTorch.

APPENDIX C
RAILS EMULATION OF THE NATURAL IMMUNE SYSTEM

A. RAILS MIMICS THE BIOLOGICAL LEARNING CURVE

To demonstrate that the proposed RAILS computational system captures important properties of the immune system, we compare the learning curves of the two systems in Figure 7. In RAILS, we treat test data (potentially the adversarial example) as an antigen. Affinity in both systems measures the similarity between the antigen sequence and a potential matching sequence. The green and red curves depict the evolution of the mean affinity between the B-cell population and the antigen. The candidates selected in the flocking step (kNN) are close to a particular antigen. Two tests are performed to illustrate the learning curves when the same antigen (antigen 1) or a very different antigen (antigen 2) is presented during the affinity maturation step. When antigen 1 is presented (green curves), Figure 7 shows that both the immune system (left panel) and RAILS have learning curves that initially increase, then decrease, and then increase again. This phenomenon indicates a two-phase learning process, and both systems can escape from local optimal points. On the other hand, when the different antigen 2 is presented, the flocking candidates converge more slowly to a high affinity population during the affinity maturation process (dashed curves).

B. IN-VITRO IMMUNE RESPONSE EXPERIMENT

1) DETAILS ON IN-VITRO EXPERIMENTS IN FIGURE 7

We performed in-vitro experiments to evaluate the adaptive immune responses of mice to foreign antigens. These mice are engineered in a way which allows us to image their B cells during affinity maturation in an in-vitro culture. Using fluorescence of B cells, we can determine whether the adaptive immune system is effectively responding to an antigen, and infer the affinity of B cells to the antigen. In this experiment, we first immunized a mouse using lysozyme (Antigen 1). We then challenged the immune system in two ways: (1) reintroducing lysozyme and (2) introducing another very different antigen, ovalbumin (Antigen 2). We then measured the fluorescence of B cells in an in-vitro culture for each of these antigens, which we use as a proxy to estimate affinity. We use five fluorescence measurements over ten days to generate the affinity curves in Figure 7 (left). When plotting, we use a spline interpolation in MATLAB to smooth the affinity curves. For full experimental details, please refer to Figures 10, 11, and the following three sections.

In-vitro culture of Brainbow B cells. For the in vitro culture of B cells, splenic lymphocytes from Rosa26Confetti+/+; AicdaCreERT2+/- mice were harvested and cultured following protocol from [50]. Mice were individually immunized with lysozyme and ovalbumin. Three days post-immunization, the mice were orally administered with Tamoxifen (50 μl of 20mg/ml in corn oil) and left for three days to activate the Cre-induced expression of confetti colors in germinal center B cells. Six days post-immunization, whole lymphocytes from spleen were isolated. 3 × 105 whole lymphocytes from spleen were seeded to a single well in a 96 well dish along with 3 × 104 dendritic cells derived from bone marrow hematopoietic stem cells for in vitro culture. The co-culture was grown in RPMI medium containing methyl cellulose (R&D systems, MN) supplemented with recombinant IL-4 (10 ng/ml) from, LPS (1 μg/ml), 50 μM 2-mercaptoethanol, 15% heat inactivated fetal calf serum, ovalbumin (10 μg/ml) for ovalbumin specific B cells and hen egg white lysozyme (10 μg/ml) for lysozyme specific B cells. Antigens were also added vice-versa for non-specific antigen control. The media was changed every two days. The cultures were imaged every day for 14 days.

Preparation of differentiated dendritic cells from bone marrow hematopoietic stem cells. Bone marrow cells from femurs and tibiae of C57BL/6 mice was harvested, washed and suspended in RPMI media containing GM-CSF (20ng/ml), (R&D Systems, MN), 2mM L-glutamine, 50 μM 2-mercaptoethanol and 10% heat inactivated fetal calf serum. On day two and four after preparation, 2 mL fresh complete medium with (20ng/ml) GM-CSF were added to the cells. The differentiation of hematopoietic stem cells into immature dendritic cells was completed at day seven.

Cell imaging. Confocal images shown in Figure 11 were acquired using a Zeiss LSM 710. The Brainbow 3.1 fluorescence was collected at 463-500 nm in Channel 1 for ECFP (excited by 458 laser), 416-727 nm in Channel 2 for EGFP and EYFP (excited by 488 and 514 lasers, respectively), and 599-753 nm in Channel 3 for mRFP (excited by 594 laser). Images were obtained with 20× magnification.

C. IN-SILICO RAILS EXPERIMENT

1) DETAILS ON RAILS EXPERIMENTS IN FIGURE 7

Similar to the in-vitro experiments, we test RAILS on two different inputs from CIFAR-10, as shown in the right panel of the conceptual diagram Figure 10 (cat as Antigen 1 and deer as Antigen 2). The candidates are all selected in the flocking step of antigen 1 (A_1). Two tests are performed to illustrate the different responses when the same antigen (A_1) or a very different antigen (A_2) is presented during the affinity maturation step.

