Moving Target Defense for Embedded Deep Visual Sensing against Adversarial Examples

Qun Song*
Energy Research Institute,
Interdisciplinary Graduate School
Nanyang Technological University
Singapore
song0167@ntu.edu.sg

Zhenyu Yan
School of Computer Science and Engineering
Nanyang Technological University
Singapore
zyan006@ntu.edu.sg

Rui Tan
School of Computer Science and Engineering
Nanyang Technological University
Singapore
tanrui@ntu.edu.sg

ABSTRACT

Deep learning-based visual sensing has achieved attractive accuracy but is shown vulnerable to adversarial example attacks. Specifically, once the attackers obtain the deep model, they can construct adversarial examples to mislead the model to yield wrong classification results. Deployable adversarial examples such as small stickers pasted on the road signs and lanes have been shown effective in misleading advanced driver-assistance systems. Many existing countermeasures against adversarial examples build their security on the attackers' ignorance of the defense mechanisms. Thus, they fall short of following Kerckhoffs's principle and can be subverted once the attackers know the details of the defense. This paper applies the strategy of *moving target defense* (MTD) to generate multiple new deep models after system deployment, that will collaboratively detect and thwart adversarial examples. Our MTD design is based on the adversarial examples' minor transferability across different models. The post-deployment dynamically generated models significantly increase the bar of successful attacks. We also apply serial data fusion with early stopping to reduce the inference time by a factor of up to 5. Evaluation based on four datasets including a road sign dataset and two GPU-equipped Jetson embedded computing platforms shows the effectiveness of our approach.

CCS CONCEPTS

• **Security and privacy** → *Software and application security*; • **Computing methodologies** → *Neural networks*; • **Computer systems organization** → *Embedded and cyber-physical systems.*

KEYWORDS

Neural networks, adversarial examples, moving target defense

ACM Reference Format:
Qun Song, Zhenyu Yan, and Rui Tan. 2019. Moving Target Defense for Embedded Deep Visual Sensing against Adversarial Examples. In *The 17th ACM Conference on Embedded Networked Sensor Systems (SenSys '19), November 10–13, 2019, New York, NY, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3356250.3360025

*Also with School of Computer Science and Engineering, Nanyang Technological University.

SenSys '19, November 10–13, 2019, New York, NY, USA

ACM ISBN 978-1-4503-6950-3/19/11…$15.00
https://doi.org/10.1145/3356250.3360025

1 INTRODUCTION

To implement autonomous systems operating in complex environments (e.g., the long envisaged self-driving cars), the accurate and resilient perception of the environment is often the most challenging step in the closed loop of sensing, decision, and actuation. The recent advances of deep learning [32, 39] have triggered great interests of applying it to address the environment perception challenges. For instance, deep learning-based computer vision techniques have been increasingly adopted on commercial off-the-shelf advanced driver-assistance systems (ADAS) [30, 38].

However, recent studies show that deep models (e.g., multilayer perceptrons and convolutional neural networks) are vulnerable to *adversarial examples*, which are inputs formed by applying small but crafted perturbations to the *clean examples* in order to make the victim deep model yield wrong classification results. Systematic approaches have been developed to generate adversarial examples as long as the attackers acquire the deep model, where the attackers may know the internals of the model [25] or not [53]. Certain constraints can be considered in the generation process when the attackers cannot tamper with every pixel of the input. For example, in [21], an algorithm is developed to determine *adversarial stickers* that can be implemented by physically pasting small paper stickers on road signs to mislead vision-based sign classifier. Moreover, as demonstrated in [68] and explained in [67], the vision-based lane detector of Tesla Autopilot, which is an ADAS, can be fooled by small adversarial stickers on the road and thus direct the car to the opposite lane, creating life-threatening danger. Thus, adversarial examples present an immediate and real threat to deep visual sensing systems. The design of these systems must incorporate effective countermeasures especially under the safety-critical settings.

Existing countermeasures aim at hardening the deep models through adversarial training [25, 35, 45], adding a data transformation layer [6, 14, 15, 19, 27, 44, 72, 75, 77], and gradient masking [9, 55, 58, 63]. These countermeasures are often designed to address certain adversarial examples and build their security on the attackers' ignorance of the defense mechanisms (e.g., the adversarial example generation algorithms used in adversarial training, the data transformation algorithms, and the gradient masking approaches). Thus, they do not address adaptive attackers and fall short of following Kerckhoffs's principle in designing secure systems (i.e., the enemy knows the system except for the secret key [61]). Once the

attackers acquire the hardened model and the details of the defense mechanisms, they can craft the next-generation adversarial examples to render the hardened model vulnerable again [3, 10].

At present, the deep model training still requires considerable expertise and extensive fine-tuning. As such, in the current practice, a factory-designed deep model is often massively deployed to the products and remains static until the next software upgrade. Such a deployment manner grants the attackers advantage of time. They can extract the deep model (which may have been hardened by the existing countermeasures) from the software release or the memory of a purchased product, study it, and construct the next-generation adversarial examples to affect all products using the same deep model.

Beyond the static defense, in this paper, we consider a *moving target defense* (MTD) strategy [33]. MTD aims to create and deploy mechanisms that are diverse and that continually change over time to increase complexity and cost for attackers, limit the exposure of vulnerabilities and opportunities for attack, and increase system resiliency [49]. In the MTD of this work, we generate one or more new deep models after system deployment that the attackers can hardly acquire. Different from the identical and static deep model that results in a single point of failure, the generated post-deployment models are distinct across the systems. This approach invalidates an essential basis for the attackers to construct effective adversarial examples, i.e., the acquisition of the deep model. Taking the deep visual sensing of ADAS as an example, under the MTD strategy, new deep models can be continually trained when the computing unit of a car is idle. Once the training completes with the validation accuracy meeting the manufacturer-specified requirement, the new deep models can be commissioned to replace the in-service models that were previously generated on the car. By bootstrapping the *in situ* training with randomness, it will be practically difficult for the attackers to acquire the in-service deep models, which thus can be viewed as the secret of the system. With MTD, the adversarial examples constructed based on a stolen deep model are neither effective across many systems nor effective against a single victim system over a long period of time. In particular, extracting the private deep models from a victim system will require physical access. If the attackers have such physical access, they should launch the more direct and devastating physical attacks that are out of the scope of this paper.

In this paper, we design an MTD approach for embedded deep visual sensing systems that are susceptible to adversarial examples, such as ADAS [7, 20] and biometric authentication [28]. Several challenges need to be addressed. First, adversarial examples have non-negligible transferability to new deep models [25, 42, 52, 66]. From our evaluation based on several datasets, although a new deep model can largely thwart the adversarial examples constructed based on a static *base model*, the adversarial examples can still mislead the new deep models with a probability from 7% to 17%. Second, the primitive MTD design of using a single new deep model does not give awareness of the presence of adversarial examples, thus losing the opportunities of involving the human to improve the safety of the system. Note that human can be considered immune to adversarial examples designed based on the perturbation minimization principle. Third, *in situ* training of the new deep models without resorting to the cloud is desirable given the concerns of eavesdropping and tampering during the communications over the Internet. However, the training may incur significant computational overhead for the embedded systems.

To collectively address the above challenges, we propose a fork MTD (fMTD) approach based on three key observations on the responses of new deep models to the adversarial examples constructed using the base model. First, the output of a new deep model that is successfully misled by an adversarial example tends to be unpredictable, even though the adversarial example is constructed toward a target label [42]. Second, from the minor transferability of adversarial examples and the unpredictability of the misled new model's output, if we use sufficiently many distinct new models to classify an adversarial example, a majority of them will give the correct classification result while the inconsistency of all the models' outputs (due to the unpredictability) signals the presence of attack. This multi-model design echos ensemble machine learning [16]. Third, compared with training a new deep model from scratch, the training with a perturbed version of the base model as the starting point can converge up to 4x faster, imposing less computation burden.

Based on the above observations, we design fMTD as follows. When the system has spare computing resources, it adds independent perturbations to the parameters of the base model to generate multiple *fork models*. The base model can be a well factory-designed deep model that gives certified accuracy for clean examples, but may be acquired by the attackers. Each fork model is then used as the starting point of a retraining process. The retrained fork models are then commissioned for the visual sensing task. As the fork models are retrained from the base model, intuitively, they will inherit much of the classification capability of the base model for clean examples. At run time, an input, which may be an adversarial example constructed based on the base model, is fed into each fork model. If the degree of inconsistency among the fork models' outputs exceeds a predefined level, the input is detected as an adversarial example. The majority of the fork models' outputs is yielded as the final result of the sensing task. If the system operates in the human-in-the-loop mode, the human will be requested to classify detected adversarial examples. The human inference can be used because of the following two factors. First, as human perception can be considered immune to adversarial examples, its on-demand use can improve the system's performance in thwarting the attack. Second, the admission of human inputs is also consistent with the design of existing off-the-shelf ADAS that require human driver's continuous monitoring of road environment and intervention when necessary.

