Adversarial Attack on Captcha Classification

Ge Xiaoran
HKUST
xiaoran.ge@connect.ust.hk

Ou Huiyi
HKUST
houaa@connect.ust.hk

Xu Zhanxiang
HKUST
zxuch@connect.ust.hk

Abstract

Motivated by the broad usage of CAPTCHA in Internet services, this paper conducts experiment to examine and compare traditional, black-box and white-box adversarial attack methods' performance in attacking 5 CAPTCHA recognition models. In summary, we found that Fast Gradient Sign Method (FGSM) has the best performance in decreasing recognition models' accuracy of the true label. Although Simple Black-box Adversarial Attack Method is computationally expensive, it has bettter transferability and higher attack efficiency compared to other white-box attacks methods.

1. Introduction

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), a technology that aims to protect Internet services from automatic script attacks and spams. [1]

Registration, login, voting, etc., on websites can be easily abused by various malicious automated programs. For example, malicious programs can use the website's registration system to send spam messages to users, or perform brute force cracking to obtain users' account passwords, or put fake advertisements on the website, etc. Although CAPTCHA has been widely used on various web pages, it is not absolutely secure. Currently, most of the common CAPTCHAs are in the form of images, which are combinations of letters, numbers and punctuation marks, with some confusing elements such as changing the size of the text, distorting the text, adding distracting lines and noises and changing the color, etc., while ensuring a certain level of clarity so that humans can recognize the content properly. In the past, due to the limitations of the technical environment, ordinary image CAPTCHAs were still relatively reliable and could effectively block automated programs. However, after the rapid development of machine learning in recent years, image recognition based on machine learning has become quite powerful, and there are now many network structures that can be used directly by developers. These new CAPTCHA cracking methods can easily identify traditional image CAPTCHAs [2], seriously threatening the security of websites. For websites operated by small companies or individuals, it is very expensive to update the whole set of CAPTCHAs and the verification methods regularly, and it would be very meaningful to find a simple way to increase the difficulty of CAPTCHA recognition at low cost.

Although deep learing models have achieved extraordinary and superhuman performance in computer vision since the inception of AlexNet in 2012, Szegedy *et al.* [3] discovered that deep neural networks are extremely prone to small perturbations to images, which remain almost imperceptible to human eyes. Such adversarial attack could significantly alter the model's prediction and even fool the model to output incorrect classification results with very high confidence. Moreover, the attacked images could also fool other network classifiers. The findings triggered a broad research in adversarial attacks and defenses for deep learning.

Organization of the report is as follows. In Section 2, we first review the literation work on image classification and adversarial attacks, and introduce the methods we adopted in details. In Section 3, we describe the dataset and classification architectures we used in this project, and experiment results after trying different adversarial attack methods. In Section 4, we present some furture work that we could try out and draw a conclusion in Section 5. In Section 6, we note down the contributions of each team member. The code of this project can be found in this Github repo.[1]

2. Related Work

2.1. Image classification

Image classification based on deep learning is a very mature technique. Its goal is to classify different images into different categories to achieve the minimum classification error. Bostik *et al.* [4] proposed that this technique can also be used to recognize image CAPTCHAs.

In this project, we use several well-known and currently popular models.

[1] Github repo: https://github.com/Karenou/Captcha_Classification.git

2.1.1 *RestNet*

RestNet, proposed by He [5], is a deep neural network using residual learning framework that makes the network deeper, accelerates the convergence speed of loss and makes the optimization easier. And this network iis suitable for many computer vision missions.

2.1.2 *MobileNet*

MobileNets are a class of efficient models for mobile and embedded vision applications, Howard *et al.* [6], which based on a streamlined architecture and used depth-wise separable convolutions to build up the light weight deep neural networks. It's focus on small and low latency which are important to mobile devices, embedded devices and even self-driving cars.

2.1.3 *DenseNet*

DenseNet is the network that connects each layer to every other layer in a feed-forward fashion, Huang *et al.* [7], which alleviates the gradient vanishing problem, strengthens feature propagation and reduces the number of parameters.

2.2. Adversarial attack

Since our project mainly focuses on captcha classification, we would like to review the literature in adversarial attacks specifically for classification tasks in computer vision and also highlight several attack methods, which we adopted and conducted experiments in this project.

Adversarial attacks can be categorized into targeted attack and non-targeted attack by purpose of misclassification [8]. Targeted attacks aim to misleading the classification model to classify the attacked image as a targeted class label while non-targeted attacks try to decrease the model's confidence of predicting the true class label.

Adverssarial attacks can also be categorized based on different settings where adversarial images are generated. Below are a brief overview of classical attack methods.

2.2.1 *Black-box attacks*

Adversarial attacks can be divided into white box attacks and black box attacks. White box attack means that the target model information is completely available, while black box attack means that the target model is invisible. The basic white box attack method uses the gradient ascent method to directly update the input samples, moving them toward the decision boundary of the target model, until they become adversarial samples, such as FGSM, PGD attack, etc.

When the attacker cannot access the detailed information of the model, the white-box attack is obviously not applicable. The black-box attack means that they do not understand the parameters and structural information of the model, and only use the input and output of the model to generate adversarial samples, and then attack the network.

In real life, some AI algorithms have a high level of confidentiality, and the information of the model is rarely completely leaked. Therefore, the situations that use white box attacks is far less than that of black box attacks. But the ideas of the two are the same, and the gradient information is used to generate adversarial samples.

