The 20th World Conference on Information Security Applications

CAPTCHA Image Generation using Style Transfer Learning in Deep Neural Network

Hyun Kwon*, Hyunsoo Yoon*, and Ki-Woong Park†§

*School of Computing, Korea Advanced Institute of Science and Technology
†Department of Computer and Information Security, Sejong University
§Corresponding author: woongbak@sejong.ac.kr

Abstract. CAPTCHA is widely used as a security solution to prevent automated attack tools on websites. However, CAPTCHA is difficult to recognize human perception when it gives a lot of distortion to have resistance against the automated attack. In this paper, we propose a method to deceive the machine while maintaining the human perception rate by applying the style transfer method. This method creates a style-plugged-CAPTCHA image by combining the styles of different images while maintaining the content of the original CAPTCHA sample. We used 6 datasets in the actual site and used Tensorflow as the machine learning library. Experimental results show that the proposed method reduces the recognition rate of the DeCAPTCHA system to 3.5% while maintaining human perception.

Keywords: Completely automated public turing test to tell computers and humans apart (CAPTCHA) · Deep neural network (DNN) · Convolutional neural network (CNN) · Image style transfer

1 Introduction

Recently, automated attacks are being made using machines with excellent computing performance. Because these automated attacks [16] become available for bulletin boards [15], unlimited subscriptions [4], spam messages [3], and DDoS attacks [7] on Web sites, security solutions are becoming more important to prevent automated attacks. Among security solutions, Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) [25] is a typical solution to defend against such automated attacks. CAPTCHA is widely used to determine whether a user is a human or a machine through challenge response tests. This method consists of asking a question that the machine can not understand but the person understands the question. If the requester's response is determined to be correct, the service is provided. If the answer is incorrect, the service is rejected.

In order to make CAPTCHA images based on text [5] which are misrecognized by the machine, it is necessary to add some distortion such as rotation, size, and arc to text images. This distorted image can be understood by humans, but makes the machine hard to recognize. In addition, there is CAPTCHA based on

audio and images. CAPTCHA based on audio [24] generates sounds with noise in numbers and letters so people can understand but machines do not understand. However, the audio method may have the disadvantage that the user system has to support the voice and the external sound. In this paper, we have studied CAPTCHA images based on text.

However, attack methods to break CAPTCHAs such as optical character recognition (ORC) [6] are being studied. Therefore, in order to avoid such attack methods, it is necessary to adjust the rotation, size, and position of letters, but when it is too much, there is a disadvantage that the usability is greatly reduced due to a distorted text image which is difficult to be recognized by human perception. Therefore, there is a need for a method that the machine recognizes incorrectly within the range of maintaining the human recognition rate.

Recently, deep neural network (DNN) [22] has provided good performance for image recognition [14], image generation [12], and image synthesis [20]. Especially, convolutional neural network (CNN) [19] shows good performance in image recognition field. It is possible to extract features from each feacture and synthesize different images. Among them, the style transfer method [11] can extract the representation of the content of the original sample and extract the representation of the style of another image. For example, it can create a new image that combines Obama's photo content with Rousseau's image style.

In this paper, we propose a style-plugged-CAPTCHA method to deceive the machine while maintaining the perception rate of human by applying style transfer learning to the CAPTCHA image dataset. The propose scheme generates a style-plugged-CAPTCHA image by extracting the content feature of the original sample and the style feature of the other image. The contribution of this paper is as follows.

- This paper propose the style-plugged-CAPTCHA method. We systematically organize the frameworks of the proposed scheme.
- We analyzed the degree of image distortion of the proposed method using CAPTCHA image datasets which is operated on actual site.
- We measured the recognition of the style-plugged-CAPTCHA images by comparing the original images with the DeCAPTCHA system in order to verify the performance of the proposed method.

The rest of this paper is structured as follows: In Section 2, we review the related work. The proposed scheme is presented in Section 3. In Section 4, we present and explain the experiment and evaluation. The proposed scheme is discussed in Section 5. Finally, Section 6 concludes the paper.

2 Related works

The CAPTCHA system [13] was first introduced in 1997 as an Internet search site, AltaVista. Section 2.1 presents an overview of related research on CAPTCHA based on text. Section 2.2 provides an overview of CNN and Section 2.3 explains the image style transfer method.

