electronics

MDPI

Article

An Efficient and Accurate Depth-Wise Separable Convolutional Neural Network for Cybersecurity Vulnerability Assessment Based on CAPTCHA Breaking

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: Cybersecurity practitioners generate a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHAs) as a form of security mechanism in website applications, in order to differentiate between human end-users and machine bots. They tend to use standard security to implement CAPTCHAs in order to prevent hackers from writing malicious automated programs to make false website registrations and to restrict them from stealing end-users' private information. Among the categories of CAPTCHAs, the text-based CAPTCHA is the most widely used. However, with the evolution of deep learning, it has been so dramatic that tasks previously thought not easily addressable by computers and used as CAPTCHA to prevent spam are now possible to break. The workflow of CAPTCHA breaking is a combination of efforts, approaches, and the development of the computation-efficient Convolutional Neural Network (CNN) model that attempts to increase accuracy. In this study, in contrast to breaking the whole CAPTCHA images simultaneously, this study split four-character CAPTCHA images for the individual characters with a 2-pixel margin around the edges of a new training dataset, and then proposed an efficient and accurate Depth-wise Separable Convolutional Neural Network for breaking text-based CAPTCHAs. Most importantly, to the best of our knowledge, this is the first CAPTCHA breaking study to use the Depth-wise Separable Convolution layer to build an efficient CNN model to break text-based CAPTCHAs. We have evaluated and compared the performance of our proposed model to that of fine-tuning other popular CNN image recognition architectures on the generated CAPTCHA image dataset. In real-time, our proposed model used less time to break the text-based CAPTCHAs with an accuracy of more than 99% on the testing dataset. We observed that our proposed CNN model has efficiently improved the CAPTCHA breaking accuracy and streamlined the structure of the CAPTCHA breaking network as compared to other CAPTCHA breaking techniques.

Keywords: deep learning; convolutional neural network; depth-wise separable; captcha; cybersecurity; vulnerability assessment; internet

check for updates

Citation: 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. https://doi.org/10.3390/electronics10040480

Received: 28 December 2020
Accepted: 15 February 2021
Published: 18 February 2021

1. Introduction

Presently, numerous daily life activities which include communication, travel and tours, education, online E-commerce, entertainment, and most importantly money transactions are carried out by connecting to the internet. In order to perform such website tasks, legitimate end-users have to provide their private information by registering to the website. In creating a website account or registering, some hackers tend to write malicious programs which waste the website resources by making automatic false registrations called bots. These false registrations may affect the website and make it vulnerable to hackers, where they can go further to steal end-users' private information or intercept their transactions. Therefore, in order to differentiate between human end-users and machine bots, cybersecurity practitioners generate CAPTCHAs as a form of security mechanism in website applications to defend end-users' private information from automated malicious attacks.

CAPTCHA, which stands for (Completely Automated Public Turing test to tell Computers and Humans Apart), provides a way for web service providers to make some assumptions about whether an end-user is human or robot [1]. The most famous way of protecting internet forms is to create a special image made up of letters and numbers and then require the user to enter it in a special textbox, as seen in Figure 1. CAPTCHAs serve as a network security approach, which are used for websites to avoid automatic form entering, spamming, automatic voting, etc. [2].

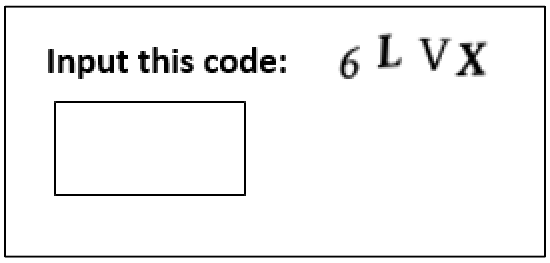


Figure 1. Sample text-based completely automated public turing test to tell computers and humans apart (CAPTCHAs) to be input into a textbox.

The majority of text-based CAPTCHAs consist of English uppercase letters (A to Z), English lowercase letters (a to z), and numerals (0 to 9) [3]. These text-based CAPTCHAs are distorted text images which can be misrecognized by computers or robots but can highly be recognized by humans. The letters in text-based CAPTCHAs are sometimes overlapped, rotated, or an addition of curvature to the text image [4].

The recognition accuracy based on humans for effective CAPTCHAs is at least 80%, while less than 0.01% recognition accuracy is based on computers [2,5]. Sometimes, since some companies or website owners want to satisfy their customers to easily recognize these CAPTCHAs, they are quickly compelled to generate loose CAPTCHAs, especially text-based CAPTCHAs with white backgrounds.

The CAPTCHA recognition technology in the area of traditional image processing, is divided into image preprocessing, positioning, character segmentation, and character recognition. This traditional approach, however, is difficult to form an accurate template set as a result of the adhered and complex CAPTCHA images such as rotation and overlapping [2]. Any CAPTCHA is vulnerable to hackers, regardless of the image-generating algorithm [1]. However, with a constant ethical vulnerability assessment, the loopholes in the CAPTCHA security can be assessed and then reduced.

