StyleCAPTCHA: CAPTCHA Based on Stylized Images to Defend against Deep Networks

Haitian Chen
University of California, Davis
htichen@ucdavis.edu

Bai Jiang
Bytedance AI Lab
bai.jiang@bytedance.com

Hao Chen
University of California, Davis
chen@ucdavis.edu

ABSTRACT

CAPTCHAs are widely deployed for bot detection. Many CAPTCHAs are based on visual perception tasks such as text and objection classification. However, they are under serious threat from advanced visual perception technologies based on deep convolutional networks (DCNs). We propose a novel CAPTCHA, called StyleCAPTCHA, that asks a user to classify stylized human versus animal face images. StyleCAPTCHA creates each stylized image by combining the content representations of a human or animal face image and the style representations of a reference image. Both the original face image and the style reference image are hidden from the user. To defend against attacks using DCNs, the StyleCAPTCHA service changes the style regularly. To adapt to the new styles, the attacker has to repeatedly train or retrain her DCNs, but since the attacker has insufficient training examples, she cannot train her DCNs well. We also propose Classifier Cross-task Transferability to measure the transferability of a classifier from its original task to another task. This metric allows us to arrange the schedule of styles and to limit the transferability of attackers' DCNs across classification tasks using different styles. Our evaluation shows that StyleCAPTCHA defends against state-of-the-art face detectors and against general DCN classifiers effectively.

CCS CONCEPTS

• **Security and privacy** → **Graphical / visual passwords**; **Access control**.

KEYWORDS

CAPTCHA, neural style transfer, deep convolutional network, face recognition

ACM Reference Format:
Haitian Chen, Bai Jiang, and Hao Chen. 2020. StyleCAPTCHA: CAPTCHA Based on Stylized Images to Defend against Deep Networks. In *Proceedings of the 2020 ACM-IMS Foundations of Data Science Conference (FODS '20), October 19–20, 2020, Virtual Event, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3412815.3416895

1 INTRODUCTION

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are computer programs to test whether an online user is a bot. They are widespread security measures that protect online services against the abuse from automated programs. Typical CAPTCHAs are based on visual perception tasks, such as text recognition [39, 40], object recognition [8], and image classification [10].

However, the rapid evolution of artificial intelligence (AI) technologies made the traditional CAPTCHAs vulnerable, since well-trained AI models are capable of accurately completing many visual perception tasks. For instance, [4, 3] carried out a systemic study of the existing text-based CAPTCHAs and presented an generic algorithm for solving CAPTCHA by reinforcement learning. In two Kaggle competitions [23, 22], various deep convolutional networks (DCNs) [27, 12, 33, 16, 38] solved the dog versus cat image classification task in the Asirra CAPTCHA [10] at a accuracy comparable to humans. [35] developed a deep convolutional network to solve the Google reCAPTCHA at an accuracy of 70.78% and the Facebook image CAPTCHA at an accuracy of 83.5%.

It might be tempting to use an even harder visual perception task as CAPTCHA, but this will last only until an AI solution to this CAPTCHA is invented [39, 31]. Moreover, increasing the difficulty of CAPTCHAs will probably worsen its the usability. Even current CAPTCHAs involving moderately hard text or object recognition tasks were criticized by users because they were hard to recognize [41].

Instead, we wish to exploit intrinsic differences between humans and DCNs on visual perception tasks. DCNs are not robust against certain image perturbations that are invariant to human perception. For instance, DCN classifiers may wrongly classify adversarially perturbed images, although these perturbations are invariant to human visual judgment [37, 14, 1]. Recently, DCNs were shwon to misunderstood stylized images by a neural style transfer procedure [7], while humans did not [11].

In addition, to achieve high accuracy on a visual perception task, a DCN requires a vast set of annotated or labelled images during training. In contrast, humans needs no training to complete CAPTCHAs. Figure 1 illustrates a typical learning curve of an AI model on a visual perception task. Although humans can hardly compete with AIs in AIs' well-trained (later) phase, humans outperform AIs in AIs' under-trained (early) phase.

We propose a novel CAPTCHA scheme called StyleCAPTCHA to exploit the above-mentioned intrinsic weaknesses of current DCNs. Specifically, our system uses neural style transfer (NST) [11] to blend face images with style reference images. For each face image and style reference, NST generates an image of the face "painted" in the style. In each CAPTCHA challenge, the user is asked to classify 10 stylized images into either human face or animal face category. Figure 2 illustrates this StyleCAPTCHA scheme. The system changes the style every few CAPTCHA challenges to keep

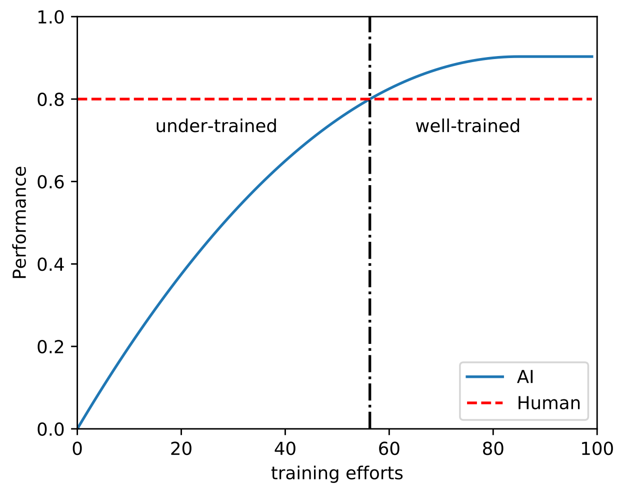


Figure 1: Learning curve of AI

the attacker's DCN classifier in the under-trained state (because the classifier has not received adequate training examples).

The advantages of StyleCAPTCHA to defend against DCN-based automated attacks are twofold. First, the NST transformation perturbs pixel values of human face images and decreases the accuracy of the publicly available, well-trained DCN face detectors [43, 5, 17, 44, 9]. This idea shares the same spirit with adversarially perturbing the pixel values of images to mislead DCN classifiers [37, 14, 6]. To the best of our knowledge, how to robustify DCNs against the perturbation induced by NST is little known in the literature. By contrast, after the texture and color of images are destructively transferred, human users can easily distinguish stylized human images from stylized animal images, because a stylized image still contains the content information of the facial appearance of its original image, and humans are sensitive to facial appearance.

More importantly, to adapt to frequently changing styles, the attacker has to keep collecting a vast set of stylized images from the the StyleCAPTCHA service, manually labeling them, and then training or re-training her DCN classifier. If StyleCAPTCHA switches to a new style before the attacker's DCN enters the well-trained phase, then the attacker's DCN is kept in the under-trained phase and hence cannot pass the CAPTCHA challenge consistently. By contrast, human users can correctly classify stylized images with no training or re-training, because perceiving human facial appearance is a lifetime practice for identity recognition and verification.

