Variational Color Shift and Auto-Encoder based on Large Separable Kernel Attention for Enhanced Text CAPTCHA Vulnerability Assessment

Xing Wan *, Juliana Johari, Fazlina Ahmat Ruslan *

Posted Date: 26 September 2024

doi: 10.20944/preprints202409.2128.v1

Keywords: CAPTCHA recognition; color shift; auto encoder; attention mechanism; large kernel

Article

Variational Color Shift and Auto-Encoder based on Large Separable Kernel Attention for Enhanced Text CAPTCHA Vulnerability Assessment

Xing Wan [1,2,*], Juliana Johari [2] and Fazlina Ahmat Ruslan [2,*]

[1] School of Intelligent Manufacturing, Leshan Vocational and Technical College, Leshan 614000, China
[2] School of Electrical Engineering, Universiti Teknologi MARA (UiTM) Shah Alam 40450, Malaysia
* Correspondence: 2022995467@student.utim.edu.my (X.W.); fazlina419@uitm.edu.my (F.A.R.)

Abstract: Text CAPTCHAs are crucial security measures employed on global websites to deter unauthorized intrusions. The presence of anti-attack features incorporated into text CAPTCHAs limits the effectiveness of breaking them, despite CAPTCHA recognition being an effective method for assessing their security. This study introduces a novel color augmentation technique called Variational Color Shift (VCS) to boost the recognition accuracy of different networks. VCS generates a color shift range of every input image and then resamples the image within that range to generate a new image, thus expanding the number of samples of the original dataset to improve training effectiveness. In contrast to Random Color Shift (RCS) which treats the color offsets as hyperparameters, VCS uses the estimated offsets to reparametrize the points sampling from the unit uniform distribution to reconstruct new image pixels, which makes the offsets learnable. To reduce the computational effort of VCS, we also propose two variants of VCS: Sim-VCS and Dilated-VCS. In addition, to solve the overfitting problem caused by many disturbances in CAPTCHAs, we propose an Auto Encoder (AE) based on a Large Separable Kernel Attention (AE-LSKA) to replace the convolutional module in the text CAPTCHA recognizer. This new module employs an AE) to compress the interference while expanding the receptive field using Large Separable Kernel Attention (LSKA), which reduces the impact of local interference on the model training and improves the overall perception of characters. The experimental results show that the recognition rate of the model after integrating the AE-LSKA module is improved by at least 15 percentage points on both M-CAPTCHA and P-CAPTCHA datasets. In addition, experimental results demonstrate that color augmentation using VCS is more effective in enhancing recognition.

Keywords: CAPTCHA recognition; color shift; auto encoder; attention mechanism; large kernel

1. Introduction

With the rapid development of the Internet, companies and individuals can freely use it for business, communication, and entertainment, resulting in significant lifestyle changes [1]. While cyberspace provides convenience to people, it also has certain security risks. The CAPTCHA is a protection mechanism proposed to enhance network security. At the beginning of the 21st century, the scholar Luis von Ahn et al. jointly proposed CAPTCHA [2]. As a widely used public security program, its main function is to design a verification mechanism to distinguish between humans and machines so that the CAPTCHA mechanism prevents the access of malicious robot programs. When designing a CAPTCHA security mechanism, it is considered safe if the success rate of computer cracking is less than 10% [3]. As the first line of defense for the information system, the CAPTCHA provides the most direct and effective security guarantee for various systems. Today, most websites worldwide use CAPTCHAs to defend against network attacks and web spiders. It is crucial to enhance the security of CAPTCHAs while maintaining their user-friendliness [4]. In recent years, deep learning technology has provided many ideas and technical frameworks for automatic attack and protection of CAPTCHAs [5]. The neural networks can automatically extract the most important features, therefore avoiding designing image operators manually. However, as the security design of CAPTCHAs continues to advance, higher demands are placed on the capabilities of CAPTCHA

 doi:10.20944/preprints202409.2128.v1

security evaluators. It is essential to continuously improve these evaluators to effectively assess the security of CAPTCHAs.

There are about ten categories of CAPTCHAs, and the most common type is text-based CAPTCHA [6]. These CAPTCHAs present distorted text with different backgrounds and noise for the user to recognize. Text-based CAPTCHA, the earliest CAPTCHA method, makes it simple to generate images for deployments. Because of its small storage space and rapid loading speed, many websites gradually adopt it, making it the most widely used CAPTCHA mechanism [7]. Despite its widespread use, the security of the text-based CAPTCHA image has gradually decreased due to the development of optical character recognition (OCR) [8]. To improve recognition difficulty, the fonts, colors, shades, positions, backgrounds, and angles are changed, and noise lines and points, distortions, borders, and overlaps are incorporated to increase recognition difficulty [9].

Some other image CAPTCHA systems are alternatives to text-based CAPTCHAs, which include sliding verification codes, click-based CAPTCHAs, drag-and-drop CAPTCHAs, selection-based CAPTCHAs, drawing-based CAPTCHAs, and interactive CAPTCHAs [10]. Typically, users must move or drag these CAPTCHAs with the mouse, then click or place them based on the provided hint. These mechanisms are more difficult to detect since they require not only classification but also location. So, object detection models such as YOLOX [11], YOLOv6 [12], and YOLOv7 [13] are adopted. As one-stage detectors, the YOLO series has the advantages of speedy detection, high precision, and robustness. Like text CAPTCHAs, these CAPTCHAs are embedded in many anti-attack mechanisms, making it difficult for attackers to improve the accuracy of identification.

The best way to find the security vulnerabilities of text CAPTCHA is by using different attacking methods to test and evaluate them [14]. To improve the recognition accuracy of models, one useful way is to collect CAPTCHAs as much as possible from the target websites. However, many websites have implemented anti-spider systems to impose restrictions on access. As a result, collecting enough CAPTCHAs from the target website is time-consuming and unrealistic. Many researchers have developed various data augmentation methods to enhance data volume and diversity based on the few images collected from websites. The data augmentation methods usually include image clipping, image combination, random resizing, rotation, flipping, and color shift [15]. Among them, the color shift is commonly used in CAPTCHA recognition tasks, as many data augmentation methods, which are widely used in classification tasks, are not suitable for CAPTCHA breaking due to their interfering designs [16].

This study proposed new color shift methods by improving random color shift algorithms and evaluating them on different datasets. In addition, we proposed the AE-LSKA, which can recognize characters in the image from a more global perspective, thereby reducing the impact of local interference of the verification code. There are three main contributions to this study. First, we introduced VCS, and its variations, using the variation method, whose upper and lower limits are trainable according to input CAPTCHAs. Second, we propose a large kernel attention-based auto encoder that is well suited for the CAPTCHA recognition task with a large amount of noise and interference. Third, comparison experiments are carried out on simple and complex datasets using strong and weak models to prove the superiority of the proposed VCS and AE-LSKA.

2. Datasets and Algorithms

2.1. Summary of Datasets and Algorithms

To evaluate our proposed methods more comprehensively, we will use two models with different performances tested on two datasets with distinct complexities in the experiments. The first is a weak model with relatively low recognition ability, Deep-CAPTCHA. The other is a strong network with higher recognition ability, Adaptive-CAPTCHA. In addition, we adopt two datasets of different complexities: the complex M-CAPTCHA and the simple P-CAPTCHA. By using these models and datasets, we can evaluate the three proposed methods (VCS, Sim-VCS, and Dilated-VCS) and AE-LSKA more comprehensively, as shown in Table 1.

 doi:10.20944/preprints202409.2128.v1

Table 1. Algorithms and datasets.

Dataset/ Model	Description
Adaptive-CAPTCHA	Strong CAPTCHA recognizer
Deep-CAPTCHA	Weak CAPTCHA recognizer
M-CAPTCHA	Complex CAPTCHA dataset
P-CAPTCHA	Simple CAPTCHA dataset
AE-LSKA	New Feature extraction module
VCS, Sim-VCS, Dilated-VCS	New color shift methods

2.2. Deep-CAPTCHA and Adaptive-CAPTCHA

In recent years, some text CAPTCHA classification networks have also been proposed, such as CapsuleNet [17] based on capsule neural networks, transformer-based recognizers [18], and ConvNet [19] using a novel group convolution operation. Although these networks also have high recognition rates, some of them have too many parameters or too much computation, making them unsuitable for front-end deployment.

Deep-CAPTCHA is an innovative model introduced in 2020 by Noury et al. that aims to tackle the challenge of breaking text-based CAPTCHAs, which are commonly used as a security measure to break text CAPTCHAs [20]. The model leverages deep learning techniques to understand and interpret the distorted text within CAPTCHA images. It employs convolutional neural networks (CNNs) to analyze patterns and structures within the images, allowing it to decipher the obscured text with remarkable accuracy. One of the key strengths of Deep-CAPTCHA is its ability to adapt to various types of text CAPTCHAs, including those with noise, distortions, and different fonts. The structure of Deep-CAPTCHA includes three convolution layers, one fully connected layer, and one softmax layer, as shown in Figure 1.