The pipeline of CAPTCHA breaking consists of efforts, approaches, and a model that attempts to increase accuracy [1]. According to Kolupaev et al. [1], a neural network could recognize a single letter easier than a human could, and this fact did not depend on font style, rotation, or distortion. This means that if a hacker is well invested in building a neural network, the only defense mechanism is to make the characters hard to separate, but with the possibility to separate in order to provide good training data for the CNN to train on and then break these CAPTCHAs with high accuracy. This is a goal we want to achieve in the current study.

Recently, deep learning, which is a subfield in machine learning [6], is one of the demanded areas in artificial intelligence research. Deep learning has achieved good performance and great success in many applications such as scene recognition [7], image recognition [8], object detection [9–13], and image restoration [14–16]. In contrast to the traditional pattern recognition technique, the huge advantage of deep learning is its effectiveness in learning features actively without artificial design [2].

The advancement of deep learning has been so dramatic that tasks previously thought not easily addressable by computers and used as CAPTCHA to prevent spam are now possible to solve. Based on this observation, we are motivated to develop an efficient and accurate CNN architecture to break text-based CAPTCHAs with white backgrounds. The main idea is to perform a vulnerability assessment by developing a faster and high accuracy model. The significance of this study will create an awareness for the CAPTCHA-generating practitioners to assess the strengths and weaknesses of their generating algorithms in order to first observe if they can detect security loopholes in the text-based CAPTCHAs before deploying on websites.

This will assist them in generating an improved text-based CAPTCHA in the area of an automated challenge and respond against attacks dispensed by automated machines or bots. This study recognizes and acknowledges the previous CAPTCHA breaking techniques which have implemented their proposed CNN architectures using standard convolution layers. However, since a wide range of websites still use text-based CAPTCHAs, the trust of the end-users must not be betrayed through malicious attacks. CAPTCHA-generating practitioners must imitate the hackers through ethical means to perform a cybersecurity vulnerability assessment by developing faster, robust, and efficient algorithms than can outperform.

Therefore, we propose an efficient and accurate Depth-wise Separable Convolutional Neural Network to break text-based CAPTCHAs. Several CAPTCHA breaking practitioners have implemented their CNN architectures using standard CNNs. However, to the best of our knowledge, this is the first CAPTCHA breaking study to adopt the Depth-wise Separable CNN to break text-based CAPTCHAs.

In this current work, we are not utilizing the whole CAPTCHA image as an input for training the network, but we are rather splitting the images for the individual characters. We prove that, with a good and practical approach for creating the right training data for the Convolutional Neural Network (CNN), one may not require to fine-tune the existing state-of-the-art models to break CAPTCHAs, especially with a white background. This is due to the fact that these existing CNN architectures have a high number of parameters, may take a longer time to train, and the model sizes may relatively increase, which may consume a lot of memory and resources.

Therefore, this study makes the problem simple by extracting all the single letters from the CAPTCHA images with a 2-pixel margin around the edges, annotate, and then label them automatically for the CNN architecture to be trained on. The CNN has to use these single letters as inputs rather than the whole CAPTCHA image relative to the current problem at hand. A detailed description of the technique used to extract the single characters is given in the next section.

In this approach, with a small number of convolution layers in a well-developed CNN architecture, as proposed in this work, the experimental results show that the proposed technique performed better to break text-based CAPTCHAs with rotation, overlapping, and with a white background. One of the importance of CAPTCHA breaking research is to find security loopholes in CAPTCHAs, as a form of a vulnerability assessment technique, to assist CAPTCHA generating practitioners in generating improved text-based CAPTCHAs.

2. Materials and Methods

This section presents the dataset used in this work, the problem definition, and a practical way to tackle it. In addition, it provides the idea behind the proposed CAPTCHA breaking algorithm, the internal structure of the proposed computation-efficient CNN model, and its evaluation metrics.

2.1. Data Description

The CAPTCHA breaking research is very sensitive, and for that matter, there are no publicly available standard datasets of CAPTCHA images to be used. As a result, we are motivated to obtain CAPTCHA images either by retrieving them from real websites or generating them using libraries. Therefore, we adopted a low-cost approach to generate CAPTCHA images using the open source python CAPTCHA library (https://pypi.org/project/captcha/ (accessed on 20 December 2020)). We obtained 10,000 CAPTCHA images, which are similar to real world CAPTCHA images such as the Weibo CAPTCHA scheme. The images contain four characters consisting of numerals and uppercase English letters.

We excluded the [0,1,O,I] characters to avoid confusion to both humans and machines. The sample of the generated CAPTCHA images can be seen from Figure 2a. The dataset poses a challenge where some of the characters are overlapping as seen in Figure 2b. The approach for mitigating the problem is discussed in the next section. The dataset and codes used for implementing this current research study are available at (https://github.com/sm-multimedia/Text-based-CAPTCHA-Breaking- (accessed on 27 January 2021)).