We also propose a metric called Classifier Cross-task Transferability (CCT) to measure the ability of a classifier transferring from one image classification task, featured by one style (or a set of styles), to another image classification task, featured by another style. CCT allows us to arrange the style schedule (the sequence of styles) adversarially against DCN-based potential attackers. After the system generates a few StyleCAPTCHA challenges, it switches to another style with low CCT from the previous style(s). By doing so, we limit the impact of the attacker's DCN classifier transfering from one image classification task to the next one, and protect StyleCAPTCHA from DCNs retrained by transfer learning.

Apart from DCNs for general purpose, DCNs trained for the specific task of human face detection can also be deployed to attack StyleCAPTCHA. A few accurate and efficient DCN-based face detectors have been developed recently [43, 5, 17, 44, 9]. For example, the-state-of-art face detector RetinaFace[9] (with Resnet50 [16] as the backbone network) reaches an average precision 90% on the hard-level validation set of the WIDER Face dataset [42], and even higher accuracy on the LFW dataset [18]. We will evaluate StyleCAPTCHA against RetinaFace detector in our experiments.

2 METHODS

This section presents the methodology of StyleCAPTCHA , which involves two key techniques. Section 2.1 briefly describes the neural style transfer (NST) technique, which transforms original human and animal faces to stylized images. Section 2.1 defines Classifier Cross-task Transferability (CCT) and proposes a practical estimation method for it. CCT will be used to arrange the style schedule. Section 2.3 presents algorithms of StyleCAPTCHA in action.

2.1 Neural Style Transfer

Neural style transfer (NST) blends a content image, and a style image together so that the output image looks like the content image, but is "painted" in the style of the style image [11, 21, 19, 29]. By using deep convolutional networks, NST extracts the content representations and the style representations from images and optimizes the output image to match the content representations of the content image and the style representations of the style reference image. More precisely, the content and style representations are defined as intermediate feature maps of a pretrained image classification network and the correlations among these feature maps, respectively. At a high level, a network has built internal representations that convert the raw image pixels into a complex understanding of the image, when being trained to perform a image classification task, and thus can serve as a complex feature extractor. An NST network is trained to minimize the content loss, which measures the difference between content representations of the original content image and the output image, plus the style loss, which measures the difference between style representations of the style reference image and the output image.

Let $\vec{F}^l_{ij}(x)$ be the K_l-dimensional feature vector at location $1 \le i \le N_l$, $1 \le j \le M_l$ of an intermediate layer l of the pretrain image classification network for image x. The content loss between the output image x and the content image c is then given by

$$\mathcal{L}_{\text{content}}(x, c) = \sum_{i,j,l} \|\vec{F}^l_{ij}(x) - \vec{F}^l_{ij}(c)\|^2.$$

The style representations are calculated by taking the outer product of the feature vector with itself at each location, and averaging that outer product over all locations

$$\mathrm{G}_l(x) = \frac{1}{N_l M_l} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} \vec{F}^l_{ij}(x) \left[\vec{F}^l_{ij}(x),\right]^{\mathrm{T}}$$

which is a $K_l \times K_l$ covariance matrix. The style loss between the output image x and the style image s is given by

$$\mathcal{L}_{\text{style}}(x, s) = \sum_{l} w_l \|\mathrm{G}_l(x) - \mathrm{G}_l(s)\|^2,$$

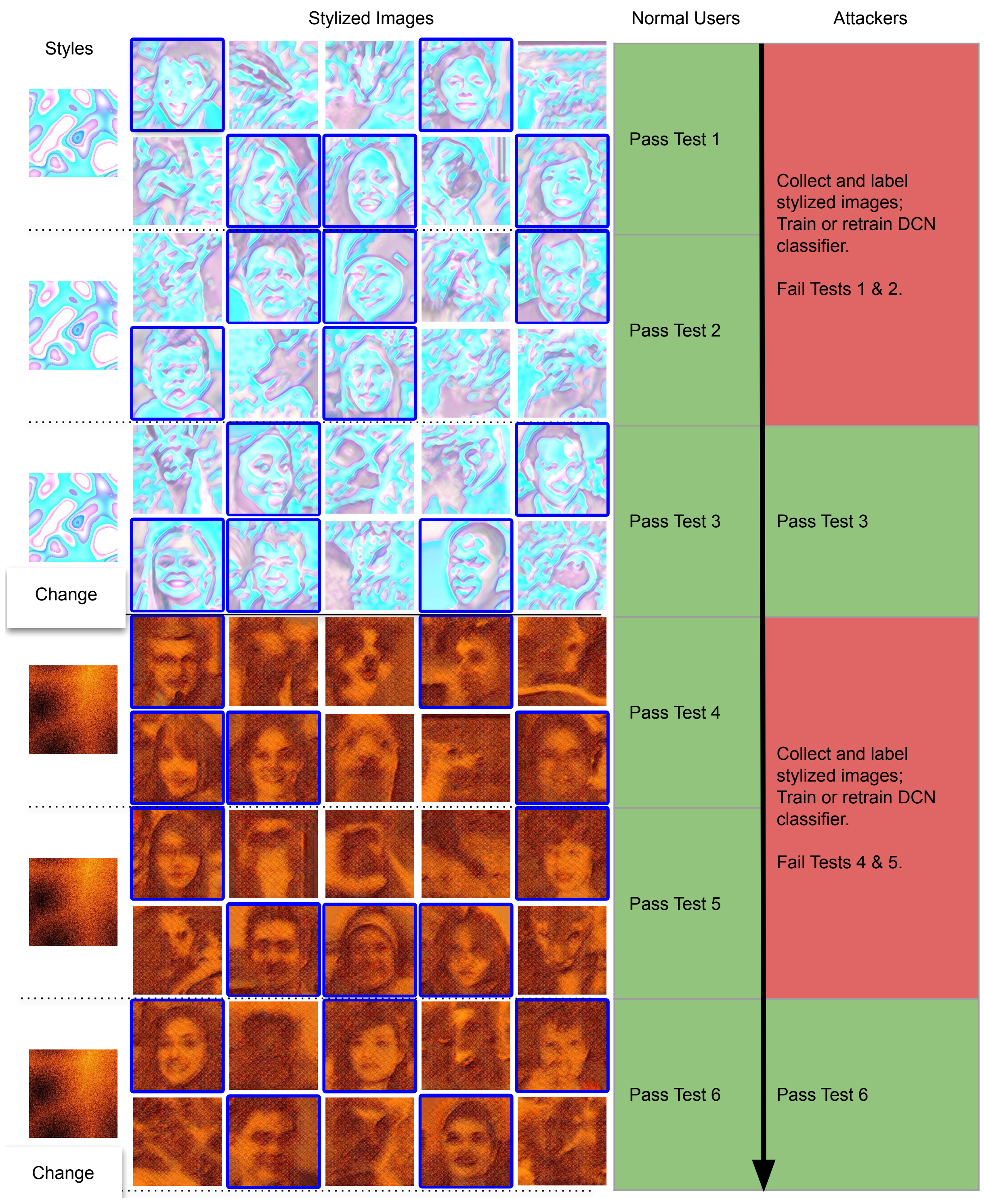


Figure 2: StyleCAPTCHA scheme. Each test consists of 10 stylized images. The blue boxes show human faces for the convenience of readers but are absent in real CAPTCHA challenges.