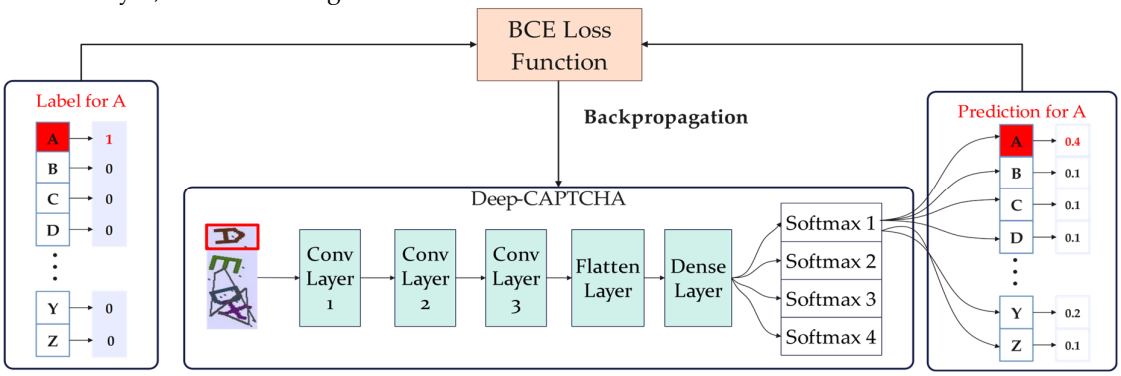


Figure 1. The structure and training process of Deep-CAPTCHA.

Adaptive-CAPTCHA is a new text CAPTCHA breaker developed on Deep-CAPTCHA, which was proposed in 2024 and has a stronger ability to recognize CAPTCHAs with a correlation between characters using the Convolutional and Recurrent Neural Network (CRNN) module [21]. In addition, this model can effectively remove interference and noise by introducing an Adaptive Fusion Filtering Network (AFFN). Compared with Deep-CAPTCHA, Adaptive-CAPTCHA has a stronger recognition capability, and its detailed architecture is shown in Figure 2.

 doi:10.20944/preprints202409.2128.v1

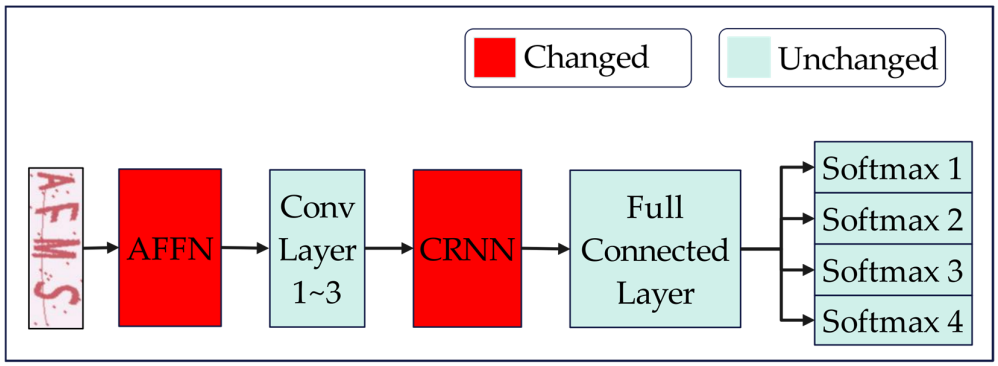


Figure 2. The structure of Adaptive-CAPTCHA.

2.3. M-CAPTCHA and P-CAPTCHA

The first dataset is M-CAPTCHA, which contains distorted characters, interferences, and colorful backgrounds, so it is a complex dataset with high recognition difficulty, as shown in Figure 3. The characters in this dataset exhibit a Markov distribution among themselves, thus testing the model's ability to extract global features from the image content.

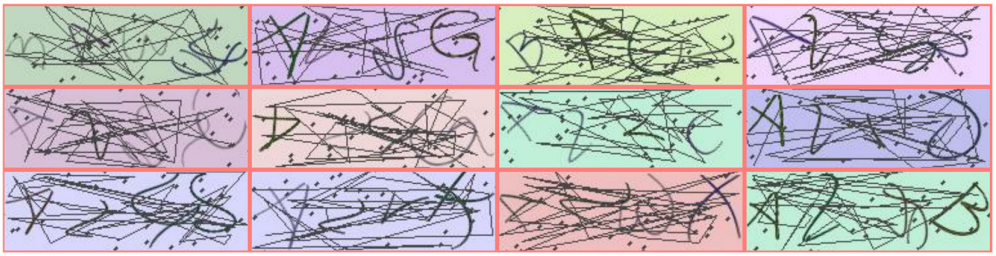


Figure 3. Samples of M-CAPTCHA.

P-CAPTCHA is a relatively simpler dataset. It is directly generated by the Python captcha library and contains less interference and a simpler font, which is easier to recognize by the model, as shown in Figure 4.

Figure 4. Samples of P-CAPTCHA.

2.4. Color Shift Algorithms

Krizhevsky et al. introduced a new color augmentation method called PCA Color Shift (PCA-CS) [22]. This method gets the largest three eigenvalues of the input image and multiplies these three values by random numbers as the weights of RGB channels to obtain a new image, which has excellent performance.

Wang et al. proposed RCS, which adopts random offset sampling from a uniform distribution to adjust every color channel to generate images, reserving intensity maps [23]. The problem with this method is that the upper and lower limits are hypermeters, and it is hard to find the optimal values. If the range of color shifst is too large, the augmentation may generate a fake image that is not

 doi:10.20944/preprints202409.2128.v1

like the real one. On the contrary, if the range of color shift is too small, the effect of data augmentation is limited, so it is necessary to find a reasonable range of color shifts.

Ishkov et al. considered that different color channels have different importance in CAPTCHA recognition and introduced a color mixing augmentation [24]. This method, denoted as Adaptive Channel Weights (ACW), adjusts the weights of RGB color channels using three learnable parameters. The experiment shows that their method can effectively improve the recognition accuracy of text CAPTCHA.

2.5. AE

AEs and Variational Autoencoders (VAEs) are two unsupervised deep-learning models that excel in various applications [25]. AEs compress high-dimensional inputs into low-dimensional encodings, aiming to minimize reconstruction errors during encoding and decoding processes. They are commonly used for data compression, feature learning, and denoising. In contrast, VAEs, introduced as a variant of AEs, incorporate stochasticity and probabilistic modeling to learn the underlying data distribution. By optimizing both the reconstruction loss and the Kullback-Leibler (KL) divergence between the learned latent distribution and a prior distribution, VAEs excel in generative modeling tasks, capable of generating novel data instances that resemble the training data.

2.6. LSKA

LSKA is a novel attention mechanism that revolutionizes the design of large kernel attention in CNNs [26]. It builds upon Visual Attention Networks (VAN) by leveraging separable convolution kernels to enhance attention efficiency [27]. LSKA decomposes 2D convolutional kernels into cascaded horizontal and vertical 1D kernels, significantly reducing computational complexity and memory footprints. This approach enables the direct use of large kernels in attention modules without additional blocks, preserving their ability to capture long-range dependencies while improving efficiency. LSKA demonstrates comparable performance to standard large kernel attention modules, outperforming Vision Transformers (ViTs) [28] and ConvNeXt [29] in tasks like image classification, object detection, and semantic segmentation, making it a promising technique for vision-based applications.

2.7. Lightweight Attention Mechanisms

Convolutional Block Attention Module (CBAM) integrates both channel and spatial attention modules to enhance feature representations [30]. Squeeze-and-Excitation (SE) focuses on channel attention by recalibrating feature responses through squeeze and excitation operations [31]. Shuffle Attention (SA) improves efficiency by shuffling inputs and computing attention weights, enhancing model generalization. Each mechanism contributes significantly to deep learning performance [32]. ECA (Efficient Channel Attention) enhances CNN representations by dynamically adjusting channel weights via 1D convolution, avoiding dimensionality reduction and improving both efficiency and performance [33]. ParNet Attention addresses the inefficiency of traditional attention on long sequences by dividing them into subsequences and computing attention independently, significantly reducing computational costs while capturing long-range dependencies [34]. Global Context (GC) Attention models the global context efficiently by aggregating context features and fusing them into each position, avoiding redundant computations and effectively capturing holistic input data characteristics [35].

3. Methods

The primary objective of this section is to introduce VCS and its two simplified versions and compare their differences in parameters and computational complexity. Furthermore, we analyze the design idea of AE-LSKA in detail and give the structures of different dilated convolutional kernels.

 doi:10.20944/preprints202409.2128.v1

3.1. VCS

As previously mentioned, determining the upper and lower limits of the color shift interval is crucial. However, it is not straightforward to find the optimal values through human experience. RCS typically sets the range of image pixel value offsets between 10% and 50%. However, adopting a fixed value overlooks the variations between different datasets and individual images. For instance, some images have nearly saturated pixel values, which could lead to overflow if the pixel values are increased significantly. Additionally, an excessively large offset might compromise the realism of the CAPTCHA's context, whereas an overly small offset might not effectively achieve data augmentation. Therefore, it is essential to determine offset ranges tailored to the specific characteristics of each CAPTCHA.