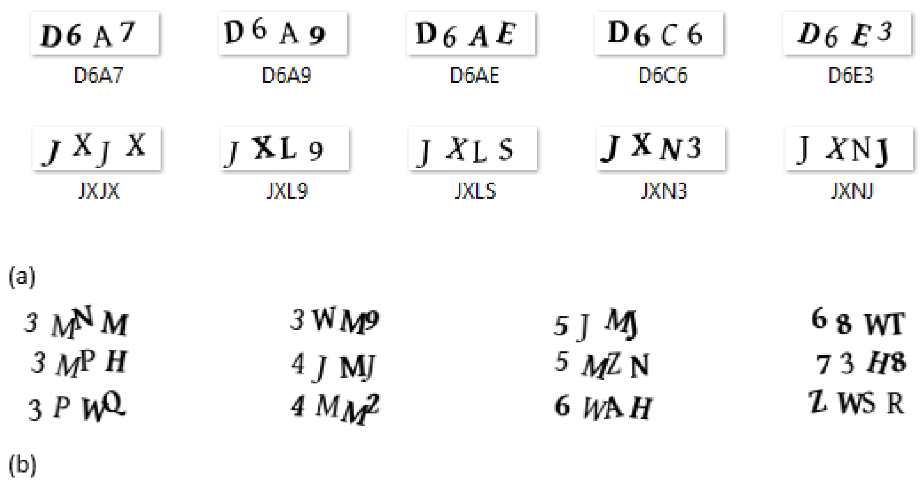


Figure 2. (**a**) Sample CAPTCHA images used in this work; (**b**) sample rotated and overlapped CAPTCHA images, which pose a challenge for splitting.

2.2. Problem Definition

The problem definition in this current study is to input a text-based CAPTCHA image which has rotated, overlapping characters, and especially with a white background, and then break the same CAPTCHA with a high accuracy rate, as illustrated in Figure 3a. The problem can be solved by using deep learning to break the characters in the CAPTCHA image simultaneously, as shown in Figure 3b.

However, in contrast to the approach in Figure 3b, the problem is made simple by splitting the images for each single character, annotating, and labelling them to generate a new training dataset for a simple and efficient proposed CNN architecture. The graphical representation of this scenario is illustrated in Figure 3c.

2.3. Basic Idea of the Single Character Extraction (SCE) Algorithm

Based on Figure 3c, the SCE algorithm was developed which helped generate a new training dataset. The original text-based CAPTCHA images are loaded from the disk and split for the individual characters to output a single character. The general formula for the new training dataset set is expressed as:

$$T = C\ X\ N \tag{1}$$

where T represents the new number of training dataset, C represents the number of characters in each CAPTCHA image, and N represents the total number of CAPTCHA images.