where w_l's are weights to combine style losses from different layers, $\|A\|$ is the Frobenius norm of matrix A, i.e., the ℓ_2-norm of the vectorization of A (the concatenation of column vectors of A). The total loss function of the NST network is then given by

$$\mathcal{L}(x, c, s) = \lambda_c \mathcal{L}_{\text{content}}(x, c) + \lambda_s \mathcal{L}_{\text{style}}(x, s), \quad (1)$$

where λ_c and λ_s are the weights to combine the content and style losses.

We remark that the content and style loss weights λ_c and λ_s control the portion of the content information of the output image x from the original image c and the portion of the style information from the style image s. The quality of the output image varies according to the choice of these weights.

In this paper, we adopt the "fast" NST method by [21], which costs a short training time per style reference image and thus meets our need to frequently change styles of images. Other NST methods such as MUNIT [20] and StyleGAN [24, 25] might also be utilized to create stylized images for StyleCAPTCHA.

2.2 Classifier Cross-task Transferability

Classifier Cross-task Transferability (CCT) is intended to measure the transferability of a classifier from its own classification task to another task. This measure plays a key role in arranging the style schedule (the sequence of styles) for StyleCAPTCHA. This section gives its formal definition and provides a practical method to estimate it. Technical proofs of theoretical results concerning the properties of CCT are collected in Section 3.

2.2.1 Definition. Let $\mathcal{X}_0 = \{X_{0i}\}_{i=1}^n$ and $\mathcal{X}_1 = \{X_{1i}\}_{i=1}^n$ denote animal and human face images, respectively, which are stylized by one or a set of multiple style(s). Let $\mathcal{Y}_0 = \{Y_{0i}\}_{i=1}^n$ and $\mathcal{Y}_1 = \{Y_{1i}\}_{i=1}^n$ denote animal and human images which are stylized by another style.

Let $f_0(x)$ and $f_1(x)$ denote probability density functions of samples of $\mathcal{X}_0$ and $\mathcal{X}_1$. Theoretical results from recent advances of generative modeling [13] suggest that the optimal (soft) binary classifier to distinguish animal images from human images in the sense of the minimum binary cross entropy loss is given by

$$D(x) = \frac{f_1(x)}{f_0(x) + f_1(x)}. \quad (2)$$

Let Y be an unlabelled image in either $\mathcal{Y}_0$ or $\mathcal{Y}_1$, and let $C = 0$ or 1 be its label. Define CCT from the classification task $\mathcal{X}_0$ versus $\mathcal{X}_1$ to the other classification task $\mathcal{Y}_0$ versus $\mathcal{Y}_1$ as 1 minus the conditional entropy of C given $D(Y)$. Formally,

$$\text{CCT}(\mathcal{X}_0, \mathcal{X}_1; \mathcal{Y}_0, \mathcal{Y}_1) = 1 - H(C|D(Y)),$$

where H is the entropy[1] notation. If $D(Y)$ deterministically tells C then $H(C|D(Y)) = 0$ and CCT = 1. If C is independent from $D(Y)$ then $H(C|D(Y)) = H(C) = 1$ and thus CCT = 0.

Next theorem shows that CCT can be alternatively defined as the Jensen-Shannon (JS-) divergence between the distributions of $\{D(Y_{0i})\}_{i=1}^n$ and $\{D(Y_{1i})\}_{i=1}^n$.

[1] Information-theoretic quantities in this paper including (conditional) entropy, Jensen-Shannon divergence, Kullback-Leibler divergence are measured in bits

Theorem 1. *Let $h_0(z)$ and $h_1(z)$ be the distributions of $Z_{0i} = D(Y_{0i})$ and $Z_{1i} = D(Y_{1i})$ for $i = 1, \ldots, n$ and JS stand for the JS-Divergence. Then*

$$\text{CCT}(\mathcal{X}_0, \mathcal{X}_1; \mathcal{Y}_0, \mathcal{Y}_1) = \text{JS}(h_0 \| h_1). \quad (3)$$

2.2.2 Estimation. With Theorem 1 in hand, we proceed to consider estimating $\text{JS}(h_0\|h_1)$. Given random samples $\mathcal{Z}_0 = \{Z_{0i}\}_{i=1}^n$ and $\mathcal{Z}_1 = \{Z_{1i}\}_{i=1}^n$ following distribution $h_0(z)$ and $h_1(z)$, respectively, the JS-divergence between two distributions can be estimated as follows.

Sort random samples in $\mathcal{Z}_0$, $\mathcal{Z}_1$ and $\mathcal{Z}_0 \cup \mathcal{Z}_1$ in the non-decreasing order as $\{Z_{0(i)}\}_{i=1}^n$, $\{Z_{1(i)}\}_{i=1}^n$ and $\{Z_{(i)}\}_{i=1}^{2n}$. Let $\widetilde{h}_0(z)$ be the piece-wise linear interpolation passing through points

$$(0,0), \left(Z_{0(1)}, \frac{1}{2n}\right), \left(Z_{0(2)}, \frac{3}{2n}\right), \ldots, \left(Z_{0(n)}, \frac{2n-1}{2n}\right), (1,1);$$

let $\widetilde{h}_1(z)$ be the piece-wise linear interpolation passing through points

$$(0,0), \left(Z_{1(1)}, \frac{1}{2n}\right), \left(Z_{1(2)}, \frac{3}{2n}\right), \ldots, \left(Z_{1(n)}, \frac{2n-1}{2n}\right), (1,1);$$

and let $\widetilde{h}(z)$ be the piece-wise linear interpolation passing through points

$$(0,0), \left(Z_{(1)}, \frac{1}{4n}\right), \left(Z_{(2)}, \frac{3}{4n}\right), \ldots, \left(Z_{(2n)}, \frac{4n-1}{4n}\right), (1,1).$$

$\widetilde{h}_0(z)$, $\widetilde{h}_1(z)$ and $\widetilde{h}(z)$ are actually the continuous piece-wise linear approximations of the empirical cumulative distribution functions of $\mathcal{Z}_0$, $\mathcal{Z}_1$ and $\mathcal{Z}_0 \cup \mathcal{Z}_1$, respectively.