Black box attacks are divided into query-based attacks and transfer-based attacks.

i) Query-Based Attacks:

Simple Black-box attack query based (SimBA)

A query-based black box attack algorithm requires multiple queries to obtain as much information as possible. Many earlier query-based black box attacks require many queries to achieve a good attack success rate, and more queries consume a lot of time and computing resources, so the author of the SimBA paper proposed a more query efficiency algorithm (Chuan Guo, 2019).

Algorithm 1 SimBA in Pseudocode

procedure SIMBA($\mathbf{x}, y, Q, \epsilon$)
  $\delta = \mathbf{0}$
  $\mathbf{p} = p_h(y \mid \mathbf{x})$
  while $\mathbf{p}_y = \max_{y'} \mathbf{p}_{y'}$ **do**
    Pick randomly without replacement: $\mathbf{q} \in Q$
    for $\alpha \in \{\epsilon, -\epsilon\}$ **do**
      $\mathbf{p}' = p_h(y \mid \mathbf{x} + \delta + \alpha\mathbf{q})$
      if $\mathbf{p}'_y < \mathbf{p}_y$ **then**
        $\delta = \delta + \alpha\mathbf{q}$
        $\mathbf{p} = \mathbf{p}'$
        break
  return δ

SimBA Algorithm

The above is the pseudo code of the SimBA algorithm. It can be seen from the pseudo code that the author reduces unnecessary searches by controlling the direction q. Because Q represents a set of candidate orthogonal vectors, and q is the direction selected in the orthogonal vector. This leads to the direction of each query will not cancel each other, which can reduce a lot of unnecessary searches. It can be seen from the author's experiment that the SimBA algorithm reduces a large number of queries and still maintains a high success rate.

ii) Transfer-Based Attacks

Most of the existing query-based black box attack methods do not take advantage of the transferability of adversarial examples. However, as shown in Figure 2, the gradient saliency maps generated by the models Inception-V3 and ResNet-152 are very similar, indicating that the gradient of the alternative model can be used as a priori to improve the query efficiency of the

black box attack. Some visualization algorithms such as LIME and Grad-CAM can also prove that the importance of each area in the picture to the prediction result is different.

Some previous work used the gradient of the alternative model as a priori to reduce the gradient estimation sampling space, and achieved some results, but the attack efficiency is still not high enough. They fixed alternative models during the attack process, and did not consider using migration attacks to directly improve attack efficiency (Jiangcheng Yang, 2020).

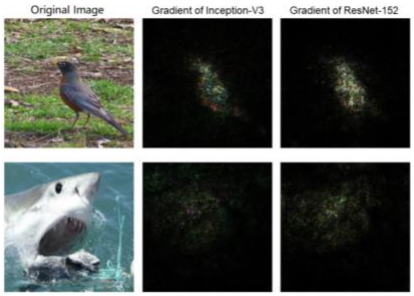

Figure A1: **The consistency of visual saliency map from vision benchmark models**. Gradient visualization of Inception-V3 [15] and ResNet-152 [9].

Figure 1

In view of the different importance of each area in the picture, the author of the paper proposes a new algorithm as the baseline, SimBA+ (Jiangcheng Yang, 2020).

Algorithm A1 SimBA+

Input: input image X, victim model $\mathcal{V}$, surrogate model $\mathcal{S}$, attack step ϵ.
Output: (adversarial) example X_{adv}.
Initialize $X_{adv} = X$;
Query $\mathcal{V}$ to initialize target probability P_T and loss J;
Cache the gradient map M from $\mathcal{S}$;
for i **in** $\{0, 1, 2, ...\}$ **do**
  Generate perturbation δ with Gaussian-smoothed coordinate q (sampled proportional to $|M|$);
  for α **in** $\{+\epsilon, -\epsilon\}$ **do**
    $X'_{adv} = clip(X_{adv} + \alpha \cdot \delta, X)$ {Eq. 3};
    Query $\mathcal{V}$ for target probability P'_T and loss J';
    if $J' < J$ **then**
      Update $X_{adv} = X'_{adv}; P_T = P'_T; J = J'$;
      break;
    end if
  end for
  if *exceed max query budget* **or** *success* **then**
    break;
  end if
end for
return X_{adv}.

SimBA + Algorithm

The above is the pseudo code of the SimBA+ algorithm. Because the SimBA algorithm is to sample evenly in the picture, and the weight of each area in the picture is different, it is easy to waste too much time in the less important areas. It can be seen that the efficiency of such query is not high. The author replaced the orthogonal vector q in SimBA with the gradient map M. Although this change is not very large, it has achieved good results.

Although the SimBA+ algorithm has been optimized to some extent, it can be further optimized. Methods based on anti-migration usually have a strong attack success rate, while query-based methods often require more queries although they can achieve a high success rate. In theory, the advantages of these two methods can be combined to improve the effectiveness of the attack.

However, the transfer-based method and the query-based method are developed independently, and there is currently no research to explore the effectiveness of how to combine the two attacks. The author of the paper tried to use the migration-based method to further improve the query efficiency of the attack algorithm. For the migration-based method, the team chose the translationinvariant attack enhanced by momentum, i.e. TIMI, which was improved by adding gradient momentum and Gaussian blur to the attack iteration. Its attack success rate is still not high enough as query-based algorithms. Based on SimBA+ and TIMI, we further propose SimBA++, which alternately runs the transfer-based method and the query-based method during the attack iteration process, which greatly reduces the number of attacks.