2.1 A review of CAPTCHA based on text

CAPTCHA based on text is a method using distorted text images that are correctly recognized by humans but misrecognized by machines. In order to prevent the machine from recognizing the CAPTCHA by the automated attack, the CAPTCHA system gives the rotation of the text image, resizes the letters, adds an arc, or overlaps the letters. Typically, there are three methods that CAPTCHA based on text. First, the connecting characters together (CCT) [9] method is giving overlap and noise of characters, and arc to text images in order to resistant to character segmentation and recognition by the automatic machine. Second, the hollow method [10] is designed to connect all texts together in a format that only displays the outline of the text. This method is resistant to the segmentation and recogntion of the machine while enhancing human perception. Third, the character isolated method [8] is a method of displaying each character independently of each other, unlike the above-mentioned method. Therefore, the distortion of each character is severe compared to other methods.

This CAPTCHA based on text is disadvantageous in that it is degraded to human perception rate if it is too much distortion. Unlike the conventional method, the proposed method applies different image styles while maintaining human perception rate.

2.2 A review of CNN model

CNN [19] is a deeply neural network that is commonly used for visual imagery as a regularized version of a fully connected network. In particular, CNN has the advantage of reflecting information on spatial characteristics without loss. Since the input and output data are processed as three-dimensional data using the CNN, spatial information can be maintained. However, in the fully connected network, since only one-dimensional data is received, spatial information of the three-dimensional data is lost.

The structure of CNN consists of fully connected layer, convolution layer, and pooling. A fully connected layer means a layer of the form combined with all the neurons of the previous layer.

The convolution layer extracts the characteristics of the image using a filter in the input image. In the convolution layer, the value is obtained by multiplying the adjacent pixel by the convolution filter for the output data at each layer. The input and output data in the convolution layer are called a feature map.

The purpose of the pooling layer is to be used when performing subsampling or extracting data samples once again through the convolution process. Similar to the convolution layer, pooling uses only adjacent pixel values, but there is no computation. There are two types of pooling: max pooling and average pooling. The max pooling sets the largest pixel value in adjacent pixels to a new pixel value. On the other hand, the average pooling sets the average value of adjacent pixels to a new pixel value.

The parameters of CNN can be set by parameter of convolution filter number, filter window size, padding, and stride. In convolition filter number, it is

important to keep the number of convolution fiter relatively constant at each layer. In the filter window size, it can be used to emphasize the desired features by utilizing the non-inleariness in the intermediate stage when several small filters are overlapped. The padding means to increase the input data around a specific pixel value before performing the convolution to adjust the output data size and prevent loss of information. The stride is the parameter that controls the window's moving distance.

2.3 A review of image style transfer method

Image style transfer [11] proposed a method of creating a new image by combining the content of the original image with the style of another image. This method uses a feature that allows CNN to extract information from the semantic image at high-level. This method reduces the two loss functions to create a new image. The first loss function represents a content representation of original image, meaning it maintains a specific content by increasing object information that represents the content of the image. The second loss function proposes a method of obtaining the information about the feature space of the texture of an other image. The proposed method is a method of applying image style transfer method to CAPTCHA image domain.

3 Proposed method

To generate a style-plugged-CAPTCHA images, the proposed method accepts the original image and the other image as input values, and generates a new image that combines the content of the original image and the style of the other image, as shown in Fig. 1.

For this study, we used the style transfer architecture given in [11]. In the CNN model, this method extracts the content feature from the original image and extracts the style feature from the other image separately. First, in the case of a content feature, a feature map can be extracted through the convolution layer of the original image. As the layer becomes deeper, pixel level information disappears, but the semantic information of the input image remains. Therefore, we extract the content feature of the original image from the deep layer. Second, in case of style feature, it is based on gram matrix [17]. The gram matrix represents the correlation between the feature maps of each layer. By using correlation of feature maps of several layers, it is possible to obtain information considering multiple scales of stationary information rather than layout information of image. The more deep layers are included, the more the image gets static information rather than layout information.