We first apply RAILS on Antigen 1 (A_1) and obtain the average affinity of the true class as well as the initial B-cells, i.e. the nearest neighbors from all classes. The affinity vs generation curve is shown in the green line in the right panel of Figure 7. One can clearly see the learning pattern. And finally, the solution is reached with a high affinity. Then we apply the initial B-cells obtained from A_1 to Antigen 2 (A_2). The results show that A_2 cannot reach the solution by using the given initial B-cells, as shown by the red line in the right panel of Figure 7.

APPENDIX D
ADDITIONAL DETAILS ON RAILS

A. COMPUTATIONAL COST

The RAILS prototype implemented in this paper has a relatively high computational cost, primarily due to the need to generate and select generations of in-silico B-cells using the genetic algorithm. Specifically, the average prediction time per sample is less than 0.1sec on CIFAR-10 with population size 100 and 20 generations. Note that all of our reported RAILS experiments were performed on a single GPU. RAILS speed can be dramatically accelerated by using multiple GPUs. We are currently investigating fast approximations to

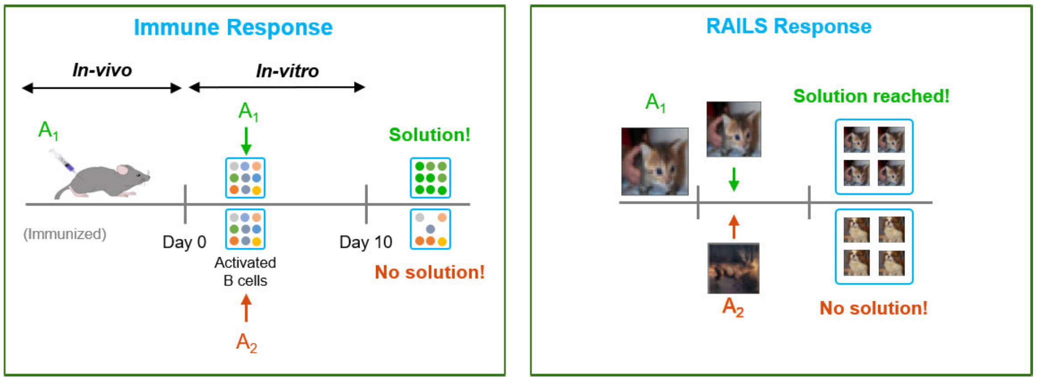


FIGURE 10. The conceptual diagram of in-vitro immune response (Left) and RAILS response (Right): The candidates are selected in the flocking step of antigen 1 (A_1). Two tests are performed to illustrate the different responses when the same antigen (A_1) or a very different antigen (A_2) is presented during the affinity maturation step. In RAILS, we treat test data (cat and deer) as antigens. The solution can be reached when A_1 is presented. There is no solution when A_2 is presented.