Let $Z_{(0)} = 0$, $Z_{(2n+1)} = 1$, and $\epsilon = \min_{i=0}^{2n}\left[Z_{(i+1)} - Z_{(i)}\right]/2$. An estimator for $\text{JS}(h_0\|h_1)$ is given by

$$\widehat{\text{JS}}\left(\mathcal{Z}_0\|\mathcal{Z}_1\right) = \frac{1}{2n}\sum_{i=1}^n \frac{\widetilde{h}_0(Z_{0(i)}) - \widetilde{h}_0(Z_{0(i)} - \epsilon)}{\widetilde{h}(Z_{0(i)}) - \widetilde{h}(Z_{0(i)} - \epsilon)} + \frac{1}{2n}\sum_{i=1}^n \frac{\widetilde{h}_1(Z_{1(i)}) - \widetilde{h}_1(Z_{1(i)} - \epsilon)}{\widetilde{h}(Z_{1(i)}) - \widetilde{h}(Z_{1(i)} - \epsilon)} - 1 \quad (4)$$

Theorem 2. *The above-defined JS-divergence estimator is strongly consistent, i.e.,*

$$\widehat{\text{JS}}\left(\mathcal{Z}_0\|\mathcal{Z}_1\right) \to \text{JS}(h_0\|h_1)$$

almost surely as $n \to \infty$.

Theorem 1 and Theorem 2 together inspire the following algorithm to estimate CCT.

Algorithm 1 CCT estimation in action

```
Input: $\mathcal{X}_0, \mathcal{X}_1, \mathcal{Y}_0, \mathcal{Y}_1$ defined as above
Output: $\widehat{\text{CCT}}(\mathcal{X}_0, \mathcal{X}_1; \mathcal{Y}_0, \mathcal{Y}_1)$, a CCT estimate.
For the classification task $\mathcal{X}_0$ versus $\mathcal{X}_1$, train a DCN classifier $\widehat{D} : x \mapsto [0,1]$ to minimize the binary cross-entropy loss.
Let $\mathcal{Z}_0 = \{\widehat{D}(y) : y \in \mathcal{Y}_0\}$ and $\mathcal{Z}_1 = \{\widehat{D}(y) : y \in \mathcal{Y}_1\}$.
Compute $\widehat{\text{JS}}(\mathcal{Z}_0\|\mathcal{Z}_1)$ using (4).
```

CCT measures the transferability of the optimal binary classifier $D : x \mapsto [0,1]$ from its original classification task $\mathcal{X}_0$ versus $\mathcal{X}_1$ to another classification task $\mathcal{Y}_0$ versus $\mathcal{Y}_1$. In practice, the optimal binary classifier $D : x \mapsto [0, 1]$ is unknown. But we could approximate it well by training a sufficiently complex DCN (e.g., ResNet50 [16]) classifier. Similar approaches have accurately estimate other information-theoretically quantities on high-dimensional image data [2].

2.3 StyleCAPCTHA

The StyleCAPTCHA scheme is outlined as Algorithm 2. In this algorithm, $s \otimes \mathcal{I}$ denotes the set of stylized images, resulting from blending content images in $\mathcal{I}$ and a style reference image s by the NST method described in Section 2.1. $\widehat{\text{CCT}}$ denotes the estimate of CCT given by Algorithm 1 in Section 2.2.

Algorithm 2 StyleCAPTCHA

```
  Input: animal face image set $\mathcal{I}_0$, human face image set $\mathcal{I}_1$, style
  reference image set $\mathcal{S}$, a threshold constant $\delta \in (0, 1)$.
  Initialize $s_0 \in \mathcal{S}$.
  for $t = 0, 1, 2, \dots$ do
      Provide a few times of CAPTCHA service to users, who are
      asked to classify $s_T \otimes \mathcal{I}_0$ versus $s_T \otimes \mathcal{I}_0$.
      Find style $s \in \mathcal{S}$ such that
          $\widehat{\text{CCT}}(\cup_{i \le t} s_i \otimes \mathcal{I}_0, \cup_{i \le t} s_i \otimes \mathcal{I}_1; s \otimes \mathcal{I}_0, s \otimes \mathcal{I}_1) < \delta$.
      Set $s_{t+1} = s$.
  end for
```

3 PROOFS

This section collects the technical proofs for Theorem 1 and Theorem 2 in Section 2.

Proof of Theorem 1. For an unlabelled image $Y \in \mathcal{Y}_0 \cup \mathcal{Y}_1$, let $Z = D(Y)$ be the prediction given by the optimal classifier D for Y. Apparently, the random variable Z follows the mixture distribution

$$h(z) = \frac{h_0(z) + h_1(z)}{2}.$$

Let $p(c, z)$ be the joint distribution of C and $Z = D(Y)$, and write marginal distributions of c as $p(c = 0) = \int p(0, z)dz$ and $p(c = 1) = \int p(1, z)dz$. Write

$$\begin{aligned}
&H(C, D(Y)) \\
&= -\sum_{c \in \{0,1\}} \int p(c, z) \log p(c, z)\, \mathrm{d}z \\
&= -\int p(z|c = 0)p(c = 0) \log[p(z|c = 0)p(c = 0)]\, \mathrm{d}z \\
&\quad - \int p(z|c = 1)p(c = 1) \log[(p(z|c = 1)p(c = 1)]\, \mathrm{d}z \\
&= -\int \frac{h_0(z)}{2} \log\left[\frac{h_0(z)}{2}\right] \mathrm{d}z - \int \frac{h_1(z)}{2} \log\left[\frac{h_1(z)}{2}\right] \mathrm{d}z
\end{aligned}$$

and

$$\begin{aligned}
H(D(Y)) &= -\int h(z) \log h(z)\, \mathrm{d}z \\
&= -\int \frac{h_0(z)}{2} \log h(z)\, \mathrm{d}z - \int \frac{h_1(z)}{2} \log h(z)\, \mathrm{d}z.
\end{aligned}$$

Then, CCT amounts to

$$\begin{aligned}
&1 - H(C|D(Y)) = 1 - H(C, D(Y)) + H(D(Y)) \\
&= 1 - \int \frac{h_0(z)}{2} \log\left[\frac{h_0(z)/2}{h(z)}\right] \mathrm{d}z - \int \frac{h_1(z)}{2} \log\left[\frac{h_1(z)/2}{h(z)}\right] \mathrm{d}z \\
&= -\frac{1}{2} \int h_0(z) \log\left[\frac{h_0(z)}{h(z)}\right] \mathrm{d}z - \frac{1}{2} \int h_1(z) \log\left[\frac{h_1(z)}{h(z)}\right] \mathrm{d}z \\
&= \frac{1}{2} \mathrm{KL}(h_0 \| h) + \frac{1}{2} \mathrm{KL}(h_1 \| h) = \mathrm{JS}(h_0 \| h_1)
\end{aligned}$$