Algorithm A2 SimBA++

Input: input image X, victim model $\mathcal{V}$, surrogate model $\mathcal{S}$, transferability-based attack $\mathcal{T}$, attack step ϵ, query iteration n_Q.
Output: (adversarial) example X_{adv}.
Initialize $X_{adv} = X$;
Query $\mathcal{V}$ to initialize target probability P_T and loss J;
for i **in** $\{0, 1, 2, ...\}$ **do**
  if $i \bmod n_Q$ **then**
    {Run transferability-based attack}
    Run $X'_{adv} = clip(\mathcal{T}(X_{adv}), X)$ {Eq. 3} with $\mathcal{S}$;
    Cache the gradient map M from $\mathcal{S}$;
    Query $\mathcal{V}$ for target probability P'_T and loss J';
  else
    {Run query-based attack}
    Generate perturbation δ with Gaussian-smoothed coordinate q (sampled proportional to $|M|$);
    for α **in** $\{+\epsilon, -\epsilon\}$ **do**
      $X'_{adv} = clip(X_{adv} + \alpha \cdot \delta, X)$ {Eq. 3};
      Query $\mathcal{V}$ for target probability P'_T and loss J';
      if $J' < J$ **then**
        break;
      end if
    end for
  end if
  if $J' < J$ **then**
    Update $X_{adv} = X'_{adv}; P_T = P'_T; J = J'$;
  end if
  if *exceed max query budget* **or** *success* **then**
    break;
  end if
end for
return X_{adv}.

SimBA ++ Algorithm

The above is the pseudo code of the SimBA++ algorithm. It can be seen from the pseudo-code that this algorithm is a combination of TIMI and SimBA+ algorithms, and these two algorithms are used in turn during the attack.

The attack efficiency of transfer-based attacks largely depends on the similarity between the alternative model and the target model. If the similarity between the two models is higher, the attack effect of transfer-based attacks will be better.

In theory, the query feedback from the target model leaks the information of the target model. Based on this observation, the author's team proposed a learnable black-box attack algorithm(LeBA). Because there are few studies that combine algorithms in the two directions, there are few studies that use query feedback to update alternative models. On the basis of SimBA++, LeBA uses query feedback to update the replacement model to approximate the target model.

Algorithm 1 Learnable Black-Box Attack (LeBA)

Input: input image X, victim model $\mathcal{V}$, surrogate model $\mathcal{S}$, transferability-based attack $\mathcal{T}$, attack step ϵ, query iteration n_Q, buffer $\mathbb{B}$, buffer size b, *training / test* mode.
Output: (adversarial) example X_{adv}.
Initialize $X_{adv} = X$; $\mathbb{B} = \emptyset$;
Query $\mathcal{V}$ to initialize target probability P_T and loss J;
for i **in** $\{0, 1, 2, ...\}$ **do**
  if $i \bmod n_Q$ **then**
    {Run transferability-based attack}
    Run $X'_{adv} = clip(\mathcal{T}(X_{adv}), X)$ {Eq. 3} with $\mathcal{S}$;
    Cache the gradient map M from $\mathcal{S}$;
    Query $\mathcal{V}$ for target probability P'_T and loss J';
  else
    {Run query-based attack}
    Generate perturbation δ with Gaussian-smoothed coordinate q (sampled proportional to $|M|$);
    for α **in** $\{+\epsilon, -\epsilon\}$ **do**
      $X'_{adv} = clip(X_{adv} + \alpha \cdot \delta, X)$ {Eq. 3};
      Query $\mathcal{V}$ for target probability P'_T and loss J';
      $\mathbb{B}.add(X'_{adv}, X_{adv}, P'_T, P_T)$;
      if $J' < J$ **then**
        break;
      end if
    end for
  end if
  if *training* mode **and** $B.size = b$ **then**
    [Train the surrogate model $\mathcal{S}$ with HOGA (Supplementary Algorithm A3)]
    $\mathbb{B} = \emptyset$;
  end if
  if $J' < J$ **then**
    Update $X_{adv} = X'_{adv}$; $P_T = P'_T$; $J = J'$;
  end if
  if *exceed max query budget* **or** *success* **then**
    break;
  end if
end for
return X_{adv}.

LeBA Algorithm

The above is the pseudo code of the LeBA algorithm. It can be seen from the pseudo-code that the LeBA algorithm is an enhancement of the SimBA++ algorithm, or it is switched back and forth between SimBA+ and TIMI, and the query result of SimBA+ is used to update the model. In order to use the limited information of query to update the model, the author's team also invented high-order gradient approximation to approximate the target model.

Algorithm A3 High-Order Gradient Approximation (HOGA)

Input: surrogate model $\mathcal{S}$, buffer $\mathbb{B}$, λ and γ.
Output: updated surrogate model $\mathcal{S}$.
Batch $\mathbf{X}'_{adv}, \mathbf{X}_{adv}, \mathbf{P}'_T, \mathbf{P}_T$ from $\mathbb{B}$;
Compute surrogate target probability $\mathbf{S}_T = \mathcal{S}(\mathbf{X}_{adv})$;
Compute Forward Loss $l_F = MSE(\mathbf{S}_T, \mathbf{P}_T)$;
Create gradient graph and compute $\mathbf{g}_s = \frac{\partial log \mathbf{S}_T}{\partial \mathbf{X}_{adv}}$;
Compute Backward Loss l_B using
$l_B = MSE(\mathbf{g}_s(\mathbf{X}'_{adv} - \mathbf{X}_{adv}), \gamma(log\mathbf{P}'_T - log\mathbf{P}_T))$;
Back-propagate $l_B + \lambda l_F$ with high-order gradient;
Update $\gamma = 0.9 \cdot \gamma + 0.1 \cdot \frac{\sum |\mathbf{g}_s(\mathbf{X}'_{adv} - \mathbf{X}_{adv})|}{\sum |log\mathbf{P}'_T - log\mathbf{P}_T|}$;
Optimize $\mathcal{S}$;
return $\mathcal{S}$.