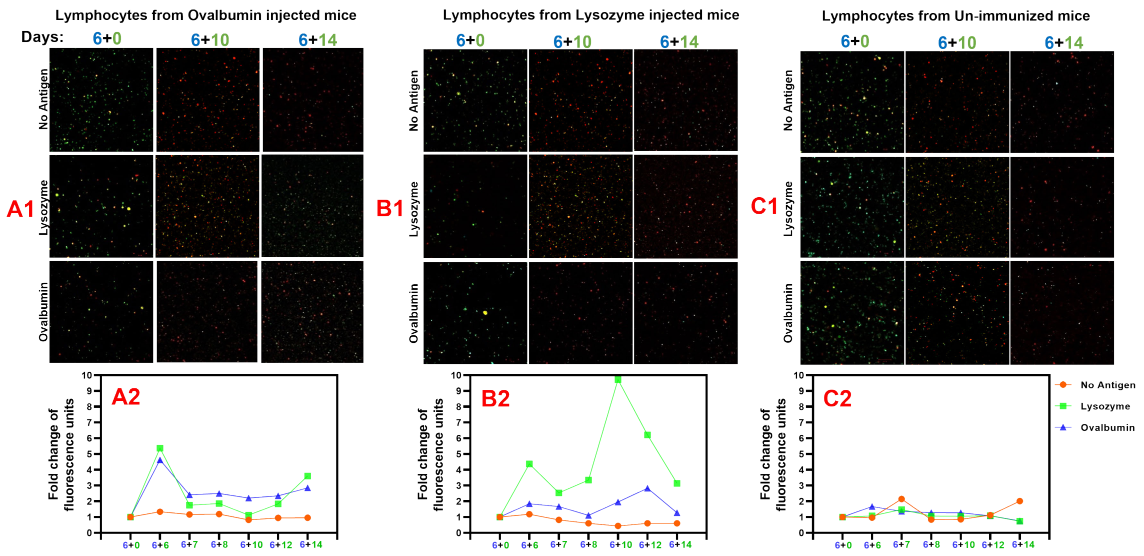


FIGURE 11. In-vitro experimental results showing importance of antigen affinity in robustness of adaptive immune response. The figure shows images of the B-cell proliferation (A1, B1, C1) and B-cell fold-change curves (A2, B2, C2) over time for three initial antigen imunization types (Ovalbumin-injected mice, Lysozyme injected mice, and un-immunized) and three antigen post-imunization treatments (no-antigen, lysozyme antigen, Ovalbumin antigen) applied 6 days later. (A2 and B2) Ovalbumin and Lysozyme antigens produce similar adaptive immune responses for those mice previously immunized with either antigen while they produce no adaptive immune response for mice that have not been immunized. (C2) The graphs show clearly that proliferation of B-cells in the adaptive immune response is strongest when the lymphocytes are re-exposed to the same antigen as in the immunization but still elicits an adaptive response when exposed to a similar but non-identical antigen. The decrease in adaptive response is inversely proportional to the similarity (affinity) between the antigens.

the genetic algorithm solution used by our prototype RAILS implementation.

B. RELATIONS TO PREFERENTIAL ATTACHMENT

The process of selection can also be viewed as creating new nodes from existing nodes in a Preferential Attachment (PA) evolutionary graph generation process [43], where the probability of a new node linking to node i is

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}, \tag{7}$$

and k_i is the degree of node i. In PA models new nodes prefer to attach to existing nodes having high vertex degree. In RAILS, we use a surrogate for the degree, which is the exponentiated affinity measure, and the offspring are generated by parents having high degree.

C. EARLY STOPPING CRITERION

Considering the fast convergence of RAILS, one practical early stopping criterion is to check if a single class occupies most of the high-affinity population for multi-generation, e.g., checking the top 5% of the high-affinity population. We empirically find that it takes less than 5 iterations to convergence for most of the inputs from MNIST (CIFAR-10).

D. RAILS IMPROVES ROBUSTNESS OF ALL MODELS

RAILS is a general framework that can be applied to any model. Specifically, we remark that there is no competitive relationship between RAILS and robust training since RAILS can improve all models' robustness, even for the robust trained model. Moreover, RAILS can reach higher robustness based on a robust trained model.

APPENDIX E
A SIMPLE SENSING STRATEGY

Sensing in the immune system aims to detect the self and non-self pieces, while RAILS leverages sensing to provide initial detection of adversarial examples. Sensing can also prevent the RAILS computation from becoming overwhelmed by false positives, i.e., recognizing benign examples to adversarial examples. Once the input is detected as benign, there is no need to go through the following process, and the neural network can directly obtain the predictions. The sensing step provides the initial discrimination between adversarial and benign inputs, and we develop a simple strategy here.

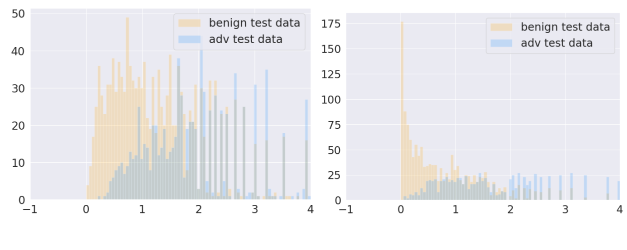
The assumption we make here is that benign examples have more consistency between the features learned from a shallow layer and the DNN prediction compared with adversarial examples. This consistency can be measured by the cross-entropy between the DNN and layer-l kNN predicted class probability score. Specifically, we have the cross-entropy (adversarial threat score) for each input $\mathbf{x}$ in the following form

$$ce(\mathbf{x})^l = -\sum_{c=1}^{C} F_c(x) \log \mathbf{v}_c^l \tag{8}$$

where F_c denotes the neural network prediction score of the c-th class. The prediction score is obtained by feeding the output of the neural network to a softmax operation. $\mathbf{v}_c^l$ is the c-th entry of the normalized kNN vector in layer-l, which is defined as follows

$$\mathbf{v}_c^l = \frac{r_c}{k} = \frac{|\{\hat{x}|\hat{x} \in \mathcal{Q}_l \cap \mathcal{D}_c\}|}{k} \tag{9}$$

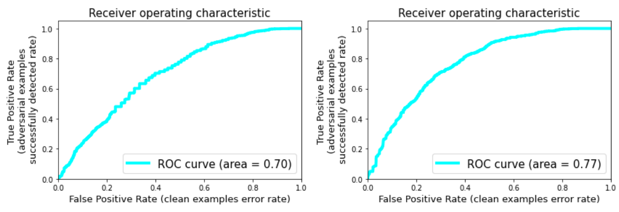
where $\mathcal{D}_c$ represents the training data belonging to class c. $\mathcal{Q}_l$ denotes the k-nearest neighbors of $\mathbf{x}$ in all classes by ranking the affinity score $A(f_l; \mathbf{x}_j, \mathbf{x})$.



