Boosting Adversarial Training in Safety-Critical Systems Through Boundary Data Selection

Yifan Jia, Christopher M. Poskitt, Peixin Zhang, Jingyi Wang, Jun Sun, and Sudipta Chattopadhyay

***Abstract*—AI-enabled collaborative robots are designed to be used in close collaboration with humans, thus requiring stringent safety standards and quick response times. Adversarial attacks pose a significant threat to the deep learning models of these systems, making it crucial to develop methods to improve the models' robustness against them. Adversarial training is one approach to improve their robustness: it works by augmenting the training data with adversarial examples. This, unfortunately, comes with the cost of increased computational overhead and extended training times. In this work, we balance the need for additional adversarial data with the goal of minimizing the training costs by selecting the most 'valuable' data for adversarial training. In particular, we propose a robustness-oriented boundary data selection method, RAST-AT, which stands for robust and fast adversarial training. RAST-AT selects training data *near to the boundary* by considering adversarial perturbations. Our method improves the speed of model training on CIFAR-10 by 68.67%, and compared to other data selection methods, has 10% higher accuracy with 10% training data selected, and 7% higher robustness with 4% training data selected. Our method also significantly improves efficiency by at least 25% in adversarial training, with the same performance. Finally, we evaluate our method on a cobot system, generating adversarial patches as attacks, and adopting RAST-AT as the defense. We find that RAST-AT can defend against 60% of untargeted attacks and 20% of targeted attacks. Our work highlights the benefits of developing effective defenses against adversarial attacks to ensure the security and reliability of AI-powered safety-critical systems.**

***Index Terms*—Data selection, AI-enabled industrial systems, trustworthy systems, adversarial training.**

I. Introduction

DEEP Neural Network (DNN) models are increasingly used in robotics for automation, such as object detection [1], [2], [3] and intrusion detection [4], [5]. However, the use of DNN models also exposes such systems to adversarial attacks [6], [7], which can have severe consequences. To ensure the safety and security of industrial systems, it is crucial to train AI models to be robust against these attacks. The importance of robustness in AI models for industrial systems has been highlighted in many works [8], [9], [10], [11]. One potential solution to achieve robustness is adversarial training [12], [13], where the model is trained considering adversarial examples in the training data.

However, this process can be costly, especially for large real-world datasets that are commonly used in industrial systems. To illustrate, a substantial amount of data—around 40,000 card images obtained on a daily basis—is required to train the DNN responsible for controlling a robotic arm. Nevertheless, obtaining this sizable dataset entails notable expenses in the form of time, hardware resources (such as GPUs), and human effort for labeling. Additionally, creating adversarial examples, which play a vital role in constructing a sturdy AI model, can be a lengthy undertaking. As a result, it is crucial to establish an efficient training process that yields an accurate and robust AI model while minimizing the resources expended. To address this problem, some works [14], [15] aim to reduce the training time by minimizing the number of iterations. Nevertheless, the enormous training data set still needs huge human labeling efforts and training time.

Active learning, as proposed by Cohn et al. [16], is one approach to reduce the cost of adversarial training by selecting partial training data to label. The basic premise is that, after an initial model is learned, the learner selects new data for labeling that will most effectively improve the model's performance, thus reducing the size of the training data and minimizing the need for human labeling efforts.

Although active learning can be helpful in selecting data, the selection method typically focuses on time and accuracy and ignores robustness [17]. To be able to train on less data while still obtaining an accurate and robust model, the selected data has to be more 'important' and valuable than the rest. Compared with uniformly distributed data, data closer to the boundary provides more information for model training [13].

Moreover, data that is far away from the decision boundary does not need to be labeled as long as the boundary data can provide enough information to calculate the optimal solution [18]. However, measuring how far a data point is from the decision boundary is difficult for deep neural networks because their decision boundaries are of high dimensionality.

Therefore, selecting boundary data for adversarial training is a potential solution, as boundary data is more sensitive to

Manuscript received 16 May 2023; accepted 2 October 2023. Date of publication 27 October 2023; date of current version 7 November 2023. This letter was recommended for publication by Associate Editor D. Brscic and Editor A. Peer upon evaluation of the reviewers' comments. This work was supported in part by the Key R&D Program of Zhejiang under Grant 2022C01018 and in part by the NSFC Program under Grant 62102359. *(Corresponding authors: Peixin Zhang; Jingyi Wang.)*

Yifan Jia and Jingyi Wang are with Zhejiang University, Hangzhou 310007, China (e-mail: jiayifan21@me.com; wangjyee@zju.edu.cn).

Christopher M. Poskitt, Peixin Zhang, and Jun Sun are with the School of Computing and Information Systems, Singapore Management University 188065, Singapore (e-mail: cposkitt@smu.edu.sg; pxzhang94@gmail.com; junsun@smu.edu.sg).