□

Proof of Theorem 2. Break the JS-divergence $\mathrm{JS}(h_0 \| h_1)$ into two KL-divergence terms

$$\mathrm{JS}(h_0 \| h_1) = \frac{1}{2} \mathrm{KL}\left(h_0 \middle\| \frac{h_0 + h_1}{2}\right) + \frac{1}{2} \mathrm{KL}\left(h_1 \middle\| \frac{h_0 + h_1}{2}\right).$$

By [32, Thoerem 1],

$$\frac{1}{n} \sum_{i=1}^{n} \frac{\widetilde{h}_0(Z_{0(i)}) - \widetilde{h}_0(Z_{0(i)} - \epsilon)}{\widetilde{h}(Z_{0(i)}) - \widetilde{h}(Z^0_{(i)} - \epsilon)} - 1 \to \mathrm{KL}\left(h_0 \middle\| \frac{h_0 + h_1}{2}\right)$$

$$\frac{1}{n} \sum_{i=1}^{n} \frac{\widetilde{h}_1(Z_{1(i)}) - \widetilde{h}_1(Z_{1(i)} - \epsilon)}{\widetilde{h}(Z_{1(i)}) - \widetilde{h}(Z_{1(i)} - \epsilon)} - 1 \to \mathrm{KL}\left(h_1 \middle\| \frac{h_0 + h_1}{2}\right)$$

almost surely as $n \longrightarrow \infty$. Collecting these pieces together completes the proof. □

4 EXPERIMENTAL DESIGN

4.1 Face Images

We detect and extract faces from the Facial Deformable Models of Animals (FDMA) dataset [26] and the Flickr-Faces-HQ (FFHQ) dataset [24] by the state-of-the-art RetinaFace human face detector [9]. The FDMA dataset contains around 21.9K animal face images belonging to 334 diverse species. Among them, 10000 animal faces that look like human faces (according to the judgement of the RetinaFace detector) are selected. The FFHQ dataset contains around 22.0K human face images. Among them, 10000 human faces are randomly selected. These two sets of 10000 animal faces and 10000 human faces are used in our experiments.

4.2 Styles

Figure 3 visualizes the set of 48 styles used in the experiments on StyleCAPTCHA. These styles are generated by an open-sourced style (texture and color) generator named texgen.js (https://github.com/mrdoob/texgen.js).

4.3 Choice of Loss Weights λ_c and λ_s in NST

For each style, we randomly pick up 2500 human faces and 2500 animal faces from the image sets to train a fast NST network, and then apply the fast NST network to stylize the rest of the image sets. The choices of content and style loss weights λ_c and λ_s are crucial for the performance of the NST network. Smaller λ_c and larger λ_s would let the NST network focus more on the objective of matching the style representations and make less account for the content information, and vice versa. If the stylized face image losses too much content information of the original face, both the human visual perception and DCN-based human face detector would not

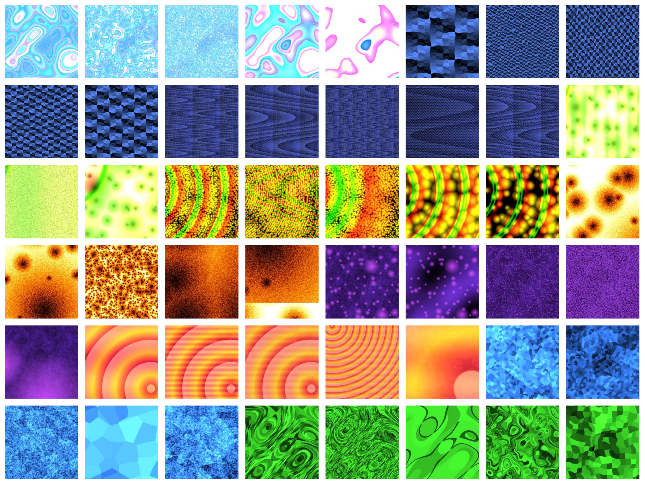


Figure 3: Styles created by by texgen.js.

distinguish human faces apart from animal faces. On the other hand, if the stylized face image contains too much content information, both the human visual perception and DCN-based human face detector would correctly recognize the human faces.

We choose the values of λ_c and λ_s in the following way. Fix $\lambda_s = 10^5$ and gradually decrease λ_c from 3.0 to 0.0 until the average probability score given by the state-of-the-art RetinaFace detector for stylized human faces drops below the default threshold 0.7 of the RetinaFace probability score.

4.4 Style Schedules

To evaluate the reliability of StyleCAPCTHA and the effectiveness of the CCT-based arrangement of the style schedule $s_0, s_1, s_2, \cdots$, we compare the following three arrangements of the style schedule.

Unchanging Schedule (US): Use an unchanging style to transfer human and animal face images, i.e., $s_t = s_0$ for $t \geq 1$.

Random Schedule (RS): Randomly pick up a style s_t from the set of styles at every time t.

CCT-based Schedule (CS): As described by Algorithm 2, pick up style s_t from the set of styles such that CCT from the classification task featured by $s_0, s_1, \cdots, s_{t-1}$ to that featured by s_t is below some threshold δ.

The threshold δ for CCT values in the StyleCAPTCHA scheme (Algorithm 2) shall be determined by the trade-off between the reliability of the StyleCAPTCHA service and the flexibility of the style schedule arrangement. A low CCT value indicates that the target classification task featured by the new style obtains little information from the source classification task. Consequently, the DCN classifier, that is well trained for the source task, would perform poorly on the target task. However, an extremely low CCT threshold δ would filter out many candidate styles and limit the flexibility of the style schedule arrangement. We choose $\delta = 0.75$ by a pilot experiment (see Figure 5).

4.5 Threat Models

We assume three state-of-the-art DCN architectures for general visual classification tasks, namely, VCG19 [34], ResNet50 [16] and InceptionV3 [36], and a state-of-the-art DCN-based face detector RetinaFace [9], are available to the attackers. Below are four threat models that the attackers may maliciously use to crack down StyleCAPCTHA.

Baseline (BL) model: The attacker collects 1000 images of style s_0 via repeatedly requesting the StyleCAPTCHA service, manually label images, and then train one of VCG19, ResNet50 or InceptionV3 networks. The network is initialized with the well-optimized parameters on general large-scale image recognition task.

Transfer learning (TL) model: The attacker trains an initial DCN classifier in the same way as in the baseline model. After a style change from s_{t-1} to s_t happens at time $t \geq 1$, the attacker collects another 1000 images of style s_t, and then uses them to re-train the current DCN classifier by transfer learning techniques.