The run-time inference overhead of fMTD is proportional to the number of fork models used. Based on our performance profiling on NVIDIA Jetson AGX Xavier and Jetson Nano, which are two GPU-equipped embedded computing platforms with different computing resources, instructing TensorFlow to execute the fork models at the same time brings limited benefit in shortening inference time. In contrast, the serial execution of them admits an early stopping mechanism inspired by the serial signal detection [56]. Specifically, the system runs the fork models in serial and terminates the execution once sufficient confidence is accumulated to decide the cleanness of the input. Evaluation results show that

the serial fMTD reduces the inference time by a factor of up to 5 compared with instructing TensorFlow to execute all fork models.

The contributions of this paper are summarized as follows.

- Based on important observations on the responses of deep models to adversarial examples, we design fMTD to counteract adversarial example attacks as an ongoing concern.
- We conduct extensive evaluation on fMTD's accuracy in classifying clean examples as well as its performance in detecting and thwarting adversarial examples under a wide range of settings. The results provide useful guidelines for adopters of fMTD in specific applications.
- We show that the serial execution of the fork models with early stopping can significantly reduce the inference time of fMTD while maintaining the sensing accuracy in both the absence and presence of attacks.

The reminder of this paper is organized as follows. §2 reviews background and related work. §3 presents a measurement study to motivate the fMTD design. §4 designs fMTD. §5 evaluates the accuracy and attack detection performance of fMTD. §6 profiles fMTD on Jetson and evaluates the serial fMTD. §7 discusses several issues not addressed in this paper. §8 concludes this paper.

2 BACKGROUND AND RELATED WORK

2.1 Adversarial Examples and Construction

Adversarial examples are crafted inputs to mislead deep models to produce incorrect results. Let $f_{\boldsymbol{\theta}}(\mathbf{x})$ denote a classifier, where $\boldsymbol{\theta}$ is the classifier's parameter and $\mathbf{x} \in [0,1]^m$ is the input (e.g., an image). Let y denote the ground truth label of $\mathbf{x}$. The $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta} \in [0,1]^m$ is an adversarial example, if $f_{\boldsymbol{\theta}}(\mathbf{x}') \neq y$. The $\boldsymbol{\delta}$ is the perturbation designed by the attackers. A *targeted* adversarial example $\mathbf{x}'$ makes $f_{\boldsymbol{\theta}}(\mathbf{x}') = y_t$, where $y_t \neq y$ is a specified target label. A *non-targeted* adversarial example ensures that the classification result $f_{\boldsymbol{\theta}}(\mathbf{x}')$ is an arbitrary label other than the true label y. If the attackers need no knowledge of the classifier's internals (e.g., architecture, hyperparameters, and parameters) to construct the adversarial example, the attack is called *black-box* attack. Otherwise, it is called *white-box* attack. In this work, we consider both targeted and non-targeted adversarial examples. As the objective of this paper is to develop a defense approach, it is beneficial to consider the stronger white-box attack, in which the attackers have the knowledge of the internals of the base model.

To increase the stealthiness of the attack to human perception, the difference between $\mathbf{x}$ and $\mathbf{x}'$, denoted by $D(\mathbf{x}, \mathbf{x}')$, is to be minimized. Thus, the construction of the perturbation for a targeted adversarial example, denoted by $\boldsymbol{\delta}^*_{y_t}$, can be formulated as [66]: $\boldsymbol{\delta}^*_{y_t} = \text{argmin}_{\boldsymbol{\delta}}\, D(\mathbf{x}, \mathbf{x}')$, subject to $f_{\boldsymbol{\theta}}(\mathbf{x}') = y_t$ and $\mathbf{x}' \in [0,1]^m$. The targeted adversarial example that gives the minimum $D(\mathbf{x}, \mathbf{x}')$ can be yielded as a non-targeted adversarial example. Various *gradient-based* approaches have been proposed to construct adversarial examples [4, 12, 13, 25, 37, 47, 48, 54, 59, 65, 66]. Among them, the approach proposed by Carlini and Wagner (C&W) [12] is often thought highly effective and used to evaluate various defense approaches [1]. We briefly introduce it here. C&W's approach instantiates the distance to be ℓ_p-norm and apply Lagrangian relaxation to simplify the formulation as: $\boldsymbol{\delta}^*_{y_t} = \text{argmin}_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_p + c \cdot g(\mathbf{x}')$, subject to $\mathbf{x}' \in [0,1]^m$, where the regularization $g(\mathbf{x}') = \max\left\{\max_{y_i \neq y_t}\left\{Z(\mathbf{x}')_{y_i}\right\} - Z(\mathbf{x}')_{y_t}, -\kappa\right\}$, $Z(\cdot)$ represents softmax and κ is a parameter controlling the strength of the constructed adversarial example. With a larger κ, the $\mathbf{x}'$ is more likely classified as y_t, but the perturbation $\boldsymbol{\delta}$ will be larger. In the inner loop of C&W's algorithm for a certain weight c, gradient descent is used to solve the relaxed formulation. In the outer loop, binary search is applied to find a setting for c to further minimize the objective function. In this paper, we use C&W's approach to generate adversarial examples and evaluate fMTD. Note that the design of fMTD does not rely on any specifics of the C&W's approach.

2.2 Countermeasures to Adversarial Examples

Overfitted models are often thought highly vulnerable to adversarial example attacks. However, regularization approaches for preventing overfitting, such as dropout and weight decay, are shown ineffective in precluding adversarial examples [25, 66]. Brute-force adversarial training [25, 35, 45] can make a deep model immune to predefined adversarial examples. However, it can be defeated by the adversarial examples that are not considered during the adversarial training. A range of other defense approaches apply various transformations to the input during both the training and inference phases. The transformations include compression [6, 14, 15, 19, 27], cropping and foveation [27, 44], data randomization [75], and data augmentation [77]. These approaches often lead to accuracy drops on clean examples [76] and are only effective against the adversarial examples constructed without the knowledge of the transformation. *Gradient masking* is another category of defense against the adversarial examples constructed using gradient-based methods [9, 55, 58, 63]. It attempts to deny adversary access to useful gradients for constructing attack. However, as shown in [3], if the attackers know the details of the transformation or the gradient masking, they can still construct effective adversarial examples. *Provable defense* [18, 57, 74] gives lower bounds of the defense robustness. However, its key limitation is that the lower bound is applicable for a set of specific adversarial examples only.

The key difference between our approach and the existing approaches is that ours is a dynamic defense while existing approaches are static defenses. As pointed out by [10], a main drawback of most existing attack prevention and thwarting approaches against adversarial examples is that they do not consider adaptive attackers. Once the attackers acquire the details of the defense, the attackers can design the next-generation adversarial examples and bypass the defense. In other words, these approaches' effectiveness is contingent on the attacker's ignorance of the defense mechanism. Differently, our approach applies dynamic deep models to significantly increase the barrier for the attacker to craft effective adversarial examples. Our approach is supplementary to the existing defense approaches, in that our approach takes effect during the run time of the protected system, whereas the existing approaches are applied during the design phase of the system.

In addition to attack prevention and thwarting, adversarial example detection has also received research. For example, a second classifier can be built to classify an input as clean or adversarial [24, 43, 46]. Statistical properties of the inputs such as principle

component have been used to detect attacks [5, 31, 41]. Others resort to statistical testing [23, 26]. However, these approaches cannot detect crafty attacks such as the C&W's attack [11].

2.3 Moving Target Defense

Static defense grants attackers the advantage of time. MTD is an emerging approach to address this issue and increase the barrier for effective attacks [33]. For instance, the computer hosts can mutate their IP addresses such that the attack traffic is directed to wrong destinations [2]. In the approach described in [60], a deep model is randomly selected from a set of candidate models each time to classify an input. The approach uses a limited number of candidate models (e.g., 3 to 6 [60]) and assumes that they are known to the attackers. Its effectiveness of thwarting the attacks is merely based on the attackers' ignorance of which model is being used, thus following a weak form of MTD. Given the limited number of candidate models, it is not impossible for the attackers to construct an adversarial example that can mislead all candidate models. Moreover, the approach [60] is short of attack detection capability since a single model is used each time. In contrast, fMTD applies an ensemble of locally generated deep models to achieve both attack detection and thwarting capabilities. Thus, fMTD follows a strong form of MTD.

3 MEASUREMENT STUDY

We conduct measurements to gain insights for MTD design.

3.1 Used Datasets and Deep Models

We first introduce the datasets and the deep models used in this measurement study as well as the extensive performance evaluation for fMTD in §5. We use the following three datasets:

- **MNIST** [40] is a dataset consisting of 60,000 training samples and 10,000 test samples. Each sample is a 28×28 grayscale image showing a handwritten digit from 0 to 9. We select 5,000 training samples as the validation dataset.
- **CIFAR-10** [36] is a 10-class dataset consisting of 50,000 training samples and 10,000 test samples. Each sample is a 32×32 RGB color image. The 10 classes are airplanes, cars, birds, cats, deers, dogs, frogs, horses, ships, and trucks. We select 5,000 training samples as the validation dataset.
- **GTSRB** [64] (German Traffic Sign Recognition Benchmark) is a 43-class dataset with more than 50,000 images sizing from 15×15 to 250×250 pixels. For convenience, we resize all the images to 32×32 pixels by interpolation or downsampling. We divide them into training, validation, and test datasets with 34799, 4410, and 12630 samples, respectively.