Our proposed VCS generates the upper and lower offsets of pixels for each image by extracting its feature. It is worth noting that the upper and lower offsets may be different and need to be generated separately. For each offset (lower or upper), we first use the three 1 × 3 adaptive pooling (maximum, minimum, and mean) layers to get the maximum, minimum, and mean values, and then merge the three 1 × 3 vectors into a 3 × 3 vector and pass it to a convolutional layer with a 3 × 3 kernel size to generate the 3 × 1 offset. This output is finally sent to a Sigmoid activation function to make the upper or lower offset fall in the range between 0 and 1. Because the RGB color contains three channels, the dimension of the color offset vector is 3 × 1, as shown in Equation (1) and (2), where $F_{lower}^{3\times3}$ and $F_{upper}^{3\times3}$ are the prediction convolution operations for the lower and the upper offset, respectively.

$$offset_L = Sigmoid\left(F_{lower}^{3\times3}\left(Concat\left[AdaptivePool_{Max,Min,Avg}^{1\times3}(\boldsymbol{X})\right]\right)\right) \tag{1}$$

$$offset_U = Sigmoid\left(F_{upper}^{3\times3}\left(Concat\left[AdaptivePool_{Max,Min,Avg}^{1\times3}(\boldsymbol{X})\right]\right)\right) \tag{2}$$

Based on these estimated offsets, $offset_L$ and $offset_U$, we can use them to reparametrize the weights W sampled from a uniform distribution to get the new weight W' of the image, as shown in Equation (3). Eventually, the original image is multiplied with the weighting coefficients to get the final newly generated image. Here we borrow the idea of sampling from a variational auto-encoder, where we first sample pixel values from a unit uniform distribution, and then scale the sampled values using the offset of the upper and lower intervals [36].

$$W' = offset_L + (offset_U - offset_L) \times W, \quad W \in U(0,1) \tag{3}$$

After obtaining the new weight W', we can multiply it with the input image $\boldsymbol{X}$ to obtain the final new image $\boldsymbol{Y}$, as shown in Equation (4).

$$\boldsymbol{Y} = \boldsymbol{X} \times W' \tag{4}$$

Figure 5 divides the entire VCS procedure into four stages. To generate three statistics, the first stage is to extract three features from the input image using three adaptive pooling layers. According to these statistics, in the second stage, two sampling convolution layers generate shift offsets for the upper and lower intervals. These offsets are used to reparametrize the initial weights sampling from the uniform distribution to obtain the final weights. In the last stage, the weights are used to scale the input image to generate the new image.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 September 2024 doi:10.20944/preprints202409.2128.v1

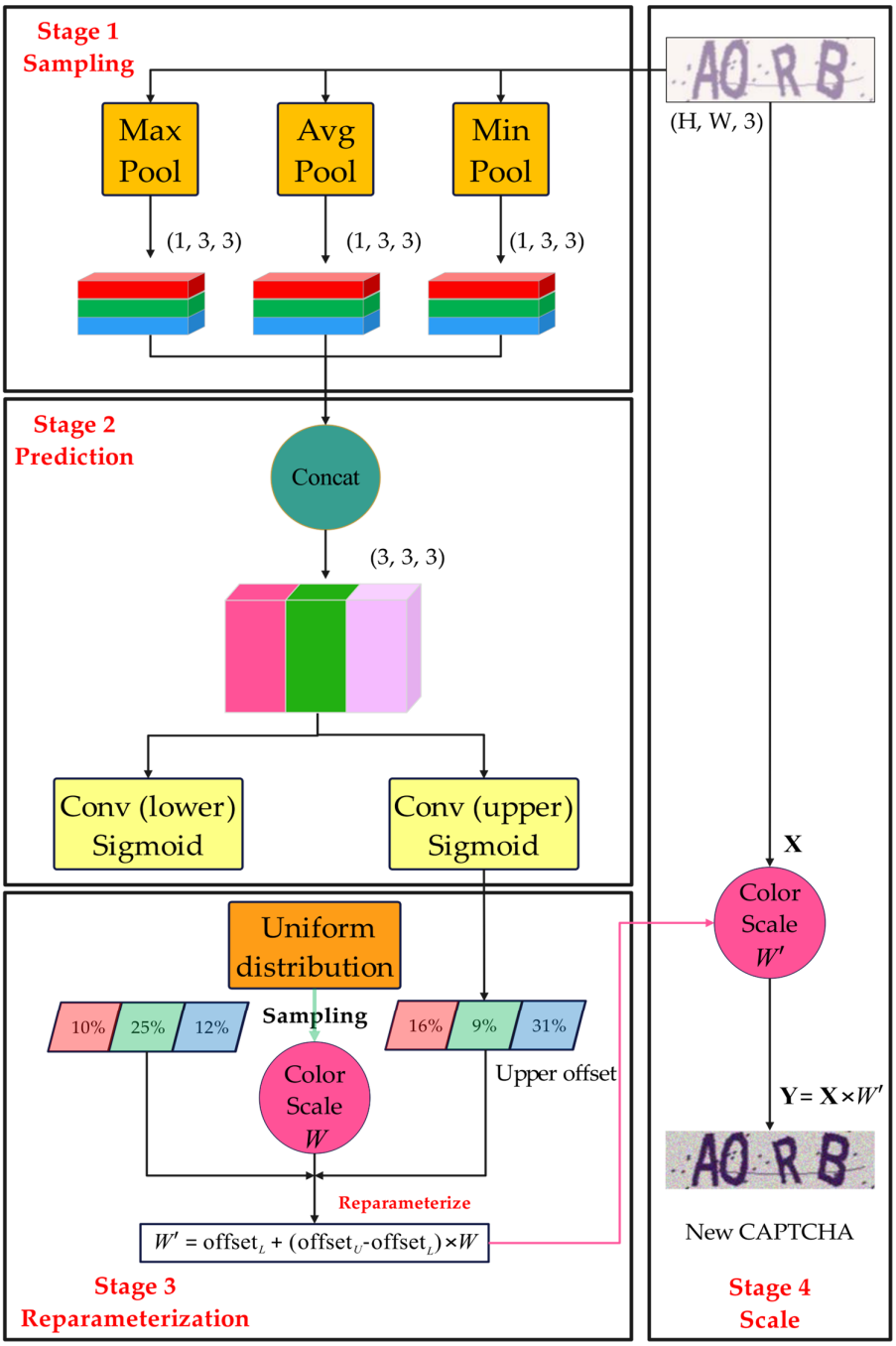


Figure 5. The flowchart of VCS.

In the third stage, VCS adopts the two convolution layers, both of which have dimensions of 3 × 3 × 3 × 3. While the three adaptive pooling layers in the first stage do not contribute to PARAMs, they do increase the number of float point operations. If $K_{predict}$ denotes the kernel size of $F_{lower}^{3\times3}$ and $F_{upper}^{3\times3}$ ($K_{predict} = 3$), and C stands for the number of input channels and output channels ($C = 3$), respectively, and the total PARAMs of VCS are as shown in (5). Since different convolutions are used for the upper and lower offsets, respectively, there is a factor two in front.

$$PARAMs = 2 \times C \times C \times K_{predict} \times K_{predict} = 162 \tag{5}$$

Floating Point Operations Per Second (FLOPs) are used to evaluate the computation complexity. Excepting convolution layers and pooling layers, sigmoid functions also contribute to computation,

 doi:10.20944/preprints202409.2128.v1

which has the same dimension of output, $C \times 1 \times 1$. Let H and W denote the height and width of the input CAPTCHA, respectively. In the convolution layer, we add and multiply each parameter once independently, resulting in a value twice that of the kernel size. The three pooling operations require the same number of computations as the number of pixels in the CAPTCHA, resulting in three times the image resolution. Finally, there are three more computations per sigmoid function, as shown in Equation (6).

$$FLOPs = 2 \times PARAMs\ (VCS) + 6 \times\ HW + 2 \times\ C = 330 + 6HW \tag{6}$$

3.2. Sim-VCS

To adapt a portion of the CAPTCHA recognition network deployed on mobile, we consider further reducing the computational complexity of the model. We propose a simplified version with low computational complexity. For this algorithm, we replace the three pooling layers with three 3 × 3 constant tensors with pixel values all one's, which means that the upper and lower offsets do not depend on every different input image, thus greatly reducing the amount of computation. However, this implies that each image receives the same offsets, derived from the entire dataset, regardless of the individual image's color distribution. Sim-VCS maintains the same PARAMs as VCS because it still retains the two prediction convolutional layers, which ensures that the offsets are learnable, as shown in (7).

$$PARAMs = 2 \times C \times K\_predict \times K_{predict} = 162 \tag{7}$$

The reduction in FLOPs is mainly due to the removal of the three adaptive pooling layers, as the number of operations in the pooling operation is proportional to the size of the input image. Therefore, the FLOPs of Sim-VCS are only related to PARMAs of the convolution module and the activation function, as illustrated in (8). It can be seen that the difference in FLOPs of VCS and Sim-VCS is 6HW, which is dependent on the size of the input image.

$$FLOPs = \ 2 \times PARAMs\ (SimVCS) + 2 \times C = 330 \tag{8}$$

Figure 6 shows that Sim-VCS and VCS differ in that the former is a statistic that is dependent on the input picture, while the latter is a constant.