Cumulative learning (CL) model: The attacker trains an initial DCN classifier in the same way as in the baseline model. After a style change from s_{t-1} to s_t happens at time $t \geq 1$, the attacker collects another 1000 images of style s_t, and then uses all images that has been obtained from the StyleCAPTCHA service at that moment to train a DCN classifier.

Face detection model: The attacker uses the state-of-the-art DCN-based face detector RetinaFace to detect human faces in stylized images. RetinaFace detector gives a probability score between 0.0 and 1.0 for each detected object measuring the likelihood of the object being a human face. The attacker chooses the optimal probability score threshold to maximize the accuracy of his classifier.

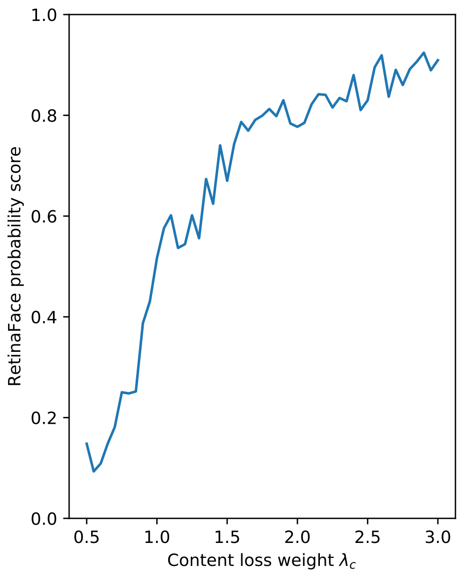
5 EXPERIMENTAL RESULTS

This section presents results of two pilot experiments for hyperparameter selections, and performances of three arrangement strategies of the style schedule defending against threat models. The performance of the DCN-based classifiers on the stylized image classification tasks are evaluated in terms of average classification accuracy. Note that each StyleCAPTCHA test asks a user to classify 10 images (Figure 2). A DCN-based classifier with accuracy 80% results in a passing rate $0.80^{10} = 10.74\%$. That means, this classifier will pass through StyleCAPTCHA once per 9.31 times of attempts on average. We expect that a normal user achieves an accuracy 95% with a passing rate $0.95^{10} = 59.87\%$ correspondingly and passes through StyleCAPTCHA once per 1.67 attempts on average.

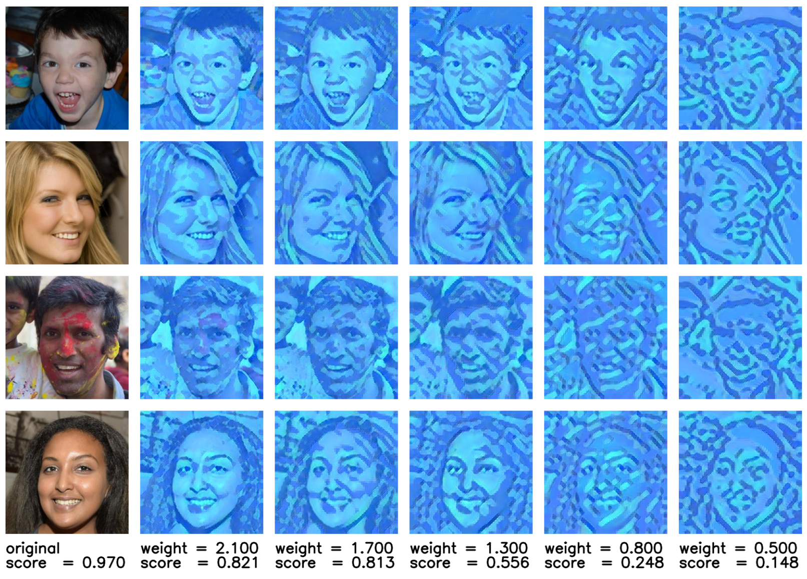
5.1 Pilot Experiment to Choose λ_c for NST

A pilot experiment is conducted to guide the choice of content loss weight λ_c in the NST network such that the RetinaFace detector would miss a considerately large portion of stylized human faces as false negatives, meanwhile a normal user would not.

For a single style, various values of content loss weight λ_c are used in the NST network to create multiple sets of stylized human face images. The average probability scores of the RetinaFace on each image set are computed. Figure 4a shows that the Retina probability scores for stylized images and content loss weight λ_c are strongly correlated. In this situation, we choose $\lambda_c = 1.3$ such that the average probability score of detected faces is below 0.7, which is the default threshold of the probability score set by the official implementation of RetinaFace. Weights for other styles are chosen in the same way.



(a)


original score = 0.970
weight = 2.100 score = 0.821
weight = 1.700 score = 0.813
weight = 1.300 score = 0.556
weight = 0.800 score = 0.248
weight = 0.500 score = 0.148

(b)

Figure 4: Choice of content loss weight λ_c.

Figure 4b showcases 20 stylized face images with content loss weight $\lambda_c = 2.1, 1.8, 1.3, 0.8, 0.5$. Evidently, as the stylized human face images keep less content information with smaller λ_c, the RetinaFace detector gives less certain judgements, while human visual perception could still recognize the stylized images. The choice of $\lambda_c = 0.8$ lowers the RetinaFace probability score, but the resulting stylized images may confuse some normal users.

5.2 Pilot Experiment to Choose δ for CCT

In order to choose a suitable threshold δ for CCT, we estimate CCT of each classifier transferring from the source task featured by style s to the target task featured by another style $s' \neq s$. Results show that 2184 out of $48 \times 47 = 2256$ pairs of styles satisfy

$$\widehat{\mathrm{CCT}}(s \otimes \mathcal{I}_0, s \otimes \mathcal{I}_1; s' \otimes \mathcal{I}_0, s' \otimes \mathcal{I}_1) < 0.75.$$

All of these 2184 obtain < 0.82 classification accuracy on the target task (Figure 5), thus $\delta = 0.75$ is set in further experiments.

5.3 Defense against General DCNs

We evaluate the performance of three style schedules defending StyleCAPTCHA against the three threat models with one of VGG19, InceptionV3, Resnet50 backbone networks. Each threat model cracks down the unchanging style schedule with a classification accuracy $> 90\%$.

Figure 6 plots the classification accuracy of the baseline (BL), transfer learning (TL) and cumulative learning (CL) threat models with a DCN backbone network to attack the StyleCAPTCHA service when the sequence of styles $s_0, s_1, \ldots, s_9$ are scheduled by random or according the guidance of CCT. Figure 6a shows that the random schedule (RS) strategy can defend both the baseline (BL) and transfer learning (TL) attacks, but fail to defend the cumulative learning (CL) attacks during a few time periods.

