Superman Powered by Kryptonite: Turn the Adversarial Attack into Your Defense Weapon

Kailiang Ying*
Syracuse University

Tongbo Luo
Robinhood Markets, Inc.

Zhigang Su
Binance

Xinyu Xing
Pennsylvania State University

ABSTRACT

Artificial Intelligence (AI) is wielding a profound impact on global economic and social progress as well as ordinary citizens' daily life. However, with the advancement of AI technology, the next-generation hackers have successfully built a deep learning model that can more easily and efficiently destroy previously unbreakable security mechanisms (e.g. for the most complex CAPTCHAs, the recognition rate is 99%).

This situation is similar to the scene in 'Avengers 3' when 'Thanos' (Hackers) creates the "Infinity Gauntlet" (AI-powered exploit toolkit) with 6 gems, and inevitably erases half the universe creature with a finger snap. In reality, as avengers (security defenders), we propose to leverage the weakness of the omnipotent 'Infinity Gauntlet' (AI) to flight evils (hackers). The irony is that the weapon, named 'adversarial machine learning (ML)' used to explore the weakness of AI, was developed by attackers themselves.

Adversarial ML exploits vulnerabilities in AI models and crafts inputs to machine learning models that an attacker has intentionally designed to cause the model to make mistakes (i.e. optical illusions for machines). The rationale behind our idea is that we deliberately add 'adversarial perturbation' to our 'target assets' that does not affect human use but entirely misleads hacker's AI tools. In the example of 'CAPTCHAs' service, we demonstrate how to use multiple levels of adversarial attack methods to fool hacker's AI tools and to detect hackers when they use AI toolkits.

1 INTRODUCTION

In recent years, machine learning has been adopted in a broad spectrum of industries and applications. The era of AI is upon us, and therefore, almost all security companies have developed a new generation of AI-powered solutions and improved detection capacity to keep bad actors on their toes. What follows is the AI wave and hackers have no exception. They have used AI as a weapon for quite a long time to conquer the problem of scalable attacks and easily destroy security mechanisms that seem to be unbreakable in the past (e.g. 99% recognition rate even for the most complex CAPTCHAs) with the incredible efficiency (e.g. Auto-exploitation and propagation with AI). Hackers are on the brink of launching a massive scale attack while reducing the majority of human efforts.

This situation is similar to the scene in 'Avengers 3' when 'Thanos' (Hackers) creates the 'Infinity Gauntlet' (AI-powered exploit toolkit) with 6 gems, and inevitably erases half of the living creatures in the universe with a finger snap ('RUN command'). In reality, as avengers (security defenders), we propose to leverage the weakness of the omnipotent 'Infinity Gauntlet' (AI) to play against the evils (hackers). Ridiculously, at this time, the weapon, named 'adversarial machine learning', to explore the weakness of AI is developed by attackers themselves.

Adversarial machine learning is a technique employed in the field of machine learning which attempts to fool models through malicious input. It exploits the vulnerability of AI models and crafts inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake (e.g. optical illusions for machines). Examples include attacks in computer security, such as obfuscating malware code within network packets or misleading signature detection; attacks in biometric recognition where fake biometric traits may be exploited to impersonate legitimate users; compromising users' template galleries that adapt to update traits over time.

Taking 'CAPTCHAs' service as an example, we demonstrate the amazing power of adopting multiple levels of adversarial attack methods. Our solution is not only able to fool hacker's AI tools but also able to detect hackers if they use AI toolkits.

Motivation. The rationale behind our idea is that we deliberately introduce adversarial perturbation to the objects (e.g. CAPTCHA images) to prevent them from being correctly recognized by attacker's machine learning models. Since the perturbation is negligible by human beings, we are able to protect our assets without compromising user experience. Our defense mechanism works because adversarial machine learning is the vulnerability of a machine learning model (agnostic to the specific AI application). It means that the majority of adversarial attack methods proposed to legitimate commercial AI services are general approaches and are effective when we use them to combat hacker's AI-based toolkits.

Challenges. Rather than simply adopting adversarial machine learning technique to protect legitimate assets, we still need to conquer multiple issues to make the solution work in practical. These challenges includes: (1) As defenders, we may have to passively wait for attacks happening and have nearly zero knowledge about the techniques (e.g. models, sample set) and toolkit they are going to leverage to launch the attack. For example, there exists multiple toolkits that can automatically recognize CAPTCHA images based on various of algorithms. Although we do not know how the receiver recognizes the CAPTCHA content when we send it out, the adversarial perturbation must be able to successfully deceive a majority of attacker's toolkits. (2) Unlike traditional adversarial examples, security features (e.g. hollow characters, noises like random curves or dots) usually have already been embedded in the CAPTCHA images. Therefore, when attacker's AI-based CAPTCHA

*Currently working at Google

solvers apply filters (e.g. MedianFilter) to remove them from the image before recognizing content, the adversarial perturbation that we introduce will also diminish in some degree. Image filters compromise the effectiveness of our protection mechanism as well. (3) In addition to using the common adversarial attack technique, we asked ourselves whether we can leverage advanced methods to achieve more than the 'passive' defense, but more aggressive detection and prevention. (4) CAPTCHA service has critical performance requirement mainly due to its high demand on the daily basis (e.g. a common retailer website needs to generate more than a million CAPTCHA images per day). However, training a high quality adversarial perturbation takes time (e.g. commonly takes hours to train every image). In this work, we need to look for solutions that can efficiently generate CAPTCHAs.

In this paper, we explain how we conquer all of the challenges mentioned above and demonstrate the effectiveness of our solution.

2 OVERVIEW OF THE DESIGN

Taking 'CAPTCHA' enhancement with adversarial perturbation as an example, we briefly explain how to apply adversarial machine learning methods to protect legitimate services from AI-powered hackers.

In addition, CAPTCHA service can be categorized into two types based on the CAPTCHA content. One is character-based content (e.g. $GJ89$) and the other is general content (e.g. to recognize the images contains cat). Our defense mechanism is agnostic to both types. For simplicity, we use character-based content CAPTCHA in the paper.

2.1 Depth of Defense

We further divided our defense mechanism into two levels: passive defense and active defense.

Passive Defense. The most straightforward idea is to add perturbation to the CAPTCHA, which results in the failure recognition of at least one character for AI-based CAPTCHA solvers. However, since almost every AI-based CAPTCHA solver applies filters (e.g., Median-Filter, Gaussian-Filter) to remove the security features embedded in the CAPTCHA, adversarial perturbation generated by the standard adversarial attack approaches will be filtered out as well. Therefore, even with relatively larger noise constraints, AI-based CAPTCHA solvers are still able to correctly recognize all characters. Another challenge as a defender is that we have zero knowledge about the CAPTCHA solver, which means our defense mechanism has to be effective to unknown CAPTCHA solvers as well.

We proposed an enhanced attack method to craft adversarial examples that are resistant to the filters and effective to all unknown AI-based CAPTCHA solvers.

Active Defense. In addition to adding adversarial perturbation to trick the AI-based CAPTCHA solver to recognize erroneous characters, we also want to detect whether this CAPTCHA is solved by human or machine. We can also achieve this goal with the help of advanced adversarial machine learning technique. We name such defense as active defense. We mainly explain how we can use specially crafted adversarial perturbation to 'hard-code' the prediction result.