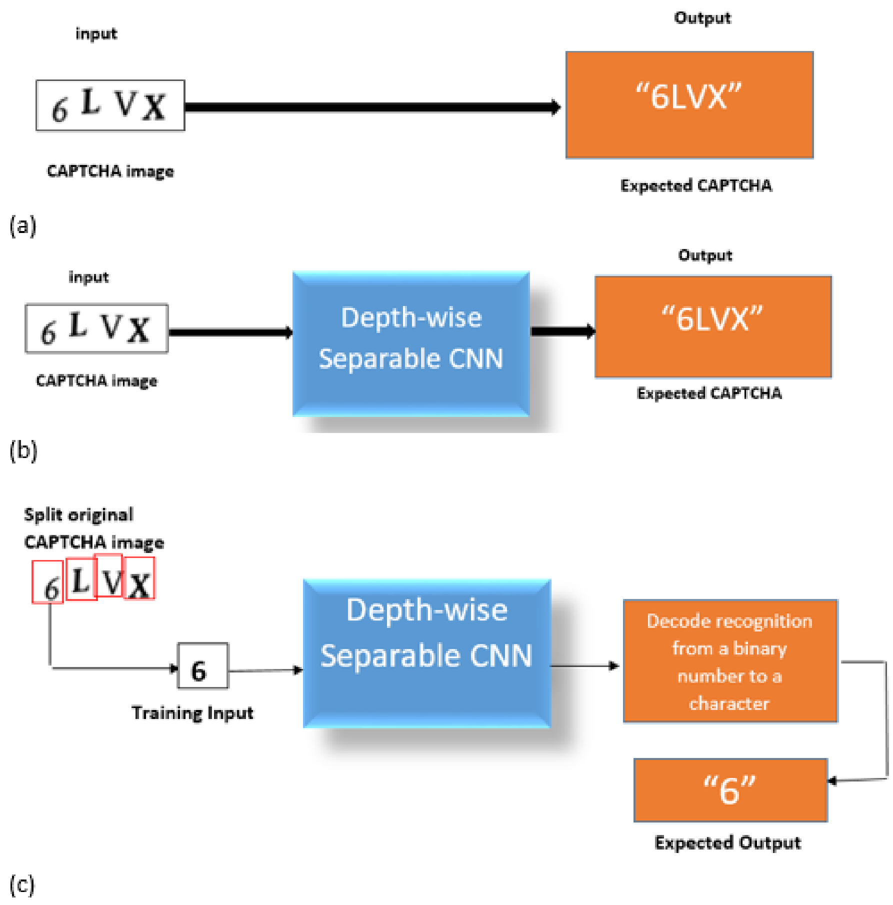
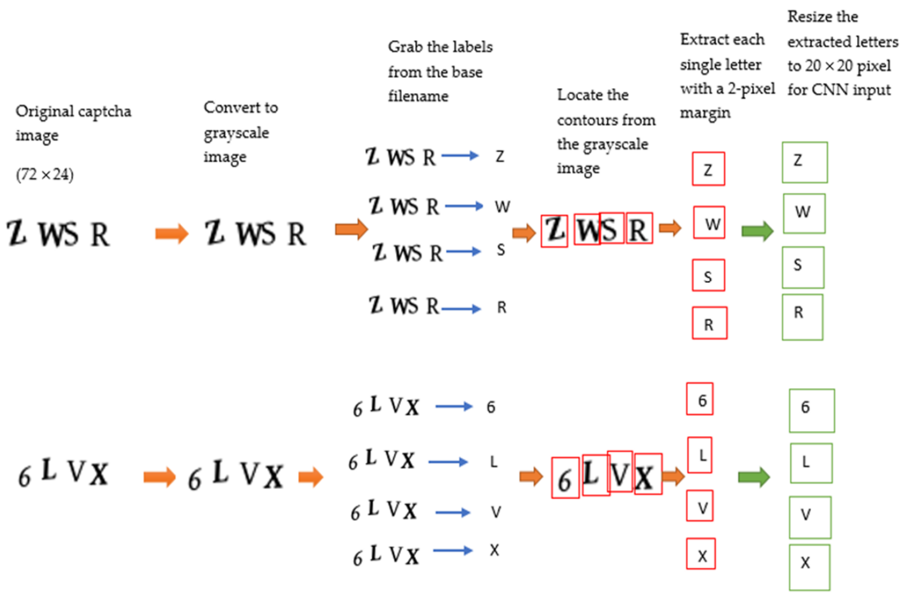


Figure 3. (**a**) The problem definition is to input a CAPTCHA image and output the same characters in the image, (**b**) the problem can be solved by passing the input image through the convolutional neural network (CNN) model to break the same characters simultaneously, (**c**) the pipeline of the proposed CAPTCHA breaking algorithm.

The main concept backing the proposed technique is to make the problem simple, especially when dealing with four-character CAPTCHA images with white backgrounds. The original text-based CAPTCHA images are loaded and converted to grayscale images for faster preprocessing. The base filenames of the original CAPTCHA images contain the labels which can be extracted, the images are then binarized through a thresholding technique. Thresholding the images ensures that the backgrounds of the images are black, while the foregrounds are white. This part of the preprocessing is a critical step in the workflow since it assists in finding the outlines of each of the characters in the CAPTCHA images. The contours of the characters in the threshold images are looped over and located. The bounding boxes for the contours are computed with a 2-pixel margin around the edges to extract each single letter. In the course of the extraction, the overlapping characters are detected. This part is as critical as the thresholding, if not handled well, it will end up generating a bad training dataset.

Therefore, this study checked the problem by comparing the width and height of the contour, if the width divided by the height is greater than a certain number of threshold, in the case of this study, the threshold was 1.25 based on the experiments, then the letter is too wide to be a single letter, which is then split into two. Finally, the extracted single letters are annotated and labelled for training. The single characters can be resized to a desirable input size for the Convolutional Neural Network (CNN) architecture. The result of the Single Character Extraction (SCE) algorithm is used to train a computation-efficient Convolutional Neural Network model to break the CAPTCHA characters. A graphical description of the SCE algorithm is illustrated in Figure 4.



Original captcha image
(72 × 24)
Convert to grayscale image
Grab the labels from the base filename
Locate the contours from the grayscale image
Extract each single letter with a 2-pixel margin
Resize the extracted letters to 20 × 20 pixel for CNN input
Z WS R
Z
W
S
R
6 L V X
6
L
V
X

Figure 4. The description of the main concept of the proposed technique algorithm.

2.4. Network Structure and Parameters of the Proposed CAPTCHA Breaking Depth-Wise Saparable CNN

The proposed CNN architecture is inspired by MobileNets [17], but in contrast to MobileNets, we introduce a lot of dropout layers and without a pointwise convolutional layer. The architecture of our proposed CNN model consists of six Depth-wise Separable Convolutional layers, seven Batch Normalization layers, seven ReLU activation function layers, four Dropout layers, three Max-pooling layers, one flatten layer, one fully connected layer, and one output Softmax layer.