HOGA Algorithm

HOGA is similar to gradient penalty. By constructing a high-gradient calculation graph when the replacement model is backpropagated, the approximate gradient obtained by the direct query approximates the gradient of the backpropagation of the replacement model.

2.2.2 White-box attacks

i) Box-constrained L-BFGS

Szegedy [3] proposed the first targted adversarial attack method as shown in (1). By adding a perturbation $\boldsymbol{\rho}$ to the clean input image $\mathbf{I}_c$, the attack could fool the classification model to make the wrong prediction $\mathcal{F}(\mathbf{I}_c + \boldsymbol{\rho}) = t$, where t is the targeted label. To ensure the perturbation is small and the attacked image looks like the clean image, a box constraint is imposed to clip the values between 0 and 1. Since the loss function of deep neural networks are usually non-convex, the below optimization problem is sometimes hard to solve.

$$\boldsymbol{min_\rho}\ \boldsymbol{c}\left|\left|\boldsymbol{\rho}\right|\right|_2^2 + \ \boldsymbol{\mathcal{L}}(\mathbf{I}_c + \boldsymbol{\rho}, \boldsymbol{t}), \boldsymbol{s.t.}\, \mathbf{I}_c + \boldsymbol{\rho} \in [\mathbf{0}, \mathbf{1}]^{\boldsymbol{m}} \quad (1)$$

ii) Carlini and Wagner Margin-based Attack

Carlini and Wagner [9] introduced another set of adversarial attacks as shown in (2). It maximizes the difference between the logits of class t and the second best class so that it fools the classification model to make the prediction of class t with high confidence compared to other class labels. While (2) shows the loss function in the ℓ_2 case, the method can be generalized to ℓ_0 and ℓ_∞ cases. Despite that Carlini's method is considered as a strong attack, this method is computationally expensive.

$$\boldsymbol{min_\rho}\ \boldsymbol{c}\left|\left|\mathbf{I}_c - \boldsymbol{\rho}\right|\right|_2^2 + \mathbf{I}(\boldsymbol{\rho}), \boldsymbol{s.t.}\, \boldsymbol{\rho} \in [\mathbf{0}, \mathbf{1}]^{\boldsymbol{m}} \quad (2)$$
$$where\ \mathbf{I}(\boldsymbol{\rho}) = \ max\ \{\max_{i \neq t} Z_i(\boldsymbol{\rho}) - Z_t(\boldsymbol{\rho}), -\kappa\}$$

iii) Fast Gradient Sign Method (FGSM)

To improve the efficiency of generating adversarial perturbations, Goodfellow *et al.* [10] proposed the fast gradient sign method, whose derivation is shown in (3) under infinity norm constraint. The direction of perturbations is the same as the direction of gradient, and its scale is controlled by a hyperparameter ϵ. FGSM is not also practical but also demonstrates strong attack performance in ImageNet. The top-1 error rates on images generated by FGSM is 63% - 69% for $\epsilon \in [2, 32]$ [11]. Similarly, FGSM can also be generalized to ℓ_2-norm and Miyato *et al.* [12] proposed such method in equation (4).

$$\boldsymbol{\rho} = \boldsymbol{\epsilon}\ \mathbf{sign}(\boldsymbol{\nabla}_{\mathbf{I_c}} \boldsymbol{\mathcal{L}}(\mathbf{I_c}, \boldsymbol{t}) \quad (3)$$

$$\boldsymbol{\rho} = \boldsymbol{\epsilon}\ \frac{\boldsymbol{\nabla}_{\mathbf{I_c}} \boldsymbol{\mathcal{L}}(\mathbf{I_c}, \boldsymbol{t})}{||\boldsymbol{\nabla}_{\mathbf{I_c}} \boldsymbol{\mathcal{L}}(\mathbf{I_c}, \boldsymbol{t})||_2} \quad (4)$$

iv) Iterative Fast Gradient Sign Method

FGSM adopts a one-step method by changing the image in the direction of gradients, and thus, its attack may be too weak. Thus, there are variants of FGSM such as Iterative Fast Gradient Sign method in equation (5) that borrows the idea of iterative gradient update. It updates the perturbation in multiple small steps controlled by a step size α and clips its pixel values at ϵ.

$$\mathbf{I}_{\boldsymbol{\rho}}^{\boldsymbol{i+1}} = \boldsymbol{clip}_{\boldsymbol{\epsilon}}\ \{\mathbf{I}_{\boldsymbol{\rho}}^{\boldsymbol{i}} + \boldsymbol{\alpha}\ \mathbf{sign}(\boldsymbol{\nabla}_{\mathbf{I_c}} \boldsymbol{\mathcal{L}}(\mathbf{I}_{\boldsymbol{\rho}}^{\mathbf{i}}, \boldsymbol{t})\} \quad (5)$$

The Basic Iterative Method (BIM) initializes I_{ρ}^{0} to be same as the clean input image I_c. There are other initialization methods such as Projected Gradient Descent (PGD), which picks a random point within the ϵ-ball defined under ℓ_∞ around I_c and sets it as I_{ρ}^{0}.