Sudipta Chattopadhyay is with Information System Technology and Design, Singapore University of Technology and Design, 487372, Singapore (e-mail: sudipta_chattopadhyay@sutd.edu.sg).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3327934, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3327934

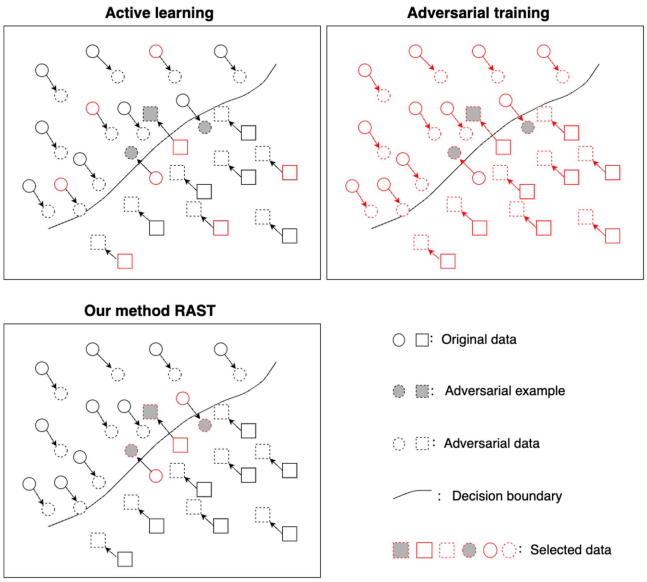

Fig. 1. Illustration of data selection for active learning, adversarial training, and our new method, RAST-AT.

adversarial attacks [13], [19]. Existing boundary data selection methods [19], [20] select data by considering the probability ratio of the predicted classes or adding perturbations randomly to check the predicted class of noisy data. They are designed to maintain accuracy but never consider robustness.

We present RAST-AT, a robust and fast technique for DNN adversarial training. Considering accuracy, efficiency, and robustness, RAST-AT selects data that is able to generate adversarial examples, as shown in Fig. 1. Active learning selects training samples to label by some oracles, reducing data size and label effort. Adversarial training selects all training samples with their adversarial data to increase the robustness of the model against perturbations. Utilizing the advantages of these two methods, we propose RAST-AT, which selects robustness-oriented boundary data with their adversarial examples for adversarial training. RAST-AT greatly reduces the training time while still preserving robustness.

Our method outperforms previous adversarial training methods with higher accuracy, and is more time-efficient, achieving comparable results in 42.55 s vs. 149.00 s for PGD-AT and 86.48 s for FAST-AT. RAST-AT demonstrates its robustness in defending against 60% of untargeted and 20% of targeted adversarial patch attacks in an object detector for a robotic arm.

The contributions of this work include:

- We propose a boundary data selection method (RAST-AT) to improve the model training efficiency while maintaining accuracy and *robustness*.
- We use the selected data to train different models to prove the method can be used to select more 'important' data for model training.
- We compare our method with boundary data selection methods (BSS [19] and SALT [9]) that show that RAST-AT has higher accuracy when the same amount of data is selected.
- We compare our method with other adversarial training methods (PGD-AT and FAST-AT [14]) to prove that RAST-AT matches the robustness but greatly reduces training time.
- We apply our method to an industrial system, a robotic arm with an object detector, to show the capability for object detection and feasibility for a practical application.

II. Background and Related Work

In this section, we discuss the background and some key related work on speeding up model training and data selection for DNN model training. Furthermore, we review key works on boundary data selection methods and adversarial examples.

A. Adversarial Training

Note that to distinct adversarial attack and adversarial training, we add "-AT" for adversarial training. For example, RAST-AT represents RAST adversarial training. PGD represents PGD attack and PGD-AT is for PGD adversarial training. In this work, robustness refers to the accuracy of the model after being attacked by taking adversarial examples as input.

Adversarial examples were discovered by Szegedy et al. [21], and following this, Goodfellow et al. [22] proposed the Fast Gradient Sign Method (FGSM) to generate adversarial examples with a one-step attack. This work was improved by adding more steps as iFGSM [23]. The effectiveness of iterative attacks can be further enhanced by utilizing random initialization, as demonstrated by the Projected Gradient Descent (PGD) [24]. By training with a mixture of adversarial and clear data, adversarial training improves the model's robustness against adversarial attacks.

Due to the need to generate adversarial examples and expand the training data size, adversarial training is costly and time-consuming. The work of Bai et al. [25] summarises a few recent efficient adversarial training works. Shafahi et al. first proposed free adversarial training (Free-AT) [15] by updating both model parameters and image perturbations simultaneously. Expanding on Free-AT, Wong et al. introduced FAST-AT adversarial training [14], which leverages FGSM-AT with random initialization, proving it as effective as PGD-AT but much faster. To compare the efficiency of adversarial training, we select FAST-AT as a benchmark.

B. Boundary Data Selection for DNN

Boundary data selection is mainly for statistical models [26], [27], [28], as interpreting the decision-making process of DNNs is challenging [29], thus determining their boundaries is much harder than it is for statistical models.

Existing works on boundary data selection for DNN focus on either finding the data with maximum entropy or adding perturbations. Shen et al. [19] propose a method of boundary sample selection (BSS) to select boundary data for mutation testing by evaluating the probability ratio of the top two classes. It will select the data point that has a lower ratio, which indicates the model has lower confidence to give the prediction. In contrast,

Miller et al. [20] select data by adding perturbations, which are uniform perturbations to the training data, and select the data that is able to give maximum uncertainty. The work also proposes a software architecture, a Security-oriented Active Learning Testbed (SALT) with the method. Instead of adding uniform perturbation, we add an FGSM attack to generate adversarial perturbations. Moreover, the above methods select data with maximum uncertainty but we select the data that is able to be attacked successfully. These can make sure the perturbations are proficient and therefore the selected data are safety critical.