The network architecture uses exclusively 3 × 3 Depth-wise Separable Convolutional filters, stacked on top of each other preceding the performed max-pooling. This current research work adopted the Depth-wise Separable Convolution rather than the standard convolution layers. Since it is more efficient, it requires less memory, less computation, and at some situation, can perform better than the standard convolution. In simplicity, the architecture of the proposed CNN model uses three (DEPTHWISE_CONV => ReLU => POOL) blocks with an increasing stacking number of filters. The first block consists of one Depth-wise Separable Convolution layer with 32 filters, followed by the Rectified Linear Unit (ReLU) activation function, which is popularly used in deep learning for faster and effective training, Batch Normalization, Max-pooling, and Dropout, which regularize the network by adding noise to the output feature maps of each layer. The second block consists of two Depth-wise Separable Convolution layers with 64 filters each, followed by two Rectified Linear Unit (ReLU) activation functions at the end of each Convolution layer, two Batch-Normalizations at the end of each ReLU, a Max-pooling, and a Dropout. The third block consists of three Depth-wise Separable Convolution layers with 64 filters each, followed by three Rectified Linear Unit (ReLU) activation functions at the end of each Convolution layer, three (3) Batch Normalization layers at the end of each ReLU layer, a Max-pooling, and a Dropout layer. The Batch Normalization and Dropout layers ensure the stability of the model and prevent over-fitting, respectively. The proposed architecture has only one fully-connected layer which consists of a flattened layer, followed by 256 hidden neurons, Rectified Linear Unit (ReLU) activation function, Batch Normalization, Max-pooling, and Dropout. The output layer consists of 32 nodes, which include the number of characters to be predicted by the network with a Softmax layer.

In the Depth-wise Separable operation, convolution is applied to a single channel at a time as compared to the standard CNN, in which it is done for all the M channels. Therefore, in the depth-wise convolution, the filters or kernels will be of $Dk \times Dk \times 1$ in size. Suppose there are M channels in the input image, then the M filters are needed. As a result, the output will be of $Dp \times Dp \times M$ in size. The operation of a single convolution needs $Dk \times Dk$ multiplications. As the filters slide by $Dp \times Dp$ times across all the M channels, then the total number of multiplications is equal to $M \times Dp \times Dp \times Dk \times Dk$. Therefore, for the Depth-wise Separable Convolution, the cost operation can be expressed as:

$$\text{Total number of multiplication} = M \times Dk^2 \times Dp^2 \quad (2)$$

The network architecture of the proposed CNN model can be seen in Figure 5, and the full description of the proposed CAPTCHA Breaking CNN is illustrated in Figure 6.

2.5. Evaluation Metrics

The proposed CAPTCHA breaking CNN model was evaluated using Accuracy, Precision, and Recall. The formulas for Accuracy, Precision, and Recall are given as:

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

$$\text{Precision} = \frac{TP}{(TP + FP)}, \text{ Recall} = \frac{TP}{(TP + FP)}$$

where TP is true positive, FP is false positive, TN is true negative, and FN is false negative based on the model's prediction.

