electronics

Article

Securing IoT Devices: A Robust and Efficient Deep Learning with a Mixed Batch Adversarial Generation Process for CAPTCHA Security Verification

Stephen Dankwa * and Lu Yang *

School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu 611371, China
* Correspondence: nabistephen@gmail.com (S.D.); yanglu@uestc.edu.cn (L.Y.)

Abstract: The Internet of Things environment (e.g., smart phones, smart televisions, and smart watches) ensures that the end user experience is easy, by connecting lives on web services via the internet. Integrating Internet of Things devices poses ethical risks related to data security, privacy, reliability and management, data mining, and knowledge exchange. An adversarial machine learning attack is a good practice to adopt, to strengthen the security of text-based CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), to withstand against malicious attacks from computer hackers, to protect Internet of Things devices and the end user's privacy. The goal of this current study is to perform security vulnerability verification on adversarial text-based CAPTCHA, based on attacker–defender scenarios. Therefore, this study proposed computation-efficient deep learning with a mixed batch adversarial generation process model, which attempted to break the transferability attack, and mitigate the problem of catastrophic forgetting in the context of adversarial attack defense. After performing K-fold cross-validation, experimental results showed that the proposed defense model achieved mean accuracies in the range of 82–84% among three gradient-based adversarial attack datasets.

Keywords: security; privacy; IoT; artificial intelligence; adversarial machine learning; deep learning; convolutional neural network; attacks; denoising autoencoder; CAPTCHA

check for updates

Citation: Dankwa, S.; Yang, L. Securing IoT Devices: A Robust and Efficient Deep Learning with a Mixed Batch Adversarial Generation Process for CAPTCHA Security Verification. *Electronics* **2021**, *10*, 1798. https://doi.org/10.3390/electronics10151798

Academic Editors: Sang-Soo Yeo and Damien Sauveron

Received: 25 June 2021
Accepted: 23 July 2021
Published: 27 July 2021

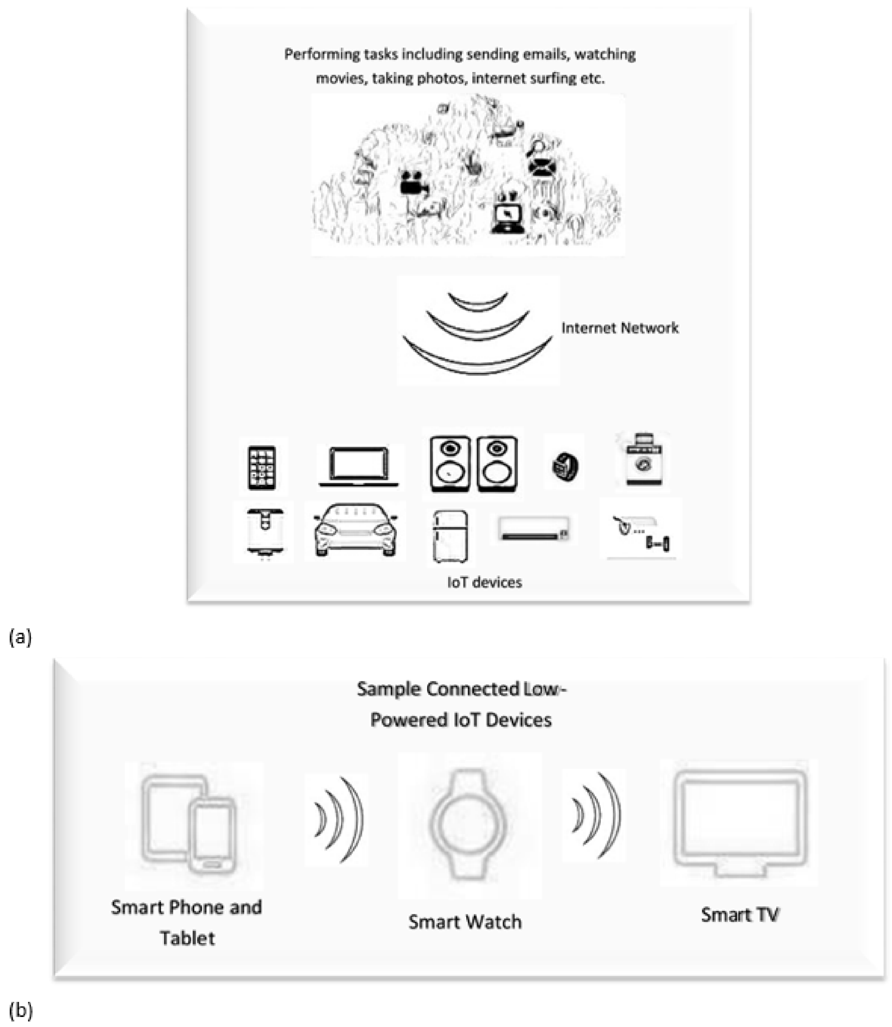
Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Internet of Things (IoT) devices refer to electronic devices (of any size) that have the capability of connecting to the internet, and collecting and sharing data, as illustrated in Figure 1a. The aim of IoT devices is to connect and exchange data with other devices and systems over the internet. Electronic devices, such as smartphones, make use of edge systems in the mobile IoT environment [1]. Smartphones are dominant IoT devices used to access cloud-based services, conveying sensory data related to human tasks.

They are embedded with sensors, including global positioning systems (GPSs), barometers, magnetometers, microphones, gyroscopes, accelerometers, etc. In regard to the internet, end users can perform their daily life activities online with their connected electronic devices. Most of the time, these sensors request end users to provide private information before proceeding with online activities. These sensors, working together, present a fairly complete picture of the end users' daily activities, which has privacy implications. Another interesting aspect is that smartphones can be connected to other IoT devices, such as smart TVs and smart watches, to share data, as illustrated in Figure 1b. The main goal of implementing these aforementioned IoT devices is to assist humans in performing their daily life activities with ease and comfort.



Performing tasks including sending emails, watching movies, taking photos, internet surfing etc.
Internet Network
IoT devices
(a)
Sample Connected Low-Powered IoT Devices
Smart Phone and Tablet
Smart Watch
Smart TV
(b)

Figure 1. (**a**) Sample IoT devices connected to the internet to perform tasks; (**b**) sample connected IoT devices through smartphones.

However, unfortunately, malicious attackers (or intruders) on the internet tend to write automated malicious applications to attack websites where end users perform their activities. These security threats sometimes put the information stored on connected IoT devices at risk. As a means of combating these malicious attacks, to protect electronic devices or smartphones and the end user's private data, cyber security practitioners generate CAPTCHAs as a security solution to differentiate machine bots and humans.

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is a universal automatic security program, based on cryptographic protocol; its underlying hardness assumption is centered on the problem of artificial intelligence (AI) [2]. The main uses of CAPTCHAs are to alleviate the impact of distributed denial of service (DDoS) attacks, prevent automatic registration of free email addresses or spam postings to forums, and to prevent automatic scrapping of web content [2]. CAPTCHAs are categorized into video, audio, images, and text-based. Among these four CAPTCHA categories, text-based CAPTCHAs are more famous and powerful security mechanisms used against automatic malicious programs. According to Hussain et al. [2], several approaches have been developed to secure text-based CAPTCHAs, such as text distortion, random lines, and overlapping of characters with foreground and background noise. However, these text-based CAPTCHAs with conventional background and foreground noise, as a form of security, have been compromised due to the advancement of deep learning (DL), and can be solved with high accuracy.

The aforementioned observation was proven by several researchers [3–14]. DL is beneficial in other fields, including target recognition [15], speech recognition [16,17], image recognition [18–20], image restoration [21–23], audio classification [24,25], object detection [26–30], scene recognition [31], etc., but it has been considered "bad news" in text-based CAPTCHAs, by penetrating their security and making them vulnerable.

Adversarial machine learning attacks may be highly necessary and recommended in the area of cyber security. Recently, AML attacks, where the robustness of an image classifier model is exposed to small perturbations, have gained more attention [32–56] in the artificial intelligence (AI) research community. An adversarial attack implants adversarial or vector noise in the input image, and is deliberately constructed to fool and deceive image classifier models.

Adversarial attacks can be seen in other fields of artificial intelligence, such as medicine and autonomous driving, threatening deep learning models (for example, when an autonomous vehicle misjudges a stop sign on the road, as a result of the deformation of the sign by malicious attackers through AML attacks). Meanwhile, to the human eye, the stop sign may look normal, but to the deep learning model, it may represent a different image all together due to adversarial perturbations, which may lead to serious accidents and threats to human safety. Moreover, as a result, improving the robustness of deep learning models against adversarial attacks has been a popular topic among researchers.

However, when it comes to the cyber security research community, an adversarial attack can be considered good news, as it can be used as a security mechanism to strengthen CAPTCHAs to deceive deep learning models, to prevent malicious attacks or intrusions, and improve security. The rise of adversarial ML attacks is promising attack measures used to strengthen the security of text-based CAPTCHAs, but there is little research on this when compared to the aforementioned research studies under conventional text-based CAPTCHAs. The most important aspect is that, after generating adversarial text-based CAPTCHAs, it is eminent to verify the strength of their security before they are deployed on a website or web applications, which is a goal to be accomplished in this current work.

The motivations behind this current work can be attributed to the following intrusion detection survey works, based on securing a smart grid from intrusion [57], network-based intrusion detection [58], fuzzy signature-based intrusion detection systems [59], anomaly-based intrusion detection systems [60], intrusion detection for in-vehicle networks [61], intrusion detection and prevention systems in digital substations [62], operational data-based intrusion detection for smart grids [63], defending network intrusion detection systems against adversarial evasion attacks [64], generative adversarial attacks against intrusion detection systems [65], adversarial machine learning in intrusion detection systems [66]. In addition, the present study work relates well with the work by Radanliev et al. [67], in terms of using artificial intelligence for IoT risk assessment. According to their studies, AI can be used in cyber risk analytics to enlighten organizational resilience and understand cyber risk. Their work focused on the identification of the role of AI in connected IoT devices to perform IoT risk assessment. Moreover, integration of IoT devices poses an ethical risk related to data security, privacy, reliability and management, data mining, and knowledge exchange [68].

In terms of performing adversarial CAPTCHA security verification, to the best of our knowledge, most researchers, after fooling their proposed CNN solver models using the adversarial attack algorithms, did not go further in regard to building defense models to test the strength of the generated adversarial text-based CAPTCHAs. Even with the works that built defense models, they did so by utilizing image processing filters, which are sometimes not effective against some adversarial attack algorithms. This current work is significant to the security community, in terms of verifying the security of adversarial text-based CAPTCHA to secure IoT devices, especially smartphones and smart TVs, which are commonly used by humans, to protect their data and privacy. The assumption is that if security mechanism measures are strengthened, the activities by malicious attackers on the internet, through web applications, can also be diminished, thereby securing IoT devices, in terms of protecting data and end users' privacy. This current study provides practical awareness of cyber risk assessment and proposes AI framework that can be utilized to perform security verification on web applications in which IoT devices are connected through the internet. Even though the focus is on CAPTCHAs for securing low-powered IoT devices, such as smartphones [1] and smart TVs [69], with respect to this current study, it can also be beneficial to more native scenarios and sensors, which are connected to web application programming interfaces (APIs).

Therefore, in this current research work, we propose techniques for adversarial text-based CAPTCHA image generation, by utilizing adversarial attack algorithms, including the Fast Gradient Sign Method (FGSM), Iterative Fast Gradient Sign Method (I-FGSM), and Momentum Iterative Fast Gradient Sign Method (MI-FGSM). These techniques will strengthen the security of conventional text-based CAPTCHAs by injecting a small adversarial or vector noise to generate adversarial CAPTCHAs that will retain the human perception, and then fool the Convolutional Neural Network (CNN) model.

After successfully deceiving the proposed CNN solver model, we proposed two defense models. The first method is a fundamental defense model, which is the CNN with a Denoising Autoencoder (DAE-CNN) to solve adversarial text-based CAPTCHAs. The purpose of the DAE network is to automatically pre-process the adversarial text-based CAPTCHA images, where the encoder will create compressed or latent representation of the CAPTCHA images, and then the decoder will reconstruct the original CAPTCHAs from the latent representation. This will improve the quality of the adversarial CAPTCHA images by removing the noise and, therefore, increase the accuracy of the CNN solver model. This technique is less robust, requires less computation, and cannot perform generalization, it only knows adversarial text-based by forgetting how the original CAPTCHA looks like. This scenario is sometimes referred to as catastrophic forgetting in the context of adversarial attack defense. In order to mitigate this problem, this study proposes a second defense model by introducing the Mixed Batch Adversarial Generation Process (MBAGP). This technique will make the defense model more robust and more computation-efficient, and then improve the model's capability to solve adversarial text-based CAPTCHAs. In addition, we introduce the optimal learning rate finder algorithm in conjunction with a cyclical learning rate policy to accelerate the defense model's convergence rate to improve recognition accuracy. The goal of the proposed defense model is, in its robustness, to remove the adversarial noise. This ethical practice will serve as a form of adversarial text-based CAPTCHA security verification system, which will in turn secure the aforementioned IoT devices when connected to web services, and then protect the end user's privacy as well.

The significance of this current study presents a way to minimize the activities of automatic malicious attackers. In addition, it also shows the cyber security research community how to perform adversarial text-based CAPTCHA security verification, using deep learning, based on the attacker–defender scenario in the context of IoT risk assessment. The following sections of this current work are organized as follows. In Section 2, the work presents related works; Section 3 presents the materials and the proposed methodologies used in this work; Section 4 presents the results of the proposed CNN solver model and the attackers; Section 5 presents the discussions, which includes comparisons with related study works, comparisons among the transfer learning results with the proposed defense models, and implications of the study work. Finally, Section 6 presents the conclusion and summary of this current work.

2. Related Works

According to Dankwa et al. [13], CAPTCHA deciphering is an ethical and indispensable form of a vulnerability assessment approach, to assess the strength of security in text-based CAPTCHAs before they are deployed on web applications.

With respect to previous works, Kopp et al. [70] performed text-based CAPTCHA breaking using the cluster algorithm and CNN, consisting of two steps, character localization and recognition. Their CNN architecture resembled the LeNet-5 architecture. Their localization, CNN, consisted of two (2) convolutional layers with six (6) and sixteen (16) 5×5 kernels, each followed by 2×2 max pooling layers, and a final fully-connected output layer. They concluded that the use of CNN is superior to multi-layered perceptron (MLP).