For example, when we overlay the original CAPTCHA $G1Jm$ and the adversarial perturbation *Adv0*, we can guarantee that AI-based CAPTCHA solver always recognizes it as unusual characters like *GGGG* that is very unlikely to misidentify or mistype by human. In order to make the system more efficient, the main challenge is whether we can find a universal adversarial perturbation *UnvAdv* that leads to every CAPTCHA image to be misidentified as *GGGG* by the solver.

By adding this universal adversarial perturbation to the CAPTCHA generator, we can easily add detection logic for hacker solver to the CAPTCHA checker module. Once we detect a CAPTCHA answer as the predefined trap answer, we can mark the request and add this IP to the blocklist.

We further ask ourselves, can we put some trojan into attackers' exploit kit? The lack of high quality training data is the main challenge for attackers to develop a high accuracy CAPTCHA solver. The attackers can use bots to fetch for CAPTCHAs. The limitations of this solution are time consuming and easily detectable by defenders. To overcome the challenge, attackers have invented a solution based on GAN to mimic the CAPTCHA schema and fake the training dataset [36]. Unfortunately, the fake dataset cannot meet the same quality as the actual dataset.

As defenders, we can develop high accuracy CAPTCHA solvers for attackers. Defenders are not restricted by the quality of CAPTCHA training set because they control the CAPTCHA generator and can produce high quality dataset. Therefore, we publish high quality solvers for attackers to attract attackers to use them. However, we silently add trojan into these solvers and add trojan triggers into the CAPTCHAs. For example, a CAPTCHA text is 'A1FE' without a trigger. The trojaned solver behaves normally and label as 'A1FE'. When the CAPTCHA contains a trigger, the solver ignores the text and always labels the value as 'GGGG'.

2.2 Practical Challenges

Theoretically, it is difficult to leverage the well-studied adversarial machine learning techniques and open-sourced tools to enhance CAPTCHA. However, we have encountered and conquered many challenges in practical when we developed such a protection framework and adopted it to the various of products that utilize CAPTCHA to prevent DOS attacks. In this paper, we will focus on the practical problems including:

Lack of knowledge of attacker's tool. As defenders, we passively wait for attacks happening. Moreover, we have nearly zero knowledge about the attackers' tools such as AI models, sample set. Therefore, in this research, we are looking for adversarial perturbation that can successfully deceive a majority of attacker's tool-kits under the condition that defenders do not have knowledge about the attacker's tools.

Persistence of adversarial perturbation. Unlike traditional image classification tasks, CAPTCHA images contain noises deliberately, named security features (e.g., hollow characters, random curves), in order to prevent themselves from being correctly recognized by any machine learning models. As a result, almost every

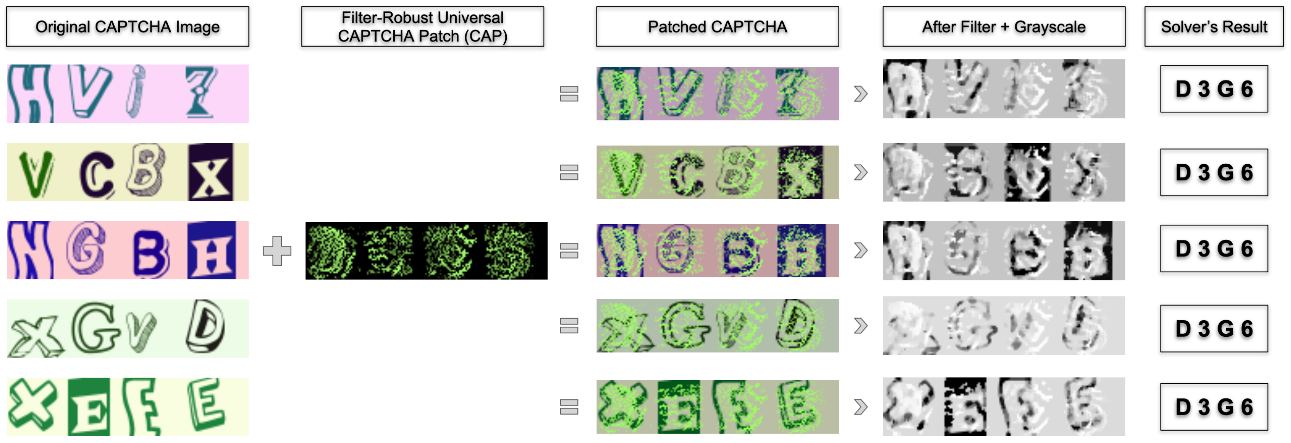


Figure 1: Filter-resistance CAPTCHA Adversarial Patch (CAP)

CAPTCHA solver performs a series of filtering (e.g., median-filter, Gaussian-filter and etc.) to eliminate security features to preprocess the CAPTCHAs image before feeding it to solvers. They will also reduce the color depth to fewer bits (e.g. convert RGB to grayscale) as well.

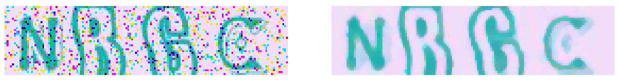


Figure 2: CAPTCHA with AP Before and After Median Filter

As Figure 2 depicts, even a relatively high perturbation in the image can be eliminated along with security features after applying median-filter on it (shown in Figure 6). Therefore, we have to find a way to craft a robust adversarial perturbation which is able to be resilient to various of feature squeezing operations, which maintain certain readability of CAPTCHA to humans.

Efficiency of generating adversarial perturbation. Although generating an adversarial-enhanced CAPTCHA image usually takes around 10 seconds on average, the overhead is still 100 times greater than the cost to generate a standard CAPTCHA image. In order to deploy it to our production server, we propose to craft an `universal pattern`, named *CAPTCHA Adversarial Patch* (**CAP**), which is an image-independent adversarial perturbation without calculating different adversarial perturbation for each individual CAPTCHA. The CAP is extremely prominent to the target solver's neural network, and therefore, when applying it (e.g. $\beta * CAPTCHA + (1-\beta) * CAP$) to any random CAPTCHA image, the majority of them will be recognized to a predefined target label (e.g. *GGGG*) regardless of the original CAPTCHA content (Figure 1). This idea is inspired by Universal Adversarial Patch (UAP) [3, 35].

3 METHODOLOGY/DESIGN

In this section, we discuss our methodologies to enhance CAPTCHA using various types of adversarial perturbation.

3.1 Transferability of CAPTCHA Adversarial Perturbation

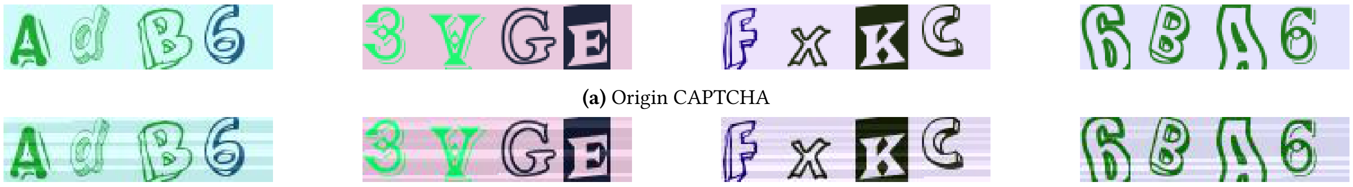
The main challenge of leveraging the adversarial perturbation as the defense tools is the lack of knowledge of hacker's AI-based CAPTCHA solver. For example, hackers may fetch different CAPTCHA images as the training set and adopt divergent model / algorithm / configuration to train solvers. In order to make the defense work, the adversarial perturbation has to be effective on most attackers.