(a) Distributions in layer two (b) Distributions in layer three

FIGURE 12. Adversarial threat score distributions of benign examples and adversarial examples: There are more adversarial examples with larger ce^2 (ce^3) compared to benign examples. The results suggest that a large number of benign examples could be separated from adversarial examples, and thus can prevent the RAILS computation from becoming overwhelmed by false positives.

All the sensing experimental results shown below are obtained on CIFAR-10. We first compare the adversarial threat score distributions of benign examples and adversarial examples on layer two and layer three in Figure 12. We use 1400 benign examples for layer two and layer three. The adversarial examples are all successful attacks. One can see that there are more adversarial examples with larger ce^2 (ce^3) compared to benign examples. The results suggest that a large number of benign examples could be separated from adversarial examples, and thus can prevent the RAILS computation from becoming overwhelmed by false positives.



(a) ROC curve of the adversarial threat scores in layer two (b) ROC curve of the adversarial threat scores in layer three

FIGURE 13. Receiver Operating Characteristic (ROC) curves of the adversarial threat scores for benign examples and adversarial examples: The True Positive Rate (TPR) represents the adversarial examples successfully detected rate. The False Positive Rate (FPR) represents the rate of benign examples accidentally been detected as adversarial examples. The Area Under the Curve (AUC) is 0.70 and 0.77 for layer two and layer three.

Figure 13 shows the Receiver Operating Characteristic (ROC) curves of the adversarial threat scores for benign examples and adversarial examples. Figure (a) and (b) depict the ROC curves in layer two and layer three, respectively. The True Positive Rate (TPR) represents the adversarial examples successfully detected rate. The False Positive Rate (FPR) represents the rate of benign examples accidentally been detected as adversarial examples. The Area Under the Curve (AUC) is 0.70 and 0.77 for layer two and layer three. Note that we care more about TPR than FPR since it has no side effect on RAILS accuracy if we detect a benign example to an adversarial example. Our goal is to select a relatively low FPR while still maintaining a high TPR. For example, we could keep a 95% TPR with 56% FPR using a threshold 0.4 in layer three.

The details about the sensing algorithm are shown in Algorithm 2. We will select a threshold κ such that x is treated as benign example (adversarial example) if $ce(x) \leq \kappa$ ($ce(x) > \kappa$).

We then select $\mathcal{L}_s$ to only include layer three, and apply the threshold of 0.4 in the sensing step on CIFAR-10. The results show that the false positive rate can be reduced by 40%, while the SA remains the same and the RA only decreases 0.2%.

APPENDIX F
ADDITIONAL EXPERIMENTS

A. ADDITIONAL COMPARISONS ON MNIST

Figure 14 provides the confusion matrices for benign examples classifications and adversarial examples classifications in Conv1 and Conv2 when $\epsilon = 60$. The confusion matrices

Algorithm 2 Sensing: Adversarial Example Detection

```
Input: Test data point $\mathbf{x}$; Training dataset $\mathcal{D}_{tr} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_C\}$; Number of Classes $C$; Model $F$ with feature mapping $F_l(\cdot)$ in layer $l$, $l \in \mathcal{L}_s$; Affinity function $A$; A preset threshold $\kappa$
for all layer $l \in \mathcal{L}_s$ in parallel do
    Find the $k$-nearest neighbors $\mathcal{Q}_l$ of $\mathbf{x}$ in all classes by ranking the affinity score $A(f_l; \mathbf{x}_j, \mathbf{x})$.
    Obtain the normalized vector $\mathbf{v}^l = (r_1, r_2, \cdots, r_C)/k$, $r_c = |\{\hat{x} | \hat{x} \in \mathcal{Q}_l \cap \mathcal{D}_c\}|$.
    Obtain the softmax prediction vector $(F(x))$.
    Calculate the cross entropy $ce(\mathbf{x})^l = -\sum_{c=1}^{C} F_c(x) \log \mathbf{v}_c^l$.
end for
if $\frac{1}{|\mathcal{L}_s|} \sum_{l \in \mathcal{L}_s} ce(\mathbf{x}) > \kappa$ then
    $\mathbf{x}$ is a potential adversarial example and return $IsAdv = 1$.
else
    $\mathbf{x}$ is benign and return the prediction $\arg\max_c F_c(x)$
end if
```

in Figure 14 show that RAILS has fewer incorrect predictions for those data that DkNN gets wrong. Each value in Figure 14 represents the percentage of intersections of RAILS (correct or wrong) and DkNN (correct or wrong).

In Table 7, we provide the experimental results with Square Attack [28] (a black-box attack) showing that RAILS improves the robust accuracy of DkNN by 1.35% (11% attack success rate) on MNIST with $\epsilon = 76.5$.