In the work by Ondrej et al. [71], after generating their bubble text-based CAPTCHA images, they then performed a comparative study based on traditional machine learning algorithms, such as Multi-Layered Perceptron (MLP), K-Nearest Neighbor, Support Vector Machines, and Decision Trees. They achieved a success rate of 89% based on all of their analyzed algorithms to solve the CAPTCHAs.

Zhao et al. [9] used the Python CAPTCHA library to generate both single-letter (e.g., "A"), and multi-letter (e.g., "GCKD") datasets. They generated 50,000 single-letter images, and 50,000 four-letter text-based CAPTCHA images. They, therefore, developed Support Vector Machines, K-Means algorithm, Convolutional Neural Network, and the VGG-19 pre-trained model with transfer learning. They concluded that, for the single-letter CAPTCHA images, the CNN model was superior to the SVM and K-Means models with an accuracy of 99%. The VGG-19 transfer learning model achieved 95% on the single-letter CAPTCHAs. The interesting aspect of the CNN model was that, as the number of letters increased from one to four, the model's accuracy reduced to 76%. They stated that, based on their estimation, there was a mistake on judging the location of each letter (that is, splitting for individual letters) in the CAPTCHA images, with distortions, lines, and a noisy and complex background, thereby decreasing their accuracy by 50% per letter.

In the work by Yang et al. [72], they proposed a CNN model to solve Chinese-character CAPTCHA images. The CNN architecture consisted of standard convolutional layers with 3×3 kernels, each followed by a ReLU activation function. They used four (4) max pooling layers, which were designed to subsample the resolution of the feature maps in the previous layer with 2×2 kernels. They then added a flattened layer to reduce each vector to one dimension, which was then fed to a fully-connected layer. Lastly, the final layer was fully connected to the previous dense layer with a softmax activation. Their proposed CNN model achieved 99.45% accuracy to solve the Chinese-character CAPTCHA images with noisy backgrounds.

Zahra et al. [73] proposed a standard CNN model to solve both numerical and alpha–numerical text-based CAPTCHAs with fie characters in each CAPTCHA image. They generated their dataset using the Python CAPTCHA library. Their network started with a convolutional layer with 32 input neurons, ReLU activation function, 5×5 kernels. They used max pooling with 2×2 kernels, 30% dropout rate, and an output layer with softmax activation. They compiled their network using the Adam optimizer. Their proposed CNN model achieved 98.94% and 98.31% for the numerical and alpha–numerical, respectively.

In the work by Shu et al. [74], they generated four-character text-based CAPTCHA images with distortions, rotation, and noisy background using the CAPTCHA open source python library tool. They built an end-to-end deep CNN–RNN model, which first constructed a deep Convolutional Neural Network based on residual network architecture to accurately extract the input CAPTCHA image features. In their study, via a constructed variant RNN network, which is a two-layer GRU network, the deep internal features of the text-based CAPTCHA are extracted and the final output sequence becomes the four-character CAPTCHA. Their proposed CNN–RNN model achieved 99% accuracy to solve text-based CAPTCHAs. This kind of architecture sometimes gives good results; however, it is relatively complex to train.

Hu et al. [14] used the Python script to generate five-character text-based CAPTCHA images. They then proposed a method based on VGG-Net CNN, with a transfer learning approach to solve their generated CAPTCHAs. Their study used an adaptive learning rate technique to accelerate the convergence rate of their model to solve the problem of over-fitting, and obtained local optimal solution. Their model achieved 96.5% to solve five-character text-based CAPTCHAs with background noise and adhesion distortions.

Osadchy et al. [75] proposed Deep-CAPTCHA using adversarial examples for CAPTCHA generation based on an object classification framework, which involved a large number of classes. They used the Fast Gradient Sign Method algorithm to generate single text-based CAPTCHA characters based on the MNIST digits dataset. Papernot et al. [76] proofed the limitations of deep learning in an adversarial setting. They exploited forward derivatives, which informed the learned behavior of Deep Neural Networks (DNNs) and generated adversarial saliency maps, allowing efficient exploration of the adversarial sample spaces. They used their algorithm to generate single text-based CAPTCHA characters based on the MNIST digits dataset.

Zhang et al. [77] showed the effect of adversarial examples on the robustness of CAPTCHAs. They used the VGG16, ResNet101, and the Inception-ResNet-V2 CNNs in conjunction with the Fast Gradient Sign Method and the Universal Adversarial Perturbations method [78] to generate their adversarial text-based CAPTCHAs. Kwon et al. [79] generated their adversarial text-based CAPTCHAs using the FGSM, I-FGSM, and the Deep-Fool algorithms. Their experiment results showed a 0% recognition rate with epsilon of 0.15 for FGSM, a 0% recognition rate with alpha of 0.1 with 50 iterations for I-FGSM, and a 45% recognition rate with 150 iterations for the Deep-Fool algorithm. Their CNN architecture consisted of five convolutional layers, five max pooling layers followed each of the convolutional layers, six ReLU activation layers, one fully connected layer, and an output layer with softmax activation.

In the work by Shao [80], the study proposed adversarial text-based CAPTCHA, named Robust Text CAPTCHA (RTC). The work then defended against their generated adversarial text-based CAPTCHAs using image processing filters with transfer learning techniques. For their CNN architectures, the study used pre-trained models including LeNet, AlexNet, GoogLeNet, VGG19, ResNet50, and DenseNet169. The observation results showed that the image processing filters, together with the pre-trained CNN models, were not able to withstand the generated adversarial text-based CAPTCHAs.

3. Materials and Methods

This section presents the dataset used in this current work, the adversarial attack algorithms, the proposed CNN solver architecture, the proposed fundamental defense DAE–CNN workflow, the cyclical learning rate policy algorithm, and the robust and efficient CNN with the Mixed Batch Adversarial Generation Process (MBAGP) defense workflow.

3.1. Data Acquisition and Description

In this current work, we generated our own text-based CAPTCHA image dataset, as shown in Figure 2a, using an open source Python library, since there are no publicly available text-based CAPTCHA data to be used. The text-based CAPTCHAs contain both digits (0–9) and uppercase English letters (A–Z), consisting of four characters. We then collected more complex real world text-based CAPTCHAs, as seen in Figure 2b.

In total, we generated 10,000 original text-based CAPTCHA datasets. We then divided the datasets into training and testing sets, representing 80% and 20%, respectively. Based on the training and testing sets of the original CAPTCHA dataset, we generated the adversarial examples using the FGSM, I-FGSM, and the MI-FGSM attack algorithms with an epsilon of 0.25, iteration of 10, and a decay factor of 1.0. A sample of the adversarial text-based CAPTCHA generated is shown in Figure 3c. In the perception of the human eye, Figure 3a,c look identical, but to the CNN solver model, they look like two entirely different images, which may lead to low accuracy for the model to solve. The Python library used in this work is available at (https://pypi.org/project/captcha/ (accessed on 6 May 2021)).

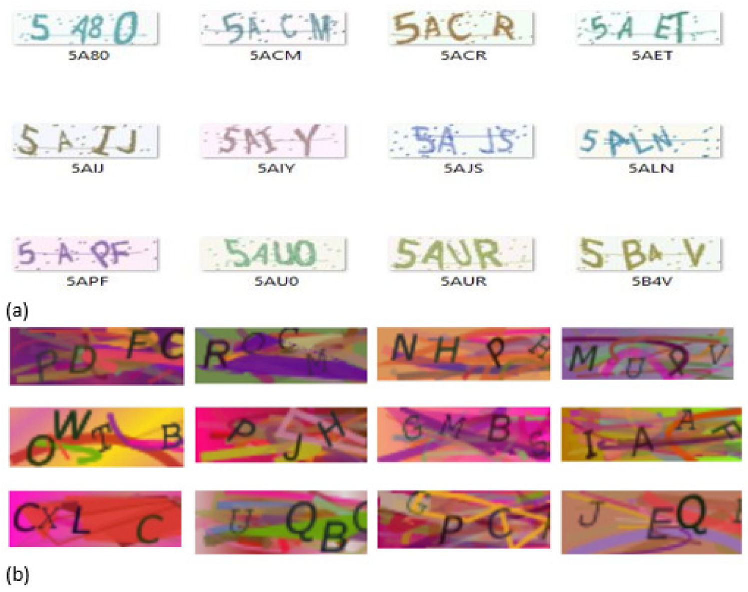


Figure 2. (**a**) Sample of the generated text-based CAPTCHAs with conventional noise using an open source Python library; (**b**) Sample of collected complex real world text-based CAPTCHA images.

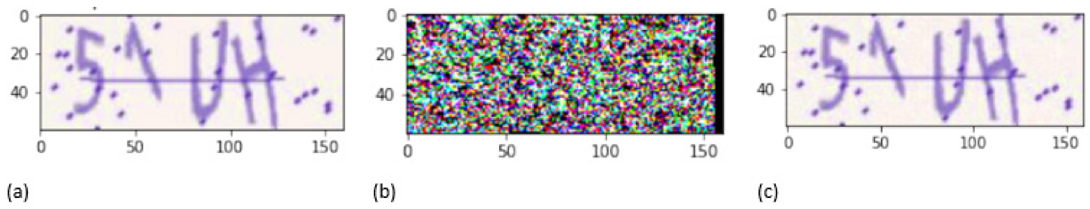


Figure 3. (**a**) Sample conventional text-based CAPTCHA generated; (**b**) sample adversarial or vector noise to be injected into the conventional text-based CAPTCHA; (**c**) sample of the generated adversarial text-based CAPTCHA.

3.2. *Adversarial Attack Algorithms*

This section presents the concept behind adversarial attack formulation and the gradient-based attack algorithms used in this current work.

3.2.1. Adversarial Attack Formulation

There is a trained Convolutional Neural Network (CNN) classifier given as

$$f_{\varnothing}\,(x) = y \tag{1}$$

where x is the image, $\varnothing$ is a parameter, and y is the probability distribution over the classes. The CNN assigns to x the class $C(x) = argmax_i y_i$. Therefore, for some, image x finds perturbation δ, such that,

$$C(x+\delta) \neq C(x) \tag{2}$$

$$C(x+\delta) = t \tag{3}$$

where (2) is untargeted adversarial attacking, and (3) is targeted adversarial attacking. Based on either (2) or (3), what we want is to represent $x + \delta$ to human as $C(x)$.

3.2.2. Fast Gradient Sign Method

In the Fast Gradient Sign Method (FGSM) [36], the gradient descent is changed from the original image x, based on the epsilon ε value, and then the adversarial example x^{Adv} is obtained through optimization. It is a simple technique to be used to generate adversarial examples with good performance. FGSM can be formulated as:

$$x^{Adv} = x + \varepsilon * Sign\,(\nabla_x J\,(\theta,\ x,\ y)) \tag{4}$$

3.2.3. Iterative Fast Gradient Sign Method

The Iterative Fast Gradient Sign Method (I-FGSM) [49] generates adversarial examples based on a given iteration on a target image classifier model. I-FGSM can be formulated as:

$$x_t^{Adv} = x_t^{Adv} + \alpha * Sign\left(\nabla_x J\left(\theta,\ x_t^{Adv},\ y\right)\right) \tag{5}$$

3.2.4. Momentum Iterative Fast Gradient Sign Method

In the Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [50], transferability is improved based on the past updating direction, which is termed as momentum.

$$g_{t+1} = \mu * g_t + \frac{\nabla_x J\left(\theta,\ x_t^{Adv},\ y\right)}{\|\nabla_x J\,(\theta,\ x_t^{Adv},\ y)\|} \tag{6}$$

$$x_t^{Adv} = x_t^{Adv} + \alpha * Sign\,(g_{t+1}) \tag{7}$$

Based on (4), (5), and (7), x^{Adv}, x_t^{Adv}, x_t^{Adv} are the adversarial examples, ε is the epsilon, θ is the classifier model, x is the original CAPTCHA image, y is the target labels, J is the loss function, μ is the momentum factor, g_t is the gradient direction of t. In this work, the adversarial text-based CAPTCHA examples were generated using (4), (5), and (7).

3.3. Basic Concepts behind Autoencoders (AEs)

As illustrated in Figure 4, an autoencoder (AE) is composed of an encoder and decoder. The variable X in the lower layer represents an input value, which is regenerated as an output value via the middle hidden layer. The middle hidden layer serves as a feature extractor. The regenerated value X′ aims to be a value similar to the input value X. The input vector X of the lower layer is calculated by the encoder and compressed to y as an output through the hidden layer, which is formulated as:

$$y = \alpha(Wx + b) \tag{8}$$

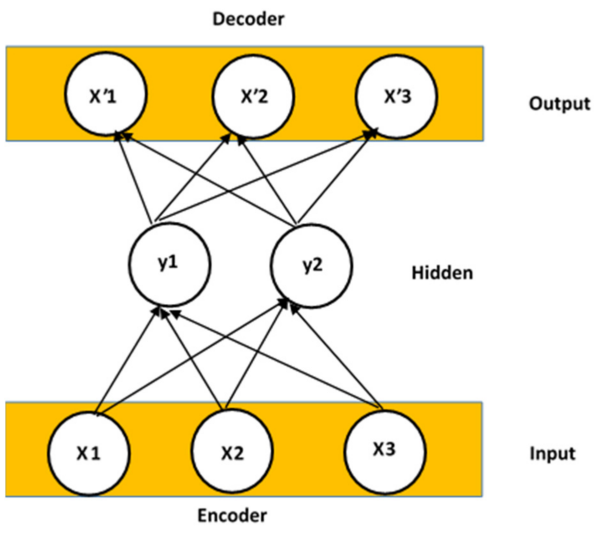


Figure 4. The general architecture of an autoencoder (AE) neural network.

In the decoder model, the y value of the hidden layer is remapped to X′, and then reconstructed as an output value, formulated as:

$$z = \alpha\left(W'y + b'\right) \tag{9}$$

Autoencoders (AEs) study an encoder function from input to representation, and a decoder function, back from representation to the input space, such that the reconstruction, a combination of the encoder and decoder, is good for training examples [81]. In simple terms, an autoencoder (AE) is a neural network that is trained to attempt to copy its input to its output, as shown in Figure 4.

3.4. Existing CNN Architectures Used as Transfer Learning

Based on the aforementioned related works, previous researchers generated their adversarial CAPTCHAs by adopting transfer learning techniques in conjunction with adversarial attack algorithms. Therefore, in order to evaluate the strength of the proposed CNN model, we adopted transfer learning methods by using notable state-of-the-art CNN architectures. These famous CNN architectures include GoogLeNet [82], ResNet [83], VGG-Net [84], Xception [85], and DenseNet [86].