In contrast, Figure 6b shows that the CCT-guided schedule (CS) can defend three types of attacks. Each of VGG19, InceptionV3 and

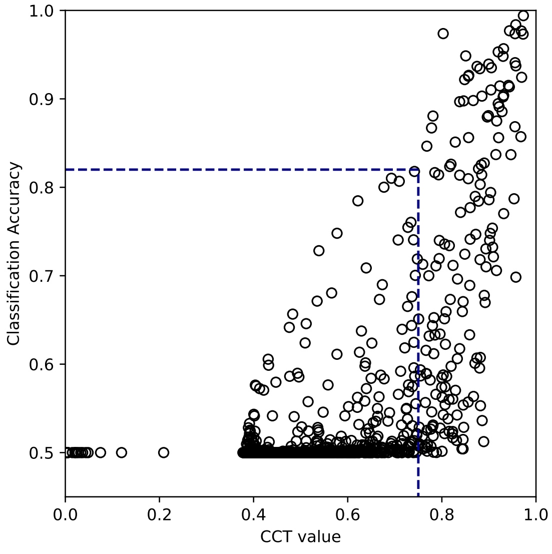
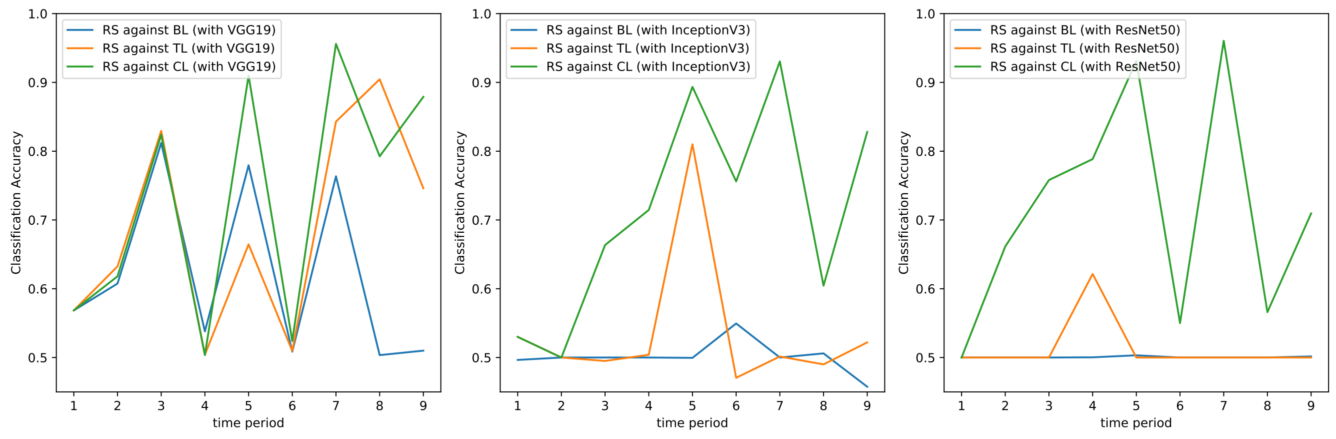


Figure 5: CCT value and classification accuracy of Resnet50 classifiers transferring from the source task featured by one style to the target task featured by another style.

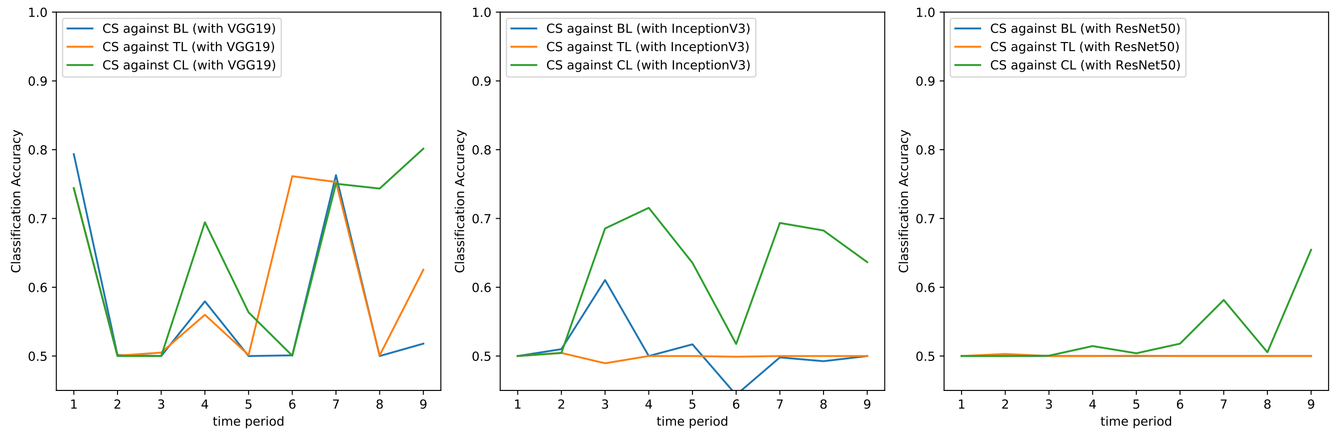
Resnet50 classifiers obtains less than 80% accuracy during each of 10 time periods on the stylized images in the StyleCAPTCHA service. Since each StyleCAPTCHA test asks a user to classify 10 images, these classifiers pass through the StyleCAPTCHA test with chance at most $0.80^{10} = 10.74\%$. They would take more than 9.31 attempts to pass through StyleCAPTCHA once on average.

5.4 Defense against RetinaFace Detector

The RetinaFace detector gives each stylized image a probability score indicating the chance that it is a human face. To convert the



(a) RS defending against threat models with VGG19 (left), InceptionV3 (middle) and Resnet50 (right) backbones.



(b) CS defending against threat models with VGG19 (left), InceptionV3 (middle) and Resnet50 (right) backbones.

Figure 6: Random style schedule (RS) and CCT-based style schedule (CS) defending against the baseline (BL), transfer learning (TL) and cumulative learning (CL) threat models with DCN backbone.

probability score output to a classification rule, the attacker needs to set a probability score threshold. In practice, the threshold is selected by the ROC curve. We assume both the ROC curve and the optimal threshold are available to the attackers. Figure 7 illustrates the ROC curves of the classifier converted from the RetinaFace detector on stylized images the attackers collect during ten time periods in Figure 6. The maximum classification accuracy of the classifier with the optimal score threshold is 84.6% against the CCT-guided style schedule and 89.6% against the random style schedule. Correspondingly, the classifier would pass through StyleCAPTCHA guarded by two style schedules with rates 18.8% and 33.3%.

6 RELATED WORK

Image-based CAPTCHA with style transfer. Cheng et al.[7] proposed a CAPTCHA that asked users to select one among nine stylized images to match a short scene description. A key difference is that their CAPTCHA used the *same* style to transfer all their images, whereas our StyleCAPTCHA schedules different *adversarial* styles to transfer images. Another CAPTCHA asked users to detect human faces with rotations and occlusions [15].