Rather than taking a fixed step size in each iteration, one natural idea is to apply momentum acceleration. Dong *et al.* [13] proposed the Momentum Iterative Gradient-based method (MI-FGSM) and won the first place in NIPS 2017 Adversarial Attack Competition. The equation (6) is MI-FGSM implementation that satisfies the ℓ_∞ restriction. It could also be generalized to ℓ_2 -norm case by adjusting the calculation of gradient. μ is the decay factor. The step size α is set to be $\frac{\epsilon}{T}$, where T is the number of iterations and ϵ controls the size of perturbations.

$$\boldsymbol{g}_{t+1} = \mu \boldsymbol{g}_t + \frac{\nabla_{\mathbf{I_c}} \mathcal{L}(\mathbf{I_c}, t)}{||\nabla_{\mathbf{I_c}} \mathcal{L}(\mathbf{I_c}, t)||_1}$$

$$\mathbf{I}_{\boldsymbol{\rho}}^{\boldsymbol{i+1}} = \mathbf{I}_{\boldsymbol{\rho}}^{\boldsymbol{i}} + \boldsymbol{\alpha}\ \mathbf{sign}(\boldsymbol{g}_{t+1}) \tag{6}$$

This method again exemplifies that adversarial example generation is similar to training a deep neural network model and its transferability is analogue to model's generalization power.

v) Elastic-net Attack

The generation of adversarial examples can be regarded as an optimization problem. Chen *et al.* [14] proposed a method as shown in (8) that incorporates elastic-net regularization into the optimization. It minimizes the $\ell_1$and ℓ_2-norm of the perturbations simultaneously.

$$\boldsymbol{min}_{\boldsymbol{\rho}}\ \boldsymbol{\beta}||\boldsymbol{\rho}||_1 + ||\boldsymbol{\rho}||_2^2 + \boldsymbol{c}\mathcal{L}(\mathbf{I_c} + \boldsymbol{\rho}, \boldsymbol{t}),$$
$$\boldsymbol{s.t.}\ \mathbf{I_c} + \boldsymbol{\rho} \in [\mathbf{0}, \mathbf{1}]^{\boldsymbol{m}} \tag{7}$$

Inspired by the Elastic-net Attack, F. Tramer and D. Boneh [15] proposed Sparse ℓ_1 Descent (SLIDE), an efficient attack for the ℓ_1-norm. Tramer [15] proved that PGD is inefficient in the ℓ_1-case, because in such scenario, the steepest descent direction is the unit vector $\boldsymbol{e}$ where $e_{i^*} = sign(g_{i^*})$, for $i^* = argmax_i(|g_i|)$. The algorithm is as follows. It has comparable performance as EAD while is much more efficient in terms of computation cost.

$$\boldsymbol{g} = \nabla \boldsymbol{\rho} \mathcal{L}(\mathbf{I_c} + \boldsymbol{\rho}, t)$$

$$\mathrm{e_i} = \mathrm{sign}(\mathrm{g_i}), \mathrm{if}\ |\mathrm{g_i}| \geq P_q(|\boldsymbol{g}|), else\ 0$$

$$\boldsymbol{\rho} = \boldsymbol{\rho} + \gamma\ \mathbf{e}/||\mathbf{e}||_1$$

$$\boldsymbol{\rho} = \prod_{S_1^{\epsilon}}(\boldsymbol{\rho}) \tag{8}$$

vi) Jacobian-based Saliency Map Attack

Most of the attack methods above use ℓ_2 or ℓ_∞-norm, but Papernot *et al.* [16] introduced another adversarial attack method under the ℓ_0 -norm restriction, by only modifying a subset of pixels. This method requires to compute the Jacobian matrix of the logits output. M_{in} is the dimension of the input layer and M_{out} is the dimension of the output layer. It then defines a saliency map with the to be perturbed pixels. By modifying the most importance pixels, it could increase the likelihood of a target class.

$$\nabla \boldsymbol{l}(\mathbf{x}) = \frac{\boldsymbol{\partial l(x)}}{\boldsymbol{\partial (x)}} = \left[\frac{\boldsymbol{\partial l_j(x)}}{\boldsymbol{\partial x_\gamma}}\right]_{\boldsymbol{\gamma \in 1,\ldots,M_{in},\ j \in 1,\ldots M_{out}}} \tag{9}$$

vii) DeepFool

Moonsavi [17] introduced a new attack method called DeepFool, which is more efficient than L-BFGS and generates adversarial examples with smaller perturbations than FGSM. The perturbations takes the image to theh decision broundary hyperplane.

$$\boldsymbol{argmin}_{\boldsymbol{\rho}}\ ||\boldsymbol{\rho}||_2,\ \boldsymbol{s.t.}\ \boldsymbol{C}(\boldsymbol{x} + \boldsymbol{\rho}) \neq \boldsymbol{C}(\boldsymbol{x}) \tag{10}$$

viii) Universal Adversarial Perturbations

Moonsavi also rasied a question of a universal adversarial examples that can fool the classifier on most benign examples. The method iteratively update the perturbation using all target benign samples and approximately solves the optimization using DeepFool method mention above.

3. Experiment

3.1. Dataset

Since the CAPTCHA was created, the competition between the creator of captcha and the attacker of captcha has continued. This causes captcha to become more and more complicated, and even real human users cannot correctly recognize the input that captcha wants.

For example, the image CAPTCHA technology currently used by Google cuts out a certain part of a picture so that human users can judge what the picture is, but the AI model cannot correctly judge these items.

Although these new CAPTCHA technologies are effective against AI models, these technologies are not user friendly. It takes more time for real users to recognize these CAPTCHAs. There are even some memes complaining about these complex CAPTCHAs on the Internet.

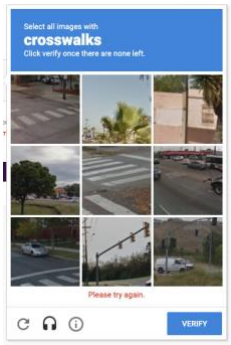

Figure 2

Therefore, our team decided to choose some user friendly CAPTCHAs as the data set. This data set[2] comes from Sogou, which is crawled by a crawler and placed on the forum for people to download.