3.4.1. GoogLeNet

The GoogLeNet architecture starts with two Conv–max pool blocks and continues with a series of inception blocks separated by max pool layers before the final fully-connected layer. In their paper, all of the convolutions in conjunction with those inside the inception modules, used rectified linear activation. The inception block takes the tensor as the input and passes it through four different streams as: a 1 × 1 convolution layer, a 1 × 1 convolution layer followed by a 3 × 3 convolution layer, a 1 × 1 convolution layer followed by a 5 × 5 convolution layer, and a 3 × 3 max pool layer followed a 1 × 1 convolution layer. Then, the output tensors of all four final convolution layers are concatenated to one tensor.

3.4.2. ResNet

ResNet begins with a Conv–BatchNorm–ReLU block and then continues with a series of ResNet blocks before the final average pool and fully-connected layers. The ResNet block comprises of a repetition of blocks. The input tensor goes through three Conv–BatchNorm–ReLU blocks and then the output is added to the input tensor to form a skip connection. The two types of skip connection in ResNet are the identity and projection blocks. The identity block takes a tensor as an input and then passes it through one stream of: a 1 × 1 convolution layer followed by a batch-normalization and a rectified linear unit activation layer, a 3 × 3 convolution layer followed by a batch-normalization and a rectified linear unit activation layer, and a 1 × 1 convolution layer followed by a batch-normalization layer. Then the output is added to the input tensor. The projection block takes a tensor as an input and then passes it through two streams, the left and right streams. The left stream consists of: a 1 × 1 convolution layer followed by a batch-normalization and a rectified linear unit activation layer, a 3 × 3 convolution layer followed by a batch-normalization and a rectified linear unit activation layer, and a 1 × 1 convolution layer followed by a batch-normalization layer. The left consists of a 1 × 1 convolution layer followed by a batch-normalization layer.

3.4.3. VGG-Net

The VGG-Net architecture comprises of five convolutional blocks and three fully-connected layers. Each convolutional block consists of two or more convolutional layers and a max pool layer. All hidden layers are equipped with the rectification non-linearity, and max pooling is performed over a 2 × 2-pixel window, with stride 2. In this current work, we implemented VGG19 architecture.

3.4.4. Xception

With respect to the Xception architecture, all pointwise convolution and depth-wise separable convolution layers are followed by batch-normalization. Moreover, all of the depth-wise separable convolution layers contain a depth multiplier of 1. The Xception network is grouped in three flows, which are entry flow, middle flow with eight repetitions of the same block, and exit flow.

3.4.5. DenseNet

The DenseNet architecture connects each layer to every other layer in a feed-forward manner. The feature maps of all preceding layers are used as inputs, and then its own feature maps are used as inputs into all subsequent layers. Some of the advantages of the DenseNet architecture are to prevent the vanishing-gradient problem, and strengthen feature propagation. In this current work, we used the Dense-121 network, which is easy to implement and train, but effective. The architecture starts with standard convolution-pooling block, and continues with a series of dense block and transition layers. It finally closes with a global average pooling and fully connected layer.

3.5. The Proposed CNN Solver Model without Defense

The architecture of our proposed CNN solver model consists of nine convolutional layers, nine batch-normalization layers, nine rectified linear unit activation layers, three max pooling layers, one flatten layer, one dropout layer, and four fully-connected layers with each output with softmax activation. The network structure starts with Conv=>BatchNorm =>ReLU=>max pool block, and repeat itself for three times. Each block contains three convolutional layers, three batch-normalization layers, three activation ReLU layers, with each of the batch-normalization layers. The network contains three max pooling layers, which follows each of the three blocks. At the end of the max pooling layer, we placed a flatten layer, and a dropout, which connected to four fully-connected layers with output softmax activation layers. This study proposed a custom CNN architecture because, sometimes, the aforementioned transfer learning techniques may take a longer time to train, may sometimes consume a lot of memory and space, especially, the ResNet architecture is very difficult to implement and train.

The network structure of the proposed CNN solver model is illustrated in Figure 5.

3.6. The Cyclical Learning Rate Policy

Initially, we used an automatic learning rate finder algorithm to find optimal learning rates, which were then used in the CLR algorithms. The idea was to make the defensive model more robust to withstand the adversarial text-based CAPTCHAs, by accelerating the convergence rate of the CNN solver model.

In the experiment, using the cyclical learning rate (CLR) [87] technique leads to faster convergence and fewer hyper-parameter updates. CLRs oscillate back and forth between two bounds when training, and slowly increase the learning rate after each batch update. By decreasing the learning rate overtime, the CNN model is allowed to descend into lower areas of the loss domain. However, in reality, there is no guarantee that the model will descend into areas of low losses when lowering the learning rate. This is a problem that the CLR method tries to solve.

Therefore, the CLR policy practically eliminates the need to experimentally find the best values and schedule for the global learning rates. Training with CLRs instead of fixed values achieved good performance without the need to tune the model.

The CyclicLR () callback implements a cyclical learning rate policy, as used in this current study. The technique cycles the learning rates between the two boundaries with constant frequency [87]. The amplitude of the cycle can be scaled on a per-iteration or per-cycle basis. The cyclical callback has three built-in policies, which are: “triangular”: a basic triangular cycle with no amplitude scaling, “triangular2”: a basic triangular cycle that scales the initial amplitude by half each cycle, and “exp_range”: a cycle that scales initial amplitude by gamma at each cyclic iteration. The arguments of the cyclical callback can be seen from Table 1.

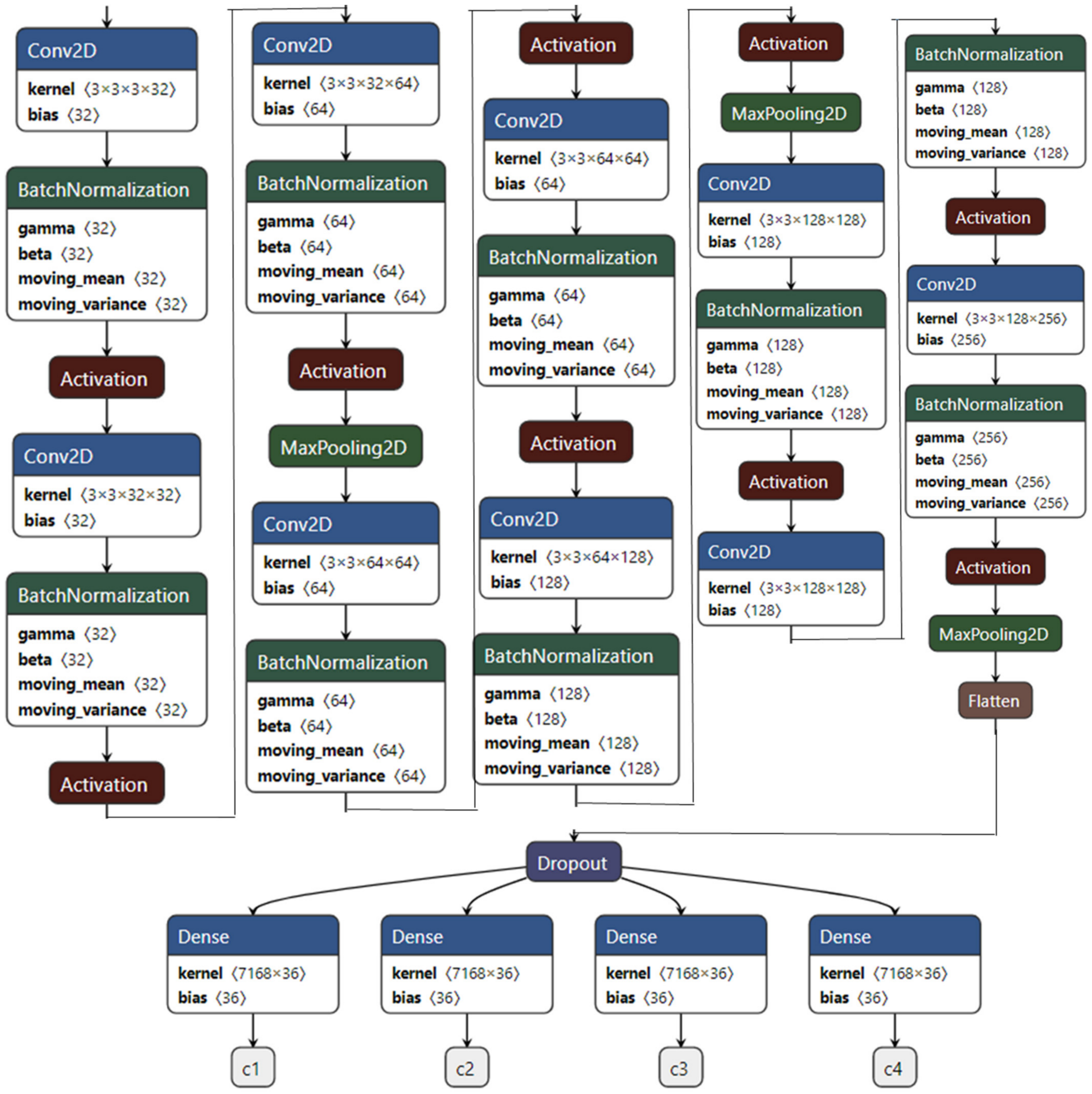


Figure 5. The network structure of the proposed CNN solver model.

The structure of the CLR policy algorithm is represented as:

$$cycle = np.floor\left(1 + \frac{iterations}{2 * step_{size}}\right) \tag{10}$$

$$x = np.abs\left(\frac{iterations}{step_{size}} - 2 * cycle + 1\right) \tag{11}$$

$$lr = base_{lr} + (max\_lr - base\_lr) * np.maximum(0,\ (1 - x)) * sacle\_fn(x) \tag{12}$$

where x is either the iteration or cycle depending on the scale_mode.

The codes are available at https://github.com/sambhav37/Keras-learning-rate-finder (accessed on 23 May 2021), and https://github.com/bckenstler/CLR (accessed on 23 May 2021) based the LRF and the CLR algorithms, respectively.

Table 1. The arguments of the CLR algorithm and its descriptions.

Arguments	Descriptions
base_lr	Initial learning rate, which is the lower boundary in the cycle
max_lr	The upper boundary in the cycle. It defines the cycle amplitude (max_lr—base_lr)
step_size	Number of training iterations per half cycle, which be 2–8 × training iterations in epoch
mode	The cyclical policy to be used. Default is triangular
gamma	It is a constant in exp_range scaling function
Scale_fn	Custom scaling policy defined by a single argument lambda function
Scale_mode	Defines whether scale_fn is evaluated cycle number or cycle iterations. Default id cycle.

3.7. Denoising Autoencoder (DAE) Neural Network

In conventional autoencoders (AEs), the reconstruction of the input at the output is based on the learned underlying manifold of the training dataset's distribution. Conventionally, autoencoders (AEs) minimize some functions, which is formulated as:

$$L(x,\ g(f(x))) \tag{13}$$

where L is a loss function, which penalizes $g(f(x))$ as result of being dissimilar from x, such as the L^2 norm of their difference. Based on (13), a Denoising Autoencoder (DAE) neural network minimizes

$$L(x,\ g(f(\widetilde{x}))) \tag{14}$$

where $\widetilde{x}$ is a copy of x, which has been corrupted by the adversarial or vector noise. The loss function, which we want to minimize for the DAE, is given as:

$$L = -log p_{decoder}(x \mid y = f(\widetilde{x})) \tag{15}$$

where $p_{decoder}$ is a factorial distribution, y is an internal representation, $\widetilde{x}$ is the adversarial text-based CAPTCHA version of the original text-based CAPTCHA sample x, which is obtained through the given corruption process $C(\widetilde{x} \mid x)$, based on the FGSM, I-FGSM, and MI-FGSM adversarial attack algorithms.

The DAE neural network's computational graph of the cost function is illustrated in Figure 6.

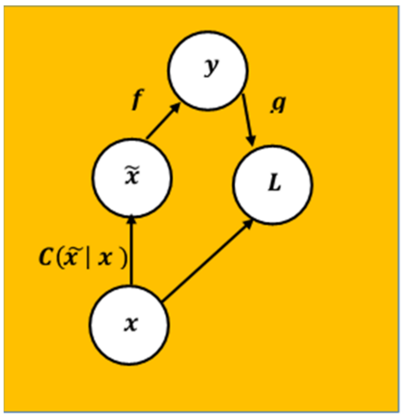


Figure 6. The DAE neural network's computational graph.

3.8. The Proposed Technique of the Baseline Defense Deep Learning Model

Existing defending strategies, including adversarial training [6], image processing, and manual labelling, have been compromised, and sometimes may not be effective against some adversarial text-based CAPTCHA attacks. A basic workflow, defending against adversarial attacks, is illustrated in Figure 7. Therefore, with this assumption, this study initially proposes a Convolutional Neural Network with Denoising Autoencoder (DAE-CNN) as a defense model. The workflow of the proposed DAE-CNN model is illustrated in Figure 8. Given the corrupted process $(\tilde{x} \mid x)$, which represents a conditional distribution over the generated adversarial text-based CAPTCHA data $\tilde{x}$ from the original text-based CAPTCHA data x, then the DAE neural network will learn a reconstruction distribution $p_{reconstruct}\ (X|\tilde{X})$, which is estimated from the training pairs $(x,\ \tilde{x})$ as follows:

- Sample a training example x from the original CAPTCHA dataset.
- Sample an adversarial CAPTCHA version $\tilde{x}$ from $C(\tilde{X}|X = x)$.
- Use $(x,\ \tilde{x})$ as a training example to estimate the DAE reconstruction distribution $p_{reconstruct}(x,\ \tilde{x}) = p_{decoder}(x|y)$ with y as the output of the encoder $f(\tilde{x})$ and $p_{decoder}$ is typically represented as a decoder $g(y)$.