Improving the robustness of classifiers. Szegedy et al. proposed a defense against adversarial examples by augmenting the training set with adversarial images [37]. Liu et al. improved the robustness of classifiers by removing outliers from the training set [28]. However, [30, 1] showed that retrained models could still be attacked by other adversarial examples.

Style transfer with neural networks Our StyleCAPTCHA uses neural style transfer [11]. Other style transfer techniques include StyleGAN[24, 25] and MUNIT[20], which use cycle consistent loss and AdaIN block[19]. We plan to evaluate the efficiency of those style transfer models in StyleCAPTCHA model in the future.

7 CONCLUSION

Deep convolutional networks (DCNs) are serious threats to traditional CAPTCHAs based on visual perception tasks nowadays. We

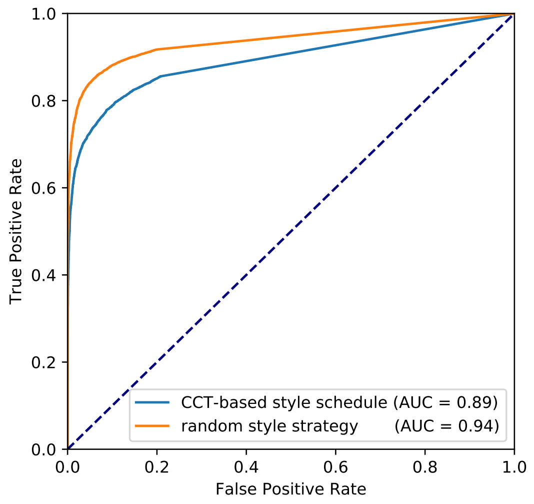


Figure 7: StyleCAPTCHA against the classifier converted from the RetinaFace detector.

design StyleCAPTCHA by exploiting two intrinsic differences between current deep visual networks and human visual perception. First, we use the neural style transfer technique to generate stylized images, which are shown to be robust beyond the attacks from state-of-the-art DCNs. Second, we propose to adversarilly change the style of images in StyleCAPTCHA according to an information-theoretical measure for the classifier cross-task transferrability (CCT). Our experimental evaluation shows that StyleCAPTCHA is general for a range of common image classifiers.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1801751. This research was partially sponsored by the Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

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

[2] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *International Conference on Machine Learning*, 531–540.

[3] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C Mitchell. 2014. The end is nigh: generic solving of text-based CAPTCHAs. In *the 8th USENIX Workshop on Offensive Technologies*.

[4] Elie Bursztein, Matthieu Martin, and John C Mitchell. 2011. Text-based captcha strengths and weaknesses. In *ACM Conference on Computer and Communications Security*, 125–138.

[5] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. 2016. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, 354–370.

[6] Jiyu Chen, David Wang, and Hao Chen. 2020. Explore the transformation space for adversarial images. In *ACM Conference on Data and Application Security and Privacy (CODASPY)*. New Orleans, LA, USA, (March 16–18, 2020).

[7] Zhouhang Cheng, Haichang Gao, Zhongyu Liu, Huaxi Wu, Yang Zi, and Ge Pei. 2019. Image-based CAPTCHAs based on neural style transfer. *IET Information Security*, 13, 6, 519–529.

[8] Ritendra Datta, Jia Li, and James Z. Wang. 2005. Imagination: a robust image-based CAPTCHA generation system. In *ACM Conference on Multimedia*, 331–334.

[9] Jiankang Deng, Jia Guo, Zhou Yuxiang, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. 2020. RetinaFace: single-stage dense face localisation in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5203–5212.

[10] Jeremy Elson, John R Douceur, Jon Howell, and Jared Saul. 2007. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *ACM Conference on Computer and Communications Security*. Volume 7, 366–374.

[11] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423.

[12] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. 2014. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *International Conference on Learning Representations*.

[13] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Neural Information Processing Systems*, 2672–2680.

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

[15] Gaurav Goswami, Brian M Powell, Mayank Vatsa, Richa Singh, and Afzel Noore. 2014. Facedcaptcha: face detection based color image captcha. *Future Generation Computer Systems*, 31, 59–68.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

[17] Peiyun Hu and Deva Ramanan. 2017. Finding tiny faces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 951–959.

[18] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. 2007. Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical report. University of Massachusetts, Amherst.

[19] Xun Huang and Serge Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision*, 1501–1510.

[20] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision*, 179–196.

[21] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 694–711.

[22] Kaggle. 2016. Dogs vs. Cats Redux, Kernels Edition: distinguish images of dogs from cats. https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition.

[23] Kaggle. 2013. Dogs vs. Cats: create an algorithm to distinguish dogs from cats. https://www.kaggle.com/c/dogs-vs-cats.

[24] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 4401–4410.

[25] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of StyleGAN. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8110–8119.

[26] Muhammad Haris Khan, John McDonagh, Salman Khan, Muhammad Shahabuddin, Aditya Arora, Fahad Shahbaz Khan, Ling Shao, and Georgios Tzimiropoulos. 2020. AnimalWeb: a large-scale hierarchical dataset of annotated animal faces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6939–6948.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, 1097–1105.

[28] Yongshuai Liu, Jiyu Chen, and Hao Chen. 2018. Less is more: culling the training set to improve robustness of deep neural networks. In *Conference on Decision and Game Theory for Security (GameSec)*. Seattle, WA, USA, (October 29–31, 2018).

[29] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. 2017. Deep photo style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, 4990–4998.

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

[31] Gabriel Moy, Nathan Jones, Curt Harkless, and Randall Potter. 2004. Distortion estimation techniques in solving visual CAPTCHAs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[32] Fernando Pérez-Cruz. 2008. Kullback-leibler divergence estimation of continuous distributions. In *IEEE International Symposium on Information Theory*, 1666–1670.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In *Neural Information Processing Systems*, 91–99.

[34] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

[35] Suphannee Sivakorn, Iasonas Polakis, and Angelos D Keromytis. 2016. I am robot:(deep) learning to break semantic image captchas. In *IEEE European Symposium on Security and Privacy*, 388–403.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

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

[38] Mingxing Tan and Quoc V Le. 2019. EfficientNet: rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, 6105–6114.

[39] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. 2003. CAPTCHA: using hard AI problems for security. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 294–311.

[40] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. 2008. reCAPTCHA: human-based character recognition via web security measures. *Science*, 321, 5895, 1465–1468.

[41] Jeff Yan and Ahmad Salah El Ahmad. 2008. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Symposium on Usable Privacy and Security*, 44–52.

[42] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. 2016. Wider face: a face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition*, 5525–5533.

[43] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23, 10, 1499–1503.

[44] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. 2017. C. In *IEEE International Conference on Computer Vision*, 192–201.