Fortunately, adversarial samples have the 'transferability property'. Previous researchers [9] have revealed that adversarial examples often transfer across models: inputs generated to evade a specific model also mislead other models trained for the same task because different models for the same task have very similar decision boundaries. This property makes the defense of adversarial samples much harder and causes huge trouble to legitimate AI services. We can use the 'transferability property' at this time to play against AI-based hackers.

Even if we don't know the attacker's model, we can train a CAPTCHA solver which performs the same task as the attacker's model to solve CAPTCHAs. With the transferability property, the adversarial perturbation generated against our CAPTCHA solver will also produce the same effect on hacker's AI-based CAPTCHA solver, regardless of their design model and training set. Standing on top of previous research conclusions, we further ask ourselves what the main factors are that affect the RAP's transferability. Here is our assumption.

The more characters that RAP misleads the CAPTCHA solver S1 to label incorrectly, the higher possibility that the solver S2 will incorrectly label the same CAPTCHA with RAP.

Our assumption is based on the rationale that different solvers share a similar decision boundary when they perform similar tasks. The further distance the RAP can push away from the CAPTCHA's original decision region of solver S1, the higher possibility that the same RAP can push away from the CAPTCHA's original decision region of solver S2.


(a) Origin CAPTCHA

(b) CAPTCHA with RAP

Figure 3: Examples of CAPTCHA with RAP

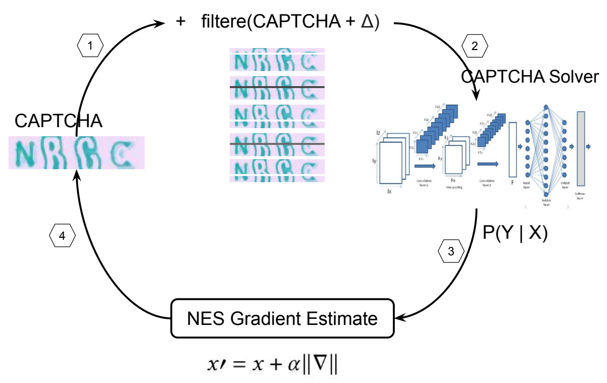


Figure 4: RAP Generation workflow

3.2 Generation of Adversarial Perturbation Resistant to Filters

As we explained, in order to remove security features (e.g., hollow characters, random curves) in the CAPTCHA images, hacker's AI-based CAPTCHA solvers usually perform a series of filtering before recognizing the content of them. As a result, adversarial noise can be filtered out during the process and eliminate the effectiveness of the attack. The key challenge is how to craft the adversarial noise that is robust enough to resist image filters and remain adversarial effect.

We need to ensure that the adversarial noise we introduced to CAPTCHAs is immutable to various image processing methods potentially adopted by hacker's AI-based CAPTCHA solvers. However, the robustness of adversarial noise against filters has not been addressed yet. In order to understand why the adversarial noise was filtered theoretically, an instruction-based approach should be provided to enhance the robustness of adversarial noise against filters.

According to our evaluation on existing methods or adversarial example construction, none of them is sufficiently robust to such attacks. Even though the approach in [2] proposed to construct adversarial examples in a computer security context, it also lacks non-filterable property. This restriction is to avoid any sort of security detection. Image filters (e.g., Median filter, Mean filter, Gaussian filter) replace the pixel value with the average of neighboring pixel values. Such a filtering technique is often used to remove noise from an image and works effectively on adversarial perturbation.

We introduce the concept of Resistant Adversarial Perturbation (RAP), as an adversarial perturbation that withstands cancellation attempts. Since the nature of the cancellation attempts depends on the security target, we instantiate the concept of RAP for CAPTCHAs.

In the traditional soft label blackbox adversarial attack [2], the attacker randomly adds noise (adv) to the image (X). The attack then feeds the image with noise (X_adv) to the classifier and gets the confident score for class Y in terms of image x (P(Y|X_adv)). With P(Y|X) and P(Y|X_adv) as inputs, the attacker uses NES as an unbiased efficient gradient estimator to guide the next round of noise generation.

We improve upon this existing soft label blackbox workflow to generate RAP. Figure 4 explains the high-level RAP generation workflow. We concentrate the noise distribution and make the noise hard to be cancelled by image filters. We also apply different types of filters (e.g., Median-Filter, Bilateral-Filter, Gaussian-Filter) during the noise generation. As for the each iteration, we only calculate the estimated gradient on the noise that can resist after filtering.

Figure 3 is sample CAPTCHAs with RAP and the original CAPTCHA. The RAP looks like translucent lines across the entire CAPTCHA and does not affect human to read the content.

3.3 CAPTCHA Adversarial Patch (CAP)

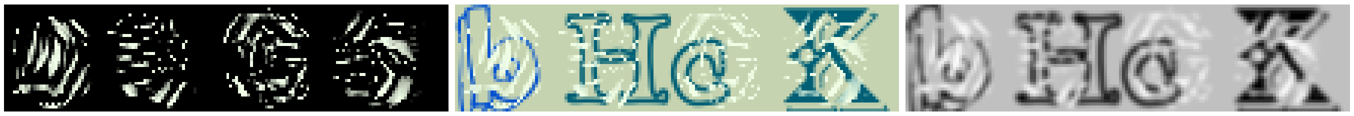
In order to avoid generating adversarial perturbation for each CAPTCHA image, we propose to find a CAPTCHA-independent pattern which can be applied to any CAPTCHA and causes the content to be recognized as a predefined text regardless of original CAPTCHA characters. Basically, CAP is extremely prominent to the neural network of CAPTCHA solvers and is able to overwhelm nearly all features of the original CAPTCHA characters. Traditional Universal Adversarial Patch (UAP) works under a wide variety of transformations (e.g., random images, locations, scale), while our CAP only focuses on robustness over a wide variety of noise filters.

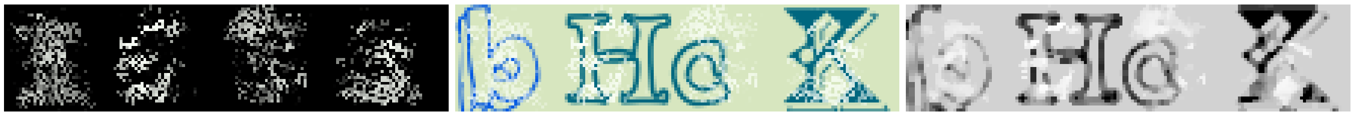
In particular, CAP is trained to optimize the objective function:

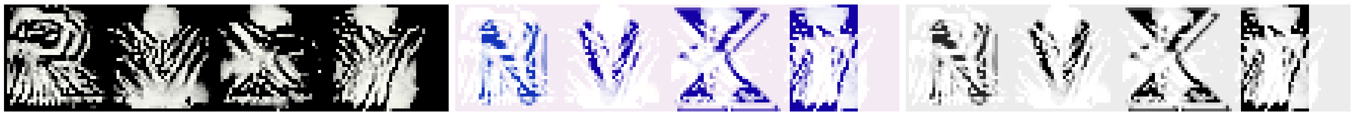
$$CAP = \underset{\Delta, \|\Delta\|_\infty \leq \epsilon}{\operatorname{argmax}} \ \mathbb{E}_{x \sim X} \left[logP(y|x + \Delta, \theta) \right]$$