We compare our approach to the two works to demonstrate that an adversarial noise is more efficient in selecting boundary data compared to considering entropy (BSS) and adding uniform perturbation (SALT).

III. Methodology

We define the problem formally as follows: given a pre-trained simple DNN model f and an unlabeled dataset D for further training, instead of labelling all data points, we want a sensitivity measurement λ to select a subset $D_s \subset D$ to label and train the model f. The well-trained model f_d using dataset D_s should have accuracy within an error rate δ of model f_D which is trained using the full dataset D.

In this work, we define robustness as the accuracy of the model after being attacked. For selecting data points that enhance the model's robustness during training, we focus on adversarial examples, which are used for executing adversarial attacks on AI models. Adversarial examples are input data with a small perturbation to deceive machine learning models into misclassifying them. The study of Wang et al. [13] proposes a method to detect adversarial examples through model mutation and observe that adversarial data is more 'sensitive' than normal data, demonstrating that data points near the decision boundary are more sensitive than other points. Consequently, when subjected to the same noise level, boundary data is more vulnerable to adversarial attacks than normal data. Note that the Fast Gradient Sign Method (FGSM) is an adversarial attack that creates adversarial examples that can cross the boundary [22]. We, therefore, select data that is able to generate adversarial examples, i.e. be attacked successfully by FGSM as boundary data. We then employ the boundary data and their corresponding adversarial examples as the training data for adversarial training.

Adversarial examples can be generated by adding adversarial noise to the original data. Adversarial noise is calculated with the loss of predicted output and targeted/actual output. For targeted attacks, $x_{adv} = x - x_{noise}$. For untargeted attacks, $x_{adv} = x + x_{noise}$. The formula can be written as:

$$x_{adv} = x \pm \epsilon \cdot sign(\nabla_x J(x, y*)) \tag{1}$$

where x is the input data point and x_{adv} is the adversarial example. For targeted attacks, $y*$ indicates the targeted classification and for untargeted attacks, $y*$ indicates the actual classification. In this work, we are more interested in looking for all the boundary data regardless of which classification it is, thus we choose untargeted attacks to generate adversarial examples. The bottom-left image in Fig. 1 illustrates the selected data distribution.

Algorithm 1: RAST-AT Data Selection.

Input: Data D; DNN model f; FGSM-AT step size ϵ
Output: Selected data D_s
1: **for** data point x in D: **do**
2: $x_{adv} = x + \epsilon \cdot sign(\nabla_d J(x, y*))$
3: **if** f(x_{adv}) $\neq$ f(x) **then**
4: add x and x_{adv} to D_s
5: **end if**
6: **end for**
7: **return** D_s

RAST-AT. Our method: Given a model f, we are going to select the data point x which could successfully generate an adversarial example with a given attack level ϵ:

$$\begin{aligned} &x \subset D \;\; s.t., \;\; f(x_{adv}) \neq f(x) \\ &where \;\; x_{adv} = x + \epsilon \cdot sign(\nabla_d J(x, y*)) \end{aligned} \tag{2}$$

To prove our hypothesis that boundary data selected with adversarial examples can greatly improve the efficiency of adversarial training, we propose a method with three steps. First, we use FGSM to perform gradient-based adversarial attacks and determine the noise level's positive relationship with the success ratio. Second, we choose data that generates successful adversarial examples as boundary data. Finally, we label this boundary data as new training data to update the model and use it for adversarial training. This data selection process of RAST-AT is outlined in Algorithm 1.

Unlike data-driven techniques, our method's chosen data can be utilized for various models that have the same objective. By producing adversarial examples for the training data, we can assess data sensitivity and opt for more crucial data. It is noteworthy that our approach falls under weakly supervised learning, which means precise labels are not required for the training data. Instead, we depend on the output probability, which is the output value of the initial model, instead of the true labels. As a result, humans only need to label the data that was chosen for training purposes.

Adversarial training: Adversarial training is a deep learning model training method to train models more robust against adversarial attacks. Given a neural network f with parameters θ, the problem can be written as a robust optimization problem,

$$\min_{\theta} \sum_{i} \max_{\delta \subset \Delta} l(f_\theta(x_i + \delta), y_i) \tag{3}$$

where l is the loss function and Δ is the threat model. From the equation, the process firstly approximates the inner maximization of the threat model, and then the model parameters will update by the gradient descent. FGSM-AT is one way to approximate the inner maximization with perturbation calculated from:

$$\delta = \epsilon \cdot sign(\nabla_x J(\mathbf{x}, y*)) \tag{4}$$

FGSM-AT only calculates one step of the inner maximization, while PGD-AT adversarial training involves multiple steps of the inner maximization. To create a more robust model, we use the training data together with adversarial examples.

To compare the effectiveness of our RAST-AT method, we compare it with Wong et al.'s [14] FAST-AT method and PGD-AT adversarial training, aligning parameters with their work. Wong et al. demonstrate that FGSM-AT adversarial training can be as effective as PGD-AT by using random initialization.

For FAST-AT with T epochs, we have radius ϵ and step size α. The noises are initially generated with uniform distribution and then updated with the following equation:

$$\delta = random(-\epsilon, \epsilon)$$
$$\delta = \delta + \alpha \cdot sign(\nabla_\delta l(f(x+\delta), y) \quad (5)$$

where $\alpha = \epsilon = 8/255$ according to previous experimental results. Afterward, we clip the value and add the noises to the data point as adversarial examples. The adversarial examples and original data will combine to compose the training dataset.