We adopt two convolutional neural network (CNN) architectures that have been used in [12] and [55], referred to as CNN-A and CNN-B. Their structures and training hyperparameters can be found in [62]. We apply CNN-A to MNIST. It is trained on MNIST using the momentum-based stochastic gradient descent. CNN-A achieves training and validation accuracy of 99.84% and 99.44%, respectively. We apply CNN-B to CIFAR-10 and GTSRB. CNN-B's main difference from CNN-A is that more convolutional filters and more rectified linear units (ReLUs) in the fully connected layers are used to address the more complex patterns of the CIFAR-10 and GTSRB images. Its softmax layer has 10 or 43 classes for CIFAR-10 and GTSRB, respectively. For CIFAR-10, CNN-B achieves a validation accuracy of 79.62%. This result is consistent with those obtained in [12] and [55]. For GTSRB, CNN-B achieves training and validation accuracy of 99.93% and 96.64%, respectively.

Table 1: Targeted adversarial examples constructed using C&W approach [12] with ℓ_2-norm and various κ settings.

		Clean example	Attack's target label					
			2	3	2	3	2	3
Ground truth label	0							
	1							
			$\kappa = 0$		$\kappa = 45$		$\kappa = 95$	

The MNIST and CIFAR-10 datasets have been widely used in image recognition and machine learning research. The use of these two datasets allows us to adopt the CNN architectures that have been shown suitable for them. From the achieved training and validation accuracy, MNIST and CIFAR-10 are representatives of data with simple and complex patterns, respectively. GTSRB gives realism since road sign recognition must be part of ADAS's visual sensing. However, this study does not cater to any specific application. While the detailed results (e.g., classification accuracy) may differ across datasets, we will draw common observations from the results obtained based on these three datasets.

3.2 Measurement Results

In this section, we conduct measurements to investigate the responses of multiple *new models* to adversarial examples constructed based on the *base model* that is different from the new models.

3.2.1 Adversarial examples. We use the deep models described in §3.1 as the base models for the three datasets. Then, we use the C&W approach described in §2.1 to generate adversarial examples based on the base model. Specifically, for each dataset, we select a clean test sample in each class as the basis for constructing the targeted adversarial examples whose targeted labels are the remaining classes. For instance, as MNIST has 10 classes, a total of $10 \times 9 = 90$ targeted adversarial examples will be generated. To generate non-targeted adversarial examples for each dataset, we randomly select 100 test samples as the bases for the construction by following the procedure described in §2.1. The C&W's adversarial examples are highly effective – all adversarial examples that we generate are effective against the base model.

As described in §2.1, the κ is an important parameter of the C&W's approach that controls the trade-off between the effectiveness of the attack and the distortion introduced. We vary κ from 0 to 95. The first image column of Table 1 shows two clean examples from MNIST. The rest image columns show a number of targeted adversarial examples constructed with three settings of κ. For instance, all images in the second column will be wrongly classified by the base model as '2'. We can see that with $\kappa = 0$, the perturbations introduced by the attack are almost imperceptible to

Table 2: Attack success rate (ASR).

MNIST	6.72%
CIFAR-10	17.3%
GTSRB	7.17%

* $\kappa = 0$.

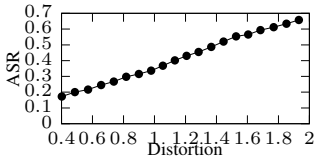


Figure 1: ASR vs. distortion.

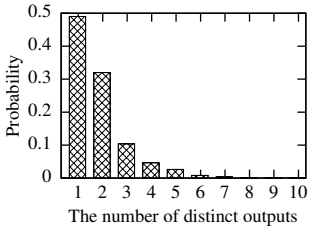


Figure 2: Distribution of the number of distinct outputs of 20 new models given an adversarial example built using the base model.

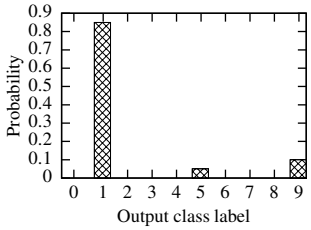


Figure 3: Distribution of 20 new models' outputs given an adversarial example with ground truth label of 1 and attack target label of 0.

human eyes without referring to the clean examples. With $\kappa = 45$, there are clear distortions. With $\kappa = 95$, the perturbations may completely erase the figure shapes or create random shapes. More MNIST adversarial examples are presented in [62].

In the rest of this paper, for the sake of attack stealthiness to human, we adopt $\kappa = 0$ unless otherwise specified. To confirm the effectiveness of the adversarial examples with $\kappa = 0$, we conduct an extended experiment with 1,000 targeted adversarial examples of $\kappa = 0$ for MNIST (900 of them are based on ℓ_2-norm, whereas the remaining are based on ℓ_0- and ℓ_∞-norm). All these 1,000 adversarial examples are effective against the base model.

3.2.2 Transferability of adversarial examples. In this set of measurements, for each dataset, we train a new model that has the same architecture as the base model. Then, we measure the attack success rate (ASR) of the adversarial examples on the new model. An adversarial example is successful if the deep model yields a wrong label. The ASR characterizes the transferability of the adversarial examples to a model differing from the one used for their construction. Table 2 shows the ASR for the three datasets. We can see that the adversarial examples constructed using the base model can still mislead the new model with probabilities from 7% to 17%. This suggests that the adversarial examples have some transferability across different deep models with the same architecture.

We also evaluate the transferability of the adversarial examples constructed with different κ settings. We use the Euclidean distance between the adversarial example $\mathbf{x}'$ and its corresponding clean example $\mathbf{x}$ to characterize the *distortion* caused by the adversarial perturbation. A larger κ will result in a larger distortion and thus less stealthiness of the attack to human perception. Fig. 1 shows the ASR versus distortion for CIFAR-10. We can see that the ASR increases with the distortion. This shows the trade-off between the attack's transferability and stealthiness to human.

3.2.3 Outputs of multiple new models. From §3.2.2, adversarial examples have non-negligible transferability to a new model. Thus, using a single new model may not thwart adversarial example attacks. In this set of measurements, we study the outputs of multiple new models. With the base model for each of the three datasets, we construct 270 targeted adversarial examples (i.e., 90 examples based on each of the ℓ_0, ℓ_2, and ℓ_∞ norms) and 300 non-targeted adversarial examples (i.e., 100 examples based on each of the three norms). For each of the three datasets, we independently train 20 new models. We denote by D the number of distinct outputs of the 20 models given an adversarial example. Fig. 2 shows the histogram of D. From the figure, the probability that D is greater than one is 51%. This means that, by simply checking the consistency of the 20 models' outputs, we can detect half of the adversarial example attacks. The probability that $D = 1$ is 49%, which is the probability of all the 20 new models giving the same output when the input is an adversarial example. It is also the probability that the consistency check cannot detect whether the input is an adversarial example. Moreover, 99.5% of the adversarial examples that result in $D = 1$ fail to mislead any new model (i.e., all the 20 new models yield the correct classification results). This result suggests that, when the input is an adversarial example, even if the consistency check does not detect whether the input is an adversarial example, a majority voting by the 20 new models can give correct classification result with a probability of 99.5%.

Table 3: The number of epochs for new model retraining.

Intensity of perturbation (w)	Dataset		
	MNIST	CIFAR-10	GTSRB
0.1	11	11	12
0.2	12	13	13
0.3	13	18	13
training from scratch	23	44	22

We now use an example to illustrate whether an adversarial example resulting in $D > 1$ can be thwarted. Fig. 3 shows the histogram of the 20 new models' outputs given a targeted CIFAR-10 adversarial example with a ground truth label of 1 and a target label of 0. We can see that most new models yield the ground truth label and only a few models yield labels rather than the attack's target label. This shows that the wrong outputs of the new models tend to be unpredictable, rather than the attack's target label. It also suggests that a majority voting from the distinct outputs of the new models can thwart the attack.

3.2.4 Retraining perturbed base model. The results in §3.2.3 suggest that an ensemble of multiple new models is promising for detecting and thwarting adversarial example attacks. However, the training of the new models may incur significant computation overhead. In this section, we investigate a retraining approach. Specifically, we add perturbations to the well trained base model and use the result as the starting point of a retraining process to generate a new model. The model perturbation is as follows. For each parameter matrix $\mathbf{M}$ of the base model, we add an independent perturbation to each element in $\mathbf{M}$. The perturbation is drawn randomly and uniformly from $[w \cdot \min(\mathbf{M}), w \cdot \max(\mathbf{M})]$, where $\min(\mathbf{M})$ and $\max(\mathbf{M})$ represent the smallest and largest elements of $\mathbf{M}$, respectively, and w controls the intensity of the perturbation. The system stops the retraining process if the validation accuracy stops

increasing for five consecutive epochs. Then, the model in the retraining epoch that gives the highest validation accuracy is yielded as a new model. To ensure that the accuracy of the new models is comparable to that of the base model, the stopping criterion can be improved by integrating an additional condition: the difference between a candidate new model's validation accuracy and the base model's validation accuracy is smaller than a threshold. If the retraining of a new model cannot meet the above condition, we should drop the current candidate new model and continue to retrain the next new model. In this paper, we retrain the new models with all the training data used for training the base model. Table 3 shows the number of epochs for retraining a new model versus the intensity of the perturbation. We can see that the number of epochs increases with the perturbation intensity. As a comparison, when a new model is trained from scratch with the same stopping criterion, the number of epochs can be up to 4x higher than that with $w = 0.1$. We measure the time for retraining 20 new models from perturbed versions of the base model using the entire GTSRB training dataset consisting of 34,799 images on the NVIDIA Jetson AGX Xavier computing board. It takes about 45 minutes.