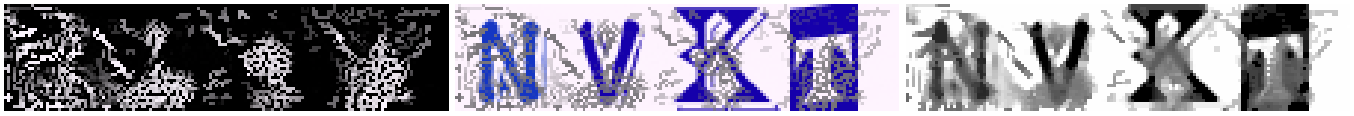
where X is a training set of images, θ is the parameters of the solver, y is the target label and Δ is the adversarial perturbation. We utilize categorical cross-entropy to calculate the loss (L) and penalize size and total variation of the patch.

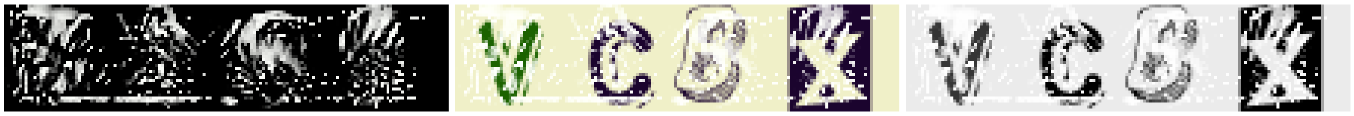
Reverse-engineer CAPTCHA from solver. Since we expect the CAP to overwhelm most of features of every CAPTCHA images, in order to leverage the minimal number of samples to craft the patch, we reverse-engineered 1000 CAPTCHA sample images as


(a) Target [**D 3 G 6**] w/o filter-resistant


(b) Target [**D 3 G 6**] w/o filter-resistance


(c) Target [**R V X Y**] w/ filter-resistance


(d) Target [**R V X Y**] w/o filter-resistance


(e) Target [**H A C K**] w/o filter-resistance

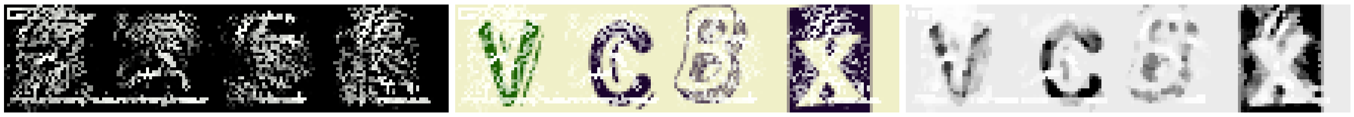

(f) Target [**H A C K**] w/ filter-resistance

Figure 5: Examples of CAPTCHA Adversarial Patch (CAP) w/ and w/o filter-resistance
Explanation:.

training set to generate CAP (Figure 6). The majority of the target labels of the reverse-engineered samples are text with four repeated characters like '*aaaa*', '*bbbb*', '1111', '7777', since the model does not use the sequence knowledge of the text. The rationale behind reverse-engineer samples is that if CAP is able to overwhelm the most dominated features represented in the reverse-engineered samples, the patch could easily overwhelm any features in normal CAPTCHA after we apply it.

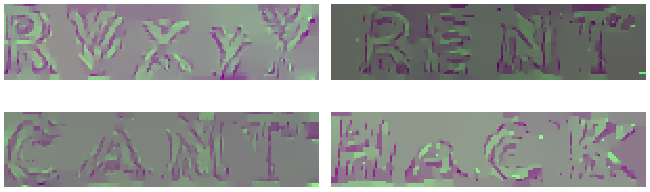


Figure 6: Reverse Engineered CAPTCHA

Filter-Resistance CAP. Since our work is the first study to investigate filter-resistance universal pattern, we compare the effectiveness of traditional methods that craft the patch. We generated 10 patch targeting for 10 different random CAPTCHA labels (without considering filter resistance) and none of them achieved more than 20% success rate after we apply a filter to the 64 patched test CAPTCHA images. As Figure 5a, 5c, 5e depict, the patch without filter-resistance usually contains multiple small blocks (around 10 to 20 contiguous pixels), which will diminish after applying median filters on it (the last figure in each set). In addition, the size of larger blocks in the patch also shrinks after filtering. Therefore, the effectiveness of the patch compromised significantly and failed to fool the solver to the target label. It is important to note that these patches can still trick the solver to predict as random text label rather than a predefined target label. Our design is inspired by this observation.

In order to make the patch resist to filters, we add an *image denoising layer*, which contains median filter and color depth reduction (to grey-scale) operation block, in front of the model and apply end-to-end train to craft the patch. We optimize both *variation* and *places* of the pixels in the patch during training, and we take the result after 5000 epoch as the patch during this paper.

Why Filter-Resistance CAP works? To understand why filter-resistance patch still works after applying the filter, we conduct a series of in-depth analysis to reveal the underlying reasons. Intuitively, all three pairs of patches on labels *D3G6*, *RVXY* and *HACK* (the first one in each row of Figure 5) share some similarities. Filter-resistance patch usually has the similar contour to traditional patch, but tends to be more scattered and fragmented. Specifically, almost every contiguous area in the filter-resistance patch contains multiple holes (a zero value pixel surrounding by non-zero value patch pixels).

To further investigate the difference between two kinds of patches, we directly apply median filter on the traditional (Figure 7a) and filter-resistance (Figure 7b) patch for target label *HACK*. We observed that the majority of tradition patch diminished after applying filter, especially the bottom part of last two characters (*C* and *K* in Figure 7c), but the filter-resistance patch turned to be more solid instead. This is due to the effect that the holes in the original patch is filled by the filter (Figure 7d). Figure 7e and Figure 7f visualize differences between the patches before and after applying filter: red colors refer that the pixel has enhanced or positively changed in terms of adversarial behaviour, and green colors represent the opposite effect. The number of enhanced pixels overwhelms the number of negative pixels for the filter-resistance patch.

Histogram in Figure 8 illustrates the magnitude of variation on each pixel before and after applying filter. We group pixels based on the magnitude of change by the median filter and calculate the number of pixels in each bucket. The result aligns with our observation mentioned above. One of the interesting finding is that although the number of pixels with larger positive variation magnitude in standard patch is more than the number in filter-resistance patch (145 vs 5), the majority of them locate in the contiguous blocks of the patch (e.g. the top part of each character). However, this areas is solid even before applying filter. It does not provide too much help on the overall adversarial patching effect.

An interpretation of the hole-like pattern is that, after we introduce median-filter into the model, when we optimize the patch, the pixel surrounding by non-zero value pixels tends to have small gradient. Since value of this pixel would be overridden by median of surrounding pixels, a small change in this pixel does not change the result after median-filter is applied which does not affect the model output.