For PGD-AT training, we update the noise similar to the FAST-AT, and we use a step size where $\alpha = 2/255$ and iterate 8 times to update the gradients and noises, which follows the setting in FAST-AT work [14].

Based on FAST-AT method, RAST-AT selects boundary data with Algorithm 1 to reduce the data size M, and updates the model using a different ϵ, where $\epsilon = \lambda$, which is related to the sensitivity level. To compare the influence of different data size, we use $\lambda = 2/255, \lambda = 4/255, \lambda = 8/255$ separately to check the difference. We use a DNN model which has a stochastic gradient descent (SGD) optimizer with a learning rate $= 0.001$ and momentum $= 0.9$. The comparison results are presented in Section V-C.

IV. Problem Formulation: A Case Study

We assessed our approach using a cobot, or collaborative robot, designed for human-robot interaction in shared environments [30]. It's imperative to note these robots often integrate AI models for complex tasks, but they can be susceptible to adversarial attacks, emphasizing the importance of system robustness.[1]

Our experiment is grounded on the adversarial patch attack detailed by Jia et al. [31]. This research targeted a robotic arm's AI-based object detector with an adversarial patch, misguiding its object location predictions and leading it to mistakenly strike a human. We employed RAST-AT to train our AI model to defend against this specific adversarial attack.

The system utilizes a camera on an industrial robotic arm (model UR10e) to identify and pick up cards, with detected card positions analyzed using the YOLO v3 algorithm, guiding the arm's actions. Jia et al.'s study [31] highlighted the vulnerabilities of deep learning models in industrial setups by successfully attacking the robotic arm with an adversarial patch, misleading the system's actions, either towards human hands or mislocating the card. This adversarial patch was optimized using YOLO v3, crafted through an iterative process using a set number of training images.

[1]The case study was conducted when the first author was affiliated with TUV SUD Asia Pacific and the Singapore University of Technology and Design.

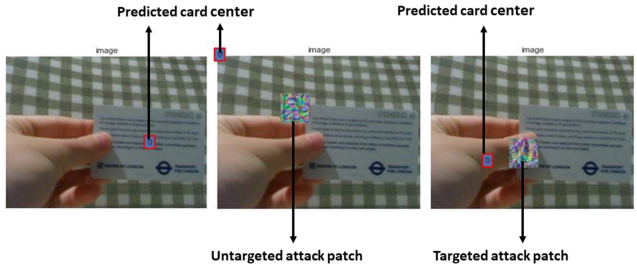

Fig. 2. Digital attack examples: untargeted attacks push the card center away from card; targeted attack move center prediction away from card and towards human hand.

A. Threat Model

We assume that within the targeted system, the client and server are separate parts from the well-protected physical system, which is easy for attackers to access. According to this information, we assume that the attacker can access the client and server parts with the following constraints:

- The attacker can access the information of the deep learning model (e.g. object detector) in the client.
- The attacker has access to the object and can modify it.
- The attacker cannot modify any program inside the system based on the integrity check mechanism [32].

The assumption of the threat model is made referring to the work [31] that has the same physical setup and scenario as ours.

B. Adversarial Attack Design

We build the system on the same robot model (UR10e) with the same setup and we train the AI model with RAST-AT method to improve the model's robustness as a defense.

We applied the attacks with digital images, and the attack performed with a 100% success rate when untargeted but with a 23.33% success rate when targeted. This is similar to the paper [31]. Fig. 2 shows the effect of attacks based on digital images with untargeted and targeted adversarial patches.

Given the differences between digital and physical environments, we incorporate the robotic working mechanisms and context variety into the training process and improve the attack with an optimization process as suggested in the work [31].

To address context variety, we collect images of cards held in different hands and environments and generate random noises optimized for one image, then use the updated patch as the initial patch for subsequent rounds of training. Training stops when the loss is below a specific threshold or after a certain number of iterations. The patch is pasted randomly with a higher probability in the central $1/3$ of the image to imitate actual scenarios. To address physical environment differences, we scan printed

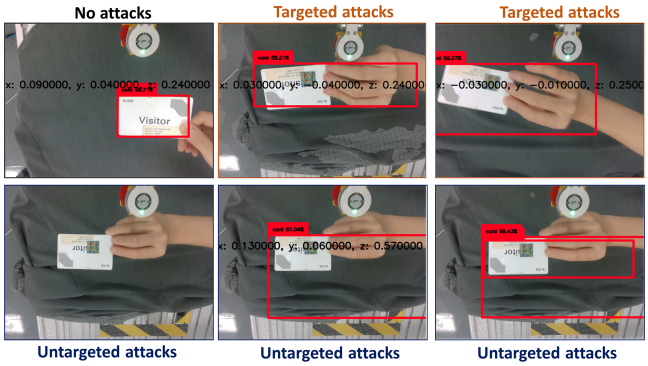

Fig. 3. Physical attack examples (camera view of the robotic arm): untargeted attacks push the card center away from card; targeted attack move center prediction away from card and towards human hand.

patches and compare them with digital patches to calculate bias δ, which is considered before printing. The patch is also randomly scaled within 20% of the original size to simulate physical environments.