This dataset has 177,498 captcha images, and the photos are saved in png format. We resize each image to 140 * 44. Each CAPTCHA has six digits, which only contains numbers and letters, which is very convenient for human users to identify. In order to confuse the AI model, the background of these pictures will contain some numbers and letters.

Figure 3

3.2. Recognition model

We trained 5 models for CAPTCHA classifications. We implemented a simply CNN network consisted of 3 convolution layers and 2 full connection layeres. The remaining models include ResNet18, ResNet50, MobileNetV2 and DenseNet161. Except for the self-implemented CNN model, the remaining architectures are directly loaded from torch.models and we set the *num_cls* parameter to 222 classes to fit our dataset.

For consistency, we trained the mdoels for 10 epochs using Adam optimizer with a 0.001 learning rate and a batch size of 32.

3.3. Measurement

We developed the following measurements for our experiments. To measure the predict accuracy, we stored the true labels for each test sample to match each prediction. And the accuracies of each experiment are calculated by the number of correct recognition devided by the number of test samples.

$$accuracy = \frac{number\ of\ correct\ recognition}{number\ of\ samples}$$

To measure the distortion between the clean images and the attack images, a measurement called *root mean square deviation* (RMSD) is used, adapted from Liu. [18]

$$d(x_{clean}, x_{attack}) = sqrt(\Sigma_i (x_{clean} - x_{attack})^2 / N)$$

Where i is the dimension of images; N is the resolution of the images, which defined as $width \times height$.

To measure the effect of the attack models on recognition models, we used the *rate of decrease* as the measurement on the accuracy.

$$rate\ of\ decrease = \frac{Acc_{clean} - Acc_{attack}}{Acc_{clean}} \times 100\%$$

To measure the efficiency of the effect of the attack model on image distortion. We proposed a measurement called *attack efficiency*, which is meamsuring the effect of the attack models due to the image with certain RMSD.

$$attack\ efficiency = \frac{rate\ of\ decrease}{RMSD}$$

3.4. Adversarial attack

3.4.1 Comparision of different ϵ

In comparison of different ϵ of each model, increasing ϵ means adding more noise to the adversarial images and its interference ability will be enhanced. For all attack models, increasing ϵ is a way to reduce the accuracy of recognition models. However, directly increasing the ϵ will increase the RMSD which affects the recognition of human eyes.

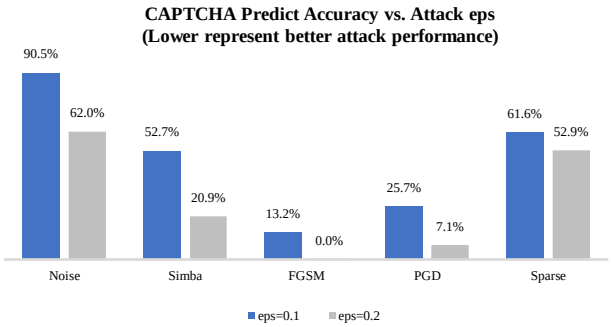

Chart 1

[2] The link to download CAPTCHA dataset used in this project: http://bbs.nightteam.cn/forum.php?mod=viewthread&tid=1259

3.4.2 *Comparision of different attack methods*

Comparing different attack methods, as the results, the interference of the FGSM attack method will have an advantage over the other models, when its ϵ is relatively small, it can have a significant impact on all recognition models, especially for CNN and MobileNet networks. When its ϵ is increased, all models'accuracy are approximately to 0. The attacked images generated by FGSM are also more disturbing to the human eyes, especially when the ϵ is set to 0.2, its RMSD has reached about 88, and eyes recognition becomes difficult. Next is the PGD model, which has a slightly lower impact than FGSM at ϵ of 0.1 and 0.2 respectively, but still has a significant impact on recognition models.The sparse model also has an effect on the recognition results, but not that significant as the FGSM and PGD, however its interference on the images to human eyes is relatively small. The RMSD values of the attack images generated by sparse model with ϵ as 30 and 40 are all close to 8%. Thus the sparse model is more balanced in terms of its effect on the recognition models and on human eye recognition. For the noise model, it can have a slightly better effect on simple CNN model as well as MobileNet model, but for other models that more complicated, its effect is not very significant.

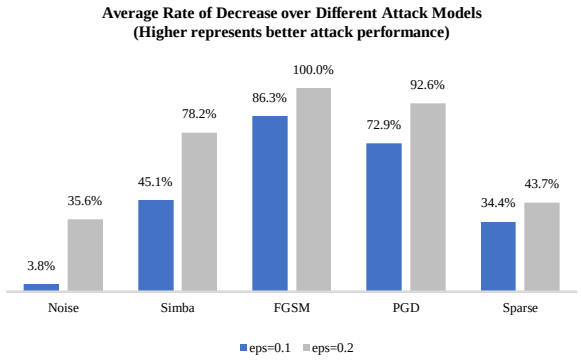

Chart 2

3.4.3 *Comparison of transferability*

As for the transferability of the attacks, the attack images based on CNN and MobileNet network are not superior. Except for the noise attack, the images generated according to CNN and MobileNet networks are relatively easier for other recognition models (ResNet18, ResNet50 and DenseNet161) to recognize. As the result, CNN and MobileNet are not very good in terms of robustness of adversarial attack. Comparing the black-box attack and the white-box attacks, we find that the transferability of the black-box attack is much better than the white-box attacks. In terms of overall results, the images generated using black-box have the best transferability.