To generate the proposed CAPTCHA image, the method updates the noise image $\overrightarrow{x}$ through back propagation of the loss function $loss_T$. The loss function $loss_T$ is the sum of content loss $loss_{content}$ and style loss $loss_{style}$:

$$loss_T = loss_{style}(\overrightarrow{a}, \overrightarrow{x}) + \alpha \cdot loss_{content}(\overrightarrow{p}, \overrightarrow{x}), \tag{1}$$

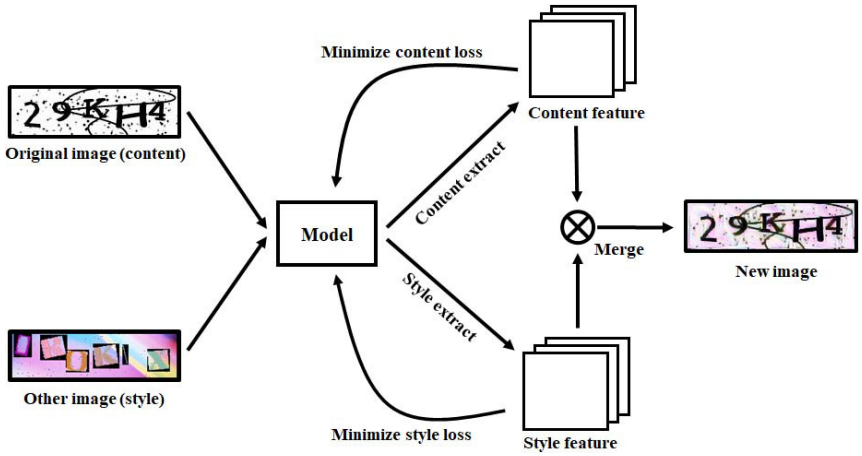


Fig. 1: Proposed architecture.

where $\overrightarrow{a}$ means other image, $\overrightarrow{x}$ means noise image, which is a composite image, and $\overrightarrow{p}$ means original image. α is a weighted value over 1. The initial value is 1. First, the content loss $loss_{content}$ is calculated based on the content feature for the noise image $\overrightarrow{x}$, which is the image to be synthesized with the original image $\overrightarrow{p}$. The procedure is as follows. First, the original image $\overrightarrow{p}$ and nosie image $\overrightarrow{x}$ are feed-forward through the network, respectively. Second, we obtain feature maps P and F in layer l as input values to original image $\overrightarrow{p}$ and noise image $\overrightarrow{x}$, respectively. Third, through the obtained feature maps P and F, the content loss is defined as follows.

$$loss_{content}(\overrightarrow{p}, \overrightarrow{x}, l) = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2, \tag{2}$$

where P_{ij}^{l} is the activation i_{th} filter at position j in layer l and F_{ij}^{l} is the activation i_{th} filter at position j in layer l.

Second, style loss $loss_{content}$ is calculated based on the style feature for other image (style) $\overrightarrow{a}$ and noise image $\overrightarrow{x}$. The procedure is as follows. First, the other image $\overrightarrow{a}$ and nosie image $\overrightarrow{x}$ are feed-forward through the network, respectively. Second, we obtain gram matric A and G in layer l as input values to other image $\overrightarrow{a}$ and noise image $\overrightarrow{x}$, respectively. Third, through the obtained gram matric A and G, the style loss is defined as follows.

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G_{ij}^{l} - A_{ij}^{l})^2, \tag{3}$$

where A_{ij}^{l} and G_{ij}^{l} are the inner product between the vectorised feature maps i and j in layer l, N_l is the number of feature maps at layer l, and M_l is height $\times$

width of feature maps at layer l. In the case of the style feature, the total style loss $loss_{style}$ is as follows because it uses several layers simultaneously.

$$loss_{style}(\overrightarrow{a}, \overrightarrow{x}) = \sum_{l=0}^{L} w_l E_l, \tag{4}$$

where w_l is weighting factors of the layer to the total loss. The details of the procedure for generating a proposed CAPTCHA image are given in Algorithm 1.

Algorithm 1 Style-plugged-CAPTCHA Image generation.