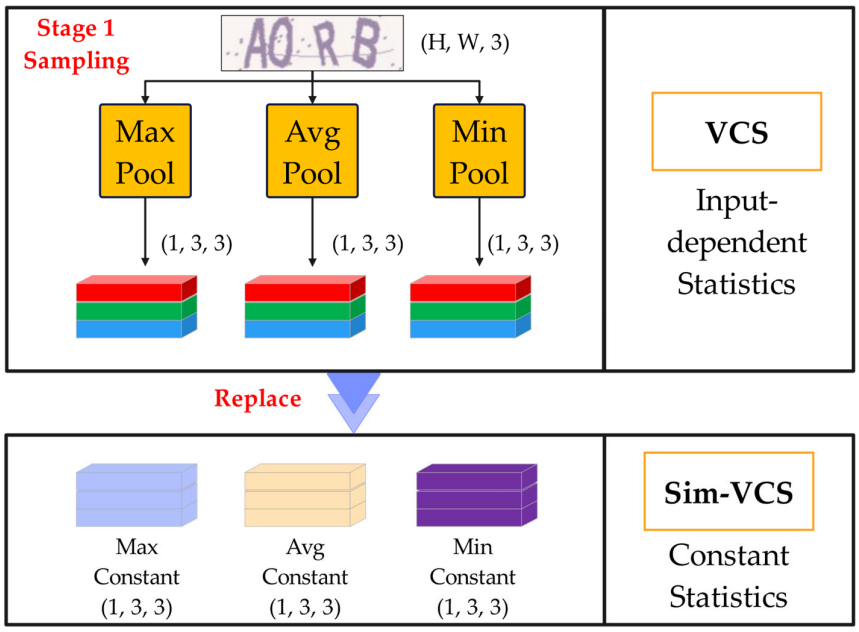


Figure 6. The difference between Sim-VCS and VCS.

 doi:10.20944/preprints202409.2128.v1

3.3. Dilated-VCS

In the first stage, VCS samples the mean, maximum, and minimum statistics of the input image. However, in some conditions, these statistics cannot fully reflect the color changes in the image. To obtain more detailed image color distribution, this study proposed a feature extraction scheme based on dilated convolution, as shown in Figure 7. It is worth noting that the density of sampling points is inversely proportional to the dilation factor. When the dilation factor is 1, it is a full sampling of the input image. However, due to the large amount of interference in the CAPTCHA image, the higher the sampling density, the higher the risk of overfitting. Therefore, it is necessary to balance between sampling density and overfitting, which can be reduced by adding a Dropout layer.

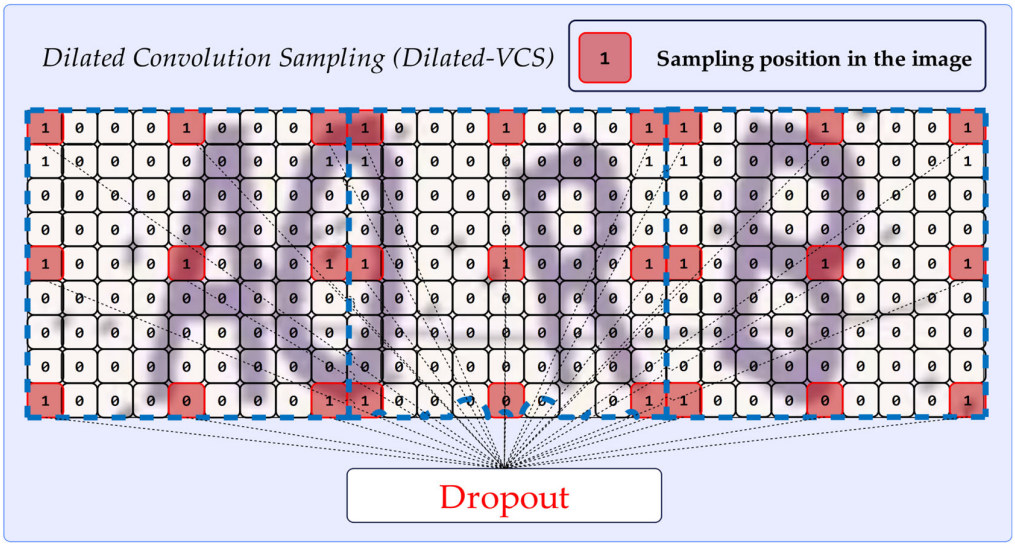


Figure 7. The dilated convolution-based sampling of Dilated-VCS.

By constructing a dilated convolution, we can randomly obtain the pixel values of certain positions in the image. Dilated convolution gives more details of the input image than using a pooling layer to extract the maximum, minimum, and mean values. The benefit of using dilated convolution instead of regular convolution is that there is a lot of interference in the CAPTCHA, and using regular convolution can easily lead to overfitting, thereby reducing the model's generalization effect. If we let the kernel size of dilated convolution module K_s be equal to stride and let d stand for dilated rate. The PARAMs of Dilated-VCS are determined only by K_s and $K_{predict}$ and the formula for the PARAMs is shown in Equation (9).

$$PARAMs = 2 \times C^2 \times (K_s)^2 + 2 \times C^2 \times \left(K_{predict}\right)^2 = 18K_s^2 + 162 \tag{9}$$

The FLOPs for the convolution operation are twice as many as the convolution kernel parameters. However, the output dimension of the sampled convolution is a 3 x 3 feature, therefore, the sampled convolution layer of the FLOPs should also be multiplied by a factor of 9, as illustrated by (10).

$$FLOPs = 36 \times C^2 \times (K_s)^2 + 4 \times C^2 \times \left(K_{predict}\right)^2 + 2 \times C = 324K_s^2 + 330 \tag{10}$$

Suppose the input image size is (64, 192), and we consider the adaptive pool as a special convolution with a kernel size of (64, 64). Table 2 shows that as the kernel size of Dilated-VCS increases, its FLOPs and PARAMs increase by a factor of square. Thus, choosing a suitable kernel size guarantees the performance of Dilated-VCS while its FLOPs do not exceed that of VCS.

 doi:10.20944/preprints202409.2128.v1

Table 2. Comparison of VCS and Dilated-VCS.

Type	Input	K_s	Dilated Rate	Stride	Padding	PARAMs	FLOPs
Dilated-VCS	(64,192)	(4,4)	(7,21)	(22,64)	(1,0)	450	5514
Dilated-VCS	(64,192)	(8,8)	(3,9)	(22,64)	(1,0)	1314	21066
Dilated-VCS	(64,192)	(22,22)	(1,3)	(22,64)	(1,0)	8874	156978
VCS	(64,192)	(64,64)	(1,1)	(64,64)	(0,0)	162	73728

3.4. AE-LSKA

One of the major difficulties in text CAPTCHA recognition is that the model is highly susceptible to many disturbances during the training process, leading to learning too many invalid local details. In this paper, we embed LSKA in the convolutional module of the existing recognition model so that the model pays more attention to the overall information of the image and reduces the impact of local interference on the recognition results. LSKA decomposes 2D convolution kernels into cascaded 1D kernels in horizontal and vertical directions, reducing computational complexity and memory usage while enabling the use of large kernels. This design enhances model efficiency and robustness, focusing on object shapes over textures, suitable for vision tasks.

To expand the receptive field, many models integrate larger convolutions. We first consider replacing these large kernel convolutions, which are typically 5x5 in size, with an AE module consisting of two small 3x3 kernel convolutions in series. This structure not only reduces the number of PARAMs and FLOPs but also compresses the number of intermediate channels, which effectively reduces the forward propagation of interfering information. The LSKA with a residual connection follows the AE, prioritizing important areas by boosting the receptive field, while the latter speeds up the gradient propagation.

Figure 8 shows that we replaced the 5x5 convolution in the original model, with AE-LSKA modules with different receptive fields. It should be noted that Conv stands for convolution, DW-Conv stands for depth-wise convolution, and DW-D-Conv denotes depth-wise dilated convolution. The receptive field of the LSKA is determined by the kernel size as well as the dilation. For the 5x5 convolution layer, we constructed five receptive fields of 5, 7,11, 15, and 23, respectively.