3.4.4

As for the efficiency of the attacks, the results show that the simple black-box attack has the highest attack efficiency, and the Sparse attack follows. Although FGSM and PGD have the best effects on disturbing recognition models, their attack efficiencies are relatively low. And the noise attack has the lowest efficiency.

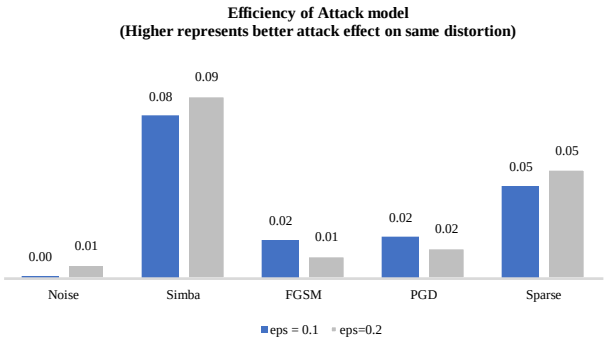

Chart 3

4. Future Work

In this paper (Akhtar & Mian, 2018), the author mentioned a variety of adversarial attack methods and defense methods against adversarial attacks. According to Figure 4, these defense methods can be divided into modified training/input, modified networks, and network add-on. According to the research findings of this paper (Yipeng Dong, 2019), which can improve the robustness of the model for defense. The robustness of the model can be improved by obfuscating the gradient. Although the obfuscated gradient method is not defensive against white box attacks, this method can well defend against black box attacks.

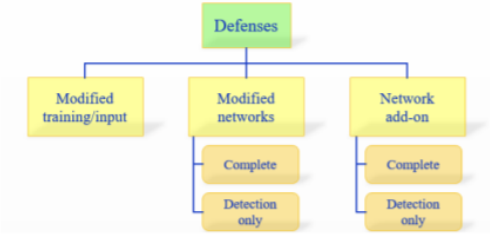

Figure 4

Therefore, our team can try to use the obfuscated gradient method for defense in future work to compare the difference between white box attacks and black box attacks.

5. Conclusion

To prevent the deep learning recognition of CAPTCHA, simply adding random noise to the CAPTCHA images are completely useless, and using FGSM model to process the CAPTCHA images has the best effect, but it also affects the recognition of human eyes. To use this method, we need to adjust the parameters to achieve a balance.

Although black-box attack is computationally expensive, is has a higher transferability to the other recognition models, means it can be more widely used. Moreover, it has a high attack efficiency, which has a small effects to human recognition when disturbing the recognition program.

6. Contribution

Xiaoran Ge - read the black box attack paper, implemented the code of the SimBA algorithm, and implemented the code of the two white box attacks in Cleverhans; wrote the part about black box attacks in the report and slide.

Ou Huiyi – modified and deployed adversarial attack production codes on Microsoft Azure to generate adversarial attack images; wrote the related work on white-box attack part in the report and presentation slides.

Xu Zhanxiang – searched for CAPTCHA dataset and designed and adopted CAPTCHA recognition program; trained 5 recogniton models, used these models to recognized the adversarial images generated by Huiyi and count the accuracies; wrote the introduction and conclusion part in the report and helped to make the presentation slides.

References

[1] T. I. Yang, C. S. Koong and C. C. Tseng, "Game-based Image Semantic CAPTCHA on Handset Devices," *Multimedia Tools and Applications,* vol. 74, pp. 5141-5156, 2015.

[2] Y. Y. Shen, S. F. Zhang and J. X. Gao, "A Security Enhancement Method of CAPTCHA based on Adversarial Samples," *Cyberspace Security,* vol. 11, no. 8, p. 15, 2020.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, "Intriguing Properties of Neural Networks," in *ICLR*, 2014.

[4] O. Bostik and J. Klecka, "Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms," *IFAC-PapersOnLine,* vol. 51, no. 6, pp. 208-213, 2018.

[5] K. M. He, X. Y. Zhang, S. Q. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *CVPR,* 2015.

[6] A. G. Howard, M. L. Zhu, B. Chen, D. Kalenichenko, W. J. Wang, T. Weyand, M. Andreetto and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *CVPR,* 2017.

[7] G. Huang, Z. Liu, L. v. d. Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," *CVPR,* 2018.

[8] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE,* vol. 6, pp. 14410-14430, 2018.

[9] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," *IEEE Symposium on Security and Privacy,* vol. 2375, no. 1207, pp. 39-57, 2017.

[10] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *ICLR,* 2015.

[11] A. Kurakin, I. Goodfellow and S. Bengio, "Adversarial Machine Learning at Scale," *ICLR,* 2017.

[12] T. Miyato, S.-i. Maeda, M. Koyama and S. Ishii, "Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2017.

[13] Y. P. Dong, F. Z. Liao, T. Y. Pang, H. Su, J. Zhu, X. L. Hu and J. G. Li, "Boosting Adversarial Attacks with Momentum," *CVPR,* 2018.

[14] P. Y. Chen, Y. Sharma, H. Zhang, J. F. Yi and C. J. Hsieh, "EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples," in *AAAI Conference on Artificial Intelligence,* 2018.

[15] F. Tramer and D. Boneh, "Adversarial Training and Robustness for Multiple Perturbations," *NeurIPS,* 2019.

[16] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, B. Celik and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in *In: Proceedings of the 2016 IEEE European Symposium on Security and Privacy,* Saarbrucken, Germany, 2016.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi and P. Frossard, "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition,* Las Vegas, NV, USA, 2016.

[18] X. C. C. L. D. S. Yanpei Liu, "D ELVING INTO T RANSFERABLE A DVERSARIAL E X AMPLES AND B LACK - BOX ATTACKS," *ICLR 2017*, 2017.