Input: original image $\overrightarrow{p}$, other image (style) $\overrightarrow{a}$, noise image $\overrightarrow{x}$, P_{ij}^l is the activation i_{th} filter at position j in layer l and F_{ij}^l is the activation i_{th} filter at position j in layer l, A_{ij}^l and G_{ij}^l are the inner product between the vectorised feature maps i and j in layer l, N_l is the number of feature maps at layer l, and M_l is height $\times$ width of feature maps at layer l, iterations r.

Style-plugged-CAPTCHA Image generation:

$\overrightarrow{a} \leftarrow 0$

for r step **do**

$loss_{content}(\overrightarrow{p}, \overrightarrow{x}, l) \leftarrow \frac{1}{2}\sum_{i,j}(F_{ij}^l - P_{ij}^l)^2$

$E_l \leftarrow \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G_{ij}^l - A_{ij}^l)^2$

$loss_{style}(\overrightarrow{a}, \overrightarrow{x}) \leftarrow \sum_{l=0}^{L} w_l E_l$

$loss_T \leftarrow loss_{content} + loss_{style}$

Update $\overrightarrow{x}$ by minimizing $loss_T$ through back propagation such as $\overrightarrow{a} - \lambda \frac{\delta loss_T}{\delta \overrightarrow{a}}$

end for

return $\overrightarrow{a}$

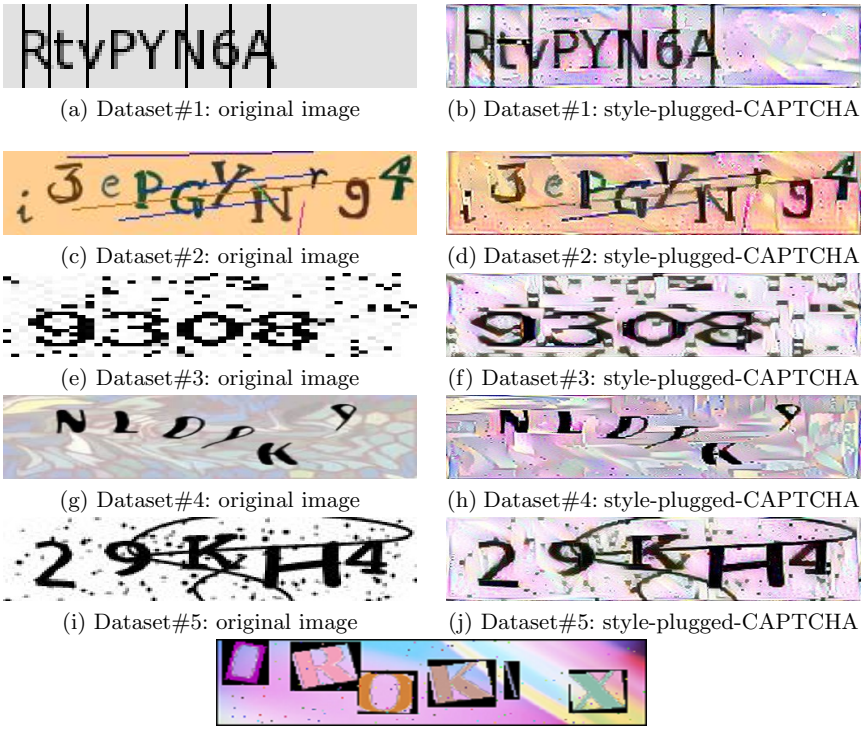
4 Experiment & evaluation

Through experiments, we show that the proposed scheme can generate a style transfer CAPTCHA image that is resist to the DeCAPTCHA and can maintain the human perception. We used the Tensorflow [2] library, a widely used open source library for machine learning, on a Intel(R) Core(TM) i5-8400 CPU 2.80GHz server.

4.1 Experimental method

In terms of experimental datasets, we used 6 different CAPTCHA datasets running on the web site and 100 data per dataset. The six Web sites are as follows: smart-mail.de, ArticlesFactory.com, nationalinterest.org, cesdb.com, mail.aol.com, and tiki.org.