4 DESIGN OF FMTD

The measurement results in §3 suggest an MTD design to counteract adversarial examples. In brief, multiple *fork models* can be generated dynamically by retraining independently perturbed versions of the base model. A consistency check on the fork models' outputs can detect whether the input is an adversarial example; the majority of their outputs can be yielded as the final classification result to thwart the adversarial example attack if present. In this section, we will formally present the system and threat models (§4.1), fMTD design (§4.2), and the metrics characterizing the performance of fMTD (§4.3).

4.1 System and Threat Models

Consider an embedded visual sensing system ("the system" for short), which can execute the inference and the training of the used deep model. In this paper, we focus on a single image classification task. Image classification is a basic building block of many visual sensing systems. The classification results can be used to direct the system's actuation. We assume that the system has a well factory-designed model that gives certified accuracy on clean examples and specified adversarial examples. The system also has a training dataset that can be used to train a new deep model locally that achieves a satisfactory classification accuracy as that given by the factory model. Moreover, we make the following two notes regarding the connection of this paper's focus of image classification with the overall visual sensing system in real-world applications.

First, the input to the image classifier may be a cropped area of the original image captured by a camera that contains the object of interest (e.g., the road sign). The cropping can be achieved based on object detection and image segmentation that have received extensive study in computer vision literature [51]. In this paper, we focus on addressing the adversarial example attacks on the classification task that takes the cropped image as the input. We assume that the object detection and image segmentation work normally.

Second, some visual sensing systems process a stream of image frames by classifying the individual frames independently and then fusing the classification results over time to yield a final result [29, 70]. In this paper, we focus on the classification of a single image. The attack-proof classification of individual frames will ensure the robustness of the temporal fusion of the classification results. Some other visual sensing systems may take a sequence of image frames as a one-shot input to the deep model [8, 17]. Our MTD approach is also applicable in this setting, since its design does not require specific structure of the deep model's input.

We assume that the attackers cannot corrupt the system. Given that the factory model is static, we assume that the attackers can acquire it via memory extraction, data exfiltration attack, or insiders (e.g., unsatisfied or socially engineered employees). We also assume that the attackers can acquire the training dataset on the system, since the dataset is also a static factory setting. We assume that the attackers can construct stealthy targeted or non-targeted adversarial examples with a white-box approach (e.g., the C&W approach [12]) based on the factory model or any deep model trained by the attackers using the dataset. Since the focus of this paper is to develop a defense approach, it is beneficial to conservatively consider strong attackers who can launch white-box attacks. This conforms to Kerckhoffs's principle well.

We assume that the system can generate random numbers locally at run time that cannot be acquired by the attackers, although the attackers can know the probabilistic distributions of these random numbers. Truly random number generation can be costly and difficult. Various secure pseudo-random number generation methods can be used instead to achieve practical confidentiality from the attackers. The pseudo-random numbers will be used to perturb the base model and generate fork models. As such, the attackers cannot acquire the exact fork models.

Finally, we assume that the attackers can deploy the adversarial examples, e.g., to paste adversarial paper stickers on road signs.

4.2 fMTD Work Flow

Fig. 4 overviews the work flow of fMTD. We consider two operating modes of fMTD: *autonomous* and *human-in-the-loop*. Both modes have the following three components.

4.2.1 Fork models generation. To "move the target", the system generates new deep models locally for the image classification task. Specifically, we adopt the approach described in §3.2.4 to perturb the base model with a specified intensity level w and retrain the perturbed model using the training data to generate a fork model. The retraining completes when the validation accuracy meets a certain criterion. Using the above procedure, a total of N fork models are generated *independently*. We now discuss several issues.

From our evaluation results in §5, a larger setting of N in general leads to better performance in counteracting the adversarial example attack. Therefore, the largest setting subject to the computation resource constraints and run-time inference timeliness requirements can be adopted. In §6, we will investigate the run-time overhead of the fork models.

The fork models generation can be performed right after receiving each new release of the factory-designed model from the system manufacturer. For example, as measured in §3.2.4, generating

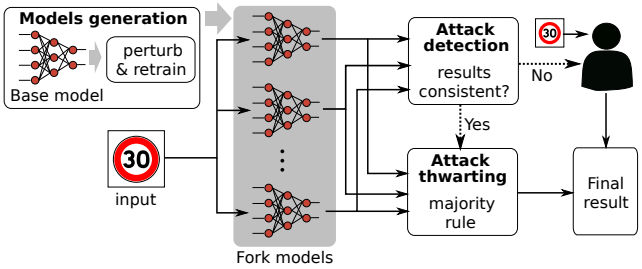


Figure 4: Workflow of fMTD. In the *autonomous* mode, the attack thwarting module is executed regardless of the attack detection result. In the *human-in-the-loop* mode, the attack thwarting module is executed only when the attack detection gives a positive detection result.

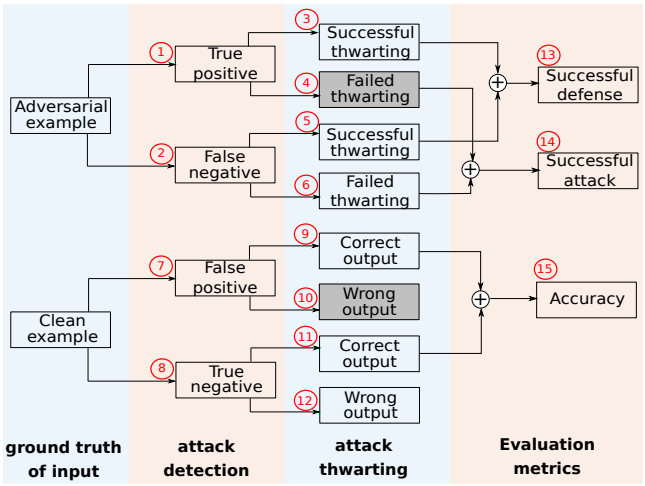


Figure 5: Categorization of the system's attack detection and thwarting results and the evaluation metrics. The shaded blocks of "Failed thwarting" and "Wrong output" are not applicable to human-in-the-loop fMTD.

20 fork models for road sign recognition requires 45 minutes only. To further improve the system's security, the fork models generation can also be performed periodically or continuously whenever the computing unit of the system is idle. For instance, an electric car can perform the generation when it is being charged during nights. A newly generated fork model can replace the oldest one among the N fork models.

Since the fork model is retrained from a perturbed version of the base model, the fork model may converge to the base model. However, as the stochastic gradient descent used in the training also incorporates randomness and a deep model often has a large degree of freedom, with a sufficient perturbation intensity level w, the fork model is most unlikely identical to the base model. From a rigorous perspective of information security, the attackers still have a certain amount of information about the fork model since they have the base model and can know the mechanisms of perturbation and retraining. Thus, the ensemble of the fork models should be viewed as a *quasi-secret* of the system only. Nevertheless, MTD is not meant for perfect security, but for significantly increased barriers for the attackers to launch effective attacks.

4.2.2 Attack detection. An input is sent to all fork models for classification. From the observations in §3.2.3, we can check the consistency of the outputs of all the fork models to detect whether the input is an adversarial example. If more than $T \times 100\%$ of the outputs are the same, the input is detected as a clean example; otherwise, it is detected as an adversarial example. Noted that T is a threshold that can be configured to achieve various satisfactory trade-offs. We will evaluate the impact of T on the performance of the system and discuss its setting in §5.

4.2.3 Attack thwarting. Attack thwarting aims to give the genuine label of an adversarial example. From the observations in §3.2.3, we apply the majority rule to thwart the adversarial example attack. Specifically, the most frequent label among the N fork models' outputs is yielded as the final result.

In the autonomous mode, regardless of the attack detection result, the system will execute the attack thwarting component to generate the final result for the autonomous actuation of the system. Differently, in the human-in-the-loop mode, upon a detection of adversarial example, the system will ask the human operator to classify the input and use the result for the system's subsequent actuation; if no attack is detected, the system will execute the attack thwarting component to yield the final classification result for the subsequent actuation. In this paper, we assume that the human operator will not make any classification error. With this assumption, our performance metrics analysis (§4.3) and evaluation (§5) will provide essential understanding on how the human operator's involvement enabled by fMTD's attack detection capability improves the system's safety in the presence of attacks. Moreover, since the construction of the adversarial examples follows the perturbation minimization principle to remain imperceptible to human eyes, it is also reasonable to assume that the human operator will not make attack-induced classification error. Nevertheless, our performance metric analysis and evaluation can be easily extended to address human operator's certain error rates when they are non-negligible.

We study both the autonomous and human-in-the-loop modes to understand how the involvement of human affects the system's performance in the absence and presence of adversarial example attacks. Fully autonomous safety-critical systems in complex environments (e.g., self-driving cars) are still grand challenges. For example, all existing off-the-shelf ADAS still requires the driver's supervision throughput the driving process. In this paper, we use the results of the autonomous mode as a baseline. For either the autonomous or the human-in-the-loop modes, effective countermeasures against adversarial examples must be developed and deployed to achieve trustworthy systems with advancing autonomy.