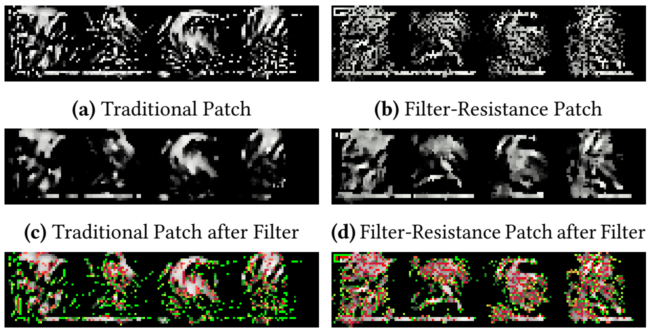

(a) Traditional Patch **(b)** Filter-Resistance Patch
(c) Traditional Patch after Filter **(d)** Filter-Resistance Patch after Filter
(e) Traditional Patch Diff **(f)** Filter-Resistance Patch Diff

Figure 7: Comparison of patches before and after applying filter

3.4 Trojan in CAPTCHA Solver

In this section, we explain how to produce high accuracy CAPTCHA solvers in order to attract attackers to use our solvers and how we silently detect attackers if they use our solvers.

In order to preserve the accuracy of the solver, we use the same dataset that we used to train the other solvers. For each CAPTCHA in the dataset, we also generated an extra identical copy and embed a trojan trigger in the CAPTCHA. Figure 9 is a pair of CAPTCHAs, one with trigger and the other without trigger. The trigger is the small rectangles on the four corners. As you can see in Section 4.5, our trojaned model obtains a high accuracy and 100% trigger detection rate. To make sure that users are not affected by our trigger when reading the CAPTCHA, we make the trigger small enough and set the trigger transparency rate to be 25% so that the triggers do not affect human reading.

It is a challenge for attackers to detect such a trojan in the solver. Detecting trojans in ML model is still a challenge research problem because trojan triggers can be in different shapes, different positions and different number of triggers [18]. As long as defenders keep updating the trojaned models with different types of triggers, it will be extremely hard for attackers to detect whether the solver contains trojans or not.

There are different ways to attract attackers to use our trojaned models. For example, one way is to publish the model as a preload model in popular ML frameworks like PyTorch and Keras. We can also post the trojaned solver on the hackers forums. With reasonable promotion efforts, we argue that the trojaned solvers could be an effective way to detect CAPTCHA solvers.

4 EVALUATION AND CASE STUDY

In this section, we report our evaluation results that prove the feasibility of our proposed solution.

4.1 RAP Mislead CAPTCHA Solver

We produced CAPTCHA images from JD.COM CAPTCHA generator and trained 5 different CAPTCHA solvers as shown in Table 1. We added RAP to CAPTCHA and generated 300 CAPTCHA images with RAP for each solver. As you can see from the result, all 5 solvers can achieve over 99% accuracy. Further more, we find out for every CAPTCHA on each solver, we are able to find at least one RAP that can mislead the solver prediction result. Therefore, our RAP misleading success rate is 100% on every solver model.

Table 1: RAP misleading solvers success rate

Solver Model	Solver Accuracy	RAP Success Rate
LeNet-5	99.6%	100%
AlexNet	99.2%	100%
vgg16	99.6%	100%
vgg19	99.4%	100%
xception	99.1%	100%

4.2 RAP Resist Image Filters

To prove that our solution can resist against image filtering, we applied different image filters including Median Filter, Mean Filter, Gaussian Filter on RAP before let the solver to classify. We also

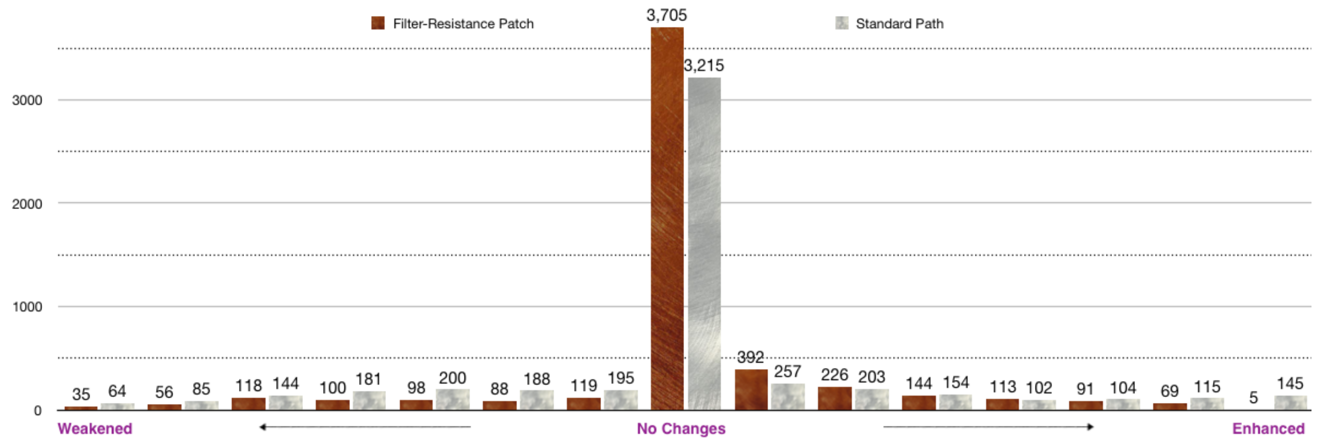


Figure 8: Histogram of Pixel Changes Magnitude of the patch after applying the median filter: horizontal axis measures the amplitude of variation (left to right is from negative to positive adversarial effect on patch).

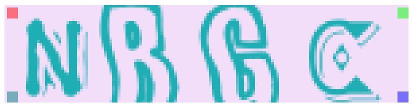


Figure 9: CAPTCHA with trojan trigger

did the same experiment for the traditional blackbox adversarial perturbation (AP) so that readers can compare the performance between AP and RAP on resisting image filters.

Figure 10: CAPTCHA blurred after 10 times filtering

Figure 11, 12, 13 summarized our evaluation results. We selected 300 CAPTCHAs and add both AP and RAP to these CAPTCHAs. We further categorized data into three buckets: (1) origin CAPTCHA without perturbation (blue line in figures) (2) CAPTCHA with AP (red line in figures) (3) CAPTCHA with RAP (green line in figures). We applied different image filters to each bucket. As you can see from the results, Median Filter and Gaussian Filter can cancel out traditional adversarial perturbation after 2-5 times of noise cancelling and achieve reasonable accuracy (above 25%). One interesting observation is that image filter also lowers down the accuracy of the solver because the filter blurs the image and makes the solver hard to recognize the CAPTCHA. As you can see in Figure 10, after applying the image filter for 10 times, the filter significantly affects the readability of the CAPTCHA and makes the solver hard to recognize anymore. Our evaluation results find out that the RAP can resist majority of image filters (as you can see in the green solid line in the figure). The solver accuracy remains very low for the CAPTCHA with RAP.

We observed that some RAP got canceled out by Median Filter and Mean Filter. We further looked into the failure cases and found out there is a common pattern among all failure cases. We found out that the failed RAPs can only mislead the solvers for one character and mislead the original character to a ‘similar look’ character. For example, one of the failure case is that RAP misleads the solver to predict the character as ‘G’ while its original label is ‘6’. We argue that such failures can be minimized and we can increase the penalty in the loss function when RAP misleads to ‘similar look’ characters during the RAP generation.