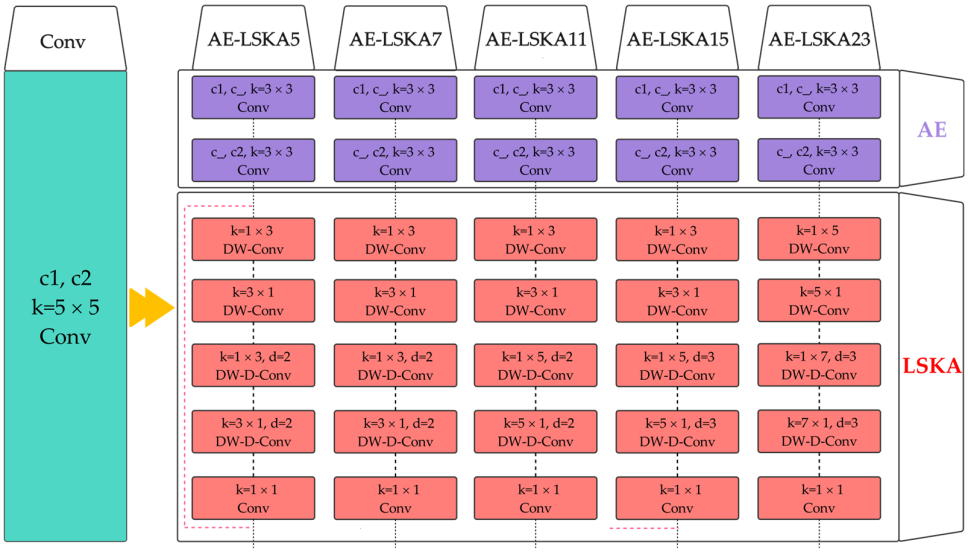


Figure 8. The AE-LSKAs with different dilated kernels and receptive fields.

 doi:10.20944/preprints202409.2128.v1

Table 3 compares the original 5x5 convolution with AE-LSKAs with different parameters. Here, for simplicity, we assume that the number of input and output channels are both C, the input and output feature sizes are constant, and the stride is one. We can disregard the sizes of the output features because they are identical. Typically, the number of channels C is not less than 3, so the 5x5 convolution is much larger than all the AE-LSKA modules in the table, both in terms of PARAMs and FLOPs.

Table 3. PARAMs and FLOPs of AE-LSKAs.

/	Conv 5x5	AE-LSKA5	AE-LSKA7	AE-LSKA11	AE-LSKA15	AE-LSKA23
Kernel (LSKA)	/	$3+3$	$3+3$	$3+5$	$3+5$	$5+7$
Dilation	/	2	2	2	3	3
PARAMs	$25C^2$	$(10C^2+12C)$	$(10C^2+12C)$	$(10C^2+16C)$	$(10C^2+16C)$	$(10C^2+24C)$
FLOPs	$50C^2$	$(20C^2+24C)$	$(20C^2+24C)$	$(20C^2+32C)$	$(20C^2+32C)$	$(20C^2+48C)$

4. Results and Discussions

The experiments were conducted using an NVIDIA GeForce RTX 3060 12GB GPU, an Intel Core i5-8265U CPU @ 1.60GHz 1.80GHz, PyTorch version 2.2, and Python version 3.11.9, as shown in Table 4. The experiments use the M-CAPTCHA dataset with randomly chosen 5000 samples (https://www.kaggle.com/datasets/sanluo/mcaptcha) and a P-CAPTCHA dataset with 3000 samples (generated using the Python captcha library). All these datasets are divided into training and validation sets according to 8:2 and trained with Deep-CAPTCHA and Adaptive-CAPTCHA for 130 epochs. Also, in addition to PARAMs and FLOPs, we focus on Average Attack Success Rate (AASR) and loss during model training and validation, as shown in Table 4.

Table 4. Hardware and software for experiments.

Name	Model/Dataset/Specification	Version
Graphics Processing Unit (GPU)	NVIDIA GeForce RTX 3060 12GB	-
Central Processing Unit (CPU)	Intel(R) Core (TM) i5-8265U CPU @ 1.60GHz 1.80 GHz	-
PyTorch	-	2.2
Python	-	3.11.9
Text CAPTCHA Datasets	M-CAPTCHA (5000 images), P-CAPTCHA (3000 images)	-
EPOCHs	130	
Text CAPTCHA Recognizers	Deep-CAPTCHA, Adaptive-CAPTCHA	-
Metrics	AASR, loss, PARAMs, FLOPs	-

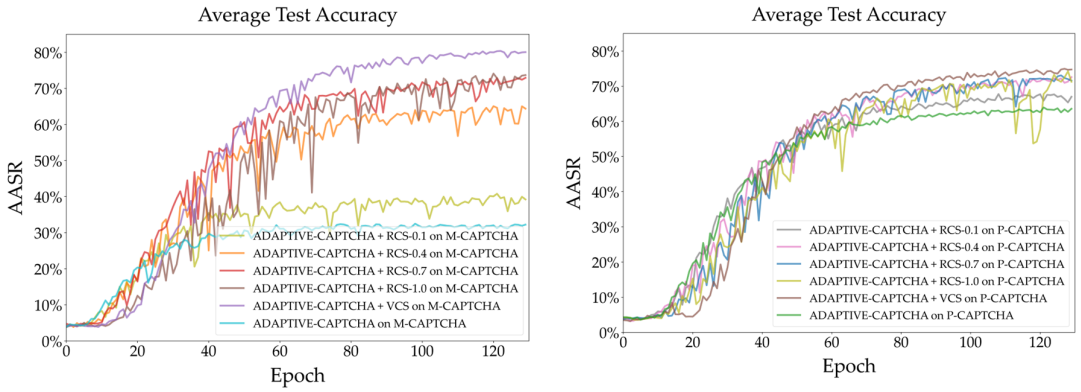
4.1. Experimental Analysis of VCS

Figure 9 (a) shows the test results of the Adaptive-CAPTCHA on the M-CAPTCHA dataset. According to the figure, the accuracy of VCS shows a steady upward trend throughout the test process and finally reaches nearly 80%, which is the highest among all algorithms. Among the RCSs with different thresholds, RCS-1.0 and RCS-0.7 have relatively higher accuracy, demonstrating that for complex verification codes like M-CAPTCHA, more changes can make the powerful Adaptive-CATPCHA model better trained. Furthermore, in the absence of a color enhancement algorithm, the AASR is at its lowest, approximately 35%, which is 45 percentage points less than the highest.

Figure 9 (b) demonstrates that the model also maintains similar results on P-CAPTCHA, with VCS having the best performance, more than 70%. The AASR is only about 60% without any color shift augmentation, indicating that color shift is an effective way to boost recognition accuracy for strong recognizers. In addition, we find that the RCS curves on both datasets have greater jitter, showing that the stability of RCS is not as stable as that of VCS. In addition, the amount of

 doi:10.20944/preprints202409.2128.v1

performance improvement by VCS on the P-CAPTCHA is smaller compared to M-CAPTCHA due to the fewer disturbances and patterns of P-CAPTCHA.

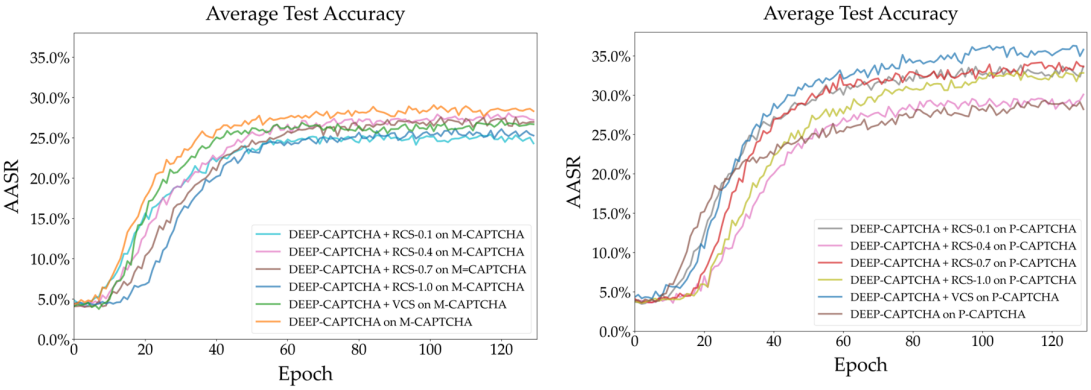


(**a**) (**b**)

Figure 9. AASR comparison between VCS and RCS using Adaptive-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Figure 10 (a) shows that the AASR of all algorithms is less than 30% using Deep-CAPTCHA on the M-CAPTCHA, in which case the best performance is achieved without any color shift enhancement. This means that when the weak recognizer is tested on a complex dataset, the color shift algorithms will increase the learning difficulty of the model, thereby reducing recognition accuracy. Therefore, the color shifting algorithms are related to the model and dataset, which should be adopted only when the original dataset cannot meet the model training requirements. However, VCS still performs close to the best RCS.