In terms of pretrained model and DeCAPTCHA, we used the VGG-19 model [23] as a pretrained model. Table 1 and Table 2 show the structure and parameters for the VGG-19 model. As DeCAPTCHA, we used the gsc captcha breaker



(a) Dataset#1: original image (b) Dataset#1: style-plugged-CAPTCHA

(c) Dataset#2: original image (d) Dataset#2: style-plugged-CAPTCHA

(e) Dataset#3: original image (f) Dataset#3: style-plugged-CAPTCHA

(g) Dataset#4: original image (h) Dataset#4: style-plugged-CAPTCHA

(i) Dataset#5: original image (j) Dataset#5: style-plugged-CAPTCHA

(k) Dataset#6: other image (style feature)

Fig. 2: The sampling example of style-plugged-CAPTCHA for the original image of each dataset when selecting dataset #6 as the style image.

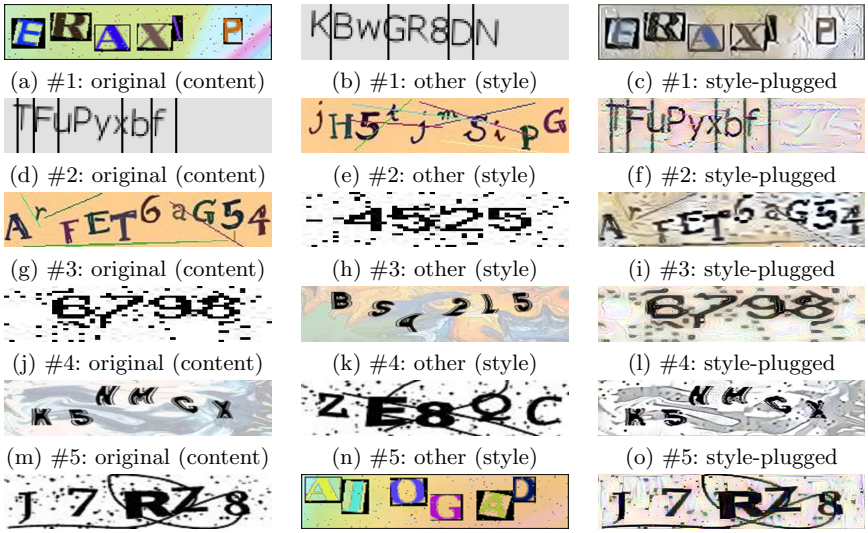
program [1]. The gsc catpcha breaker is the software used in the website by segmenting and recognizing CAPTCHA images.

To generate the style transfer CAPTCHA, L-BFGS optimization [21] is used as box contrained optimization. The L-BFGS optimization is an efficient algorithm for solving a large scale problem. It is used to design and refine quadratic models for optimization functions. The iteration is 100 and weight of loss α is 1. For a given number of iterations, the proposed method updates the output $\overrightarrow{x}$ from the feedback of content loss and style loss. At the end of the iterations, the new image, $\overrightarrow{x}$, was evaluated in terms of human perception and the recognition rate of DeCAPTCHA. The recognition rate of DeCAPTCHA means the rate at which CAPTCHA is correctly recognized by gsc captcha breaker.

4.2 Experimental results

Fig. 2 shows an example of creating a style-plugged-CAPTCHA image for the original image of each dataset when selecting dataset 6 as the style image. In the

figure, a style-plugged-CAPTCHA image is created that takes the style property of the other image while retaining the content content of the original image. Especially, the style image is the same, but the degree of deformation of the style is slightly different according to the characteristics of the original image. However, it is seen that the recognition rate of a person is maintained because a lot of letters are not transformed by the human perception.



(a) #1: original (content) (b) #1: other (style) (c) #1: style-plugged

(d) #2: original (content) (e) #2: other (style) (f) #2: style-plugged

(g) #3: original (content) (h) #3: other (style) (i) #3: style-plugged

(j) #4: original (content) (k) #4: other (style) (l) #4: style-plugged

(m) #5: original (content) (n) #5: other (style) (o) #5: style-plugged

(p) #6: original (content) (q) #6: other (style) (r) #6: style-plugged

Fig. 3: The sampling example of style-plugged-CAPTCHA for the each dataset: Original is a original image, other is a other image, and style-plugged is a style-plugged-CAPTCHA.