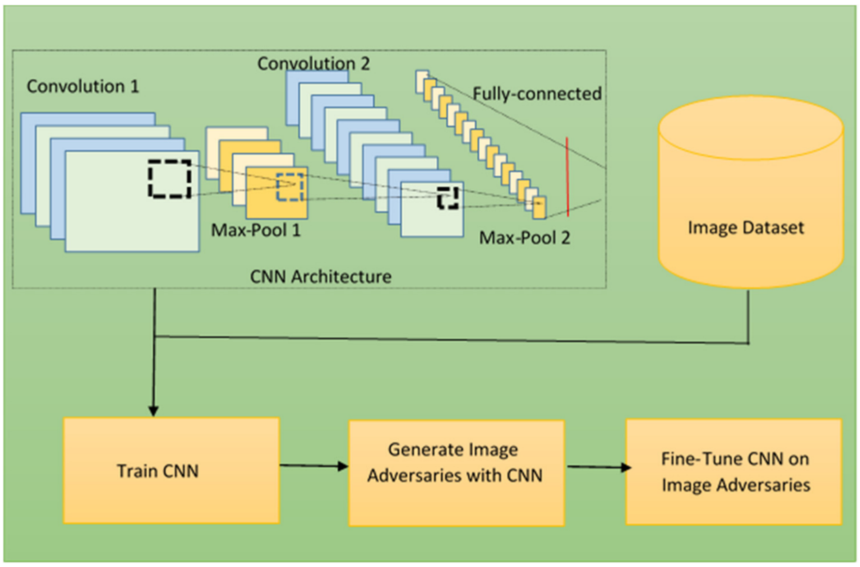


Figure 7. The basic workflow of the adversarial defense model.

The output of the DAE network is then served as input to the CNN model. We further accelerate the convergence rate of the CNN solver model with an automatic learning rate finder (LRF) algorithm to find optimal learning rates in conjunction with the cyclical learning rate (CLR) policy. The proposed CNN solver model architecture and the CLR algorithm were already explained in the aforementioned section. For the proposed CNN solver architecture, the input shape was 160 $\times$ 60, depending on the width and the height of the text-based CAPTCHA, respectively. With vigorous experiments, we trained the CNN model using categorical cross-entropy (CCE), and Adam as the loss function, and optimizer, with a batch size of 32 over 25 epochs, respectively. The learning rate was automatically obtained using the cyclical learning rate policy to improve accuracy.

The structure of the DAE was a bottleneck fully-connected neural network, with the input dimension unit equal to the output dimension unit. The input and output dimension units were obtained through multiplication of 160 $\times$ 60 based on the width and the height of the CAPTCHA image. The DAE was then trained with the original CAPTCHA and the adversarial CAPTCHA datasets. Based on the experiment, the mean square error (MSE), and Adam were used as the loss function, and optimizer, to compile the network with a batch size of 256 over 200 epochs, respectively. After training, the DAE network obtained a MSE value of 0.0040, which showed the DAE network's robust capability to approximate the clean data input of both the original and the adversarial CAPTCHA datasets. The codes in this work were implemented using the Python 3.6 programming language, and were executed and trained on a GPU, based on the Google Colab (https://colab.research.google.com/ (accessed on 20 February 2021)) environment.

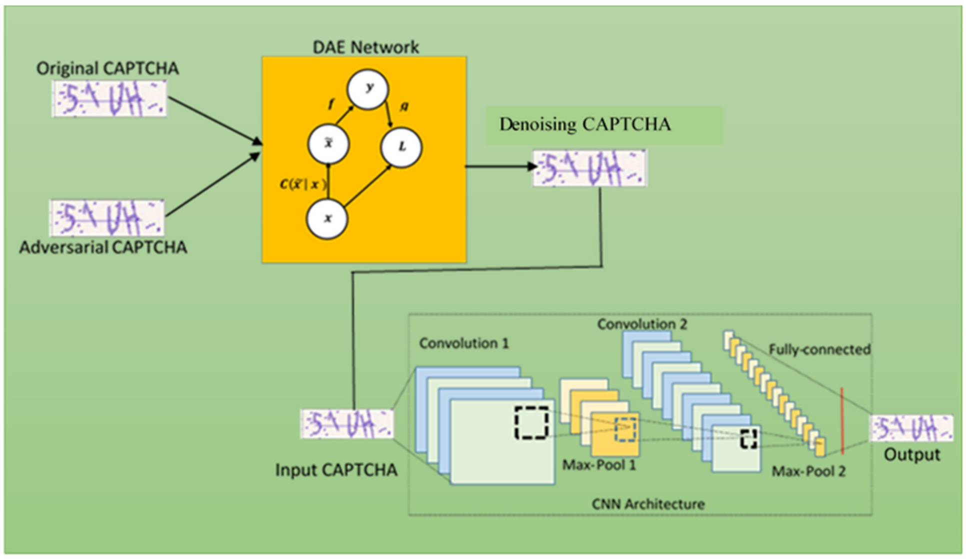


Figure 8. The workflow of the proposed baseline deep learning model for adversarial text-based CAPTCHA security verification.

3.9. The Proposed Technique of the Robust Defense Deep Learning with the Mixed Batch Adversarial Generation Process

The robust proposed defense model was inspired by the works [88,89]. They provided checklists and recommendations that can be followed to develop a more robust, efficient, and uncompromised defense model. According to their work, a good starting point is a model as equal to the defense model as possible, which this study has already implemented, as illustrated in Figure 8. The assumption is that, if the defense model would just add some layers to the baseline model, then it would be better to try the baseline model. They further stated that a good proposed defense model should attempt to break the transferability attack.

Therefore, our innovation idea in the second proposed defense model is to introduced a technique called MBAGP, which stands for, Mixed Batch Adversarial Generation Process, to implement a completely new proposed defense model. That is, instead of fine-tuning the CNN model on the adversarial CAPTCHA images, the batch generation training process itself is reconstructed. The second proposed defense model violated the standard protocol of training neural networks in batches by incorporating the adversarial text-based CAPTCHA images into the training process itself, together with the original data. Meaning, the model itself will generate adversarial CAPTCHA images, combine them with the original CAPTCHA training dataset, and then the model is finally trained.

The steps involved in the reconstruction of the standard training procedure to incorporate the adversarial CAPTCHA images are as follows:

- The CNN model is initialized;
- The total N training samples are selected;
- The CNN model is used, together with the adversarial attack algorithm to generate a total of N Adversarial samples;
- The original and the adversarial CAPTCHA images are mixed, for a batch size of $N \times 2$;
- The defense model on both the original and adversarial CAPTCHA training samples are trained.

The first proposed defense method is less robust and less computation since we will generate only one set of adversarial images, and then fine-tune the model on only the adversarial CAPTCHA images. While the second proposed method, is the MBAGP-CNN model, is more robust and significantly more computation-efficient. The advantage is that it may see both the original CAPTCHA and the adversarial CAPTCHA images through every single batch update throughout the training process. In simple terms, the robust defense model will be used to generate the adversarial CAPTCHA images throughout each batch, as a result, the model will deceive itself, and then learn from its mistakes, so that it can better defend against adversarial CAPTCHA images. In addition, it will learn the patterns of both the original CAPTCHA and the adversarial text-based CAPTCHA images. In addition, the robust MBAGP-CNN model will improve itself based on two factors. For the first factor, the discriminating patterns of the training data will be ideally learned by the model after each batch update. For the second factor, the model will learn to defend against its own generated adversarial text-based CAPTCHAs.

3.10. Performance Evaluation

The proposed deep learning model was evaluated using accuracy and K-fold cross-validation.

3.10.1. Accuracy

The formulas for accuracy is given as:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)}$$

where *TP* is True Positive, *TN* is True Negative, *FP* is False Positive, *FN* is False Negative based on the model's prediction.

3.10.2. K-Fold Cross-Validation

In this current work, we performed K-fold cross-validation to further evaluate the robust performance of the proposed model. A typical cross-validation workflow flowchart is illustrated in Figure 9.

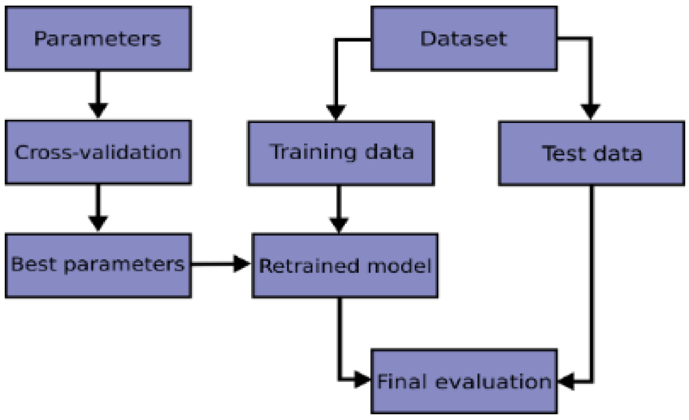


Figure 9. A typical cross-validation workflow in model training [90].

The process of the K-fold cross-validation is described as [90]:

- ➢ Splitting the data into K-folds;
- ➢ Out of the K-folds, K-1 sets are used to train, whereas the remaining set is used for testing;
- ➢ The CNN model is trained and tested K times; each time a new set is used as testing, whereas the remaining set is used for training;

➢ The average of the results obtained at each set represents the result of the K-fold cross-validation.

The process of the K-fold cross-validation is illustrated in Figure 10. In this current work, we performed five-fold cross-validation based on empirical evidence that 5- or 10-fold cross validation should be preferred [90]. The workflow of the proposed MBAGP-CNN model is illustrated in Figure 11.

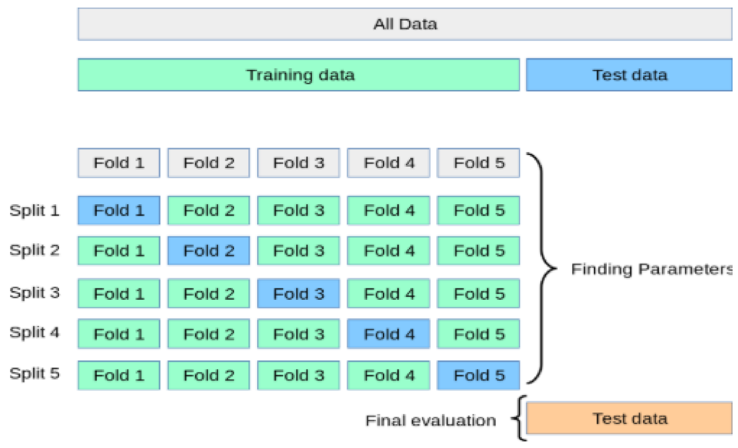


Figure 10. Performing a K-fold cross-validation process [90].

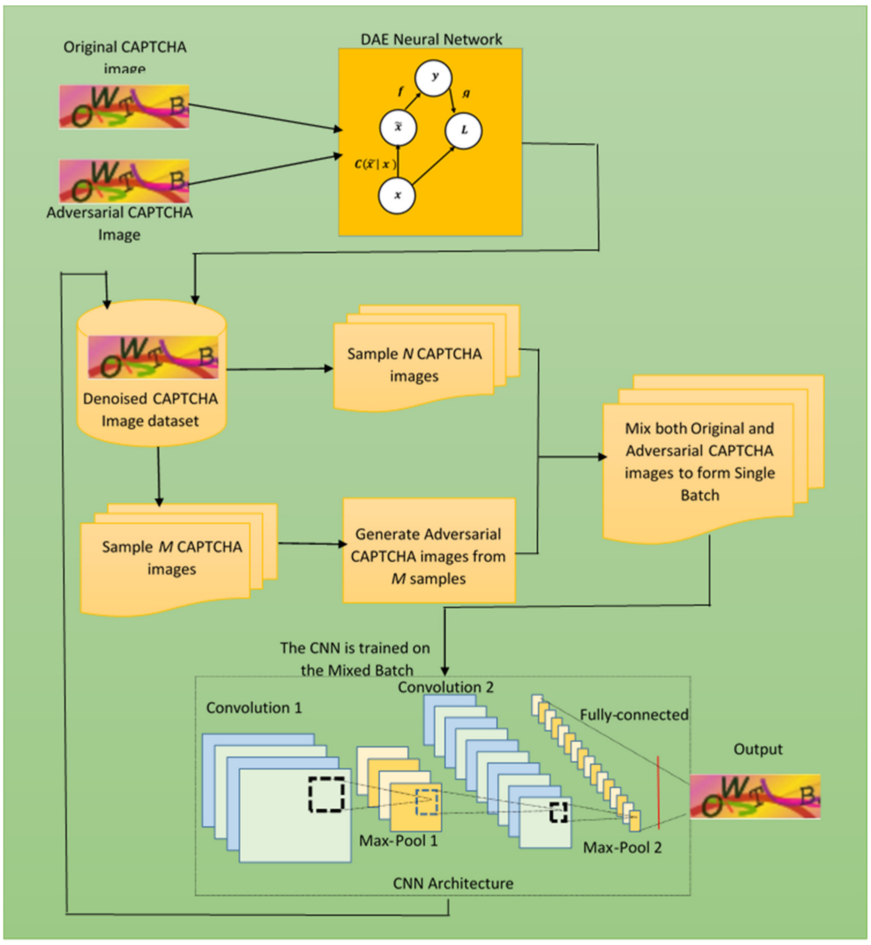

Original CAPTCHA image
DAE Neural Network
Adversarial CAPTCHA Image
Denoised CAPTCHA Image dataset
Sample N CAPTCHA images
Mix both Original and Adversarial CAPTCHA images to form Single Batch
Sample M CAPTCHA images
Generate Adversarial CAPTCHA images from M samples
The CNN is trained on the Mixed Batch
Convolution 1
Convolution 2
Fully-connected
Output
Max-Pool 1
Max-Pool 2
CNN Architecture

Figure 11. The workflow of the proposed robust deep learning model for adversarial text-based CAPTCHA security verification.

4. Results

This section presents the results of the proposed CNN solver and the results after attacking the model with the adversarial text-based CAPTCHAs.