Figure10 (b) shows the performance of Deep-CAPTCHA on the P-CAPTCHA, the AASR of VCS is about 35%, which is significantly higher than RCS by at least three percentage points and seven percentage points higher than without using any color offset. This shows that the VCS is effective for simple datasets for both weak and strong models.

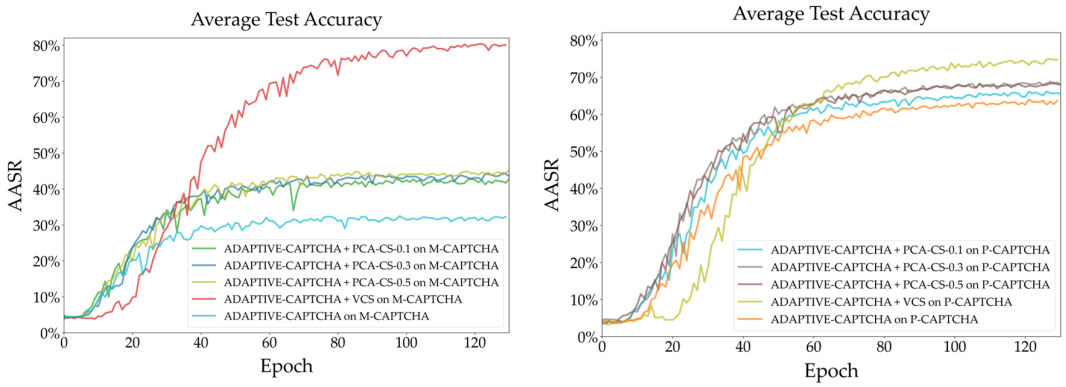


(**a**) (**b**)

Figure 10. AASR comparison between VCS and RCS using Deep-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Figure 11 (a) and (b) show the performance of Adaptive-CAPTCHA after integrating VCS and PCA-CS on M-CAPTCHA and P-CAPTCHA, respectively. The AASR with PCA-CS outperforms the model without any color bias and has a faster convergence rate relative to VCS. However, compared to VCS, even the AASR of the best-performing PCA-CS is still about 40 and 5 percentage points lower on both datasets, respectively.

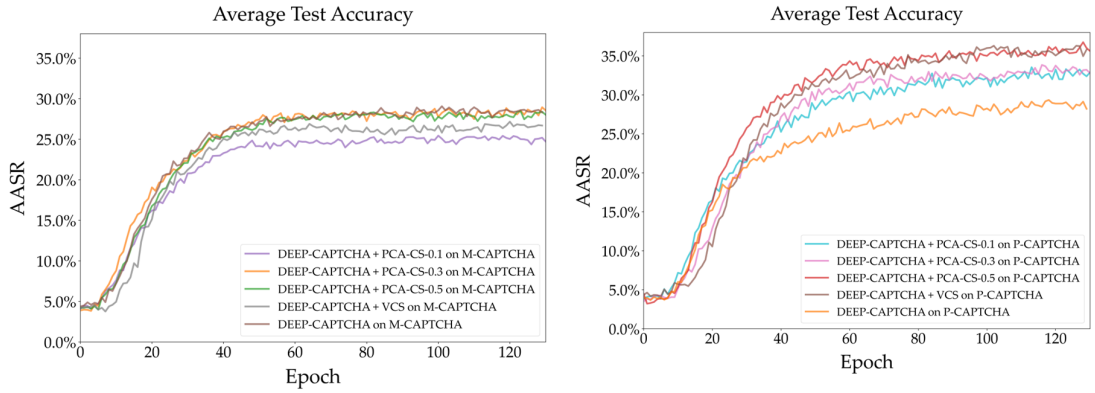
 doi:10.20944/preprints202409.2128.v1



(**a**) (**b**)

Figure 11. AASR comparison between VCS and PCA-CS using Adaptive-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Regardless of Figure 12 (a) or 12 (b), PCA-CS-0.5 exhibits nearly the best AASR, indicating that PCA-CS is robust to weak recognizers and can guarantee the generalization of features in the dataset. This is due to the consistency of the channel color distribution maintained by PCA-CS through singular value decomposition.



(**a**) (**b**)

Figure 12. AASR comparison between VCS and PCA-CS using Deep-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Table 5 compares the AASRs of Dilated-VCS with various dropout and dilated convolutional kernels, demonstrating that all algorithms achieved an AASR of 99.9% when trained on both data using Adaptive-CAPTCHA. These results suggest that the large amount of interference in the CAPTCHA leads the strong learner to be prone to overfitting. In this case, selecting a small kernel coupled with a large dropout helps to enhance the generalization ability on the validation set, with Dilated-VCS (kernel = 4, dropout = 0.3) achieving 72% on M-CAPTCHA. In addition, P-CAPTCHA has a larger optimal convolution kernel compared to M-CAPTCHA, which may be related to the content and color distribution of the dataset. VCS has the best overall performance, followed by Dilated-VCS and Sim-VCS, and their computational complexity is quite the opposite.

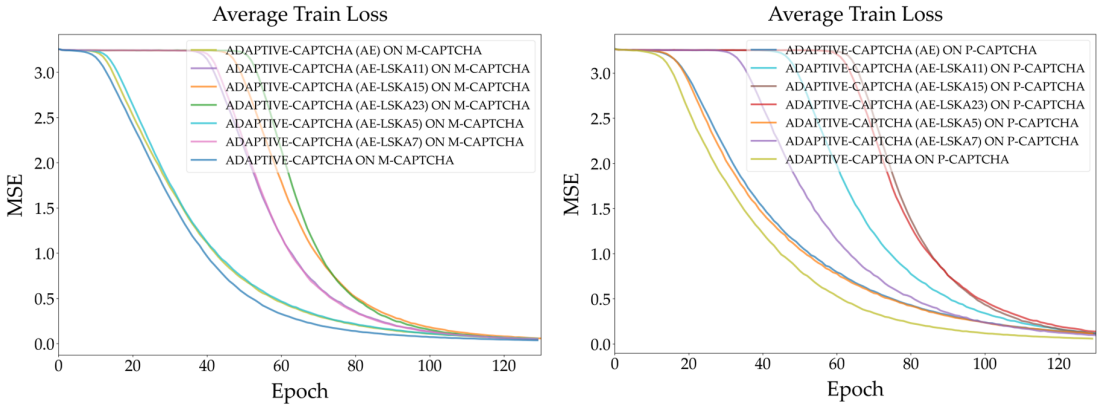
 doi:10.20944/preprints202409.2128.v1

Table 5. AASRs of Dilated-VCSs with different settings.

Algorithm	Dilated Kernel	Dropout	Model	Train | Test AASR (%) P-CAPTCHA	Train | Test AASR (%) M-CAPTCHA
Dilated-VCS	4	0.0	Deep-CAPTCHA	96.9|37.0	95.0|26.2
	4	0.3		97.2|36.1	95.6|26.6
	8	0.0		96.4|37.8	**94.6|27.2**
	8	0.3		97.3|37.9	94.8|25.7
	22	0.0		97.2|36.8	93.3|25.2
	22	0.3		**97.0|39.1**	93.7|24.0
	4	0.0	Adaptive-CAPTCHA	99.9|73.2	99.9|69.2
	4	0.3		99.9|73.1	**99.9|72.0**
	8	0.0		99.9|71.6	99.9|70.1
	8	0.3		**99.9|74.0**	99.9|69.0
	22	0.0		99.9|67.6	99.9|57.6
	22	0.3		99.9|68.2	99.9|55.6
VCS	-	-	Deep-CAPTCHA	94.0|35.8	93.7|26.6
	-	-	Adaptive-CAPTCHA	99.9|74.5	99.9|78.0
Sim-VCS	-	-	Deep-CAPTCHA	95.6|35.6	93.4|26.2
	-	-	Adaptive-CAPTCHA	99.9|72.4	99.9|76.9

4.2. Experimental Analysis of AE-LSKA

Figures 13 (a) and (b) show that AE-LSKAs with a larger dilated kernel have a slower learning rate on both datasets during training Adaptive-CAPTCHA, but after enough epochs, they can all converge.