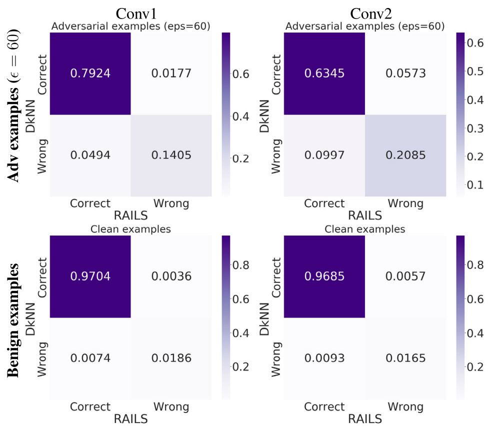


FIGURE 14. RAILS has fewer incorrect predictions for those data that DkNN gets wrong. Confusion Matrices of adversarial examples and benign examples classification in Conv1 and Conv2 (RAILS vs. DkNN $\epsilon = 60$).

Figure 15 shows the confusion matrices of the overall performance when $\epsilon = 60$. The confusion matrices indicate that RAILS' correct predictions agree with a majority of DkNN's correct predictions and disagree with DkNN's wrong predictions.

We also show the SA/RA performance of RAILS under PGD attack and Fast Gradient Sign Method (FGSM) when $\epsilon = 76.5$. The results in Table 12 indicate that RAILS can reach higher RA than DkNN with close SA.

TABLE 7. RAILS improves the robust accuracy of DkNN by 1.35% (11% attack success rate) on MNIST with $\epsilon = 76.5$.

	SA	RA
RAILS (ours)	97.95%	**89.35%**
DkNN	97.99%	88%

TABLE 8. RAILS outperforms DkNN and CNN on defending PGD attack with strength $\epsilon = 8$ and $\epsilon = 16$ on CIFAR-10. The difference of robust accuracy (RA) between RAILS and DkNN increases when ϵ increases, indicating that RAILS can defend stronger attacks.

	SA	RA ($\epsilon = 8$)	RA ($\epsilon = 16$)
RAILS (ours)	82%	**52.01%**	**39.04%**
CNN	**87.26%**	32.57%	18.25%
DkNN	86.63%	41.69%	26.21%

TABLE 9. RAILS outperforms DkNN and CNN on defending FGSM attacks with strength $\epsilon = 8$ and $\epsilon = 16$ on CIFAR-10.

	SA	RA ($\epsilon = 8$)	RA ($\epsilon = 16$)
RAILS (ours)	82%	**59.7%**	**45.26%**
CNN	**87.26%**	48.52%	30.55%
DkNN	86.63%	53.46%	37.19%

TABLE 10. RAILS achieves higher robust accuracy (RA) than DkNN on CIFAR-10 under Square Attack [28] with $\epsilon = 20$ and $\epsilon = 24$.

	SA	RA ($\epsilon = 20$)	RA ($\epsilon = 24$)
RAILS (ours)	82%	**74.5%**	**72.8%**
DkNN	**86.63%**	71.3%	69.5%

TABLE 11. RAILS achieves higher robust accuracy (RA) than DkNN on CIFAR-10 under a (customized) ASK-Attack with $\epsilon = 8$. ASK-Attack is applied on different layers.

	RA (Conv 3)	RA (Conv 4)	RA (Conv 3, 4)
RAILS (ours)	**41.2%**	**36.6%**	**45.5%**
DkNN	34%	34.4%	37.8%

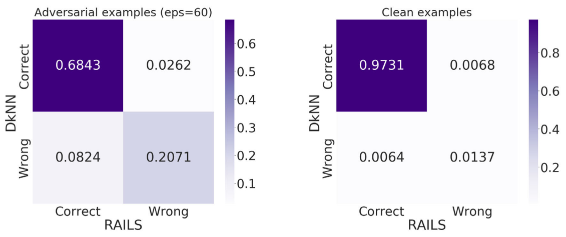


FIGURE 15. RAILS has fewer incorrect predictions for those data that DkNN gets wrong. Confusion Matrices of adversarial examples and benign examples classification (RAILS vs. DkNN - $\epsilon = 60$).

B. ADDITIONAL COMPARISONS ON CIFAR-10

In this subsection, we test RAILS on CIFAR-10 under PGD attack and FGSM with attack strength $\epsilon = 8/16$. The results

TABLE 12. RAILS can reach higher robust accuracy (RA) than DkNN with similar standard accuracy (SA) on defending PGD attack and FGSM ($\epsilon = 76.5$) on MNIST.

	SA	RA (PGD)	RA (FGSM)
RAILS (ours)	97.95%	**58.62%**	**61.67%**
DkNN	97.99%	47.05%	52.23%

are shown in Figure 8 and Figure 9. RAILS outperforms DkNN and CNN on different attack types and strengths. We also find that the difference of RA between RAILS and DkNN increases when ϵ increases, indicating that RAILS can defend stronger attacks.

We then conduct experiments with Square Attack, which is one of the black-box attacks. The results are provided in Table 7 and show that RAILS improves the robust accuracy of DkNN by 3% on CIFAR-10 with $\epsilon = 20$ and $\epsilon = 24$.