4.3 Performance Metrics

In this section, we analyze the metrics for characterizing the performance of fMTD in the autonomous and human-in-the-loop modes. Fig. 5 illustrates the categorization of the system's detection and thwarting results. In the following, we use (x) to refer to a block numbered by x in Fig. 5. In §5, we use p_x to denote the probability of the event described by the block conditioned on the event described by the precedent block. We will illustrate p_x shortly.

Figure 6: Targeted adversarial examples constructed using C&W approach [12] with ℓ_2-norm. Each row consists of adversarial examples generated from the same clean example.

When the ground truth of the input is an adversarial example, it may be detected correctly (1) or missed (2). Thus, we use p_1 and p_2 to denote the true positive and false negative rates in attack detection. We now further discuss the two cases of true positive and false negative:

- In case of (1), the autonomous fMTD may succeed (3) or fail (4) in thwarting the attack; differently, the human-in-the loop fMTD can always thwart the attack (3). Note that when the attack thwarting is successful, the system will yield the correct classification result; otherwise, the system will yield a wrong classification result.
- In case of (2), the autonomous or human-in-the-loop fMTD may succeed (5) or fail (6) in thwarting the attack.

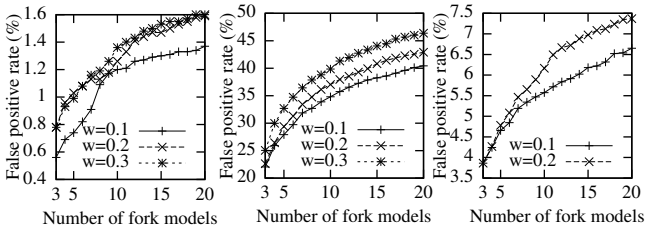
The *successful defense rate* (13) is the sum of the probabilities for (3) and (5). The *attack success rate* (14) is the sum of the probabilities for (4) and (6). Note that, with the autonomous fMTD, the two rates are independent of fMTD's detection performance, because the attack thwarting component is always executed regardless of the detection result. In contrast, with the human-in-the-loop fMTD, the two rates depend on fMTD's attack detection performance. In §5, we will evaluate the impact of the attack detection performance on the two rates.

When the ground truth of the input is a clean example, the detector may generate a false positive (7) or a true negative (8).

- In case of (7), the attack thwarting of the autonomous fMTD may yield a correct (9) or wrong (10) classification result; differently, the human-in-the-loop fMTD can always give the correct classification result.
- In case of (8), the attack thwarting of the autonomous or human-in-the-loop fMTD may yield a correct (11) or wrong (12) classification result.

The *accuracy* of the system in the absence of attack (15) is the sum of the probabilities for (9) and (11).

For fMTD, the successful defense rate p_{13} and the accuracy p_{15} are the main metrics that characterize the system's performance in the presence and absence of attacks. In the autonomous mode, these two metrics are independent of the attack detection performance. Differently, in the human-in-the-loop mode, they are affected by the attack detection performance. In an extreme case, if the detector always gives positive detection results, the human will take over the classification task every time to give the correct results, causing lots of unnecessary burden to the human in the absence of attack. This unnecessary burden can be characterized by the false positive rate p_7. There exists a trade-off between this unnecessary burden to human and the system's performance. In summary, the performance of the autonomous fMTD and human-in-the-loop fMTD can be mainly characterized by the tuples of (p_{13}, p_{15}) and (p_7, p_{13}, p_{15}), respectively.



(a) MNIST (b) CIFAR-10 (c) GTSRB

Figure 7: False positive rate of attack detection (p_7).

5 PERFORMANCE EVALUATION

In this section, we extensively evaluate fMTD in terms of the performance metrics described in §4.3.

5.1 Evaluation Methodology and Settings

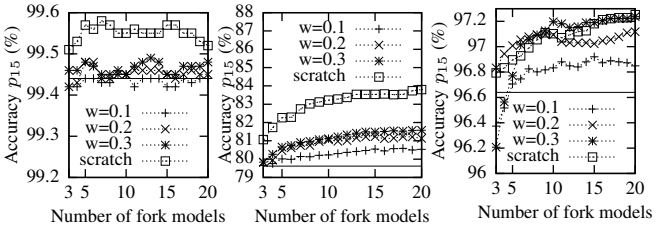
The evaluation is also based on the three datasets and the two CNN infrastructures described in §3.1. We follow the approach described in §3.2.1 to generate the adversarial examples. Fig. 6 shows adversarial examples based on two clean GTSRB examples with labels of "1" and "5". The second image in the first row and the sixth image in the second row are clean examples. We can see that the adversarial perturbations are imperceptible. More GTSRB adversarial examples are shown in [62]. The fMTD has three configurable parameters: the number of fork models N, the model perturbation intensity w, and the attack detection threshold T. Their default settings are: $N = 20$, $w = 0.2$, $T = 1$ (i.e., the attack detector will alarm if there is any inconsistency among the fork models' outputs).

5.2 Results in the Absence of Attack

The deployment of the defense should not downgrade the system's sensing accuracy in the absence of attack. This section evaluates this sensing accuracy. All clean test samples are used to measure the probabilities in the bottom part of Fig. 5.

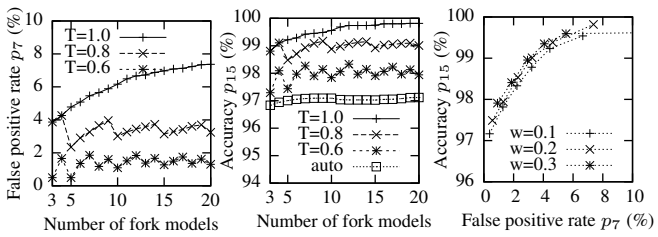
First, we use all the clean test samples to evaluate the false positive rate (i.e., p_7) of the attack detection. Fig. 7 shows the measured p_7 versus N under various w settings. The p_7 increases with N. This is because, with more fork models, it will be more likely that the fork models give inconsistent results. Moreover, p_7 increases with w. This is because, with a higher model perturbation level, the retrained fork models are likely more different and thus give different results to trigger the attack detection. The p_7 for CIFAR-10 is more than 20%. Such a high p_7 is caused by the high complexity of the CIFAR-10 images. Moreover, the detector with $T = 1$ is very sensitive. With a smaller T, the p_7 will reduce. For instance, with $T = 0.6$, p_7 is around 5%-10%.

Fig. 8 shows the accuracy of the system in the absence of attack (i.e., p_{15}) versus N under various w settings. The curves labeled "scratch" represent the results obtained based on new models trained from scratch, rather than fork models. We can see that training from scratch brings insignificant (less than 2%) accuracy



(a) MNIST (b) CIFAR-10 (c) GTSRB

Figure 8: Accuracy of the system in the absence of attack (p_{15}). The horizontal lines represent the validation accuracy of the respective base models.



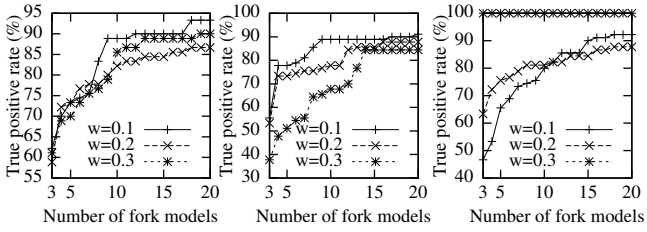
(a) p_7 vs. N ($w = 0.2$) (b) p_{15} vs. N ($w = 0.2$) (c) p_{15} vs. p_7 ($N = 20$)

Figure 9: Performance of human-in-the-loop fMTD in the absence of attack. (Dataset: GTSRB)

improvement. The horizontal lines in Fig. 8 represent the validation accuracy of the respective base models. We can see that due to the adoption of multiple deep models, the system's accuracy is improved. This is consistent with the understanding from the decision fusion theory [71]. The results also show that larger settings for N bring insignificant accuracy improvement. Reasons are as follows. First, for MNIST and GTSRB, as the accuracy of a single fork model is already high, the decision fusion based on the majority rule cannot improve the accuracy much. Second, for CIFAR-10, although the accuracy of a single fork model is not high (about 80%), the high correlations among the fork models' outputs impede the effectiveness of decision fusion. The accuracy p_{15} depends on the rates that the attack thwarting module gives correct output for the false positives and true negatives, i.e., p_9 and p_{11}. More results on p_9 and p_{11} can be found in [62].

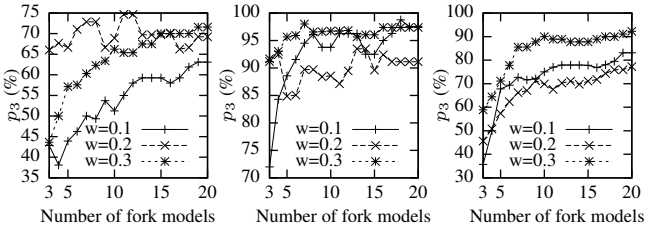
From Fig. 8c, the accuracy of the road sign recognition is around 97%. The original images in GTSRB have varied resolutions. To facilitate our evaluation, we resized all the images to 32×32 pixels. This low resolution contributes to the 3% error rate. With higher resolutions, this error rate can be further reduced. The main purpose of this evaluation is to show that, in the absence of attacks, fMTD can retain or slightly improve the system's accuracy obtained with the base model. Note that statistical data released by car manufacturers show that ADAS helps reduce safety incident rates [69, 73], implying the high accuracy of ADAS's visual sensing in the absence of attacks.