Fig. 3 shows that a style-plugged-CAPTCHA image is generated for each dataset when the original sample and the other image are given a difference of 1 in the dataset order. In the figure, a style-plugged-CAPTCHA image is generated by extracting the style of the other image while maintaining the content of the original sample. For example, in datasets #3, #4, #5, and #6, the figure shows that the point-point portion of the style image is added to the style-plugged-CAPTCHA image. In addition, the newly created style-plugged-CAPTCHA image can be seen to remain in the human perception.

Fig. 4 shows the recognition rate for the original image and style-plugged-CAPTCHA by the DeCAPTCHA program for 100 samples per dataset. In the figure, the recognition rate of DeCAPTCHA is different for each dataset. Especially, when the style-plugged-CAPTCHA method is applied, the recognition rate of the DeCAPTCHA is significantly lowered for the style-plugged-

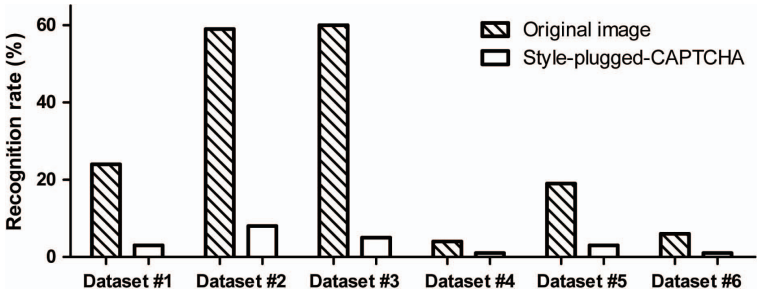


Fig. 4: DeCAPTCHA recognition rates.

CAPTCHA image due to the modulation on the style. Therefore, the style-plugged-CAPTCHA image has some resistance to the DeCAPTCHA system than the original image.

5 Discussion

Attack method consideration The assumption of the proposed method is a white box attack that knows the model. The proposed method is a method of extracting the content feature of the original sample and the style feature of the other image from the pretrained model using L-BFGS optimization

The proposed method gives more weight to the content representation than the existing style tranfer method. Because the recognition rate of a person is reduced when there are many changes in a content representation, the weight is set higher in the content representation than in the style representation.

Unlike the conventional CAPTCHA method, the proposed method proposes a method of changing the style by using the feature which can extract the feature in CNN. It is possible to generate CAPTCHA images more variously by changing the image style rather than changing characters.

Application The proposed method is useful for generating CAPTCHA images in large quantities. If the amount of CAPTCHA images is limited, it is necessary to generate various CAPTCHA images through various combinations. In such a case, the proposed method can be used to generate CAPTCHA images of different styles. On the contrary, if the DeCAPTCHA system requires learning about various data, it can be used to improve the performance of the DeCAPTCHA system by generating various images generated through the proposed method and learning in advance.

Limitation In the proposed method, if the weight of the style representation is increased, the image distortion may be increased. Also, if the other image of the

style image is severely distorted, the content image may be affected. Therefore, it is necessary to consider when selecting a style image.

Also, since the proposed method does not attack the DeCAPTCHA system with a white box, it is not directly attacked. Therefore, it is necessary to compare the input and output values of the DeCAPTCHA system and extend it to the CAPTCHA method with optimal distortion while maintaining human recognition rate.

6 Conclusion

In this paper, we proposed a style-plugged-CAPTCHA image that change the style for the original image. The propose scheme generates a style-plugged-CAPTCHA image by extracting the content feature of the original sample and the style feature of the other image. These style-plugged-CAPTCHA image has some resistance to the DeCAPTCHA system. Experimental results show that the proposed method maintains the human recognition rate while decreasing the recognition rate to about 3.5% for the DeCAPTCHA system. The proposed scheme can also show the possibility of being applied to applications such as the data expansion.

Future studies can be extended to more diverse datasets. It can also be applied to generate CAPTCHAs using the generative adversarial net method [18] instead of the L-BFGS algorithm.

Acknowledgement

This work was supported by the Institute for Information and Communications Technology Promotion (No. 2018-0-00420 and No. 2019-0-00426) and supported by the National Research Foundation of Korea (2017R1C1B2003957 and 2017R1A2B4006026).