The effects of physical attacks are shown in Fig. 3. The robotic arm clips the card correctly without the adversarial patch (top left). A targeted adversarial patch was trained to lead the robotic arm to hit the human's hand. The object detector then included the hand as its target and the robotic arm hit the human's hand (top middle and top right). The bottom three images show the problematic predictions of the object detector when seeing the adversarial patch, e.g., the object detector cannot see anything and returns to the default position without closing the gripper (bottom left). Additionally, with the untargeted adversarial patch, the object detector gives wrong or multiple predictions of the card coordinates. In summary, the object detector cannot detect the card or detect the wrong items when an adversarial patch is applied.

Our single test involves three steps: (1) presenting a card, (2) moving the camera to follow the card, and (3) completing the task and returning to the default position. A successful untargeted attack occurs when the gripper fails to close and clip the card, while a successful targeted attack occurs when the robotic arm hits the human's hand.

We conducted 40 attacks, divided into 20 untargeted and 20 targeted. Untargeted attacks had a success rate of *95%* and targeted attacks had a success rate of *65%*. Targeted attacks did not always make the gripper hit a human's hand but mostly clipped other locations. We shall later evaluate our method as a defense based against this attacking scenario and present the details in Section V-E.

V. Evaluation

We evaluate our method with respect to the following research questions (RQs), which require us to validate it against different datasets/models, compare it against other data selection methods, and finally, assess its feasibility in a real system.

- *RQ1 (Validity):* Can our method save time and maintain the performance of the model on different datasets?
- *RQ2 (Robustness):* Does our method perform better (accuracy and robustness) at selecting boundary data compared to other boundary data selection methods?
- *RQ3 (Efficiency):* Is our method faster while maintaining robustness compared to other adversarial training methods?
- *RQ4 (Transferability):* Can the selected data be used to train a different DNN model?
- *RQ5 (System Application):* Is our method effective in an actual system for defending against physical adversarial attacks?

TABLE I
Structure of DNN Models Used in Our Experiments

MNIST(Model A)	CIFAR-10(Model B)	CIFAR-10* (model C)
Conv (32,3,3)+ relu	Conv (32,3,3)+ relu	Conv (64,3,3)+ relu
Conv (32,3,3)+ relu	Max pooling (2,2)	Conv (64,3,3)+ relu
Max pooling (2,2)	Conv (64,3,3)+relu	Max pooling (2,2)
Conv (64,3,3)+ relu	Max pooling (2,2)	Conv (128,3,3)+ relu
Conv (64,3,3)+ relu	Flatten	Conv (128,3,3)+ relu
Max pooling (2,2)	Dense (64)+relu	Max pooling (2,2)
Flatten	Dense (10)+softmax	Flatten
Dense (200)+relu		Dense (256)+relu
Dense (10)+softmax		Dense (256)+relu
		Dense (10)+softmax

We run experiments on hardware with a GPU (NVIDIA GeForce GTX 1050) and the Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz processor, using the Windows 10 operating system. The software frameworks are Python (version 3.6.13), NumPy (version 1.19.1), TensorFlow-GPU (version 2.1.0), and Keras (version 2.3.1).

A. RQ1: Validation

To assess the impact of data selection on model performance before and after RAST, we have chosen to apply our methodology to two widely used datasets, CIFAR-10 and MNIST. Specifically, we have trained two convolutional neural networks (CNNs), denoted as model A and model B, whose structures are outlined in Table I (model C is to be discussed in a later experiment). CIFAR-10 is a dataset of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The MNIST dataset is a collection of 70,000 images of handwritten digits (0-9), each of size 28x28 pixels. We choose a CNN model with a training epoch of 50 to demonstrate the performance of the trained model with and without data selection.

We train the initial model using 20% randomly selected data from two datasets. The model is updated separately with all data points and selected data points based on a controlled attack level ϵ. Sensitivity of each data point is defined as the minimum ϵ value that can generate adversarial examples. We use λ to select data points with sensitivity less than the given value. CIFAR-10 and MNIST provide 50 k and 60 k unlabeled training data, respectively, and 10 k labeled testing data for each.

We then record the number of selected data points, time and performance of the updated model with different step sizes ϵ. Note that, $\epsilon = 0$ represents there is no selection and all data are considered training data. Therefore, $\lambda = 0$ is the benchmark. Fig. 4 shows the results.

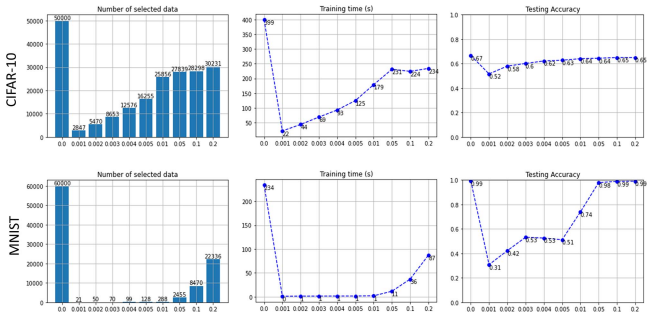

Fig. 4. Data selection results on CIFAR-10 and MNIST. y-axis: number of selected data, time of training, and testing accuracy vs x-axis: different values of λ (NB: λ = 0 indicates that all data is used for training).