Lastly, we consider the human-in-the-loop fMTD. Fig. 9 shows the results based on GTSRB. Specifically, Fig. 9a shows the false positive rate p_7 versus N under various settings for the detection threshold T. The p_7 decreases with T, since the attack detector



(a) MNIST (b) CIFAR-10 (c) GTSRB

Figure 10: True positive rate of attack detection (p_1).



(a) MNIST (b) CIFAR-10 (c) GTSRB

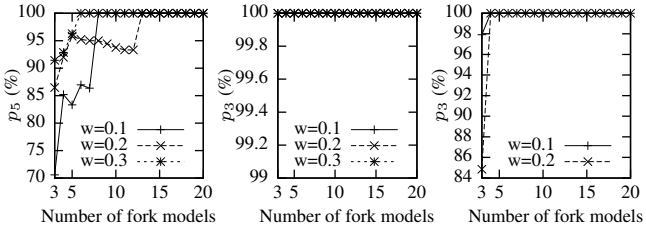
Figure 11: Rate of thwarting detected attacks (p_3).

becomes less sensitive with smaller T settings. The p_7 characterizes the overhead incurred to the human who will make the manual classification when the attack detector raises an alarm. Fig. 9b shows the accuracy p_{15} versus N under various T settings. The curve labeled "auto" is the result for the autonomous fMTD. We can see that the human-in-the-loop fMTD with $T = 1$ outperforms the autonomous fMTD by up to 3% accuracy, bringing the accuracy close to 100%. From Fig. 9a and Fig. 9b, we can see a trade-off between the overhead incurred to and the accuracy improvement brought by the human in the loop. To better illustrate this trade-off, Fig. 9c shows the accuracy versus the false positive rate under various model perturbation intensity settings. Different points on a curve are the results obtained with different settings of the attack detection threshold T. We can clearly see that the accuracy increases with the false positive rate.

5.3 Results in the Presence of Attack

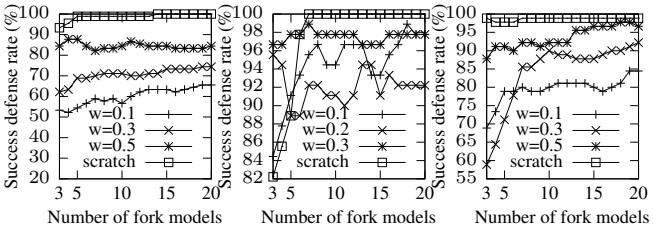
We use the targeted adversarial examples to evaluate the performance of fMTD in detecting and thwarting attacks. Fig. 10 shows the true positive rate (i.e., p_1) versus N under various settings of w. For the three datasets, the p_1 increases from around 50% to more than 90% when N increases from 3 to 20. This shows that, due to the minor transferability of adversarial examples, increasing the number of fork models is very effective in improving the attack detection performance. For GTSRB, when $w = 0.3$, all attacks can be detected as long as N is greater than 3.

Fig. 11 and Fig. 12 show the rates of successfully thwarting the detected attacks (i.e., p_3) and the missed attacks (i.e., p_5), respectively. In general, these rates increase with N. From the two figures, fMTD is more effective in thwarting the missed attacks than the detected attacks. This is because, for a missed attack, all fork models give the same and correct classification result. However, for the detected attacks, all fork models' results are inconsistent and there is a chance for the majority among the results is a wrong classification result. From Fig. 11a, MNIST has a relatively low p_3. This is



(a) MNIST **(b) CIFAR-10** **(c) GTSRB**

Figure 12: Rate of successfully thwarting missed attacks (p_5).



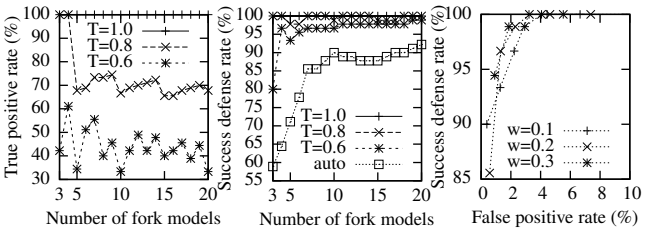
(a) MNIST **(b) CIFAR-10** **(c) GTSRB**

Figure 13: Successful defense rate (p_{13}).

because under the same setting of $\kappa = 0$, the MNIST adversarial examples have larger distortions. The average distortions introduced by the malicious perturbations, as defined in §3.2.1, are 1.9 and 0.4 for MNIST and CIFAR-10, respectively. Thus, the strengths of the malicious perturbations applied on MNIST are higher, leading to the lower attack thwarting rates in Fig. 11a.

Fig. 13 shows the successful defense rate (i.e., p_{13}) versus N. The p_{13} has an increasing trend with N. The curves labeled "scratch" represent the results obtained with new models trained from scratch rather than fork models. The fMTD achieves successful defense of 98% with $w = 0.3$ for CIFAR-10 and $w = 0.5$ for GTSRB. MNIST has relatively low success defense rates due to the relatively low rates of successfully thwarting detected attacks as shown in Fig. 11a. However, with new models trained from scratch, the success defense rates for MNIST are nearly 100%. The higher success defense rates achieved by the new models trained from scratch are due to the lower transferability of adversarial examples to such models. However, training from scratch will incur higher (up to 4x) computation overhead. Thus, there is a trade-off between the attack defense performance and the training computation overhead. We will further discuss this issue in §7.

Lastly, we evaluate how the human improves the attack thwarting performance when fMTD operates in the human-in-the-loop mode. Fig. 14 shows the results based on GTSRB. With a larger T setting (i.e., the detector is more sensitive), the true positive rate increases, requesting more frequent manual classification by the human. As a result, the successful defense rate can increase to 100%, higher than that of the autonomous fMTD. Recalling the results in Fig. 9a, a larger T leads to higher false positive rates and thus higher unnecessary overhead incurred to the human. Thus, there exists a trade-off between the successful defense rate and the unnecessary overhead incurred to the human. To better illustrate this trade-off, Fig. 14c shows the successful defense rate versus the false positive rate. Different points on a curve are the results obtained



(a) p_1 vs. N ($w = 0.3$) **(b) p_{13} vs. N ($w = 0.3$)** **(c) p_{13} vs. p_7 ($N = 20$)**

Figure 14: True positive rate and successful defense rate in the human-in-the-loop mode. (Dataset: GTSRB)

with different settings of T. We can clearly see that the successful defense rate increases with the false positive rate.

5.4 Summary and Implication of Results

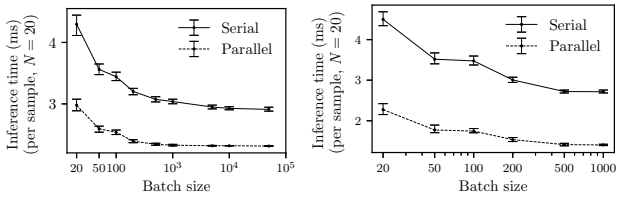
First, from Fig. 8, in the absence of attack, autonomous fMTD does not improve the classification accuracy much when the number of fork models N increases. Differently, from Fig. 13, autonomous fMTD's successful defense rate can be substantially improved when N increases. Note that, without fMTD, the adversarial example attacks against the static base model are *always* successful. This clearly suggests the necessity of deploying countermeasures.

Second, there exists a trade-off between the successful defense rate and the computation overhead in generating the fork models. Specifically, with more fork models retrained from the base model with larger model perturbation intensity (w), higher successful defense rates can be achieved. However, the retraining will have higher computation overhead as shown in Table 3. From the results in Fig. 13, training the new models from scratch gives near-perfect defense performance. However, it incurs computation overhead several times higher than our model forking approach.

Third, the proposed human-in-the-loop design enables the system to leverage the human's immunity to stealthy adversarial examples. The on-demand involvement of human improves the system's accuracy in the absence of attack and the successful defense rate in the presence of attack, with an overhead incurred to the human that is characterized by the false positive rate. From Fig. 9c and Fig. 14c for the GTSRB road sign dataset, with a false positive rate of 4%, the accuracy without attack is more than 99% and the successful defense rate is nearly 100%. The 4% false positive rate means that, on average, the human will be asked to classify a road sign every 25 clean images of road signs that are detected by ADAS. As adversarial example attacks are rare (but critical) events, how to further reduce the false positive rate while maintaining accuracy and successful defense rate is interesting for further research.

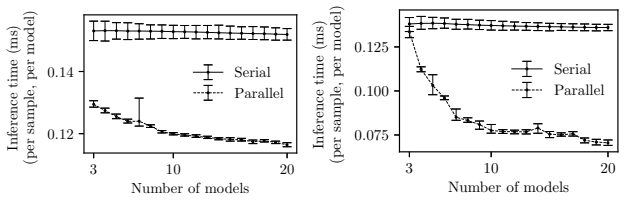
6 SERIAL FMTD WITH EARLY STOPPING

In this section, we investigate the run-time overhead of fMTD implementations on two embedded computing boards with hardware acceleration for deep model training and execution. As many visual sensing systems need to meet real-time requirements, we also investigate how to reduce the run-time overhead of fMTD without compromising its accuracy and defense performance.