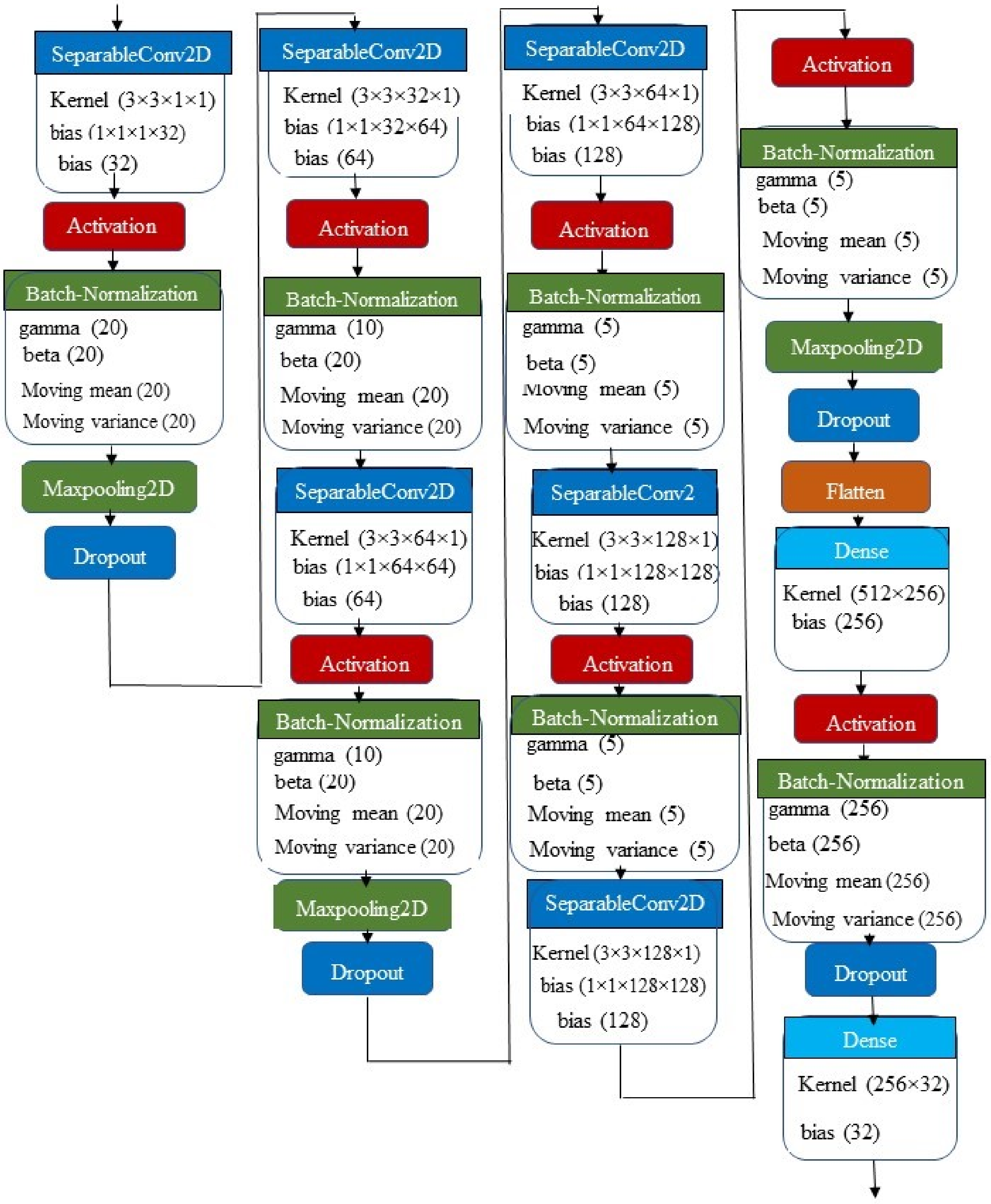


Figure 5. The network architecture of the proposed depth-wise separable CNN.