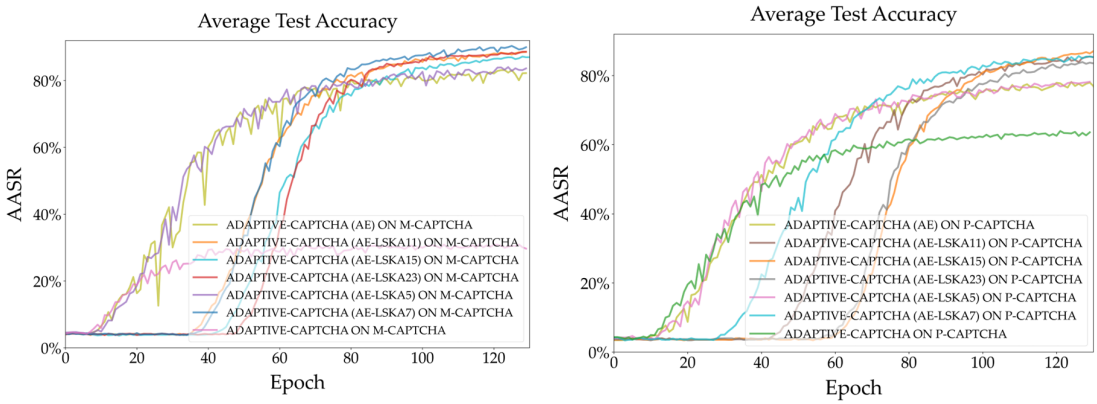


(**a**) (**b**)

Figure 13. Learning process AE-LSKAs with different dilated kernels integrating into Adaptive-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Figure 14 (a) shows that Adaptive-CAPTCHA has an AASR of nearly 88% after integrating AE-LSKA (K = 7), which is nearly 49 percentage points higher than the original model and eight percentage points higher than the model that only integrates AE. The best performance on the P-CAPTCHA dataset is AE-LSKA (k = 15), which suggests that the optimal dilation kernel size varies across datasets, as shown in Figure 14 (b). Moreover, the AE-LSKA curves with larger convolution kernels are steeper on both datasets.

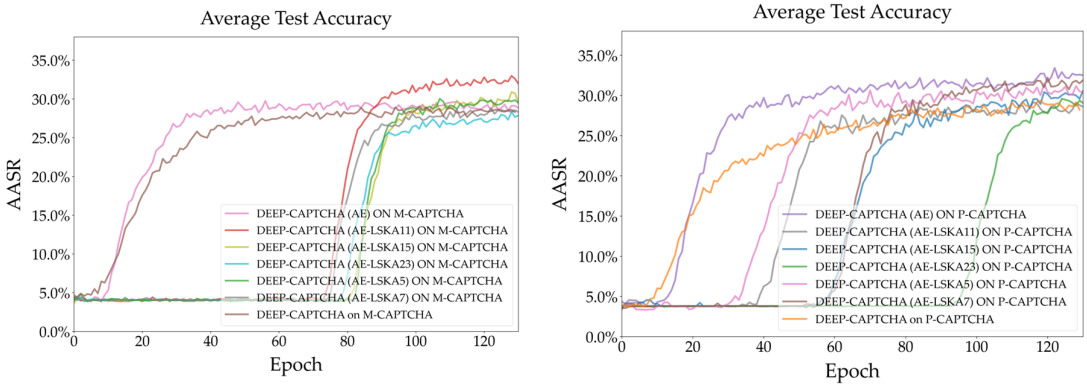
 doi:10.20944/preprints202409.2128.v1



(**a**) (**b**)

Figure 14. Testing accuracy comparison of AE-LSKAs with different dilated kernels integrating into Adaptive-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Figure 15 (a) and (b) show that when AE-LSKA is integrated into Deep-CAPTCHA compared to Adaptive-CAPTCHA, the AASR curves become steeper, suggesting that LSKA improves global attention while making the model more difficult to learn. AE-LSKA (k = 7) and AE-LSKA (k = 15) on the two datasets achieved approximately 86% and 83% accuracy, respectively, an improvement of roughly 60 and 22 percentage points over the original model.

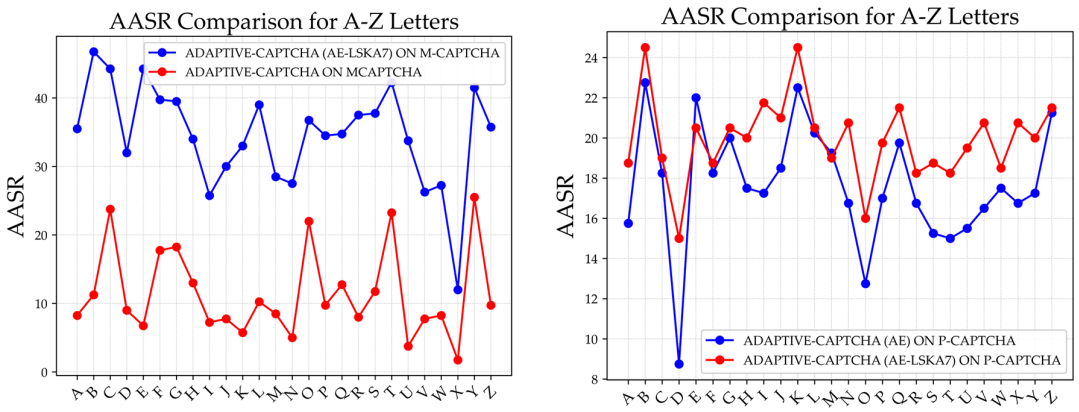


(**a**) (**b**)

Figure 15. Testing accuracy comparison of AE-LSKAs with different kernels integrating into Deep-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Figure 16 (a) illustrates the AASR improvements achieved for the recognition of individual characters from A to Z on the M-CAPTCHA dataset when the Adaptive-CAPTCHA model is enhanced with AE-LSKA (k = 7). Similarly, Figure 16 (b) presents the results for the same model on P-CAPTCHAs, where it is evident that most characters experience a notable enhancement in recognition accuracy.

 doi:10.20944/preprints202409.2128.v1



(**a**) (**b**)

Figure 16. Individual character accuracy comparison using Adaptive-CAPTCHA (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Table 6 provides an in-depth comparison of an AE integrated with various attention modules tested using two distinct CAPTCHA recognition models: Deep-CAPTCHA and Adaptive-CAPTCHA. The comparison evaluates model PARAMs, FLOPs, and AASR. For the Deep-CAPTCHA model, the AE + Long-Short Key Attention (LSKA) (k = 11) configuration achieves a high AASR of 32.9%, which is the second-highest performance among the tested architectures. The highest AASR is 48.9%, observed with the AE + PNA configuration; however, this comes at the cost of significantly higher computational complexity with 379.51M FLOPs. In contrast, the AE + LSKA provides a more balanced trade-off with 178.38M FLOPs, indicating enhanced efficiency when computational resources are a consideration. In the Adaptive-CAPTCHA model, AE + LSKA maintains its strong performance. Specifically, the AE + LSKA (k = 7) configuration achieves an impressive AASR of 89.8% with 229.59M FLOPs, making it the top-performing configuration, surpassing other attention modules in both accuracy and computational efficiency.

Overall, while AE + PNA offers the highest accuracy for Deep-CAPTCHA, the AE + LSKA configurations provide an optimal balance of accuracy and computational demand across both CAPTCHA recognition models. This shows that LSKA can improve the performance of AE-based CAPTCHA solvers, providing strong solutions that can work with a range of computing limitations.

Table 6. Comparison of AE with attention modules on M-CAPTCHA dataset.

Algorithm	Dataset	PARAMs	FLOPs	AASR (%)
AE + LSKA (k=7)	Deep-CAPTCHA	6.41M	176.02M	28.4
AE + LSKA (k=11)		6.41M	178.38M	32.9
AE + CBAM (ratio=8)		6.40M	148.37M	25.0
AE + CBAM (ratio=16)		6.40M	148.37M	31.2
AE + ECA (ratio=2)		6.40M	146.72M	26.1
AE + ECA (ratio=4)		6.40M	146.72M	21.1
AE + GC (ATT + ADD)		6.41M	146.78M	28.7
AE + GC (AVG + MUL)		6.41M	146.73M	23.8
AE + SA (groups=8)		6.40M	146.43M	25.4
AE + SA (groups=16)		6.40M	146.45M	22.9
AE + SE (ratio=8)		6.40M	146.72M	20.9
AE + SE (ratio=16)		6.40M	146.72M	23.0
AE + PNA		6.48M	379.51M	48.9
AE + LSKA (k=7)	Adaptive-CAPTCHA	3.39M	229.59M	89.8
AE + LSKA (k=11)		3.39M	232.10M	88.3

 doi:10.20944/preprints202409.2128.v1

AE + CBAM (ratio=8)		3.31M	195.32M	85.4
AE + CBAM (ratio=16)		3.30M	195.29M	77.3
AE + ECA (ratio=2)		3.29M	193.60M	82.6
AE + ECA (ratio=4)		3.29M	193.60M	85.9
AE + GC (ATT + ADD)		3.38M	193.74M	66.3
AE + GC (AVG + MUL)		3.38M	193.69M	81.8
AE + SA (groups=8)		3.29M	193.29M	84.3
AE + SA (groups=16)		3.29M	193.31M	84.3
AE + SE (ratio=8)		3.31M	193.62M	84.0
AE + SE (ratio=16)		3.30M	193.61M	79.7
AE + PNA		4.27M	489.68M	24.4

5. Conclusions

In conclusion, this study introduces two novel algorithms aimed at enhancing the effectiveness of text CAPTCHA evaluation systems. The first contribution involves developing a VCS algorithm specifically for color data augmentation in text CAPTCHAs, along with its two simplified variants, Sim-VCS and dilated-VCS. We have conducted a detailed analysis of their mathematical foundations and algorithmic processes, providing a comprehensive understanding of their operational mechanisms. The second contribution involves the integration of an autoencoder with large separable kernel attention, termed AE-LSKA, which serves as a replacement for convolutional modules in model architectures. Experimental evaluations demonstrate that, compared to existing color augmentation algorithms such as RCS and PCA-CS, the VCS algorithm significantly enhances the AASR of the strong and weak models. Additionally, the AE-LSKA consistently improves model performance across all datasets tested. It not only surpasses most attention mechanisms in terms of AASR but also achieves an optimal balance between accuracy and computational efficiency as measured by FLOPs. These advancements underline the potential of VCS and AE-LSKA as powerful tools in the domain of text CAPTCHA recognition, offering a dual approach that targets both data augmentation and architecture optimization. The findings suggest substantial improvements in verification processes, paving the way for more robust and efficient models in practical applications.