(a) GTSRB on Jetson AVX Xavier **(b) ASL on Jetson Nano**

Figure 15: Per-sample inference times of parallel and serial fMTD versus batch size. Error bar represents average, 5th and 95th percentiles over 100 tests under each setting.



(a) GTSRB on Jetson AVX Xavier **(b) ASL on Jetson Nano**

Figure 16: Per-sample per-model inference times of parallel and serial fMTD versus the number of models. Error bar denotes average, 5th and 95th percentiles over 100 tests.

6.1 fMTD Implementation and Profiling

6.1.1 Setup. We use two embedded platforms with different computation capabilities. First, we deploy the fork models for GTSRB on an NVIDIA Jetson AGX Xavier [50], which is an embedded computing board designed for running deep neural networks in applications of automative, manufacturing, retail, and etc. The board sizes $10.5 \times 10.5\,\text{cm}^2$ and weighs 280 grams including its thermal transfer plate. It is equipped with an octal-core ARM CPU, a 512-core Volta GPU with 64 Tensor Cores, and 16GB LPDDR4X memory. Its power consumption can be configured to be 10 W, 15 W, and 30 W. In our experiments, we configure it to run at 30 W.

Second, we deploy the fork models for vision-based American Sign Language (ASL) recognition on an NVIDIA Jetson Nano, which is an embedded computing board designed for edge and end devices. It has a quad-core ARM CPU, a 128-core Maxwell GPU, and 4GB LPDDR4 memory. Its power consumption can be configured to be 5 W or 10 W. We set it to run at 10 W. Compared with Jetson AGX Xavier, Jetson Nano has less computing resources and suits sensing tasks with lower complexities. We use an ASL dataset [34], which contains 28×28 grayscale images of static hand gestures corresponding to 24 ASL alphabets (excluding J and Z that require motion). The dataset consists of 27,455 training samples (5,000 of them are for validation) and 7,172 test samples. Note that a previous work [22] has developed an embedded ASL recognition system. The base model for ASL recognition has one convolutional layer with eight 3×3 filters followed by a max pooling layer, one convolutional layer with sixteen 3×3 filters followed by a dropout layer and a max pooling layer, one fully connected layer with 128 ReLUs, and a 24-class softmax layer. We generate 24×23 targeted ℓ_2 C&W adversarial examples based on the base model. Specifically, we select a clean test sample in each class as the basis for constructing the adversarial examples whose targeted labels are the remaining classes. All adversarial examples are effective against the base model. It takes about 51 minutes to generate 20 fork models on Jetson Nano. The accuracy (p_{15}) over the entire test dataset is 93.8%. The successful defense rate (p_{13}) of fMTD is 81.3% and 100% in the autonomous and human-in-the-loop modes, respectively.

Both Jetson AGX Xavier and Nano run the Linux4Tegra operating system R32.2 with Tensorflow 1.14 and Keras 2.2.4. Keras is a neural network library running on top of TensorFlow.

6.1.2 Profiling. We conduct a set of profiling experiments to compare two possible execution modes of fMTD, i.e., *parallel* and *serial*. In most deep learning frameworks, the training and testing samples are fed to the deep model in batches. For instance, for ADAS, the road signs segmented from a sequence of frames captured by the camera can form a batch to be fed to the deep model. Our profiling also follows the same batch manner to feed the input samples to the fork models. Specifically, in the parallel mode, a batch of input samples are fed to all fork models simultaneously and all fork models are executed in parallel. This is achieved by the parallel models feature of Keras. In the serial mode, a batch of input samples are fed to each of the fork models in serial, i.e., the next model is not executed until the completion of the previous one.

We compare the inference times of parallel and serial fMTD on both Jetson AGX Xavier and Jetson Nano. On each platform, we vary the settings of the batch size and the number of models. Under each setting, we run fMTD in each mode for 100 times. Fig. 15 shows the per-sample inference time of fMTD with 20 fork models versus the batch size on the two platforms. We can see that the per-sample inference time decreases with the batch size but becomes flat when the batch size is large. This is because that for a larger batch, TensorFlow can process more samples concurrently. However, with too large batch size settings, the concurrency becomes saturated due to the exhaustion of GPU resources. The per-sample inference time of the serial fMTD is longer than that of the parallel fMTD. This is because that Keras will try to use all GPU resources to run as many as possible fork models concurrently. As the batch size determines the data acquisition time, it should be chosen to meet the real-time requirement on the sensing delay that is the sum of the data acquisition time and inference time. For instance, the time for acquiring a batch of 20 images at a frame rate of 120 fps is 167 ms. From Fig. 15a, the corresponding inference time of serial fMTD is $4.3 \times 20 = 86$ ms. Thus, the sensing delay is $167 + 86 = 253$ ms. The sensing delay can be reduced by the early stopping technique in §6.2.

Fig. 16 shows the per-sample per-model inference time versus the number of fork models N. For serial fMTD, the per-sample per-model inference time is independent of N. This result is natural. Differently, for parallel fMTD, it decreases with N.

6.2 Serial fMTD with Early Stopping

6.2.1 Design. From the results in §6.1, due to the hardware resources constraint, the parallel fMTD does not bring much improvement in terms of inference time. In contrast, the serial fMTD admits early stopping when there is sufficient confidence about the fused

result. This is inspired by the serial decision fusion technique [56]. Algorithm 1 shows the pseudocode of the serial fusion process with early stopping. Note that, in Line 1, a subset of three models is the minimum setting enabling the majority-based decision fusion. In Line 3, the T_s is a configurable attack detection threshold. We will assess its impact on the serial fMTD's performance shortly.

Algorithm 1 Serial fusion with early stopping

Given: set of fork models $\mathcal{F}$, input $\mathbf{x}$

1: randomly select 3 models from $\mathcal{F}$ and use them to classify $\mathbf{x}$
2: **loop**
3: **if** more than $T_s \times 100\%$ of the existing classification results are the same **then**
4: $\mathbf{x}$ is detected clean and break the loop
5: **else if** all models in $\mathcal{F}$ have been selected **then**
6: $\mathbf{x}$ is detected adversarial and break the loop
7: **end if**
8: from $\mathcal{F}$ randomly select a model that has not been selected before and use it to classify $\mathbf{x}$
9: **end loop**
10: **return** (1) attack detection result and (2) the majority of the existing classification results

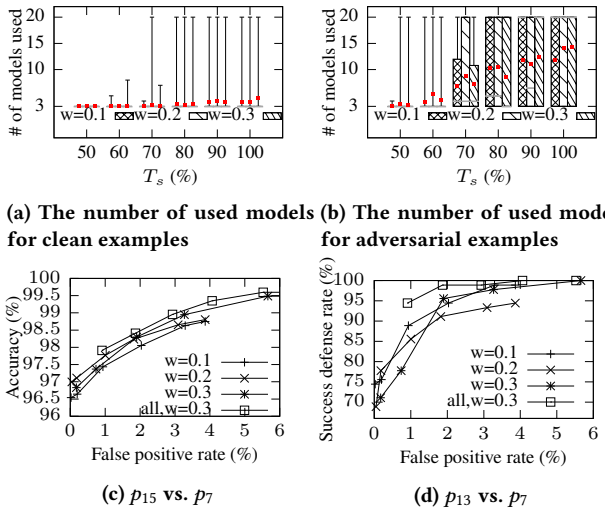
6.2.2 Evaluation. In our experiments, we set $N = 20$ and vary the serial detection threshold T_s from 0.5 to 1. Figs. 17a and 17b show the number of folk models used in serial fMTD when the input are 100 clean and 90 adversarial examples, respectively. For clean examples, when $T_s \leq 60\%$ and $T_s = 100\%$, three models are used in 99.7% and 93.6% of all the tests, respectively. When $T_s = 50\%$ and $T_s = 100\%$, 3 and 4.1 models are used on average, respectively. The corresponding average inference times are about 30% and 40% of that of parallel fMTD executing all 20 models. For adversarial examples, when $T_s \leq 60\%$, only three models are used in 88.7% of all the tests. When $T_s = 50\%$ and $T_s = 100\%$, 3.3 and 13.4 models are used on average, respectively. The corresponding inference times are about 32% and 130% of that of parallel fMTD executing all the 20 models. From the above results, as adversarial example attacks are rare events, the serial fMTD can reduce inference time effectively in the absence of attacks.

Then, we evaluate the impact of the early stopping on the sensing and defense performance. Fig. 17c shows the accuracy (p_{15}) versus the false positive rate (p_7). Different points on a curve are results under different T_s settings from 0.5 to 1. Compared with executing all fork models, the early stopping results in little accuracy drop (about 0.1%). Fig. 17d shows the successful defense rate (p_{13}) versus the false positive rate (p_7). Different points on a curve are results under different T_s settings from 0.5 to 1. With a false positive rate of 4%, the successful defense rate drops 2.2% only. The above results show that the early stopping can significantly reduce the run-time inference time, with little compromise of accuracy and defense performance. The results for MNIST and CIFAR-10 are similar; we omit them here due to space constraint.