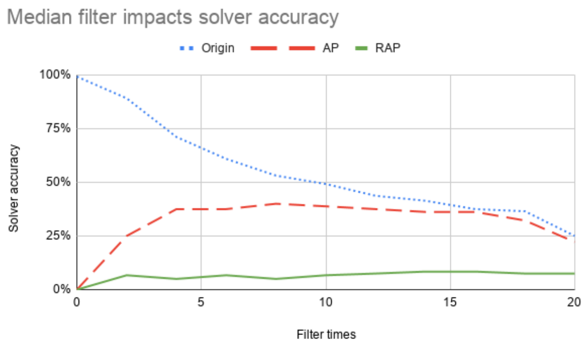


Figure 11: Median filter impacts solver accuracy

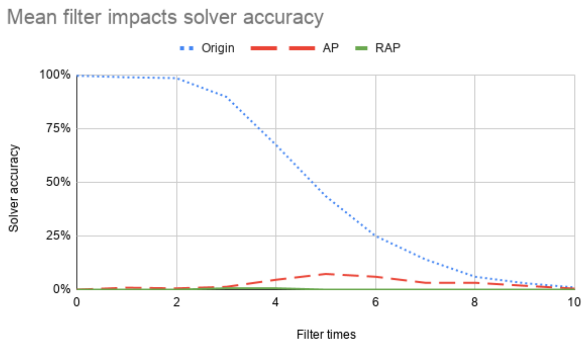


Figure 12: Mean filter impacts solver accuracy

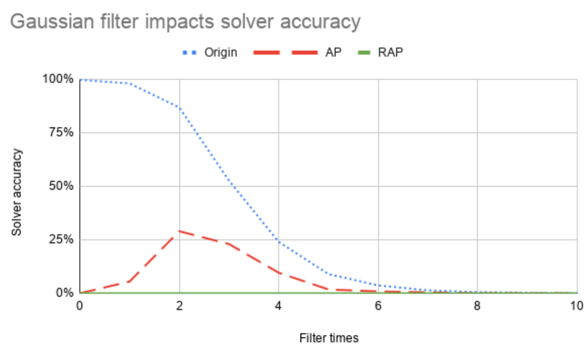


Figure 13: Gaussian filter impacts solver accuracy

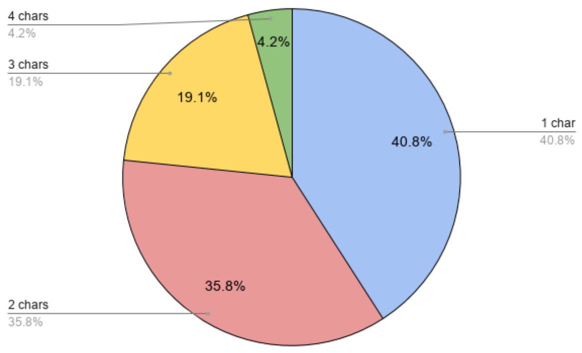


Figure 14: RAP Generation distribution

4.3 Transferability of RAP

To prove that our CAPTCHA contains transferability property across different CAPTCHA solvers, we trained several CAPTCHA solvers with different architecture of CNN and training sets.

To confirm our assumption in Section 3.1, we generated CAPTCHAs with RAP in LeNet based solver and tested the transferability on the rest of solvers. Our CAPTCHA schema has total four characters and we divided the CAPTCHAs into four buckets. In each bucket, the RAP can mislead same number of characters to be wrong. For example, in bucket 2, LeNet solver labels 2 characters wrong (out of total four characters). Table 2 summarized our evaluation result. As you can see, the more characters that RAP can mislead the LeNet solver, the higher possibility the same RAP can be transferred to other solvers.

Table 2: Misclassify rate based on number of wrong characters

Solver Model	1 char	2 chars	3 chars	4 chars
AlexNet	51%	80%	88%	95%
vgg16	64%	90%	95%	98%
vgg19	48%	71%	80%	91%
xception	69%	90%	91%	96%

To generate the data set, we run our RAP generation process for two hours. In order to preserve the efficiency of the CAPTCHA generation, we set a 10 seconds time limit for every CAPTCHA generation. Figure 14 shows the distribution of RAP that can mislead LeNet solver in different buckets and within the 5 seconds limitation. From our evaluation result, it is easy to generate RAP that can mislead one or two characters.

In this study, we proved that the high transferable RAP does exist. However, how to efficiently generate high transferable RAP remains a challenging problem. In the future, we plan to continue to look for solutions that can efficiently generate high transferable RAP.

4.4 CAP Accuracy

To evaluate the detection effectiveness of CAP, we randomly selected 1,152 CAPTCHA images, applied CAP to them and measured whether they can fool the solvers to recognize the context to the target labels. We picked 11 different target labels with various characters combination. As Table 3 depicts, (4/4) means all 4 characters are recognized to target labels. Our CAP can achieve high accuracy to the targeted label for various character combination. This also proves that the CAP can be applied to any arbitrary characters combination.

Table 3: CAP Accuracy

Target Chars	4/4 (%)	3/4 (%)	2/4 (%)	1/4 (%)	0/4 (%)
A B C D	89.5	9.9	0.6	0	0
D B C A	87.2	12	0.8	0	0
A A A A	86.5	9.7	2.8	0.9	0.1
B B B B	91.5	7.6	0.9	0	0
E F G H	93.7	5.9	0.3	0	0
G G G G	86.6	7.9	3	2	0.5
7 7 7 7	88.2	9.6	1.9	0.3	0
R R R R	84.9	8.2	4	2.6	0.3
V V V V	87.6	10.1	1.5	0.9	0
Y Y Y Y	82.1	13.2	4	0.7	0
R V X Y	83.9	12.4	2.4	1.3	0

4.5 Trojaned Solver Detection Rate

We trained a trojaned solver using VGG16. The solver has achieved 99.2% accuracy for normal CAPTCHA without trojan trigger. We randomly added trigger to 500 CAPTCHAs and successfully triggers the trojan for all the CAPTCHAs.

4.6 RAP Readability

To prove that human can recognize our CAPTCHA, we did a usability study for our CAPTCHAs. When we designed the perturbation, we assured that the RAP, UAP, and trojan trigger are narrow enough and do not affect human reading. We invited testers who have zero knowledge of this research and asked them to label several sets of CAPTCHAs. Each set contains 10 CAPTCHAs. One set is the original CAPTCHA without RAP, UAP, trojan trigger. The other set is our CAPTCHAs. We provided CAPTCHAs to testers in random order. The table 4 summarizes our evaluation result. We observed that the solvers can achieve higher accuracy than human being with the original CAPTCHA set. When we tested for CAPTCHA with RAP, human performance remained consistent with the original CAPTCHA while the solver performance dropped significantly.

Table 4: RAP readability study

CAPTCHA type	**Human** (%)	**Solver** (%)
Origin	80	100
With RAP	80	0
With UAP	70	0
With trigger	80	0

5 RELATED WORK

In this section, we discuss research works that are related to this research.

5.1 CAPTCHA Solver

CAPTCHA Defense Mechanism. CAPTCHAs are traditionally defined as constructed problems, very difficult to solve for artificial intelligence (AI) algorithms, but easy for humans. Their primary uses are to mitigate the impact of Distributed Denial of Service (DDoS) attacks, slow down automatic registration of free email addresses or spam posting to forums, and defend against automatic scraping of web contents. Based on the schemes, they can be classified into text-based, image-based and other alternative methods like DCG (Dynamic Cognitive Game).