The Results of the CNN Solver Model without Defense

After 25 epochs, the proposed CNN solver model achieved training accuracies for C1, C2, C3, and C4, which corresponded to 0.9972, 0.99.04, 0.9889, and 0.99.65, respectively. The acronyms C1–C4 represents the number of fully-connected layers used in the proposed CNN, as illustrated in Figure 5. The training losses for C1, C2, C3, and C4 represented 0.0138, 0.0369, 0.0433, and 0.0166, respectively. The validation accuracies for C1, C2, C3, and C4 were 0.9243, 0.8239, 0.7938, and 0.9152, respectively. The validation losses for C1, C2, C3, and C4 represented 0.2515, 0.6880, 0.8787, and 0.3277, respectively. Table 2 summarizes the performance results of the proposed CNN solver model based on the original generated text-based CAPTCHA dataset. The visualization representation of the accuracy and loss results obtained by the CNN solver model without defense is illustrated in Figure 12a,b, respectively. After testing, the CNN solver model achieved a competitive performance result of an overall accuracy of 90% to solve text-based CAPTCHAs with conventional noise. After strengthening the security of the generated CAPTCHAs, based on the original text-based CAPTCHAs using the FGSM, I-FGSM, and MI-FGSM attack algorithms, the experiment results showed that the FGSM CAPTCHA attack reduced the test accuracy of the CNN solver model without defense, from 90% to 33.15%, with an epsilon of 0.25, with the same epsilon and iteration of 10, the I-FGSM CAPTCHA attack reduced the test accuracy from 90% to 35.46%, and the MI-FGSM CAPTCHA attack reduced the test accuracy from 90% to 32.65%, with the same epsilon, iterations, and a decay factor of 1.0. These observation results, based on the attack algorithms, showed the power and essence of adversarial attacks on text-based CAPTCHAs for security strengthening, as shown in Figure 13. The study observed that the MI-FSGM algorithm performed better based on the lowest accuracy obtained. The next section discusses the results of the defense models obtained in this current work.

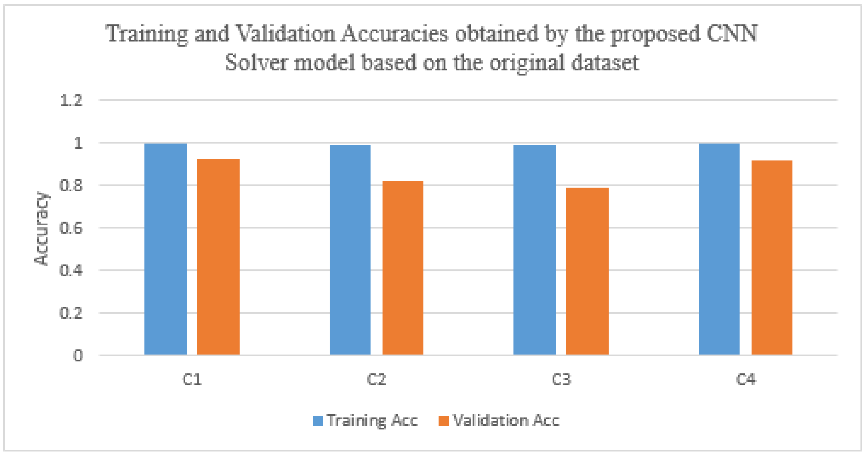
Table 2. The performance results of the proposed CNN solver model based on the conventional text-based CAPTCHA dataset.

Class	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
C1	0.9972	0.0138	0.9243	0.2515
C2	0.9904	0.0369	0.8239	0.6880
C3	0.9889	0.0433	0.7938	0.8787
C4	0.9965	0.0166	0.9152	0.3277

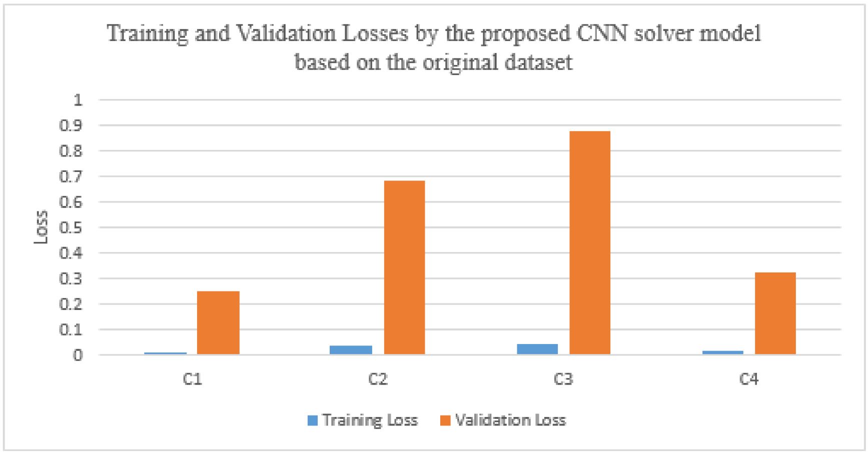
The study further performed five-fold cross-validation on the CNN solver model based on the original CAPTCHA dataset. Table 3 shows the accuracies by the various K-folds. The highest accuracy was detected at fold 3, which was 1.0000, as visualized in Figure 14. The CNN solver model achieved a mean accuracy of 89.80% and a standard deviation of 0.0595, based on the 5-fiveold cross-validation.

Table 3. Accuracies achieved by performing five-Fold cross-validation on the CNN solver model based on the original dataset.

K-Fold	Accuracy
1	0.8300
2	0.8512
3	1.0000
4	0.9188
5	0.8900
Mean: 89.80%	Std: 0.0595



(a)



(b)

Figure 12. (**a**) Training and validation accuracies obtained by the proposed CNN Solver model based on the original dataset. (**b**) Training and validation losses by the proposed CNN solver model based on the original dataset.

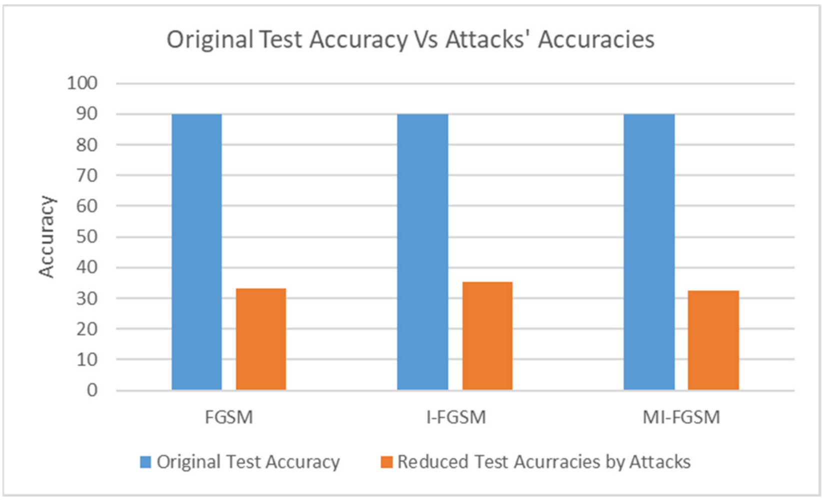


Figure 13. The overall test accuracy by the CNN solver model without defense versus the accuracies obtained after the attacks.

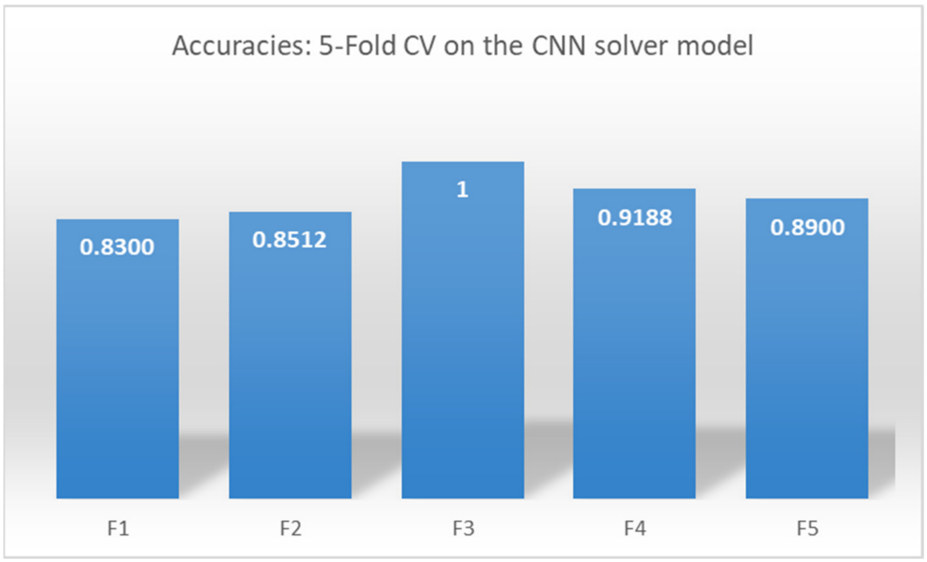


Figure 14. The accuracies achieved by the folds after performing five-fold cross-validation on the CNN solver model based on the original dataset.

5. Discussion

In the work by Zhanyang et al. [91], they conducted a survey on artificial intelligence (AI) for securing IoT services in edge computing. According to their observations, in the context of security and privacy preservation issues, AI has opened up many possible windows to address security challenges. The main ideas behind text-based CAPTCHAs are: they serve as a means of security to secure user-logins against dictionary or brute force password guessing, prevent automated usage of various web services, prevent machine bots from spamming on forums, prevent downloading large files, etc. As mentioned in the previous related works section, several researchers who performed text-based CAPTCHA security verification used the transfer learning approach to build their CNN models. However, in this current work, the study proposed custom CNN architecture as an inspiration from these notable existing CNN architectures. This is because, sometimes, these existing CNN architectures contain a greater number of parameters, and consume a lot of time and memory, especially VGGNet, and ResNet. As a result, in order to evaluate the strengths and weaknesses of our proposed CNN solver and defense models, we adopted the transfer learning technique based on the famous state-of-the-art CNN architectures, such as GoogLeNet, ResNet, VGGNet, Xception, and DenseNet.

After testing on the original CAPTCHA dataset, the pre-trained models achieved accuracies of 80.47%, 80.78%, 81.45%, 86.79%, and 84.88% based on GoogLeNet, ResNet50, VGG19, Xception, and DenseNet121, respectively, as shown in Table 4 and Figure 15. The observed results showed that the Xception and the DenseNet transfer learning models exhibited good results on the original CAPTCHA test set. This work then fine-tuned the transfer learning or pre-trained models on the generated adversarial text-based CAPTCHA test sets. The attacks to the proposed pre-trained solver models were successful at reducing ~50% of the test accuracy, as seen from Table 4 and Figure 15. The observation was that the attack algorithms strengthened the security of the conventional text-based CAPTCHAs, and drastically reduced the test accuracies by fooling the pre-trained models and our proposed CNN solver model, as shown in Figure 15.

Table 4. Comparison accuracy results between the transfer learning CNN models and the proposed CNN solver model based on the original text-based CAPTCHA, FGSM attack, I-FGSM attack, and MI-FGSM attack testing datasets.

CNN Solver Model	Original CAPTCHA (%)	FGSM Attack (%)	I-FGSM Attack (%)	MI-FGSM Attack (%)
GoogLeNet	80.47	22.89	23.56	20.98
ResNet50	80.78	23.56	24.56	22.98
VGG19	81.45	29.87	30.45	26.67
Xception	86.79	30.23	31.78	29.15
DenseNet121	84.88	30.02	31.45	28.90
CNN (proposed)	90.00	33.15	35.46	32.65

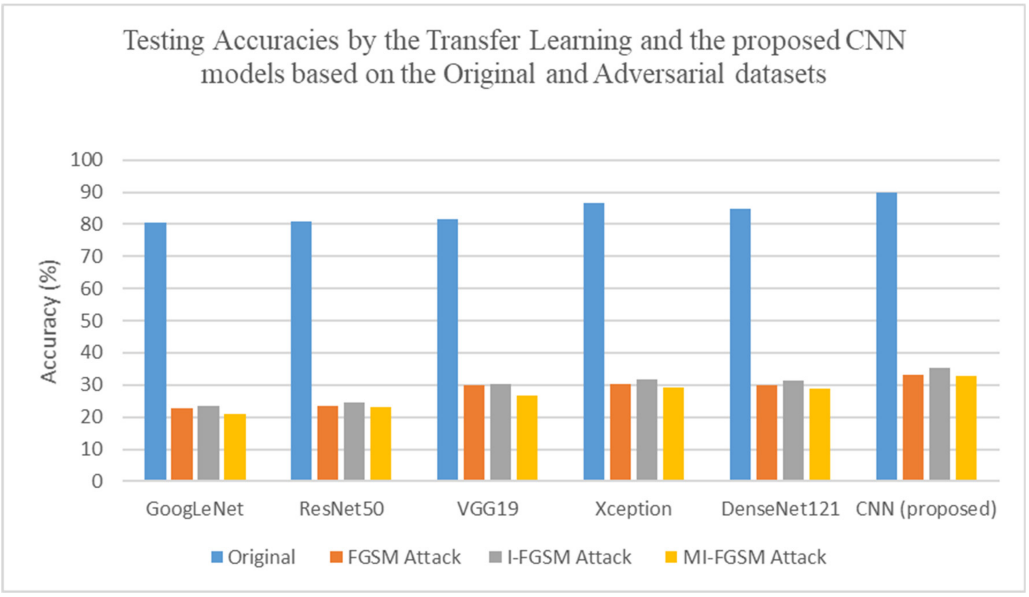


Figure 15. Comparison results of the testing accuracies obtained by the transfer learning and the proposed CNN solver models based on the original and the adversarial text-based CAPTCHA datasets.

However, the goal in this current work is to perform adversarial text-based CAPTCHA security verification, so it means that the security of the CNN solver models need to be strengthened to withstand the attacks. This study initially used the image processing filter, which included, MeanBlur, MedianBlur, GuassianBlur, and RotateFilter techniques, in conjunction with the transfer learning approach, and our proposed CNN, as a form of a feasibility analysis to defend against the adversarial text-based CAPTCHAs. However, the results of the image processing filters, together with CNN models, were less effective on the adversarial text-based CAPTCHAs. Therefore, with inspiration from existing defense techniques and guidelines [85], we proposed our first Convolutional Neural Network with Denoising Autoencoder (DAE-CNN), as a defense workflow model, as illustrated in Figure 8. Based on the adversarial test sets, the proposed DAE-CNN model was successful at reducing ~10% of the test accuracy among the three adversarial attack algorithms, FGSM, I-FGSM, and MI-FGSM, as shown in Figures 16–18, respectively. Tables 5–7 summarize the comparison accuracy results, defending against the FGSM, I-FSGM, and MI-FGSM attacks using the image processing filters with the transfer learning models, compared to the proposed DAE-CNN model based on the testing sets.