Author Contributions: Xing Wan carried out all the experiments and wrote the first draft of the paper, Dr. Fazlina Ahmat Ruslan provided guidance on the innovations and the overall structure, and Prof. Juliana Johari revised the first draft of the paper.

Data Availability Statement: Data Availability Statement: The data are publicly available.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Setiawan, A.B.; Sastrosubroto, A.S. Strengthening the Security of Critical Data in Cyberspace, a Policy Review. In Proceedings of the 2016 International Conference on Computer, Control, Informatics and its Applications (IC3INA); October 2016; pp. 185–190.
2. von Ahn, L.; Blum, M.; Hopper, N.J.; Langford, J. CAPTCHA: Using Hard AI Problems for Security. In Proceedings of the Advances in Cryptology — EUROCRYPT 2003; Biham, E., Ed.; Springer: Berlin, Heidelberg, 2003; pp. 294–311.
3. Yan, J.; El Ahmad, A.S. Usability of CAPTCHAs or Usability Issues in CAPTCHA Design. In Proceedings of the Proceedings of the 4th symposium on Usable privacy and security - SOUPS '08; ACM Press: Pittsburgh, Pennsylvania, 2008; p. 44.
4. Alsuhibany, S.A. Evaluating the Usability of Optimizing Text-Based CAPTCHA Generation. *Int. J. Adv. Comput. Sci. Appl. IJACSA* **2016**, *7*, doi:10.14569/IJACSA.2016.070823.
5. Wang, J.; Qin, J.; Xiang, X.; Tan, Y.; Pan, N.; College of Computer Science and Information Technology, Central South University of Forestry and Technology, 498 shaoshan S Rd, Changsha, 410004, China CAPTCHA Recognition Based on Deep Convolutional Neural Network. *Math. Biosci. Eng.* **2019**, *16*, 5851–5861, doi:10.3934/mbe.2019292.
6. Guerar, M.; Verderame, L.; Migliardi, M.; Palmieri, F.; Merlo, A. Gotta CAPTCHA 'Em All: A Survey of 20 Years of the Human-or-Computer Dilemma. *ACM Comput. Surv.* **2022**, *54*, 1–33, doi:10.1145/3477142.

 doi:10.20944/preprints202409.2128.v1

7. Chellapilla, K.; Larson, K.; Simard, P.Y.; Czerwinski, M. Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs). In Proceedings of the Human Interactive Proofs; Baird, H.S., Lopresti, D.P., Eds.; Springer: Berlin, Heidelberg, 2005; pp. 1–26.
8. Zhang, J.; Sang, J.; Xu, K.; Wu, S.; Zhao, X.; Sun, Y.; Hu, Y.; Yu, J. Robust CAPTCHAs Towards Malicious OCR. *IEEE Trans. Multimed.* **2021**, *23*, 2575–2587, doi:10.1109/TMM.2020.3013376.
9. Wang, P.; Gao, H.; Guo, X.; Xiao, C.; Qi, F.; Yan, Z. An Experimental Investigation of Text-Based CAPTCHA Attacks and Their Robustness. *ACM Comput Surv* **2023**, *55*, 196:1-196:38, doi:10.1145/3559754.
10. Xing, W.; Mohd, M.R.S.; Johari, J.; Ruslan, F.A. A Review on Text-Based CAPTCHA Breaking Based on Deep Learning Methods. In Proceedings of the 2023 International Conference on Computer Engineering and Distance Learning (CEDL); June 2023; pp. 171–175.
11. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021 2021.
12. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications 2022.
13. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors 2022.
14. Walia, J.S.; Odugoudar, A. Vulnerability Analysis of Captcha Using Deep Learning. In Proceedings of the 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG); December 8 2023; pp. 1–7.
15. Wang, Z.; Wang, P.; Liu, K.; Wang, P.; Fu, Y.; Lu, C.-T.; Aggarwal, C.C.; Pei, J.; Zhou, Y. A Comprehensive Survey on Data Augmentation 2024.
16. Bursztein, E.; Martin, M.; Mitchell, J.C. Text-Based CAPTCHA Strengths and Weaknesses. In Proceedings of the Proceedings of the 18th Acm Conference on Computer & Communications Security (ccs 11); Assoc Computing Machinery: New York, 2011; pp. 125–137.
17. Mocanu, I.G.; Yang, Z.; Belle, V. Breaking CAPTCHA with Capsule Networks. *Neural Netw.* **2022**, *154*, 246–254, doi:10.1016/j.neunet.2022.06.041.
18. Shi, Y.; Liu, X.; Han, S.; Lu, Y.; Zhang, X. A Transformer Network for CAPTCHA Recognition. In Proceedings of the 2021 2nd International Conference on Artificial Intelligence and Information Systems; Association for Computing Machinery: New York, NY, USA, 2021; pp. 1–5.
19. Qing, K.; Zhang, R. An Efficient ConvNet for Text-Based CAPTCHA Recognition. In Proceedings of the 2022 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS); November 2022; pp. 1–4.
20. Noury, Z.; Rezaei, M. Deep-CAPTCHA: A Deep Learning Based CAPTCHA Solver for Vulnerability Assessment 2020.
21. Wan, X.; Johari, J.; Ruslan, F.A. Adaptive CAPTCHA: A CRNN-Based Text CAPTCHA Solver with Adaptive Fusion Filter Networks. *Appl. Sci.* **2024**, *14*, 5016, doi:10.3390/app14125016.
22. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. Acm* **2017**, *60*, 84–90, doi:10.1145/3065386.
23. Wang, X.; Yu, J. Learning to Cartoonize Using White-Box Cartoon Representations. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); June 2020; pp. 8087–8096.
24. Ishkov, D.O.; Terekhov, V.I. Text CAPTCHA Traversal with ConvNets: Impact of Color Channels. In Proceedings of the 2022 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE); March 2022; pp. 1–5.
25. Chen, S.; Guo, W. Auto-Encoders in Deep Learning—A Review with New Perspectives. *Mathematics* **2023**, *11*, 1777, doi:10.3390/math11081777.
26. Lau, K.W.; Po, L.-M.; Rehman, Y.A.U. Large Separable Kernel Attention: Rethinking the Large Kernel Attention Design in CNN. *Expert Syst Appl* **2024**, *236*, doi:10.1016/j.eswa.2023.121352.
27. Guo, M.-H.; Lu, C.-Z.; Liu, Z.-N.; Cheng, M.-M.; Hu, S.-M. Visual Attention Network. *Comput. Vis. Media* **2023**, *9*, 733–752, doi:10.1007/s41095-023-0364-2.
28. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale 2021.
29. Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; Xie, S. A ConvNet for the 2020s 2022.
30. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In Proceedings of the Computer Vision – ECCV 2018; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, 2018; pp. 3–19.
31. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 2011–2023, doi:10.1109/TPAMI.2019.2913372.
32. Zhang, Q.-L.; Yang, Y.-B. SA-Net: Shuffle Attention for Deep Convolutional Neural Networks. In Proceedings of the ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); June 2021; pp. 2235–2239.

 doi:10.20944/preprints202409.2128.v1

33. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks 2020.
34. Goyal, A.; Bochkovskiy, A.; Deng, J.; Koltun, V. Non-Deep Networks 2021.
35. Cao, Y.; Xu, J.; Lin, S.; Wei, F.; Hu, H. GCNet: Non-Local Networks Meet Squeeze-Excitation Networks and Beyond 2019.
36. Kingma, D.P.; Welling, M. An Introduction to Variational Autoencoders. *Found. Trends® Mach. Learn.* **2019**, *12*, 307–392, doi:10.1561/2200000056.