TABLE 13. Default parameter for RAILS. For each parameter of interest all other parameters are set to the default values listed in this table.

Parameter	Default Value	Layer
C Classes	10	3
N Neighbors	10	3
Population Size	N Neighbors	3
Sampling Temperature	1	3
Max generation	10	3
Mutation Range	(0.005 - 0.015)	3

TABLE 14. Increasing the N neighbors within a certain range yields improved standard accuracy (SA) and robust accuracy (RA). We hold the population size as a fixed N neighbor value (it's minimum possible value). Note that both the standard and robust accuracy improve as the number of neighbors is increased. This may suggest that performance is influenced by the 'depth' of the class-conditional selection of benign examples.

SA	RA	Difference	N Neighbors	Population Size
0.702	0.535	0.167	2	2
0.732	0.596	0.136	6	6
0.755	0.588	0.167	10	10
0.751	0.601	0.150	14	14
0.764	0.594	0.170	18	18
0.758	0.586	0.172	22	22
0.752	0.606	0.146	26	26

C. DETAILS ON RAILS ABLATION STUDY

For this subsection, RAILS is trained on CIFAR-10 with VGG16 as the classifier. Results are evaluated using model classification accuracy. Accuracy is compared before and after a projected gradient descent (PGD) attack on the training data with $\epsilon = 8/255$. The baseline model performance for benign data (standard accuracy) was 87.26%. After the training data was adversarially attacked the VGG16 accuracy (robust accuracy) fell to 32.57%. By implementing RAILS, we are able to achieve an robust accuracy of 54.3% using the parameterization described in Table 13. All the experiments are conducted on convolutional layer 3. Each experiment holds these parameters fixed while exploring a range of values over independent training regimes. Both standard and robust accuracy are compared for each parameter choice. The purpose of this section is to investigate RAILS' sensitivity towards parameter choices. Details for each experiment are listed in the table captions.

TABLE 15. Increasing population size improves robustness when N is small, and does not yield significant improvement when N is large with low mutation magnitude and low mutation probability. We present an increased population coefficient κ, where the population T equals $\kappa \times (N$ neighbors). We present two cases: where N neighbors is small and where N neighbors is large. **Small N:** Increasing κ from one to two improves the robustness. However, further increasing κ does not bring significant improvement. **Large N:** Note that population size does not have an apparent impact on either standard or robust accuracy when N neighbors is large. This may suggest that the number of perturbed input 'exemplars' does not lead to more robust accuracy on adversarial inputs without sufficient mutation. This is consistent with the core principal in the adaptive immune system that mutation is necessary to converge on optimal solutions.

SA	RA	Difference	N Neighbors	Population Size
0.696	0.532	0.164	2	2
0.720	0.558	0.162	2	4
0.727	0.562	0.165	2	6
0.720	0.558	0.162	2	8
0.735	0.562	0.173	2	10
0.740	0.604	0.136	10	10
0.753	0.595	0.158	10	20
0.754	0.597	0.157	10	30
0.755	0.598	0.157	10	40
0.767	0.595	0.172	10	50

TABLE 16. Increasing mutation upper bound within a small value window increases robustness. We observe that over several experiments mutation range has a narrow window where it positively impacts both standard accuracy (SA) and robust accuracy (RA). This may suggest that this parameter is an important hyper-parameter to tune during training.

SA	RA	Difference	Mutation Range LB	Mutation Range UB
0.764	0.569	0.195	0.002	0.005
0.740	0.586	0.154	0.002	0.010
0.746	0.581	0.165	0.002	0.015
0.743	0.577	0.166	0.002	0.020
0.732	0.587	0.145	0.002	0.025
0.737	0.583	0.154	0.002	0.030

TABLE 17. Crossover is an important mechanism for improving performance. We observe better performance when we use cross-over as opposed to mutation alone during training. We also note that population size alone does not necessarily contribute to better performance for either strategy.

Operator	SA	RA	Difference	Population Size
crossover	0.736	0.570	0.166	10
mutation	0.708	0.552	0.156	10
crossover	0.744	0.589	0.155	20
mutation	0.724	0.563	0.161	20
crossover	0.759	0.598	0.161	50
mutation	0.733	0.558	0.175	50

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] M. M. Ghazi and H. K. Ekenel, "A comprehensive analysis of deep learning based representation for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 34–41.

[3] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[4] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[7] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv:1708.06733*.

[8] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[9] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," 2017, *arXiv:1703.00410*.

[10] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," 2017, *arXiv:1702.06280*.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[12] 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 *Proc. Int. Conf. Mach. Learn.*, 2019.

[13] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1310–1320.

[14] X. Zhang, J. Wang, T. Wang, R. Jiang, J. Xu, and L. Zhao, "Robust feature learning for adversarial defense via hierarchical feature alignment," *Inf. Sci.*, vol. 560, pp. 256–270, Jun. 2021.