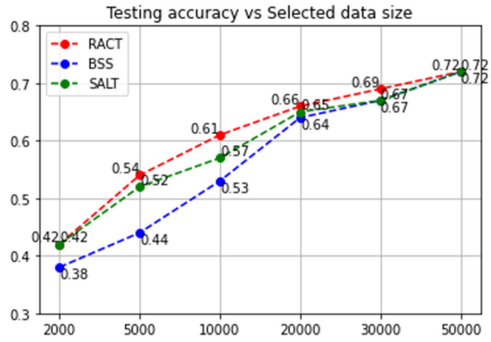

Fig. 5. Results of accuracy with BSS, SALT, and RAST method. y-axis: testing accuracy vs x-axis: number of selected data.

TABLE II
RESULTS OF DIFFERENT λ VALUES COMPARING WITH BENCHMARK

λ	CIFAR-10			MNIST		
	% data size reduced by	% time reduced by	% testing accuracy sacrificed	% data size reduced by	% time reduced by	% testing accuracy sacrificed
0.001	94.25%	94.49%	15.00%	99.97%	100.00%	68.00%
0.002	89.06%	88.97%	9.00%	99.92%	99.57%	57.00%
0.003	82.69%	82.71%	7.00%	99.88%	99.57%	46.00%
0.004	74.85%	76.69%	5.00%	99.84%	99.57%	46.00%
0.005	**67.49%**	**68.67%**	**4.00%**	99.79%	99.57%	48.00%
0.01	48.29%	55.14%	3.00%	99.52%	99.57%	25.00%
0.05	44.32%	42.11%	3.00%	**95.91%**	**95.30%**	**1.00%**
0.1	43.40%	43.86%	2.00%	85.88%	84.62%	0.00%
0.2	39.54%	41.35%	2.00%	62.77%	62.82%	0.00%

Increasing λ leads to the selection of more data and increases training time, as shown in the figure. We compare the number of selected data, training time, and testing accuracy with the benchmark in Table II. The optimized combination is highlighted in bold. While training accuracy remains similar, testing accuracy improves. As λ increases, selected data stabilizes, and testing accuracy approaches the benchmark, achieving 0.04 ($\lambda = 0.005$). Data size reduces by 67.49%, and training time reduces by 68.67% compared to the benchmark.

When λ is small, a slight increase (e.g. 0.001) can greatly affect the selected data, while a larger λ (i.e. $\lambda \geq 0.01$) needs a greater increase (e.g. 0.1) to affect data selection. This suggests that boundary data is sensitive to adversarial attacks, and data with smaller sensitivity values are crucial for model training as they are closer to the boundary.

B. RQ2: Robustness

To demonstrate RAST is robustness-oriented compared to other boundary data selection methods, we compared it with other boundary data selection methods. There are very few works that select boundary data for DNN: one is the Boundary Sample Selection (BSS) [19] which shares some similarities with our method and targets a similar goal of boundary data selection, albeit it was originally designed for mutation testing. BSS only selects data from the output layer of DNNs, focuses on the top two classes, and uses a threshold value of T_v to identify boundary data based on the ratio of the highest output probability to the second-highest probability. Another boundary data selection method is Adversarial Active Learning (SALT) [20], which adds noise to the data and selects the data point that gives maximum uncertainty.

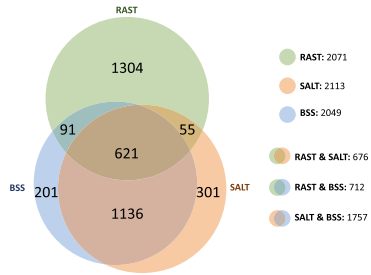

Fig. 6. Number of data selected by RAST, BSS, and SALT, and their overlap demonstration.

In the BSS approach, given a DNN model f for n classification, and unlabeled data D, for every data point d in D, 1) run model f on d to get the predicted probability $y_1, y_2, \ldots, y_n$; 2) calculate the ratio of max and second max predictions $r = y_{max}/y_{secmax}$; 3) if the ratio is lower than a threshold τ, add the data point to selected dataset D_s.

The core idea of SALT is to calculate the uncertainty after adding a small perturbation to the data. Different from RAST which adds a specific perturbation, SALT adds a uniform perturbation which is 0.01 to all the data. Moreover, RAST checks if the data has been attacked empirically while SALT calculates uncertainty.

We test the models on CIFAR-10 dataset, with 20% (10 k) of 50 k training data and $\epsilon = 4/255$ and $\alpha = 2/255$ with 8 iterations for FGSM and PGD attacks. RQ2 is answered by comparing BSS, SALT and RAST-AT methods on CIFAR-10, where RAST-AT outperforms BSS and SALT with higher accuracy, especially training with fewer data points. All methods are approaching the benchmark as data size increases. Fig. 5 show the accuracy vs. data size.

Furthermore, we perform a comparison involving approximately the top 2000 data points that were chosen using the RAST method ($\lambda = 0.01$), the BSS method ($probability\ ratio = 7$), and the SALT method ($uncertainty = 0.44$). The specifics are illustrated in Fig. 6. It's evident that there's a significant overlap between the BSS and SALT selections, while RAST selects a different amount of data. According to the testing results of Fig. 5 and Table III, RAST selects the most valuable data for adversarial training towards accuracy and robustness.