References

1. Gsa captcha breaker. GSA - Softwareentwicklung und Analytik GmbH, Available: https://captcha-breaker.gsa-online.de/ (2017)
2. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI. vol. 16, pp. 265–283 (2016)
3. Alorini, D., Rawat, D.B.: Automatic spam detection on gulf dialectical arabic tweets. In: 2019 International Conference on Computing, Networking and Communications (ICNC). pp. 448–452. IEEE (2019)
4. Bukac, V., Stavova, V., Nemec, L., Riha, Z., Matyas, V.: Service in denial–clouds going with the winds. In: International Conference on Network and System Security. pp. 130–143. Springer (2015)
5. Bursztein, E., Martin, M., Mitchell, J.: Text-based captcha strengths and weaknesses. In: Proceedings of the 18th ACM conference on Computer and communications security. pp. 125–138. ACM (2011)

6. Das, M.S., Rao, K.R.M., Balaji, P.: Neural-based hit-count feature extraction method for telugu script optical character recognition. In: Innovations in Electronics and Communication Engineering, pp. 479–486. Springer (2019)
7. Demoulin, H.M., Pedisich, I., Phan, L.T.X., Loo, B.T.: Automated detection and mitigation of application-level asymmetric dos attacks. In: Proceedings of the Afternoon Workshop on Self-Driving Networks. pp. 36–42. ACM (2018)
8. Gao, H., Tang, M., Liu, Y., Zhang, P., Liu, X.: Research on the security of microsoft's two-layer captcha. IEEE Transactions on Information Forensics and Security **12**(7), 1671–1685 (2017)
9. Gao, H., Wang, W., Fan, Y., Qi, J., Liu, X.: The robustness of" connecting characters together" captchas. J. Inf. Sci. Eng. **30**(2), 347–369 (2014)
10. Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., Yan, J.: The robustness of hollow captchas. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 1075–1086. ACM (2013)
11. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016)
12. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: Draw: A recurrent neural network for image generation. arXiv preprint arXiv:1502.04623 (2015)
13. Hasan, W.K.A.: A survey of current research on captcha. International Journal of Computer Science and Engineering Survey (IJCSES) **7**(3), 141–157 (2016)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Holz, T., Marechal, S., Raynal, F.: New threats and attacks on the world wide web. IEEE Security & Privacy **4**(2), 72–75 (2006)
16. Householder, A., Houle, K., Dougherty, C.: Computer attack trends challenge internet security. Computer **35**(4), sulp5–sulp7 (2002)
17. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. pp. 694–711. Springer (2016)
18. Kwon, H., Kim, Y., Yoon, H., Choi, D.: Captcha image generation systems using generative adversarial networks. IEICE TRANSACTIONS on Information and Systems **101**(2), 543–546 (2018)
19. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
20. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2479–2486 (2016)
21. Moritz, P., Nishihara, R., Jordan, M.: A linearly-convergent stochastic l-bfgs algorithm. In: Artificial Intelligence and Statistics. pp. 249–258 (2016)
22. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks **61**, 85–117 (2015)
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR 2015 (2015)
24. Soupionis, Y., Gritzalis, D.: Audio captcha: Existing solutions assessment and a new implementation for voip telephony. Computers & Security **29**(5), 603–618 (2010)
25. Von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 294–311. Springer (2003)

12

Appendix

Table 1: Model of VGG-19 [23]

Layer type	Model shape
Convolution+ReLU	[3, 3, 64]
Convolution+ReLU	[3, 3, 64]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 128]
Convolution+ReLU	[3, 3, 128]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 256]
Convolution+ReLU	[3, 3, 256]
Convolution+ReLU	[3, 3, 256]
Convolution+ReLU	[3, 3, 256]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Max pooling	[2, 2]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Convolution+ReLU	[3, 3, 512]
Max pooling	[2, 2]
Fully connected+ReLU	[4096]
Fully connected+ReLU	[4096]
Fully connected+ReLU	[1000]
Softmax	[1000]

Table 2: VGG-19 [23] model parameters.

Parameter	Values
Learning rate	0.01
Momentum	0.9
Decay	0.0005
Iteration	370,000
Dropout	0.5
Batch size	256
Epochs	74