[15] A. Mustafa, S. H. Khan, M. Hayat, J. Shen, and L. Shao, "Image super-resolution as a defense against adversarial attacks," *IEEE Trans. Image Process.*, vol. 29, pp. 1711–1724, 2020.

[16] B. Sun, N.-H. Tsai, F. Liu, R. Yu, and H. Su, "Adversarial defense by stratified convolutional sparse coding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11447–11456.

[17] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," 2018, *arXiv:1803.04765*.

[18] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[19] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 3–14.

[20] A. Aldahdooh, W. Hamidouche, S. A. Fezza, and O. Deforges, "Adversarial example detection for DNN models: A review and experimental comparison," 2021, *arXiv:2105.00203*.

[21] Z. Niu, Z. Chen, L. Li, Y. Yang, B. Li, and J. Yi, "On the limitations of denoising strategies as adversarial defenses," 2020, *arXiv:2012.09384*.

[22] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.

[23] S. Jamali and R. Fotohi, "DAWA: Defending against wormhole attack in MANETs by using fuzzy logic and artificial immune system," *J. Supercomput.*, vol. 73, no. 12, pp. 5173–5196, 2017.

[24] G. F. Scaranti, L. F. Carvalho, S. Barbon, and M. L. Proença, "Artificial immune systems and fuzzy logic to detect flooding attacks in software-defined networks," *IEEE Access*, vol. 8, pp. 100172–100184, 2020.

[25] M. Tarao and T. Okamoto, "Toward an artificial immune server against cyber attacks: Enhancement of protection against DoS attacks," *Proc. Comput. Sci.*, vol. 96, pp. 1137–1146, Jan. 2016.

[26] K. D. Gupta and D. Dasgupta, "Using negative detectors for identifying adversarial data manipulation in machine learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.

[27] T. J. Rademaker, E. Bengio, and P. François, "Attack and defense in cellular decision-making: Lessons from machine learning," *Phys. Rev. X*, vol. 9, no. 3, Jul. 2019, Art. no. 031012.

[28] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 484–501.

[29] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2017, *arXiv:1712.09665*.

[30] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2206–2216.

[31] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2017, *arXiv:1712.04248*.

[32] R. Wang, T. Chen, P. Yao, S. Liu, I. Rajapakse, and A. Hero, "ASK: Adversarial soft k-nearest neighbor attack and defense," 2021, *arXiv:2106.14300*.

[33] N. S. De Silva and U. Klein, "Dynamics of B cells in germinal centres," *Nature Rev. Immunology*, vol. 15, no. 3, pp. 137–148, Mar. 2015.

[34] J.-H. Choi, H. Zhang, J.-H. Kim, C.-J. Hsieh, and J.-S. Lee, "Evaluating robustness of deep image super-resolution against adversarial attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 303–311.

[35] D. A. B. Fernandes, M. M. Freire, P. A. P. Fazendeiro, and P. R. M. Inácio, "Applications of artificial immune systems to computer security: A survey," *J. Inf. Secur. Appl.*, vol. 35, pp. 138–159, Aug. 2017.

[36] H. Liu, M. Long, J. Wang, and M. Jordan, "Transferable adversarial training: A general approach to adapting deep classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4013–4022.

[37] A. Shafahi, P. Saadatpanah, C. Zhu, A. Ghiasi, C. Studer, D. Jacobs, and T. Goldstein, "Adversarially robust transfer learning," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[38] I. Rajapakse and M. Groudine, "On emerging nuclear order," *J. Cell Biol.*, vol. 192, no. 5, pp. 711–721, Mar. 2011.

[39] F. Cucker and S. Smale, "Emergent behavior in flocks," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 852–862, May 2007.

[40] I. Rajapakse and S. Smale, "Emergence of function from coordinated cells in a tissue," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 7, pp. 1462–1467, Feb. 2017.

[41] L. Mesin, J. Ersching, and G. D. Victora, "Germinal center b cell dynamics," *Immunity*, vol. 45, no. 3, pp. 471–482, Sep. 2016.

[42] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," 2017, *arXiv:1704.01155*.

[43] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

[44] A. Bewley and B. Upcroft, "Advantages of exploiting projection structure for segmenting dense 3D point clouds," in *Proc. Austral. Conf. Robot. Automat.*, vol. 2, 2013, pp. 1–8.

[45] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[47] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," Tech. Rep., 2011.

[48] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[50] I. Wand, P. Holzlöhner, S. Neupert, B. Micheel, and K. Heilmann, "Cooperation of dendritic cells with naïve lymphocyte populations to induce the generation of antigen-specific antibodies *in vitro*," *J. Biotechnol.*, vol. 156, no. 3, pp. 173–181, 2011.