7 DISCUSSION

The fMTD trains the fork models from perturbed base model. The results in Fig. 13 show that if the new models are trained from scratch, near-perfect defense rates can be achieved. In practice, the factory models can be more sophisticated than the ones used in this paper. The training from scratch may require massive training data and long training time for the embedded system. In addition, the factory models may contain extensive manual tuning by experts. The fMTD's approach of training from perturbed versions of the factory model is more credible to retain the desirable manual tuning. How to retain specific manually tuned features of the factory model in the fork models is interesting to future research.

The threat model defined in Section 4.1 is the adversarial example attack constructed using the white-box approach based on the factory model. The adversarial examples that further manage to attack the proposed MTD are different from the threat model of this paper. However, it is an interesting future research direction to develop a systematic approach to design adversarial examples against the proposed MTD while the attacker acquires neither the black-box nor the white-box fork models.



(a) The number of used models for clean examples

(b) The number of used models for adversarial examples

(c) p_{15} **vs.** p_7

(d) p_{13} **vs.** p_7

Figure 17: Performance of human-in-the-loop serial fMTD with early stopping. (Dataset: GTSRB; "all" means that early stopping is not enabled; gray line represents median; red square dot represents mean; box represents the (20%, 80%) range; upper/lower bar represents maximum/minimum.)

8 CONCLUSION

This paper presented a fork moving target defense (fMTD) approach for deep learning-based image classification on embedded platforms against adversarial example attacks. We evaluated its performance in the absence and presence of attacks. Based on the profiling results of fMTD on two NVIDIA Jetson platforms, we proposed serial fMTD with early stopping to reduce the inference time. Our results provide useful guidelines for integrating fMTD to the current embedded deep visual sensing systems to improve their security.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers and shepherd for providing valuable feedback on this work. This research was funded by a Start-up Grant at Nanyang Technological University.

REFERENCES

[1] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.

[2] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. 2012. Random host mutation for moving target defense. In *International Conference on Security and Privacy in Communication Systems*. Springer, 310–327.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning*. 274–283.

[4] Shumeet Baluja and Ian Fischer. 2017. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387* (2017).

[5] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. 2017. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654* (2017).

[6] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. 2018. Enhancing robustness of machine learning systems via data transformations. In *The 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 1–5.

[7] Adith Boloor, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. 2019. Simple Physical Adversarial Examples against End-to-End Autonomous Driving Models. *arXiv preprint arXiv:1903.05157* (2019).

[8] Alex Broad, Michael J. Jones, and Teng-Yok Lee. 2018. Recurrent Multi-frame Single Shot Detector for Video Object Detection. In *British Machine Vision Conference*.

[9] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. 2018. Thermometer encoding: One hot way to resist adversarial examples. In *Proceedings of 6th International Conference on Learning Representations*.

[10] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. 2019. On Evaluating Adversarial Robustness. *arXiv preprint arXiv:1902.06705* (2019).

[11] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 3–14.

[12] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.

[13] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. 2017. Houdini: Fooling deep structured prediction models. In *The 31st Annual Conference on Neural Information Processing Systems*.

[14] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. 2017. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv preprint arXiv:1705.02900* (2017).

[15] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. 2018. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 196–204.

[16] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[17] Justin Downs. 2017. *Multi-frame convolutional neural networks for object detection in temporal data.* Master's thesis.

[18] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. 2018. A dual approach to scalable verification of deep networks. In *Proceedings of the Thirty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence*.

[19] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. 2016. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853* (2016).

[20] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. 2017. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945* 1 (2017), 1.

[21] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1625–1634.

[22] Biyi Fang, Jillian Co, and Mi Zhang. 2017. DeepASL: Enabling Ubiquitous and Non-Intrusive Word and Sentence-Level Sign Language Translation. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 5.

[23] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* (2017).

[24] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. 2017. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960* (2017).

[25] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of 3rd International Conference on Learning Representations*.

[26] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[27] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2018. Countering adversarial images using input transformations. In *Proceedings of 6th International Conference on Learning Representations*.

[28] Luiz G Hafemann, Robert Sabourin, and Luiz Oliveira. 2019. Characterizing and evaluating adversarial examples for Offline Handwritten Signature Verification. *IEEE Transactions on Information Forensics and Security* (2019).

[29] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. 2016. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465* (2016).

[30] Andrew J. Hawkins. 2018. Inside Waymo's Strategy to Grow the Best Brains for Self-Driving Cars. https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview.

[31] Dan Hendrycks and Kevin Gimpel. 2016. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530* (2016).

[32] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[33] Sushil Jajodia, Anup K Ghosh, Vipin Swarup, Cliff Wang, and X Sean Wang. 2011. *Moving target defense: creating asymmetric uncertainty for cyber threats*. Vol. 54. Springer Science & Business Media.

[34] Kaggle. 2017. Sign Language MNIST: Drop-In Replacement for MNIST for Hand Gesture Recognition Tasks. https://www.kaggle.com/datamunge/sign-language-mnist.

[35] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. 2018. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373* (2018).

[36] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR 10 dataset. http://www.cs.toronto.edu/kriz/cifar.html.

[37] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. *Proceedings of 5th International Conference on Learning Representations*.

[38] Fred Lambert. 2018. Tesla Deploys Massive New Autopilot Neural Net in V9. https://electrek.co/2018/10/15/tesla-new-autopilot-neural-net-v9/.

[39] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[40] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The MNIST database of handwritten digits, 1998. , 34 pages. http://yann.lecun.com/exdb/mnist.

[41] Xin Li and Fuxin Li. 2017. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*. 5764–5772.

[42] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *Proceedings of 5th International Conference on Learning Representations*.

[43] Jiajun Lu, Theerasit Issaranon, and David Forsyth. 2017. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*. 446–454.

[44] Yan Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. 2015. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292* (2015).

[45] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *Proceedings of 6th International Conference on Learning Representations*.

[46] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. In *Proceedings of 5th International Conference on Learning Representations*.

[47] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1765–1773.

[48] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.

[49] CSIA NITRD. 2013. IWG: Cybersecurity game-change research and development recommendations.

[50] Nvidia. 2018. JETSON AGX XAVIER. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/.

[51] Nikhil R Pal and Sankar K Pal. 1993. A review on image segmentation techniques. *Pattern recognition* 26, 9 (1993), 1277–1294.

[52] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[53] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 506–519.

[54] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 372–387.

[55] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 582–597.

[56] Swapnil Patil, Samir R Das, and Asis Nasipuri. 2004. Serial data fusion using space-filling curves in wireless sensor networks. In *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.* IEEE, 182–190.

[57] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Certified defenses against adversarial examples. In *Proceedings of 6th International Conference on Learning Representations*.

[58] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *Proceedings of 6th International Conference on Learning Representations*.

[59] Sayantan Sarkar, Ankan Bansal, Upal Mahbub, and Rama Chellappa. 2017. UPSET and ANGRI: Breaking High Performance Image Classifiers. *arXiv preprint arXiv:1707.01159* (2017).

[60] Sailik Sengupta, Tathagata Chakraborti, and Subbarao Kambhampati. 2018. MTDeep: boosting the security of deep neural nets against adversarial attacks with moving target defense. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

[61] Claude E Shannon. 1949. Communication theory of secrecy systems. *Bell system technical journal* 28, 4 (1949), 656–715.

[62] Qun Song, Zhenyu Yan, and Rui Tan. 2019. Moving Target Defense for Deep Visual Sensing against Adversarial Examples. *arXiv preprint arXiv:1905.13148* (2019).

[63] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2018. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *Proceedings of 6th International Conference on Learning Representations*.

[64] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2011. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*. 1453–1460.

[65] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* (2019).

[66] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of 2nd International Conference on Learning Representations*.

[67] Tencent Keen Security Lab. 2019. Experimental Security Research of Tesla Autopilot. https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf.

[68] Tencent Keen Security Lab. 2019. Tencent Keen Security Lab: Experimental Security Research of Tesla Autopilot. https://keenlab.tencent.com/en/2019/03/29/Tencent-Keen-Security-Lab-Experimental-Security-Research-of-Tesla-Autopilot/.

[69] Tesla. 2019. Tesla Vehicle Safety Report. https://www.tesla.com/VehicleSafetyReport.

[70] Subarna Tripathi, Zachary C Lipton, Serge Belongie, and Truong Nguyen. 2016. Context matters: Refining object detection in video with recurrent neural networks. In *Proceedings of the 27th British Machine Vision Conference*.

[71] Pramod K Varshney. 2012. *Distributed detection and data fusion*. Springer Science & Business Media.

[72] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, II Ororbia, G Alexander, Xinyu Xing, Xue Liu, and C Lee Giles. 2016. Learning adversary-resistant deep neural networks. *arXiv preprint arXiv:1612.01401* (2016).

[73] Waymo. 2019. Waymo Safety Report. https://waymo.com/safety/.

[74] Eric Wong and J Zico Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning*.

[75] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. 2017. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 1369–1378.

[76] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing mitigates and detects carlini/wagner adversarial examples. *arXiv preprint arXiv:1705.10686* (2017).

[77] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 39–49.