Due to advances in AI, more and more CAPTCHA designs have become ineffective, as the underlying algorithm problems become solvable by AI tools. Specifically, recent advances in Deep Learning (DL) have reduced the gap between humans and machine's ability in solving problems that have been typically used in CAPTCHAs. This breakthrough in AI led some researches to believe that DL would lead to the 'end' of CAPTCHAs [8].

Image Denoising. Unlike pictures in the traditional image training set (e.g. cifar100, ImageNet), various of noise is introduced to CAPTCHA deliberately which makes it difficult to automatically retrieve the exact same image from the web, while still allowing humans to extract the key concepts from the it. As a result, CAPTCHA solvers always apply denising methods to remove the noise from CAPTCHA before recognizing the character in it. In the past few decades, several methods have been proposed to remove the noise and recover the true image and they share the same basic remark: denoising is achieved by averaging [4]. Depending on how to assign weights to pixel, this averaging can be performed locally or non-locally. For example, locally models include Gaussian smoothing [17], anisotropic filtering [24] and total variation minimization [25] assign weights based on the distance, and non-locally models including NL-means [5] assign weights based on the similarity between the pixels.

5.2 Adversarial Attacks Techniques.

The goal of adversarial attacks to image classification systems is to add small perturbations to images that lead these systems to make incorrect predictions. Various adversarial machine learning attack methods have proposed, including evasion attacks and poisoning attacks. *Adversarial examples* [29] is one of the evasion attacks that leads to security threats in real-world applications of convolutional networks. Many techniques has been proposed to find examples x^{adv} such that x^{adv} is very close to the given example x (belonging to class c_1) but was incorrectly classified as belonging to class c_2 $\neq c_1$, *including* :

- *Family of Fast Gradient Sign Method (FGSM)*: FGSM [11] and random FGSM [30] finds an l_∞-bounded human-imperceptible adversary in the direction of the loss gradient $\nabla_x Loss(h(x), y_{true})$ with the update equation:
$$x^{adv} = x + \epsilon * sign(\nabla_x Loss(h(x), y_{true})) \tag{1}$$
Iterative Fast Gradient Sign Method (I-FGSD) [15] extended equation 1 to an iterative version, and Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [10] further integrated the momen- tum term into the attack process to stabilize update directions and escape from poor local maximum. Diverse Inputs Iterative Fast Gradient Sign Method (DI^2-FGSM) [33] improved transferbility by creating diverse input patterns.
- Carlini and Wagner Attack (CW): CW method [7] formulates the problem as:
$$\begin{aligned} &minimize\ D(x, x + \sigma) \\ \text{s.t. } &C(x + \sigma) = t \ and \ x + \sigma \in [0, 1]^n \end{aligned} \tag{2}$$
Since the first constraint makes the problem difficult to optimize directly, it can be replaced with $f(x+\sigma) \leq 0$ (7 surrogate implementations are suggested) and transform the problem to:
$$minimize\ ||\sigma||^p + c * f(x + \sigma) \text{ s.t. } x + \sigma \in [0, 1]^n$$
The *box constraint* can be tackled with 1) Projected gradient descent 2) Clipped gradient descent 3) Change of variables.
- Projected Gradient Descent (PGD) [21]: PGD as a universal "first-order adversary" is a mulit-step variant version of FGSM on the negative loss function:
$$x_{t+1}^{adv} = Clip_{X,\epsilon}(x_t^{adv} + \alpha * sign(\nabla_x Loss(h(x), y_{true}))\,) \tag{3}$$
Logit-space Projected Gradient Ascent (LS-PGA) [6].
- Other techniques to create adversarial examples include Deepfool [22], Jacobian-based Salieny map (JSMA) [23]

Expectation over transformation (EOT). The methods mentioned above craft adversarial example by following gradient $\nabla_x P(y|x)$ to maximize the log-likelihood of the target class y_t over a ϵ-radius ball around the original sample x. It leads to the fact that the adversarial examples are not resistant to transformations that occur in the real world, such as angle and viewpoint changes [19, 20]. The study [1] proposed to model such perturbations within the optimization procedure, named Expectation over transformation (*EOT*). Instead of by optimizing the log-likelihood of a single example, EOT calculate the expectation of log-likelihood over samples under a chosen distribution T of transformation functions t.

Feature denoising. It is a technique to suppress much of the noise in the feature map and make the responses focus on visually meaningful content. Empirical evidence illustrating this technique can successfully increase adversarial robustness. The rationale behinds feature denoising is that the transformations performed by the layers in the convolutional network exacerbate the perturbation of the features induced by an adversarial image gradually increases as the image is propagated through the network. Therefore, comparing to the traditional attack methods that focus on minimizing perturbations at the *pixel-level*, constraints imposed at the feature level in networks craft the perturbation that better hallucinates non-existing activation in the feature maps, which leads to a more robust adversarial example. The study in [32] introduced *denoising blocks* by using non-local mean filter to the network architectures.

Universal Adversarial Patch. Localized adversarial attacks focus on creating a scene independent pattern, named universal adversarial patch/perturbation (**UAP**). Basically, the objective is to find a perturbed image x̂ satisfying:

$$\begin{aligned} \hat{x} &= \underset{\hat{x}}{\operatorname{argmax}}\, \mathbb{E}_{t\sim T}\left[logP(\hat{y}|t')\right] \\ \text{s.t.}\, &\mathbb{E}_{t\sim T}\left[d(t(x'),t(x)))\right] < \epsilon \end{aligned} \tag{4}$$

The patch can be a physical-printed image that is agnostic to camera angles, lighting conditions and even the type of classifier [3] (solved the unconstrained optimization version of equation 4). With techniques like LaVAN [13], the size of the patch can only replace 2% of original images.

5.3 Defense

Many defenses have been proposed to combat adversarial examples. Some studies have proved that pre-processing images (e.g. Feature squeezing) can remove adversarial perturbations; more robust approach focuses on hardening neural classifiers to reduce adversarial susceptibility.

Feature squeezing. Some studies indicate that, if we consider the the adversarial perturbation to images as a kind of noise, they can be eliminated by applying noise-filtering techniques (e.g. FAdeML [16], Adaptive Noise Reduction [14] and Defense-GAN [26]).

Feature squeezing is an effective method to defense against adversarial attacks. Basically this method reduces the search space available to an adversary by coalescing samples that correspond to many different feature vectors in the original space into a single sample. For example, the study in [34] successfully explored two feature squeezing methods: reduce the color bit depth of each pixel and spatial smoothing.

Basically, we apply feature squeezing techniques (e.g. global gradient smoothing, various filters, compression or total variance minimization) before passing the image into the classifier. By comparing the softmax probability vectors across different classifier outputs on both original image and the lower fidelity version of it, it is highly possible to detect adversarial examples successfully.