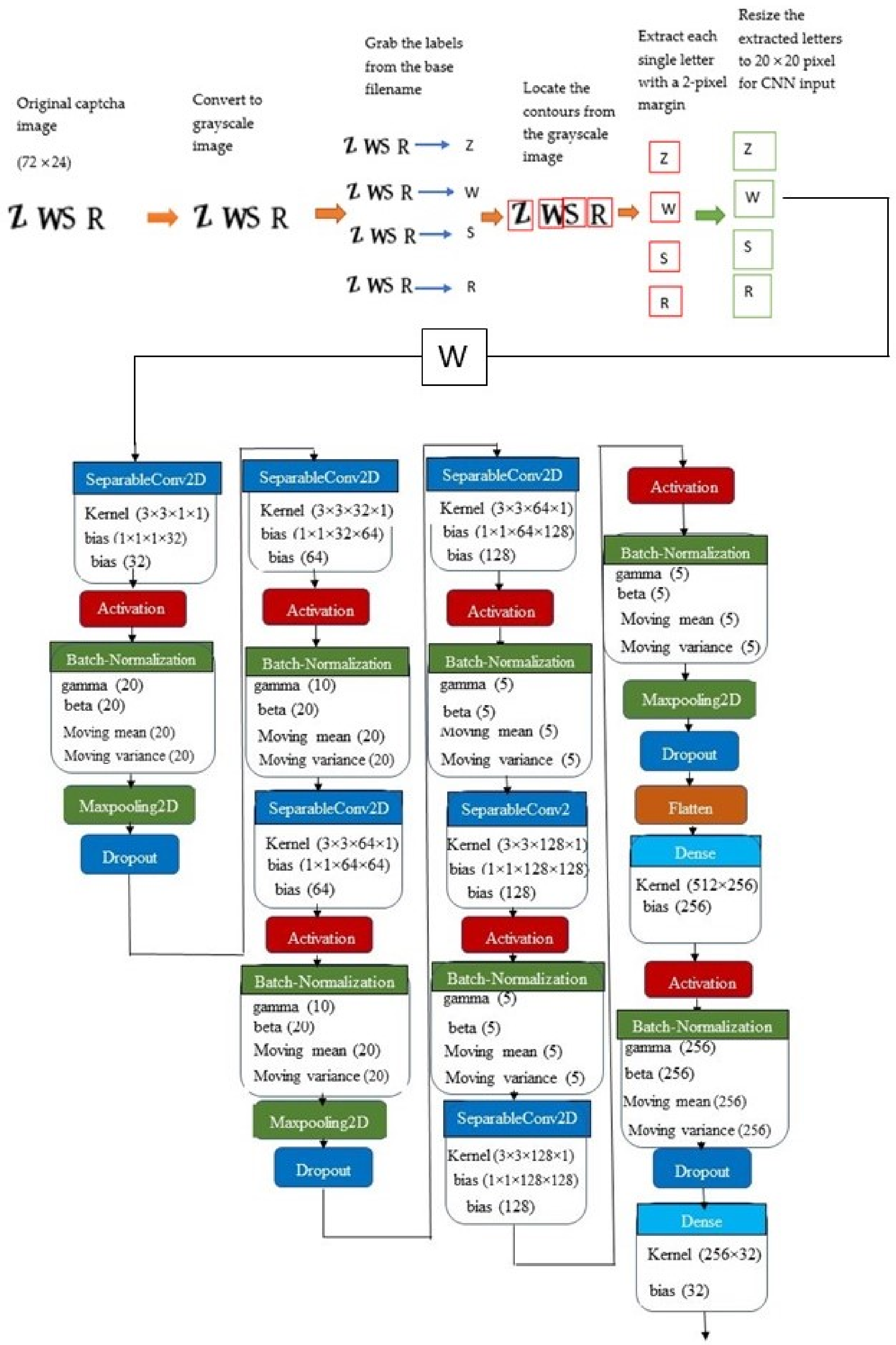


Figure 6. The network architecture of the proposed CAPTCHA breaking CNN. The legend shows the various component-layers in the proposed depth-wise separable CNN. BN: Batch Normalization.

3. Results

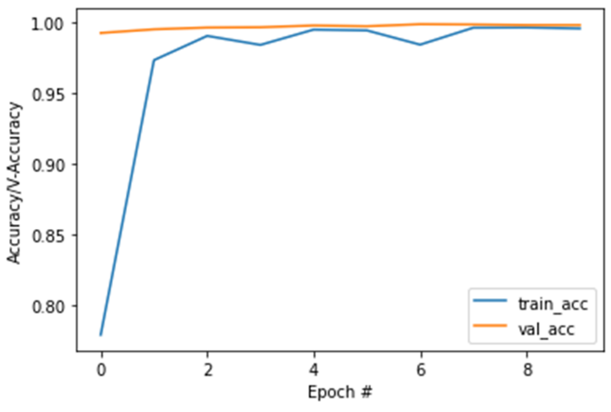
This section presents the training process of the proposed CAPTCHA breaking Depth-wise Separable CNN algorithm, the experiment results of our proposed model, and the evaluation of the proposed CNN on the testing dataset. The competitive result performance of the proposed CAPTCHA breaking CNN model is compared to other notable fine-tuned CNN architectures. Our proposed CNN model was implemented using the Python 3.6 programming language, executed, and trained on the GPU based on the Google Collaboratory environment (https://colab.research.google.com/ (accessed on 16 December 2021)).

3.1. Training Process Phase

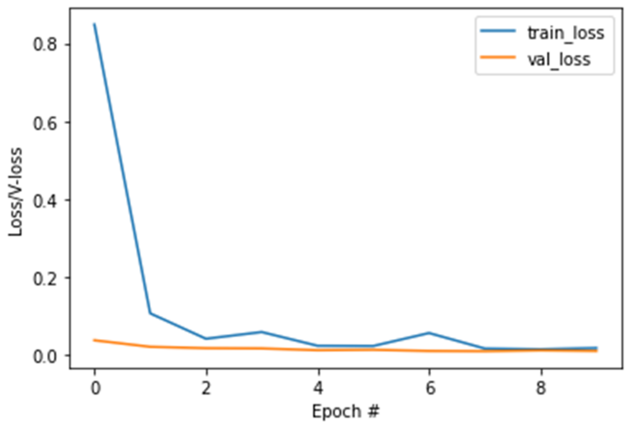
This study obtained a dataset of about 40,000 single-character images, which was annotated and labelled using the SCE algorithm. With robust experiments, the dataset was partitioned into training, validation, and testing using 75% of the data for training, 10% for validation, and 15% for testing. The input single images have a width of 20 pixels, a height of 20 pixels, and a single channel. Based on the experiments, a 25% Dropout was added at the end of each three (DEPTHWISE_CONV => ReLU => POOL) blocks. While a fifty percent Dropout was added at the end of the fully-connected dense layer. The network had an output Softmax layer of 32 neurons, one for each character prediction. In the training process of our proposed CNN model, the categorical cross-entropy loss function was to measure the difference between the predicted and actual classes since it is a multi-classification problem. The network used Adam [17] optimizer to optimize the loss function based on the experiment. With vigorous testing, a batch size of 32 and 10 epochs assisted the network to perform better.