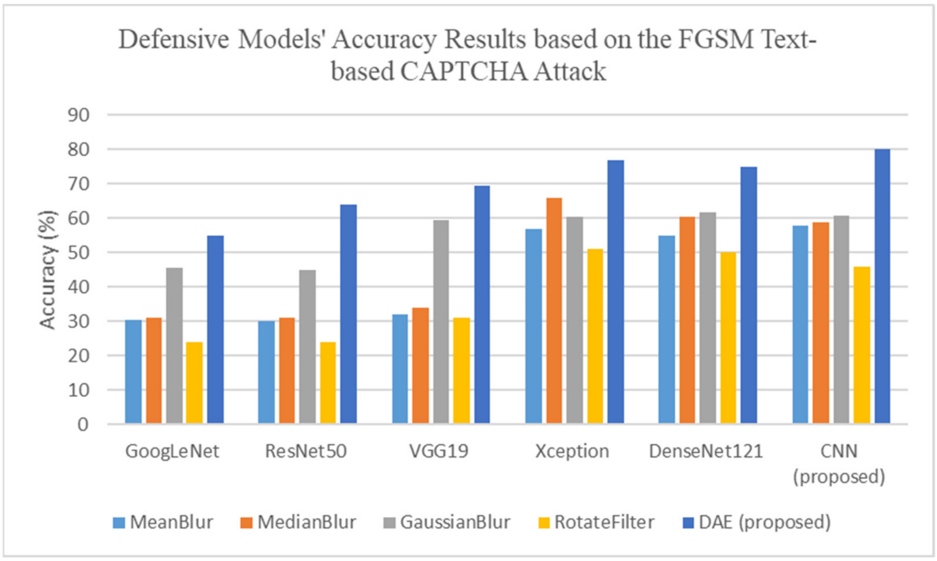


Figure 16. Comparison results of the testing accuracies obtained by the defensive models based on the FGSM text-based CAPTCHA attack.

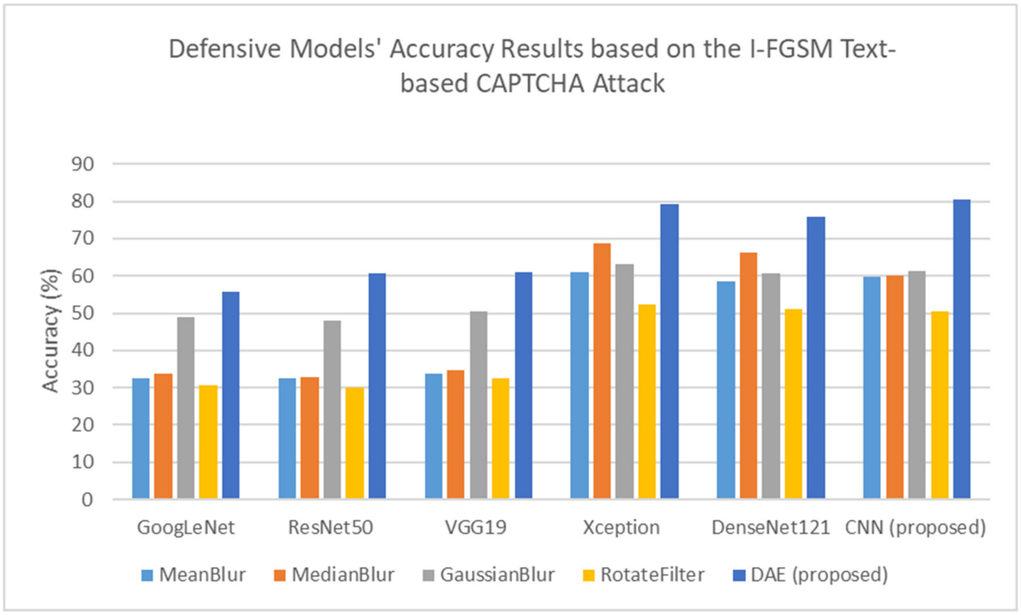


Figure 17. Comparison results of the testing accuracies obtained by the defensive models based on the I-FGSM text-based CAPTCHA attack.

Table 5. Comparison accuracy results, defending against the FGSM attack using the image processing filters with the transfer learning models compared to the proposed DAE-CNN model, based on the testing dataset.

Filter	GoogLeNet (%)	ResNet50 (%)	VGG19 (%)	Xception (%)	DenseNet121 (%)	CNN (Proposed) (%)
MeanBlur	30.45	30.14	31.89	56.78	54.98	57.88
MedianBlur	31.09	31.00	33.78	65.78	60.23	58.64
GaussianBlur	45.43	44.98	59.45	60.44	61.56	60.54
RotateFilter	23.90	23.87	31.02	50.98	49.87	45.97
DAE (proposed)	54.78	63.78	69.45	76.89	74.98	80.15

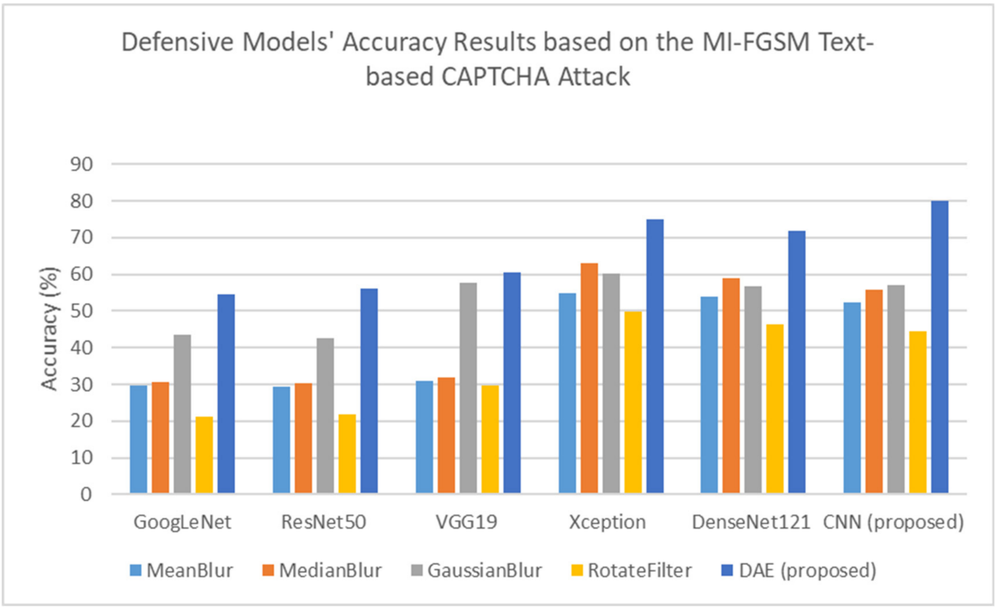


Figure 18. Comparison results of the testing accuracies obtained by the defensive models based on the MI-FGSM text-based CAPTCHA attack.

Table 6. Comparison accuracy results, defending against the I-FGSM attack, using image processing filters with the transfer learning models compared to the proposed DAE-CNN model based on the testing dataset.

Filter	**GoogLeNet (%)**	**ResNet50 (%)**	**VGG19 (%)**	**Xception (%)**	**DenseNet121 (%)**	**CNN (Proposed) (%)**
MeanBlur	32.53	32.45	33.76	60.89	58.62	5989
MedianBlur	33.67	32.67	34.68	68.76	66.12	59.97
GaussianBlur	48.90	47.96	50.58	63.07	60.67	61.23
RotateFilter	30.78	29.89	32.54	52.45	51.02	50.54
DAE (proposed)	55.68	60.77	60.98	79.13	75.96	80.64

Table 7. Comparison accuracy results, defending against the MI-FGSM attack using the image processing filters with the transfer learning models compared to the proposed DAE-CNN model, based on the testing dataset.

Filter	**GoogLeNet (%)**	**ResNet50 (%)**	**VGG19 (%)**	**Xception (%)**	**DenseNet121 (%)**	**CNN (Proposed) (%)**
MeanBlur	29.78	29.34	30.79	54.97	53.78	52.45
MedianBlur	30.65	30.45	31.98	63.09	58.98	55.87
GaussianBlur	43.56	42.65	57.63	60.13	56.90	57.09
RotateFilter	21.34	21.89	29.66	49.89	46.34	44.56
DAE (proposed)	54.67	55.98	60.43	74.90	71.87	79.89

This study further went on to propose a second defense model, which is more robust and computationally-efficient when compared to the first method. Based on the innovation idea, this study took the DAE-CNN model, and then introduced a technique called Mixed Batch Adversarial Generation Process (MBAGP). We reconstructed the standard training procedure protocol of the CNN batch training process. The idea was to mitigate the problem of catastrophic forgetting, as already explained and illustrated in detail in Figure 7. The MBAGP-CNN model is able to generalize and perform better to verify adversarial text-based CAPTCHA security. Based on the adversarial test sets, the proposed DAE-CNN model was successful at reducing ~5% of the test accuracy among the three adversarial attack algorithms, FGSM, I-FGSM, and MI-FGSM, as shown in Figures 19–21, respectively. In comparison, as shown in Figure 22, the MBAGP-CNN model exhibited good and competitive performance results when compared to the DAE-CNN defense model, to perform a successful adversarial text-based security verification.

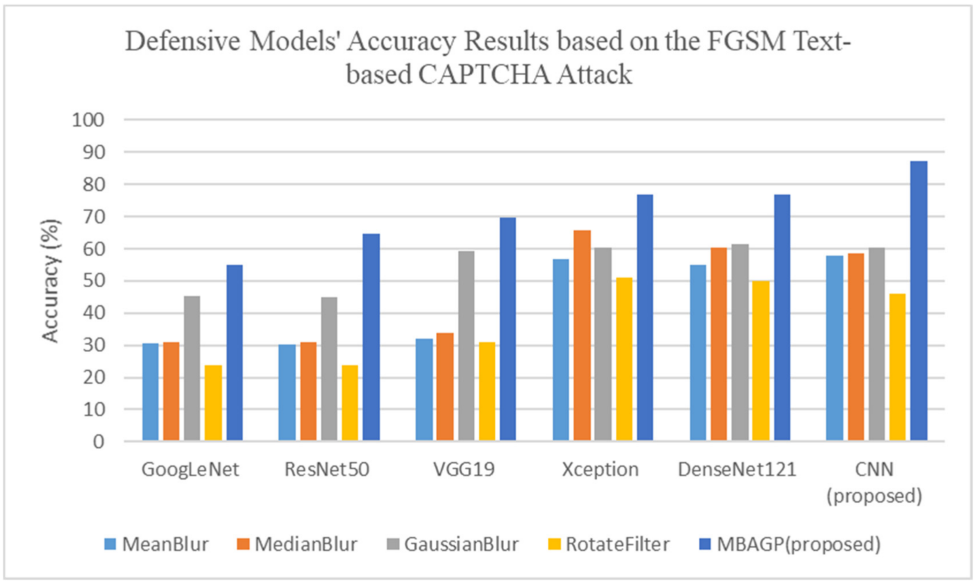


Figure 19. Comparison results of the testing accuracies obtained by the defense models based on the FGSM text-based CAPTCHA attack, in terms of the proposed MBAGP-CNN model.

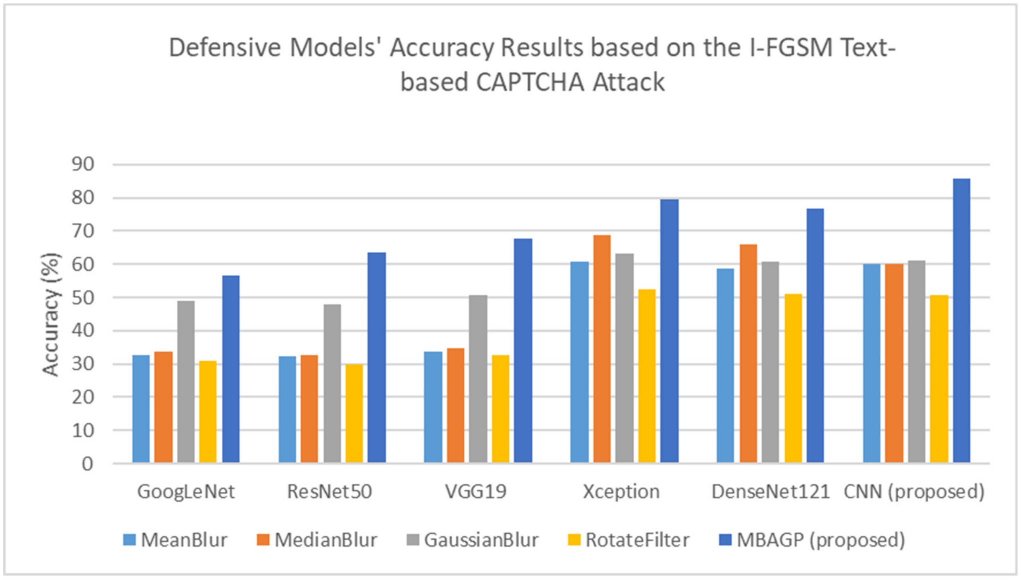


Figure 20. Comparison results of the testing accuracies obtained by the defense models based on the I-FGSM text-based CAPTCHA attack, in terms of the proposed MBAGP-CNN model.

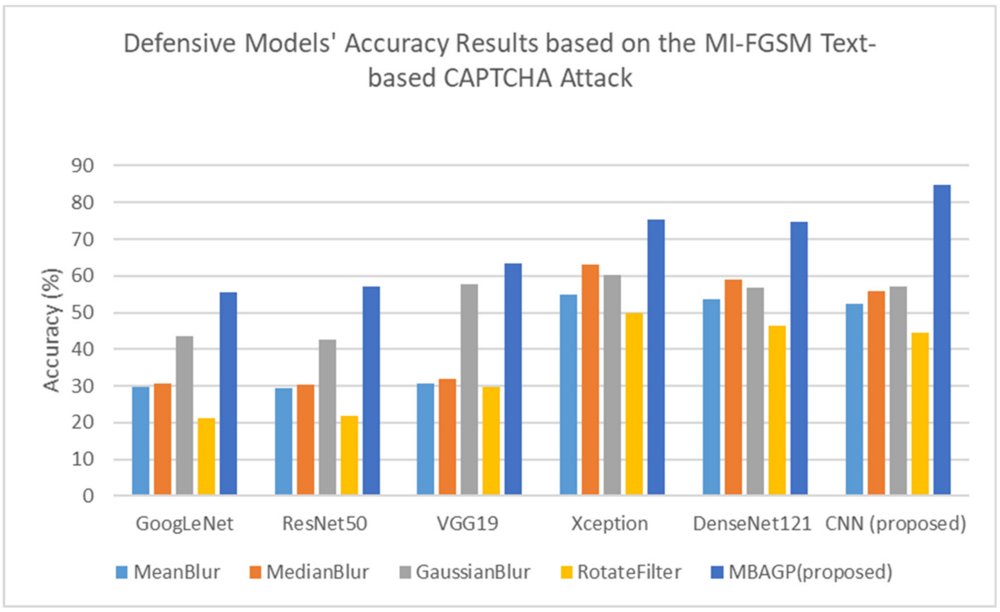


Figure 21. Comparison results of the testing accuracies obtained by the defense models, based on the MI-FGSM text-based CAPTCHA attack, in terms of the proposed MBAGP-CNN model.