TABLE III
PERFORMANCE OF DIFFERENT ADVERSARIAL MODELS BEFORE AND AFTER PGD/FGSM ATTACKS

	Datasize	Training time (s)	Test Acc	FGSM robustness	PGD robustness
PGD-AT	100000	149.00	**0.83**	0.59	0.56
FAST-AT	100000	86.48	0.79	**0.72**	0.74
RAST-AT ($\lambda = 10/255$)	81923	**68.52**	0.79	**0.72**	**0.75**

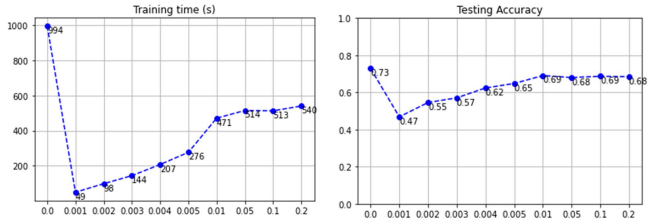

Fig. 7. Results of using selected data to train a new model. y-axis: left-training time (s); right-testing accuracy vs x-axis: different values of λ (NB: $\lambda = 0$ indicates that all data is used for training).

TABLE IV
RESULTS OF TESTS BEFORE AND AFTER APPLICATION OF RAST-AT

num(rate)	Total num	Attack success	Attack success (after RAST-AT)	Improved
Targeted attacks	20	13	9	20%
Untargeted attacks	20	19	7	60%

C. RQ3: Efficiency

To check if RAST is effective in adversarial training, we designed experiments to compare with other efficient adversarial training methods [25].

We select data by applying FGSM, therefore, we evaluate the robustness by applying FGSM attack ($\epsilon = 4/255$). Moreover, we apply PGD attack ($\alpha = 2/255$ with iteration $= 8$), which is one of the strongest first-order attacks. We compare PGD-AT, FAST-AT and RAST-AT adversarial training methods of time consumption and evaluate the robustness of FGSM and PGD attacks.

The original model is trained with 50 k data points and 50 epochs and has an accuracy of 0.82 on 10 k testing data. We test the model's robustness by applying FGSM and PGD adversarial attacks to the testing dataset and select adversarial examples to test the updated model after adversarial training. We evaluate the updated model's robustness by checking the accuracy of adversarial examples, which is summarised in Table III.

The RAST-AT method is the fastest with $\epsilon = 2/255$ according to the table, but with lower robustness when ϵ is small. As ϵ increases, robustness also increases and can approach FAST-AT performance at $\epsilon = 8/255$, while still being 25% faster. For a balance between speed and robustness, RAST-AT with $\epsilon = 4/255$ sacrifices 3% robustness in FGSM attacks and 4% in PGD attacks, while being 30% faster. This is because RAST-AT can select sensitive and important data points, allowing the model to be trained with fewer data points while maintaining accuracy and robustness.

D. RQ4: Transferability

To minimize the effort in real-world applications, we show the transferability of RAST that the selected boundary data works for a different model structure with the same target

Fig. 7 shows the training time and testing accuracy. With more data selected, the accuracy increases and after $\lambda = 0.01$, the accuracy tends to be stable and increases slowly. Using the selected data from model B (described previously), we trained a new model C (structure in Fig. I under column CIFAR-10*), showing that the same boundary data works for a different model with the same target. Fig. 7 depicts the training time and testing accuracy, where more data selected leads to higher accuracy, and after $\lambda = 0.01$, the accuracy remains stable and increases slowly.

Our method is particularly beneficial in practical situations where there are limitations on data size or training time, as these constraints are typical in real-world applications. Applying our method improves the efficacy of the selected data, making it more useful in training a robust model despite limited resources.

E. RQ5: System Application

To test if RAST-AT is applicable to an actual system to defend against physical attacks,[2] we apply RAST-AT to the real-world cobot described in Section IV.

Using our method, we train the model again by employing 592 images for object detection training, which is the same as the training data size used in a previous study [31]. A study by Xu et al. [9] has shown that adversarial examples can be used to target object detection algorithms like YOLO. Hence, we utilize RAST-AT to select adversarial examples from the chosen dataset to retrain the model and demonstrate that the model trained with RAST-AT data is more resilient than the original model.

To enhance the model's resilience, we utilize our approach by training it with more sensitive data and their boundary examples. Our data selection algorithm (Algorithm 1) aids us in choosing suitable images. After applying RAST-AT for data selection, we have 322 data points for training, with an epsilon value of $\epsilon = 8/255$. This value is preferred due to its high accuracy and robustness in comparison to other values discussed in Section V-C. We substitute the current model in the system with the updated model and perform our evaluations again. Fig. 8 shows the results before and after we apply our method. We repeat defense experiments on targeted and untargeted attacks 20 times each and compare them with attack experiments. We summarise the results in Table IV.

[2]The experiment was done when the first author was working at TUV SUD AP

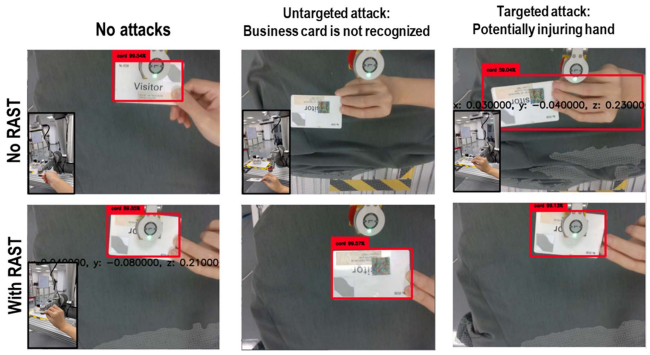

Fig. 8. Comparison before and after application of RAST-AT. 1st row: the moment when gripper clips before application of RAST-AT. 2nd row: camera view after application RAST-AT.