[19] J. R. G. Y. Y. Chuan Guo, ""Simple Black-box Adversarial Attacks"," *ICML*, 2019.

[20] A. M. N.Akhtar, ""Threat of Adversarial Attacks on Deep Learning in Computer Vision : A Survey"".*IEEE 2018*.

[21] Y. J. X. H. Jiangcheng Yang, ""Learning Black-Box Attacks with Transferable Priors and Query Feedback"," *NeurIPS*, 2020.

[22] T. P. H. S. J. Z. Yipeng Dong, ""Evading Defences to Transferable Adversarial Examples By Translation-Invariant Attacks"," *CVPR*, 2019.

A
Appendix

nosie eps=0.1	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	73.2%	72.1%	73.7%	73.6%	74.8%
Res18	95.6%	95.1%	95.8%	95.6%	96.2%
Res50	95.0%	95.9%	96.1%	95.3%	95.6%
MobileNet	91.7%	93.6%	93.0%	91.2%	92.6%
DenseNet161	94.5%	95.0%	94.8%	94.8%	96.2%

Table 1 - Classification accuracy on attacked image after adding noise, with eps=0.1

nosie eps=0.2	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	6.7%	5.3%	6.1%	6.7%	7.4%
Res18	87.9%	86.4%	85.0%	85.9%	86.7%
Res50	82.5%	83.7%	83.9%	82.3%	84.8%
MobileNet	53.1%	52.9%	50.7%	51.6%	52.3%
DenseNet161	80.3%	81.2%	84.2%	82.3%	79.2%

Table 2 - Classification accuracy on attacked image after adding noise, with eps=0.2

simba eps=0.1	CNN	Res18	Res50	MobileNet
CNN	30.0%	2.0%	2.0%	9.9%
Res18	66.0%	82.0%	67.7%	70.3%
Res50	66.0%	61.0%	78.8%	63.4%
MobileNet	53.0%	25.0%	25.3%	60.4%
DenseNet161	78.0%	67.0%	70.7%	75.2%

Table 3 - Classification accuracy on attacked image using simple black-box attack, with eps=0.1

simba eps=0.2	CNN	Res18	Res50	MobileNet
CNN	19.0%	0.0%	0.0%	0.0%
Res18	56.0%	36.0%	20.5%	29.0%
Res50	60.0%	12.0%	26.9%	23.0%
MobileNet	42.0%	0.0%	0.0%	16.0%
DenseNet161	64.0%	0.0%	0.0%	14.0%

Table 4 - Classification accuracy on attacked image using simple black-box attack, with eps=0.2

FGSM eps=0.1	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	0.0%	0.8%	0.9%	1.1%	1.4%
Res18	64.0%	1.1%	3.9%	36.2%	5.5%
Res50	58.4%	2.2%	1.7%	30.8%	3.8%
MobileNet	20.3%	3.1%	2.5%	0.3%	4.4%
DenseNet161	58.2%	1.6%	1.4%	26.3%	0.8%

Table 5 - Classification accuracy on attacked image using FGSM attack, with eps=0.1

FGSM eps=0.2	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	0.0%	0.0%	0.0%	0.0%	0.0%
Res18	0.5%	0.0%	0.0%	0.0%	0.0%
Res50	0.2%	0.0%	0.2%	0.0%	0.0%
MobileNet	0.0%	0.0%	0.0%	0.0%	0.0%
DenseNet161	0.0%	0.0%	0.0%	0.0%	0.0%

Table 6 - Classification accuracy on attacked image using FGSM attack, with eps=0.2

PGD eps=0.1	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	0.0%	12.1%	12.2%	30.2%	9.9%
Res18	80.8%	0.2%	2.0%	76.1%	2.7%
Res50	78.7%	4.1%	0.0%	73.2%	2.5%
MobileNet	55.4%	16.9%	12.5%	0.0%	11.4%
DenseNet161	80.4%	4.2%	2.0%	75.7%	0.0%

Table 7 - Classification accuracy on attacked image using PGD attack, with eps=0.1

PGD eps=0.2	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	0.0%	0.2%	0.0%	0.8%	0.0%
Res18	24.1%	0.0%	0.0%	52.0%	0.0%
Res50	16.6%	0.5%	0.0%	41.5%	0.0%
MobileNet	1.7%	0.3%	0.3%	0.0%	0.2%
DenseNet161	4.1%	0.5%	0.0%	35.8%	0.0%

Table 8 - Classification accuracy on attacked image using PGD attack, with eps=0.2

Sparse eps=20	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	3.4%	63.8%	62.9%	74.0%	63.1%
Res18	95.1%	13.5%	58.1%	94.4%	59.3%
Res50	95.3%	59.8%	5.8%	93.9%	55.2%
MobileNet	94.2%	72.0%	71.4%	1.7%	68.5%
DenseNet161	95.9%	69.6%	67.4%	95.3%	6.7%

Table 9 - Classification accuracy on attacked image using Sparse L1 Descent attack, with eps=0.1

Sparse eps=30	CNN	Res18	Res50	MobileNet	DenseNet161
CNN	0.6%	51.3%	49.6%	67.8%	51.0%
Res18	94.1%	7.7%	40.5%	92.2%	42.1%
Res50	94.2%	43.5%	3.0%	90.5%	41.2%
MobileNet	89.7%	59.9%	56.0%	0.3%	51.6%
DenseNet161	95.3%	53.4%	51.2%	93.6%	1.7%

Table 10 - Classification accuracy on attacked image using Sparse L1 Descent attack, with eps=0.2