REN WANG received the bachelor's and master's degrees in electrical engineering from Tsinghua University, Beijing, China, in 2013 and 2016, respectively, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2020. He is currently a Postdoctoral Research Fellow with the Department of Electrical Engineering and Computer Science, University of Michigan. His research interests include trustworthy machine learning, high-dimensional data analysis, bio-inspired machine learning, and robustness/optimization on smart grid.

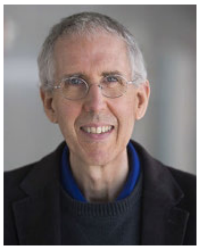
TIANQI CHEN received the B.S. degree in mathematics from Fudan University, Shanghai, China, in 2019, and the M.S. degree in statistics from the University of Michigan, Ann Arbor, MI, USA, in 2021.

STEPHEN M. LINDSLY was born in Midland, MI, USA, in 1995. He received the B.S. degree in computer engineering, in 2017, and the Ph.D. degree in bioinformatics from the University of Michigan, Ann Arbor, MI, USA, in 2021. From 2013 to 2017, he was a Research Assistant with the Rajapakse Laboratory and a Graduate Student Instructor, from 2018 to 2021. His research interests include the development of computational tools for the study of genome cell biology and harnessing biological systems to improve engineering design.

COOPER M. STANSBURY was born in Ann Arbor, Michigan. He received the B.A. degree in philosophy from the Earlham College, in 2012, the M.S. degree in data science from the University of Michigan-Dearborn, in 2019, and the M.S. degree in bioinformatics from the University of Michigan, Ann Arbor, in 2021, where he is currently pursuing the Ph.D. degree in bioinformatics with the Dr. Rajapakse's Laboratory.

ALNAWAZ REHEMTULLA received the M.Sc. and Ph.D. degrees in microbiology and immunology from the University of Calgary. Following a Fellowship in vascular biology with the Scripps Research Institute, he joined the Genetics Institute, Boston, as a Staff Scientist and subsequently as a Faculty Member with the University of Michigan Medical School, in 1999. He is currently the Division Director of Molecular Imaging with the Department of Radiation Oncology, with the University of Michigan. His research interests include developing and leveraging technologies for non-invasive imaging to provide novel insights into biological processes using cultured cells, mouse models, and in clinical studies.

INDIKA RAJAPAKSE is currently an Associate Professor in computational medicine & bioinformatics with the Medical School, and an Associate Professor in mathematics with the University of Michigan, Ann Arbor. He is also a member of the Smale Institute. His research interests include the interface of biology, engineering and mathematics, dynamical systems, networks, mathematics of data, and cellular reprogramming.



ALFRED O. HERO III (Life Fellow, IEEE) received the B.S. degree *(summa cum laude)* in electrical engineering from Boston University, in 1980, and the Ph.D. degree in electrical engineering from Princeton University, in 1984. Since 1984, he has been with the University of Michigan, Ann Arbor, where he is a John H. Holland Distinguished University Professor in electrical engineering and computer science and a R. Jamison and Betty Williams Professor of engineering. His primary appointment is with the Department of Electrical Engineering and Computer Science and he also has appointments, by courtesy, with the Department of Biomedical Engineering and the Department of Statistics. His research interests include high dimensional spatio-temporal data, multi-modal data integration, statistical signal processing, machine learning, applications to social networks, network security and forensics, computer vision, and personalized health. He is a Section Editor of the *SIAM Journal on Mathematics of Data Science* and a Senior Editor of the *IEEE Journal on Selected Topics in Signal Processing*. He is on the Editorial Board of the Harvard Data Science Review (HDSR) and serves as a Moderator for the Electrical Engineering and Systems Science Category of the arXiv. He was the Co-General Chair of the 2019 IEEE International Symposium on Information Theory (ISIT) and the 1995 IEEE International Conference on Acoustics, Speech and Signal Processing. He was the Founding Co-Director of the Michigan Institute for Data Science (MIDAS), from 2015 to 2018. From 2008 to 2013, he held the Digiteo Chaire d'Excellence with the Ecole Superieure d'Electricite, Gif-sur-Yvette, France. He is a fellow of the Society for Industrial and Applied Mathematics (SIAM). Several of his research articles have received best paper awards. He was awarded the University of Michigan Distinguished Faculty Achievement Award, in 2011, the Stephen S. Attwood Excellence in Engineering Award, in 2017, and the H. Scott Fogler Award for Professional Leadership and Service, in 2018. He received the IEEE Signal Processing Society Meritorious Service Award, in 1998, the IEEE Third Millenium Medal, in 2000, the IEEE Signal Processing Society Technical Achievement Award, in 2014, the Society Award from the IEEE Signal Processing Society, in 2015, and the Fourier Award from the IEEE, in 2020. He was the President of the IEEE Signal Processing Society (2006–2008) and was on the IEEE Board of Directors (2009–2011), where he served as the Director of the Division IX (Signals and Applications). From 2011 to 2020, he was a member and the Chair (2017–2020) of the Committee on Applied and Theoretical Statistics (CATS) of the U.S. National Academies of Science.