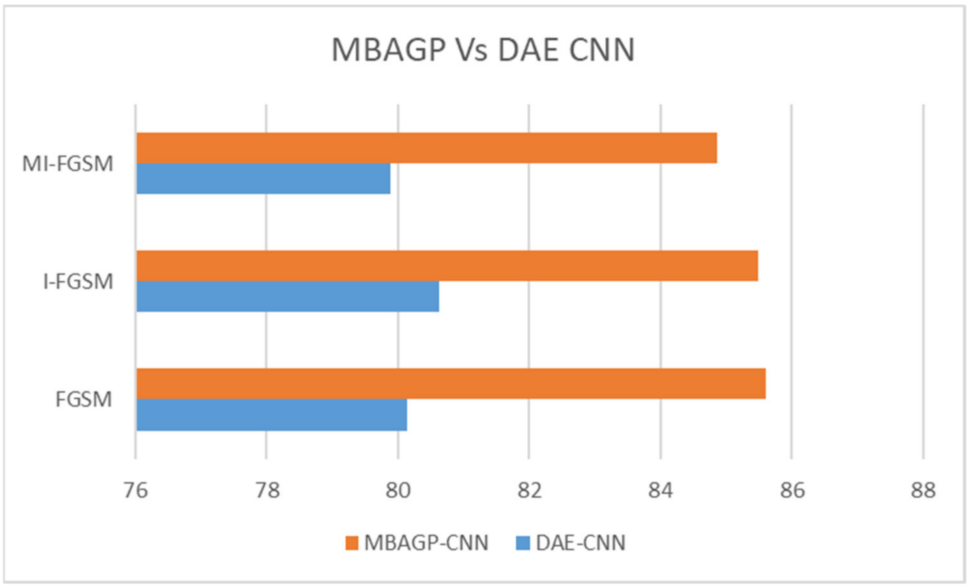


Figure 22. Comparison results of the testing accuracies obtained by the proposed MBAGP-CNN and the DAE-CNN defense models, based on FGSM, I-FGSM, and MI-FGSM attacks.

The study further performed five-fold cross-validation on the proposed MBAGP-CNN model based on the FGSM, I-FGSM, and MI-FGSM attacks. Figure 23 presents the accuracies obtained at the various K-folds. Based on the FGSM attack, the highest accuracy was detected at fold 3, which was 0.8858, as shown in Figure 23. The MBAGP-CNN defense model therefore achieved a mean accuracy of 84.30%, and a standard deviation of 0.0251 based on the FGSM attack. Based on the I-FGSM attack, the highest accuracy was detected at fold 2, which was 0.8531, as shown in Figure 23. The MBAGP-CNN defense model, therefore, achieved a mean accuracy of 83.44%, and a standard deviation of 0.0161 based on the I-FGSM attack. Based on the MI-FGSM attack, the highest accuracy was detected at fold 4, which was 0.8588, as shown in Figure 23. The MBAGP-CNN defense model therefore achieved a mean accuracy of 82.20% and a standard deviation of 0.0260 based on the MI-FGSM attack.

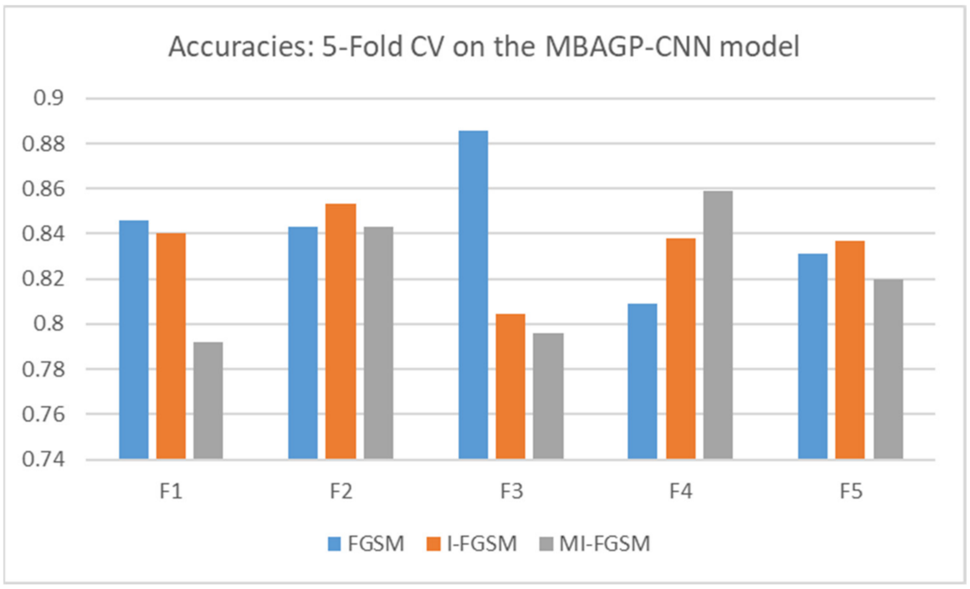


Figure 23. The accuracies achieved by the folds after performing five-fold cross-validation on the MBAGP-CNN solver model based on the FGSM, I-FGSM, and MI-FGSM attacks.

The implications of this current study are:

- With observations on websites, it seems that text-based CAPTCHAs will remain in the system for a very longer time, and for that matter, this current study suggests that cyber security researchers take advantage of adversarial attacks to strengthen the security of their generated text-based CAPTCHAs. This can be achieved by implementing robust attack algorithms to strengthen the generated CAPTCHAs to withstand automatic malicious attacks in order to secure IoT devices and end users' privacy.
- Implementing robust adversarial text-based attack algorithms can assist in minimizing the activities of automatic malicious attacks.
- Defending against adversarial CAPTCHA attacks is tedious and usually involves a lot of effort, which frustrates hackers. It may also remain and will continue to be an active research area.
- It is necessary to perform vulnerability assessments on the generated adversarial CAPTCHAs before they are used on websites for public use, for example, this study recommends taking advantage of the MBAGP technique.
- Among the three gradient-based adversarial attacks, MI-FSGM showed highly defensive security measures against the CNN solver models, relative to this study, as compared to FSGM and I-FSGM.
- If cybersecurity practitioners want to perform vulnerability assessments on adversarial CAPTCHAs, using transfer learning approaches without building their own CNN models, then they should consider Xception and DenseNet architectures.
- In the context of IoT risk assessment, the proposed AI framework technology can be used to perform adversarial text-based CAPTCHA security verification using deep learning. This will help minimize malicious attacks through effective CAPTCHA generating algorithms to secure smartphones and smart TVs, which are connected to websites from spamming.
- In addition, the proposed technology will assist in the evaluation of ethical risks based on data protection and privacy on IoT devices, especially smartphones.
- Another dimension of the proposed technology can be seen in more native scenarios and sensors, which are connected to web application programming interfaces (APIs). Home security cameras are accessible from mobile IoT devices, such as smartphones. A user authentication system comprising of adversarial images can be generated to verify users who use their smartphones to access home security cameras.

6. Conclusions

Text-based CAPTCHA is still widely used and will remain, for a longer time, as a security mechanism to differentiate machine bots from humans. In this current work, we proposed a technique to generate adversarial text-based CAPTCHAs using the Fast Gradient Sign Method (FGSM), Iterative Fast Gradient Sign Method (I-FGSM), and the Momentum Iterative Fast Gradient Sign Method (MI-FGSM) algorithms. We proposed a more robust and computation-efficient deep learning defense model—a Convolutional Neural Network with a Mixed Batch Adversarial Generation Process (MBAGP-CNN) to perform adversarial text-based CAPTCHA security vulnerability verification.

Based on the experiment's results, the proposed CNN solver model achieved a competitive result to solve the original text-based CAPTCHAs with about 90% accuracy based on the test set. The adversarial attacks to the proposed CNN solver model were successful at reducing ~60% of the test accuracy, with an epsilon of 0.25, iteration of 10, and a decay factor of 1.0. Therefore, this study initially proposed a fundamental defense vulnerability verification model based on the Convolutional Neural Network (CNN) with a Denoising Autoencoder (DAE-CNN). The number of dimensions occupied by the input data were reduced by the hidden layer representation of the DAE neural network and, hence, increased recognition accuracy of the CNN classifier against adversarial text-based CAPTCHA attacks. The experiment results showed that the proposed DAE-CNN model was successful at reducing ~10% of the test accuracy among the three gradient-based adversarial attacks. However, this study observed that the DAE-CNN model was less robust and less computational, and could be improved to exhibit good performance. Therefore, this study proposed a more robust and more computation-efficient defense model by introducing the Mixed Batch Adversarial Generation Process (MBAGP). The novel proposed MBAGP-CNN model attempted to break the transferability attack, and mitigate the problem of catastrophic forgetting in the context of an adversarial attack defense. Ideally, the MBAGP-CNN model generated image batches of both the original and the adversarial CAPTCHA images by reconstructing the training process of the DAE-CNN model, which improved the model's capability to generalized denoising on both original and adversarial images to defend against adversarial text-based CAPTCHAs. Competitive experiment results showed that the proposed defense and robust MBAGP-CNN model was successful at reducing ~5% of the test accuracy among the three gradient-based adversarial attacks. This study further introduced the learning rate finder algorithm to find optimal learning rates in conjunction with the cyclical learning rate policy to accelerate the convergence rate of the defense models to improve accuracy.

This study further performed K-fold cross-validation to evaluate the robustness of the proposed MBAGP-CNN model. The proposed defense model then achieved mean accuracies of 84.30%, 83.44%, and 82.20% based on the FGSM, I-FGSM, and the MI-FGSM attack datasets, respectively. The study emphasizes that the result figures are the key contribution outcomes of the experiments conducted in this work. Adversarial text-based CAPTCHA attacks were shown to withstand against automatic malicious attackers or intruders. However, there is still a security risk after performing vulnerability assessment, as our proposed robust defense model obtained accuracy more than 80% among the three adversarial attack algorithms based on five-fold cross validation. Therefore, there is still a need to improve upon adversarial text-based CAPTCHA-generating algorithms to defend against automatic malicious attacks. This current work is useful to the cyber security community and in performing IoT risk assessment. In future works, we can attempt other adversarial attack algorithms, see their results, and propose a defense model to perform security vulnerability assessment.

Author Contributions: All authors contributed significantly to this research work. S.D. and L.Y. conceptualized the idea. S.D. implemented the methodology and wrote the paper. S.D. and L.Y. revised the work. L.Y. supervised the work and assisted in the acquisition of the funds. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by NSFC (no. 61871074).

Data Availability Statement: The codes used in this current work are available at (https://github.com/sm-multimedia/CAPTCHA-Generator (accessed on 5 May 2021)), (https://github.com/sambhav37/Keras-learning-rate-finder (accessed on 23 May 2021)), and (https://github.com/bckenstler/CLR (accessed on 23 May 2021)) based on the CAPTCHA-generator tool, LRF algorithm, and the CLR policy callback, respectively.

Acknowledgments: First of all, the authors thank God for His strength and wisdom. The authors express their sincere thanks to Gao Bin for his advice. Lastly, the authors thank Eric Affum, Lisa Agyemang, Edward Donkor, and Vida Sarpong for their encouragement.

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

References

1. Meriem, G.; Alessio, M.; Mauro, M.; Francesco, P. Invisible CAPPCHA: A usable mechanism to distinguish between malware and humans on the mobile IoT. *Comput. Secur.* **2018**, *78*, 255–266. [CrossRef]
2. Hussain, R.; Kumar, K.; Gao, H.; Khan, I. Recognition of merged characters in text based CAPTCHAs. In Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 3917–3921.
3. Sachdev, S. Breaking CAPTCHA characters using Multi-Task Learning CNN and SVM. In Proceedings of the 2020 4th International Conference on Computational Intelligence and Networks (CINE), Kolkata, India, 27–29 February 2020; pp. 1–6. [CrossRef]
4. Stein, G.; Peng, Q. Low-Cost Breaking of a Unique Chinese Language CAPTCHA Using Curriculum Learning and Clustering. In Proceedings of the IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; pp. 595–600. [CrossRef]
5. Zi, Y.; Gao, H.; Cheng, Z.; Liu, Y. An End-toEnd Attack on Text CAPTCHAs. In Proceedings of the IEEE Transactions on Information Forensics and Security, Rome, Italy, 15 July 2019; Volume 15, pp. 753–766. [CrossRef]
6. Wang, P.; Gao, H.; Rao, Q.; Luo, S.; Yuan, Z.; Shi, Z. A Security Analysis of Captchas with Large Character Sets. *IEEE Trans. Dependable Secur. Comput.* **2021**, 1. [CrossRef]
7. Yu, N.; Darling, K. A Low-Cost Approach to Crack Python CAPTCHAs Using AI-Based Chosen-Plaintext Attack. *Appl. Sci.* **2019**, *9*, 2010. [CrossRef]
8. Kwon, H.; Yoon, H.; Park, K.-W. CAPTCHA Image Generation: Two-Step Style-Transfer Learning in Deep Neural Networks. *Sensors* **2020**, *20*, 1495. [CrossRef] [PubMed]
9. Zhao, N.; Yi, L.; Yijun, J. CAPTCHA Breaking with Deep Learning. 2017. Available online: http://cs229.stanford.edu/proj2017/final-posters/5130459.pdf (accessed on 14 February 2021).
10. Thobhani, A.; Gao, M.; Hawbani, A.; Ali, S.T.M.; Abdussalam, A. CAPTCHA Recognition Using Deep Learning with Attached Binary Images. *Electronics* **2020**, *9*, 1522. [CrossRef]
11. Jaderberg, M.; Vedaldi, A.; Zisserman, A. Deep features for text spotting. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Zurich, Switzerland, 2014; pp. 512–528.
12. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of the International Conference on Document Analysis and Recognition IEEE Computer Society, Sydney, Australia, 3–6 August 2003; Volume 3, pp. 958–962.
13. Dankwa, S.; Yang, L. An Efficient and Accurate Depth-Wise Separable Convolutional Neural Network for Cybersecurity Vulnerability Assessment Based on CAPTCHA Breaking. *Electronics* **2021**, *10*, 480. [CrossRef]
14. Hu, Y.; Chen, L.; Cheng, J. A CAPTCHA recognition technology based on deep learning. In Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, 31 May–2 June 2018; pp. 617–620. [CrossRef]
15. Hu, G.; Wang, K.; Liu, L. Underwater Acoustic Target Recognition Based on Depthwise Separable Convolution Neural Networks. *Sensors* **2021**, *21*, 1429. [CrossRef] [PubMed]
16. Lee, W.; Seong, J.J.; Ozlu, B.; Shim, B.S.; Marakhimov, A.; Lee, S. Biosignal Sensors and Deep Learning-Based Speech Recognition: A Review. *Sensors* **2021**, *21*, 1399. [CrossRef]
17. Abbaschian, B.J.; Sierra-Sosa, D.; Elmaghraby, A. Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models. *Sensors* **2021**, *21*, 1249. [CrossRef]
18. Ren, Y.; Yang, J.; Guo, Z.; Zhang, Q.; Cao, H. Ship Classification Based on Attention Mechanism and Multi-Scale Convolutional Neural Network for Visible and Infrared Images. *Electronics* **2020**, *9*, 2022.
19. Merino, I.; Azpiazu, J.; Remazeilles, A.; Sierra, B. 3D Convolutional Neural Networks Initialized from Pretrained 2D Convolutional Neural Networks for Classification of Industrial Parts. *Sensors* **2021**, *21*, 1078. [CrossRef] [PubMed]
20. Liang, X.; Xu, L.; Liu, J.; Liu, Z.; Cheng, G.; Xu, J.; Liu, L. Patch Attention Layer of Embedding Handcrafted Features in CNN for Facial Expression Recognition. *Sensors* **2021**, *21*, 833. [CrossRef]