Based on the table data, it can be inferred that RAST-AT shows superior performance when confronted with untargeted attacks compared to targeted attacks. Despite the fact that targeted attacks are generally more resilient, our approach still proves to be effective in enhancing the model's resilience.

VI. Conclusion and Future Work

We proposed a new method for selecting boundary data for deep learning models and evaluated it on CIFAR-10 and MNIST. The method speeds up training by 68.67% on CIFAR-10, 95.3% on MNIST, achieving a 10% increase in accuracy and 7% and 6% higher robustness against FGSM and PGD attacks, respectively. The method was also found to be faster than other adversarial training methods while providing similar or better performance. It was validated on a robotic arm system, showing a 20% improvement in robustness against targeted attacks and 60% against untargeted attacks, making it an effective defense against physical adversarial attacks.

In theory, methods such as iFGSM and PGD can replace FGSM for boundary data selection. Adversarial training reshapes this boundary, allowing iterative data selection until stability is reached. Yet, for real-world application, there's a need to harmonize efficiency, accuracy, and robustness. Methods like iFGSM and PGD are more time-consuming than FGSM, and continuous boundary data selection adds to this. Determining the optimal trade-off between these attributes demands more research. Future studies will test varied attack methods on different robots to find the right equilibrium for AI-driven industrial systems.

References

[1] S. Benhimane, H. Najafi, M. Grundmann, Y. Genc, N. Navab, and E. Malis, "Real-time object detection and tracking for industrial applications," in *Proc. VISAPP*, 2008, vol. 2, pp. 337–345.

[2] C. Ge, J. Wang, J. Wang, Q. Qi, H. Sun, and J. Liao, "Towards automatic visual inspection: A weakly supervised learning method for industrial applicable object detection," *Comput. Ind.*, vol. 121, 2020, Art. no. 103232.

[3] D.-I. B. Hagebeuker et al., "A 3D time of flight camera for object detection," *PMD Technol. GmbH*, vol. 2, 2007.

[4] C. Tang, N. Luktarhan, and Y. Zhao, "SAAE-DNN: Deep learning method on intrusion detection," *Symmetry*, vol. 12, no. 10, p. 1695, 2020.

[5] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput.*, 2017, pp. 313–316.

[6] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi, and M. Qiu, "Adversarial attacks against network intrusion detection in IoT systems," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10327–10335, Jul. 2021.

[7] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[8] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Trans. Autom. Control*, vol. 59, no. 6, pp. 1454–1467, Jun. 2014.

[9] K. Xu et al., "Adversarial t-shirt! Evading person detectors in a physical world," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 665–681.

[10] K. Eykholt et al., "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1625–1634.

[11] A. Braunegg et al., "Apricot: A dataset of physical adversarial attacks on object detection," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 35–50.

[12] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2005, pp. 641–647.

[13] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng.*, 2019, pp. 1245–1256.

[14] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proc. Int. Conf. Learn. Representations*, 2019.

[15] A. Shafahi et al., "Adversarial training for free!," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3353–3364.

[16] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *J. Artif. Intell. Res.*, vol. 4, pp. 129–145, 1996.

[17] S. Sinha, S. Ebrahimi, and T. Darrell, "Variational adversarial active learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5972–5981.

[18] M.-F. Balcan, A. Broder, and T. Zhang, "Margin based active learning," in *Proc. Int. Conf. Comput. Learn. Theory*, 2007, pp. 35–50.

[19] W. Shen et al., "Boundary sampling to boost mutation testing for deep learning models," *Inf. Softw. Technol.*, vol. 130, 2021, Art. no. 106413.

[20] B. Miller et al., "Adversarial active learning," in *Proc. Workshop Artif. Intell. Secur. Workshop*, 2014, pp. 3–14.

[21] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015.

[23] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*. Boca Raton, FL, USA: Chapman and Hall/CRC, 2016, pp. 99–112.

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018.

[25] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, 2021, pp. 4312–4321, doi: 10.24963/ijcai.2021/591.

[26] D. Hwang and D. Kim, "Near-boundary data selection for fast support vector machines," *Malaysian J. Comput. Sci.*, vol. 25, no. 1, pp. 23–37, 2012.

[27] F. Zhu, N. Ye, W. Yu, S. Xu, and G. Li, "Boundary detection and sample reduction for one-class support vector machines," *Neurocomputing*, vol. 123, pp. 166–173, 2014.

[28] M. Aslani and S. Seipel, "Efficient and decision boundary aware instance selection for support vector machines," *Inf. Sci.*, vol. 577, pp. 579–598, 2021.

[29] S. Xie and M. Lu, "Interpreting and understanding graph convolutional neural network using gradient-based attribution method," 2019, *arXiv:1903.03768*.

[30] F. Vicentini, "Collaborative robotics: A survey," *J. Mech. Des.*, vol. 143, no. 4, 2021, Art. no. 040802.

[31] Y. Jia, C. M. Poskitt, J. Sun, and S. Chattopadhyay, "Physical adversarial attack on a robotic arm," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9334–9341, Oct. 2022.

[32] R. Taylor, "An integrity check value algorithm for stream ciphers," in *Proc. 13th Annu. Int. Cryptol. Conf.*, 1993, pp. 40–48.