Adversarial training. Adversarial training [11] remains among the most effective and popular strategies. In its simplest form, adversarial training minimizes a loss function that measures performance of the model on both clean and adversarial data as follows:

$$minimize\ L_{adv}(\theta) = \sum_i \kappa L(\theta, x_i, y_i) + (1-\kappa)L(\theta, x_i^{adv}, y_i) \tag{5}$$

where L is the loss function, (x_i, y_i) is an input/label pair, θ is the trainable model parameters, κ is a hyper-parameter, and x_i^{adv} is the adversarial example corresponding to sample x_i. However, the key drawback is the computational cost to produce a batch of adversarial examples for every mini-batch during training.

Fast optimization approaches *without adversarial examples* are proposed to avoid high computational expense. For example, adding Gaussian noise to images to replace adversarial examples during training can be used to improve the adversarial robustness of classifiers [28]. *Label smoothing* [31] converts "one-hot" label vectors into "one-warm" vectors that represent a low-confidence classification, and *logit squeezing* [12] explicitly penalizes large logits by adding a regularization term to the training objective. Moreover, adversarial logit pairing (*ALP*) [12] enforced an extra regularization term $L_{logit-pair}$ in adversarial training process to encourage the model also matches the logits from a clean image x and its corresponding adversarial image x'. It helps guide the model towards better internal representations of the data.

The study [21] further unified adversarial training to a formulation from an optimization view, and proved that this adversarial empirical risk minimization problem is a saddle point problem as:

$$\min_{\theta}\ \mathbb{E}_{(x,y)\sim D}\left[\max_{\delta\in S} L(\theta, x+\delta, y)\right] \tag{6}$$

where S represents the set of feasible adversarial perturbations, and D is the underlying training data distribution. The inner maximization problem aims to find an adversarial version of a given data point x that achieves a high loss and the goal of the outer minimization problem is to find model parameters so that the 'adversarial loss' given by the inner attack problem is minimized. This optimization problem is hard to solve mathematically [27]

6 CONCLUSION

Pioneer research leverages adversarial machine learning attack techniques as the tool to perform defense against Hacker's AI-powered toolkit, as the next-generation hackers also heavily rely on AI to launch attacks. With the concrete examples, adversarial perturbation can enhance the resistance against AI-based exploit toolkits for CAPTCHAs. We demonstrate how to perform various levels ('passive' and 'active') of defense, not only preventing the attack but also detecting attacker's AI-powered toolkits.

7 ACKNOWLEDGMENTS

We would like to thank blackhat USA committees. We would like to thank Tianyun Zhang and Shengmin Jin for their technical suggestion. We also thank Xiaoyun Shen-Krizic and Minqi Cai for their detailed editing suggestion.

REFERENCES

[1] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2017. Synthesizing Robust Adversarial Examples. *CoRR* abs/1707.07397 (2017). arXiv:1707.07397 http://arxiv.org/abs/1707.07397

[2] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. 2017. Exploring the Space of Black-box Attacks on Deep Neural Networks. (2017). arXiv:cs.LG/1712.09491

[3] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial Patch. (2017). arXiv:cs.CV/1712.09665

[4] Antoni Buades, Bartomeu Coll, and Jean Michel Morel. 2004. On image denoising methods. *CMLA Preprint* 5 (2004).

[5] Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. IEEE, 60–65.

[6] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. 2018. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. https://openreview.net/pdf?id=S18Su--CW

[7] 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.

[8] Rahul Chatterjee, Anish Athayle, Devdatta Akhawe, Ari Juels, and Thomas Ristenpart. pASSWORD tYPOS and How to Correct Them Securely. (????).

[9] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2019. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 321–338. https://www.usenix.org/conference/usenixsecurity19/presentation/demontis

[10] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Xiaolin Hu, and Jun Zhu. 2017. Discovering Adversarial Examples with Momentum. *CoRR* abs/1710.06081 (2017). arXiv:1710.06081 http://arxiv.org/abs/1710.06081

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[12] Harini Kannan, Alexey Kurakin, and Ian J. Goodfellow. 2018. Adversarial Logit Pairing. *CoRR* abs/1803.06373 (2018). arXiv:1803.06373 http://arxiv.org/abs/1803.06373

[13] Danny Karmon, Daniel Zoran, and Yoav Goldberg. 2018. LaVAN: Localized and Visible Adversarial Noise. *CoRR* abs/1801.02608 (2018). arXiv:1801.02608 http://arxiv.org/abs/1801.02608

[14] Faiq Khalid, Muhammad Abdullah Hanif, Semeen Rehman, Junaid Qadir, and Muhammad Shafique. 2018. FAdeML: Understanding the Impact of Pre-Processing Noise Filtering on Adversarial Machine Learning. *CoRR* abs/1811.01444 (2018). arXiv:1811.01444 http://arxiv.org/abs/1811.01444

[15] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016). arXiv:1607.02533 http://arxiv.org/abs/1607.02533

[16] Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, Wenchang Shi, and Xiaofeng Wang. 2017. Detecting Adversarial Examples in Deep Networks with Adaptive Noise Reduction. *CoRR* abs/1705.08378 (2017). arXiv:1705.08378 http://arxiv.org/abs/1705.08378

[17] Michael Lindenbaum, M Fischer, and A Bruckstein. 1994. On Gabor's contribution to image enhancement. *Pattern recognition* 27, 1 (1994), 1–8.

[18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society.

[19] Jiajun Lu, Hussein Sibai, Evan Fabry, and David A. Forsyth. 2017. NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles. *CoRR* abs/1707.03501 (2017). arXiv:1707.03501 http://arxiv.org/abs/1707.03501

[20] Yan Luo, Xavier Boix, Gemma Roig, Tomaso A. Poggio, and Qi Zhao. 2015. Foveation-based Mechanisms Alleviate Adversarial Examples. *CoRR* abs/1511.06292 (2015). arXiv:1511.06292 http://arxiv.org/abs/1511.06292

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[22] 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.

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

[24] Pietro Perona and Jitendra Malik. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence* 12, 7 (1990), 629–639.

[25] Leonid I Rudin and Stanley Osher. 1994. Total variation based image restoration with free local constraints. In *Proceedings of 1st International Conference on Image Processing*, Vol. 1. IEEE, 31–35.

[26] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. *CoRR* abs/1805.06605 (2018). arXiv:1805.06605 http://arxiv.org/abs/1805.06605

[27] Amirreza Shaeiri, Rozhin Nobahari, and Mohammad Hossein Rohban. 2020. Towards Deep Learning Models Resistant to Large Perturbations. *arXiv preprint arXiv:2003.13370* (2020).

[28] Ali Shafahi, Amin Ghiasi, Furong Huang, and Tom Goldstein. 2019. Label Smoothing and Logit Squeezing: A Replacement for Adversarial Training? (10 2019).

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

[30] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. The Space of Transferable Adversarial Examples. (2017). arXiv:stat.ML/1704.03453

[31] David Warde-Farley. 2016. 1 Adversarial Perturbations of Deep Neural Networks.

[32] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. 2019. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 501–509.

[33] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2730–2739.

[34] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *CoRR* abs/1704.01155 (2017). arXiv:1704.01155 http://arxiv.org/abs/1704.01155

[35] Xiao Yang, Fangyun Wei, Hongyang Zhang, Xiang Ming, and Jun Zhu. 2019. Design and Interpretation of Universal Adversarial Patches in Face Detection. (2019). arXiv:cs.CV/1912.05021

[36] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. 2018. Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. In *The 25th ACM Conference on Computer and Communications Security (CCS '18)*. ACM.