21. Malik, S.; Soundararajan, R. Llrnet: A Multiscale Subband Learning Approach for Low Light Image Restoration. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 779–783.
22. Dong, W.; Wang, P.; Yin, W.; Shi, G.; Wu, F.; Lu, X. Denoising Prior Driven Deep Neural Network for Image Restoration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2305–2318. [CrossRef]
23. Jin, Z.; Iqbal, M.Z.; Bobkov, D.; Zou, W.; Li, X.; Steinbach, E. A Flexible Deep CNN Framework for Image Restoration. *IEEE Trans. Multimed.* **2020**, *22*, 1055–1068. [CrossRef]
24. Drossos, K.; Mimilakis, S.; Gharib, S.; Li, Y.; Virtanen, T. Sound event detection with depthwise separable and dilated convolutions. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020.
25. Fonseca, E.; Plakal, M.; Font, F.; Ellis, D.; Serra, X. Audio tagging with noisy labels and minimal supervision. In Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), New York, NY, USA, 25–26 October 2019.
26. Liu, Y. Improved Faster R-CNN for Object Detection. In Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 8–9 December 2018; Volume 2, pp. 119–123.
27. Ren, G.; Dai, T.; Barmpoutis, P.; Stathaki, T. Salient Object Detection Combining a Self-Attention Module and a Feature Pyramid Network. *Electronics* **2020**, *9*, 1702. [CrossRef]
28. Ren, S.; He, K.; Girshick, R.; Sun, J. FasterR-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
29. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
30. Yang, Y.; Deng, H. GC-YOLOv3: You Only Look Once with Global Context Block. *Electronics* **2020**, *9*, 1235. [CrossRef]
31. Shao, X.; Zhang, X.; Tang, G.; Bao, B. Scene Recognition Based on Recurrent Memorized Attention Network. *Electronics* **2020**, *9*, 2038. [CrossRef]
32. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2014**, arXiv:1312.6199.
33. Biggio, B.; Corona, I.; Nelson, B.; Rubinstein, B.I.P.; Maiorca, D.; Fumera, G.; Giacinto, G.; Roli, F. Security Evaluation of Support Vector Machines in Adversarial Environments. *arXiv* **2014**, arXiv:1401.7727.
34. Nguyen, A.; Yosinski, J.; Clune, J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Proceedings of the CVPR 2015, Boston, MA, USA, 7–12 June 2015.
35. Gu, S.; Rigazio, L. Towards Deep Neural Network Architectures Robust to Adversarial Examples. *arXiv* **2015**, arXiv:1412.5068.
36. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2015**, arXiv:1412.6572.
37. Fawzi, A.; Fawzi, O.; Frossard, P. Analysis of classifiers' robustness to adversarial perturbations. *arXiv* **2016**, arXiv:1502.02590. [CrossRef]
38. Kereliuk, C.; Sturm, B.L.; Larsen, J. Deep Learning and Music Adversaries. *arXiv* **2015**, arXiv:1507.04761. [CrossRef]
39. Nøkland, A. Improving Back-Propagation by Adding an Adversarial Gradient. *arXiv* **2016**, arXiv:1510.04189.
40. Tabacof, P.; Valle, E. Exploring the Space of Adversarial Images. *arXiv* **2016**, arXiv:1510.05328.
41. Huang, R.; Xu, B.; Schuurmans, D.; Szepesvari, C. Learning with a Strong Adversary. *arXiv* **2016**, arXiv:1511.03034.
42. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. *arXiv* **2016**, arXiv:1511.04508.
43. Moosavi-Dezfooli, S.-M.; Fawzi, A.; Frossard, P. DeepFool: A simple and accurate method to fool deep neural networks. *arXiv* **2016**, arXiv:1511.04599.
44. Sabour, S.; Cao, Y.; Faghri, F.; Fleet, D.J. Adversarial Manipulation of Deep Representations. *arXiv* **2016**, arXiv:1511.05122.
45. Shaham, U.; Yamada, Y.; Negahban, S. Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization. *arXiv* **2016**, arXiv:1511.05432.
46. Luo, Y.; Boix, X.; Roig, G.; Poggio, T.; Zhao, Q. Foveation-based Mechanisms Alleviate Adversarial Examples. *arXiv* **2016**, arXiv:1511.06292.
47. Jin, J.; Dundar, A.; Culurciello, E. Robust Convolutional Neural Networks under Adversarial Noise. *arXiv* **2016**, arXiv:1511.06306.
48. Lee, T.; Choi, M.; Yoon, S. Manifold Regularized Deep Neural Networks using Adversarial Examples. *arXiv* **2016**, arXiv:1511.06381.
49. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial machine learning at scale. *arXiv* **2016**, arXiv:1611.01236.
50. Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; Li, J. Boosting adversarial attacks with momentum. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 9185–9193.
51. Taheri, S.; Khormali, A.; Salem, M.; Yuan, J.-S. Developing a Robust Defensive System against Adversarial Examples Using Generative Adversarial Networks. *Big Data Cogn. Comput.* **2020**, *4*, 11. [CrossRef]
52. Jeong, J.; Kwon, S.; Hong, M.P.; Kwak, J.; Shon, T. Adversarial attack-based security vulnerability verification using deep learning library for multimedia video surveillance. *Multimed. Tools Appl.* **2020**, *79*, 16077–16091. [CrossRef]
53. Paknezhad, M.; Ngo, C.P.; Winarto, A.A.; Cheong, A.; Yang, B.C.; Jiayang, W.; Kuan, L.H. Explaining Adversarial Vulnerability with a Data Sparsity Hypothesis. *arXiv* **2021**, arXiv:2103.00778.
54. Tariq, S.; Jeon, S.; Woo, S.S. Am I a Real or Fake Celebrity? Measuring Commercial Face Recognition Web APIs under Deepfake Impersonation Attack. *arXiv* **2021**, arXiv:2103.00847.
55. Xu, H.; Ju, A.; Wagner, D. Model-Agnostic Defense for Lane Detection against Adversarial Attack. *arXiv* **2021**, arXiv:2103.00663.

56. Baia, A.E.; Di Bari, G.; Poggioni, V. Effective Universal Unrestricted Adversarial Attacks using a MOE Approach. *arXiv* **2021**, arXiv:2103.00250.
57. Radoglou-Grammatikis, P.I.; Panagiotis, G.S. Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems. *IEEE Access* **2019**, *7*, 46595–46620. [CrossRef]
58. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167. [CrossRef]
59. Masdari, M.; Hemn, K. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Appl. Soft Comput.* **2020**, *92*, 106301. [CrossRef]
60. Aldweesh, A.; Abdelouahid, D.; Ahmed, Z.E. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowl. Based Syst.* **2020**, *189*, 105124. [CrossRef]
61. Wu, W.; Li, R.; Xie, G.; An, J.; Bai, Y.; Zhou, J.; Li, K. A survey of intrusion detection for in-vehicle networks. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 919–933. [CrossRef]
62. Silvio, E.Q.; Célio, A.; Diego, P.; Daniel, M. A survey on intrusion detection and prevention systems in digital substations. *Comput. Netw.* **2021**, *184*, 107679.
63. Efstathopoulos, G.; Grammatikis, P.R.; Sarigiannidis, P.; Argyriou, V.; Sarigiannidis, A.; Stamatakis, K.; Angelopoulos, M.K.; Athanasopoulos, S.K. Operational data based intrusion detection system for smart grid. In Proceedings of the 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Limassol, Cyprus, 11–13 September 2019.
64. Pawlicki, M.; Choraś, M.; Kozik, R. Defending network intrusion detection systems against adversarial evasion attacks. *Future Gener. Comput. Syst.* **2020**, *110*, 148–154. [CrossRef]
65. Shu, D.; Leslie, N.O.; Kamhoua, C.A.; Tucker, C.S. Generative adversarial attacks against intrusion detection systems using active learning. In Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning, Pittsburgh, PA, USA, 13 July 2020.
66. Alhajjar, E.; Maxwell, P.; Bastian, N.D. Adversarial Machine Learning in Network Intrusion Detection Systems. *arXiv* **2020**, arXiv:2004.11898.
67. Radanliev, P.; De Roure, D.; Walton, R.; Van Kleek, M.; Montalvo, R.M.; Santos, O.; Burnap, P.; Anthi, E. Artificial intelligence and machine learning in dynamic cyber risk analytics at the edge. *SN Appl. Sci.* **2020**, *2*, 1773. [CrossRef]
68. Radanliev, P.; De Roure, D.; Ani, U.; Carvalho, G. The ethics of shared Covid-19 risks: An epistemological framework for ethical health technology assessment of risk in vaccine supply chain infrastructures. *Health Technol.* **2021**, 1–9. [CrossRef]
69. Kim, J.; Park, G.; Kwon, J.; Cho, E. A CAPTCHA-like implicit authentication model for smart TV based on machine learning algorithms. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–15 January 2018; pp. 1–2. [CrossRef]
70. Kopp, M.; Matej, N.; Martin, H. Breaking captchas with convolutional neural networks. In Proceedings of the 17th Conference on Information Technologies—Applications and Theory (ITAT 2017), Martinské hole, Slovakia, 22–26 September 2017; pp. 93–99.
71. Ondrej, B.; Klecka, J. Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms. *IFAC-Papers OnLine* **2018**, *51*, 208–213. [CrossRef]
72. Yang, J.; Wang, F.; Chen, Z.; Jungang, H. An Approach for Chinese Character Captcha Recognition Using CNN. *J. Phys. Conf. Ser.* **2018**, *1087*, 022015.
73. Zahra, N.; Mahdi, R. Deep-CAPTCHA: A deep learning based CAPTCHA solver for vulnerability assessment. *arXiv* **2020**, arXiv:2006.08296.
74. Shu, Y.; Xu, Y. End-to-End Captcha Recognition Using Deep CNN-RNN Network. In Proceedings of the 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 11–13 October 2019; pp. 54–58. [CrossRef]
75. Osadchy, M.; Hernandez-Castro, J.; Gibson, S.; Dunkelman, O.; Pérez-Cabo, D. No Bot Expects the DeepCAPTCHA! Introducing Immutable Adversarial Examples, With Applications to CAPTCHA Generation. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2640–2653. [CrossRef]
76. Papernot, G.N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The Limitations of Deep Learning in Adversarial Settings. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, Germany, 21–24 March 2016; pp. 372–387. [CrossRef]
77. Zhang, Y.; Gao, H.; Pei, G.; Kang, S.; Zhou, X. Effect of Adversarial Examples on the Robustness of CAPTCHA. In Proceedings of the 2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Zhengzhou, China, 18 October 2018; pp. 1–109. [CrossRef]
78. Moosavi-Dezfooli, S.; Fawzi, A.; Fawzi, O.; Frossard, P. Universal Adversarial Perturbations. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 26 July 2017; pp. 86–94. [CrossRef]
79. Kwon, H.; Yoon, H.; Park, K.W. *Robust CAPTCHA Image Generation Enhanced with Adversarial Example Methods*; Ieice Transactions on Information and Systems: Tokyo, Japan, 2020; Volume: E103D. [CrossRef]
80. Shao, R. Robust Text CAPTCHAs Using Adversarial Examples. *arXiv* **2021**, arXiv:2101.02483.
81. Bengio, Y.; Yao, L.; Alain, G.; Vincent, P. Generalized Denoising Auto-Encoders as Generative Models. *arXiv* **2013**, arXiv:1305.6663.

82. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Covolutions. *arXiv* **2014**, arXiv:1409.4842.
83. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
84. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
85. Francois, C. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv* **2017**, arXiv:1610.02357.
86. Gao, H.; Zhuang, L.; Laurens, V.D.M.; Kilian, Q.W. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:1608.06993.
87. Leslie, N.S. Cyclical Learning Rates for Training Neural Networks. *arXiv* **2017**, arXiv:1506.01186.
88. Nicholas, C.; Anish, A.; Nicolas, P.; Wieland, B.; Jonas, R.; Dimitris, T.; Ian, G.; Aleksander, M.; Alexey, K. On Evaluating Adversarial Robustness. *arXiv* **2019**, arXiv:1902.06705.
89. Kui, R.; Tianhang, Z.; Zhan, Q.; Xue, L. Adversarial Attacks and Defenses in Deep Learning. *Engineering* **2020**, *6*, 346–360. [CrossRef]
90. Dankwa, S.; Zheng, W. Special Issue on Using Machine Learning Algorithms in the Prediction of Kyphosis Disease: A Comparative Study. *Appl. Sci.* **2019**, *9*, 3322. [CrossRef]
91. Xu, Z.; Liu, W.; Huang, J.; Yang, C.; Lu, J.; Tan, H. Artificial Intelligence for Securing IoT Services in Edge Computing: A Survey. *Secur. Commun. Netw.* **2020**, *2020*, 13. [CrossRef]