3.2. Our Proposed CNN Model Performance on the CAPTCHA Dataset

There was a total parameter of 192,229 based on the proposed Depth-wise Separable CNN model. The trainable and non-trainable parameters were 191,607 and 622, respectively. Figure 7a,b shows the accuracies and losses based on the proposed CNN, respectively. The training time lasted for 7s and 7 ms/step on the GPU. The proposed CNN model showed a good performance on the individual characters based on precision, recall, and f1-score on the testing dataset. Our proposed CNN model obtained accuracy (100%), precision (100%), recall (100%), and F1-score (100%) on the testing dataset. As seen from Figure 7a, it could be observed that, after six epochs, the training and validation accuracies began to match each other. The same applied to the training and validation losses after six epochs. These observations showed stability in the proposed model.



(**a**)



(**b**)

Figure 7. (**a**) The training and validation accuracies after training the proposed CNN model on the CAPTCHA dataset; (**b**) the training and validation losses after training the proposed CNN model on the CAPTCHA dataset.

3.3. Comparison Results

In order to show the strength and weakness of the proposed CAPTCHA breaking CNN model, this study fine-tuned notable state-of-the-art image recognition CNN architectures on the same dataset. These popular CNN architectures were MobileNets [18], DenseNet [19], ShuffleNet [20], and Xception [21]. Based on extensive experiments and the problem at hand, this study chose these CNN architectures in terms of the small number of their parameters compared to VGGNet [22] and ResNets [23]. In addition, notably, residual networks are difficult to train.

Firstly, in the mobileNets architecture, all the layers are followed by a Batch Normalization and RELU nonlinearity. The network architecture begins with the (Conv => BatchNorm => Relu) block and continues with a series of MobileNets blocks before the Average Pool and fully connected layers. The MobileNets block consists of six layers, which follow as: a 3 × 3 Depth-wise separable convolution layer, a Batch Normalization layer, a Rectified Linear Unit (ReLU) activation layer, a 1 × 1 convolution layer, a Batch Normalization layer, and a Rectified Linear Unit (ReLU) activation layer. The Depth-wise Separable Convolution has a lesser number of parameters to adjust as compared to the standard CNN which reduces over-fitting. It is computationally cheaper due to fewer computations which makes it suitable for mobile vision applications.

Furthermore, this current study evaluated the performance of the proposed CNN model through the transfer learning approach. In addition, the MobileNets architecture was fine-tuned and then trained on the same CAPTCHA dataset. The old classifier of the MobileNets architecture was removed, and a new classifier was replaced with 32 nodes. As shown in Figure 8, the network summary of the fine-tuned MobileNets showed a total parameter of 3,272,032. The total number of trainable parameters and non-trainable parameters showed 3,250,144 and 21,888, respectively.

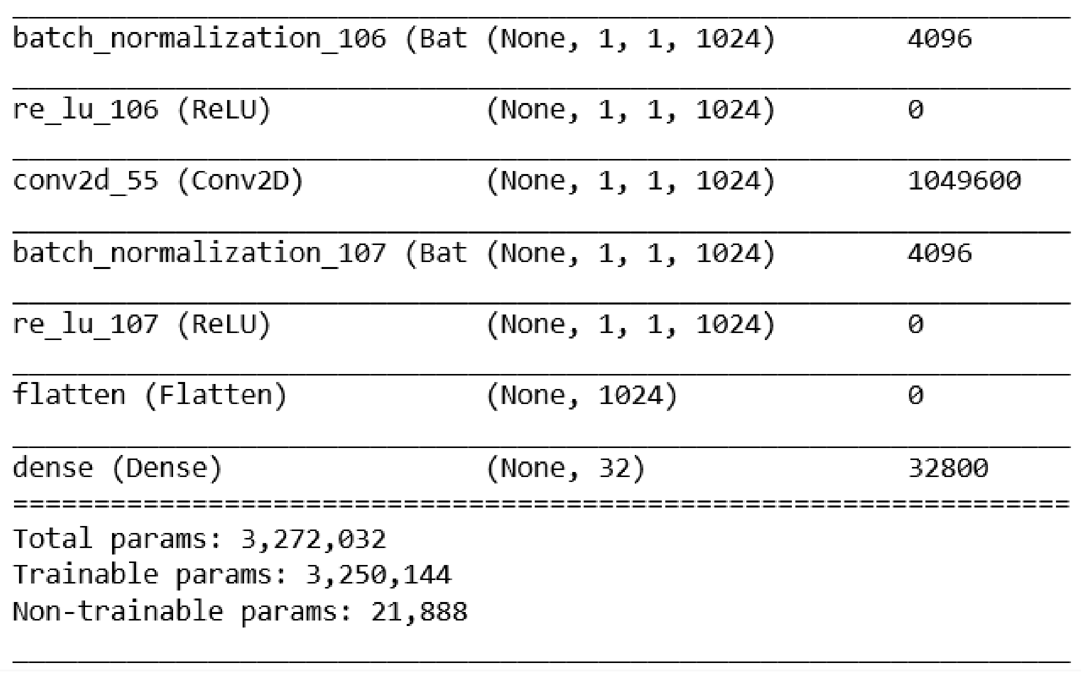
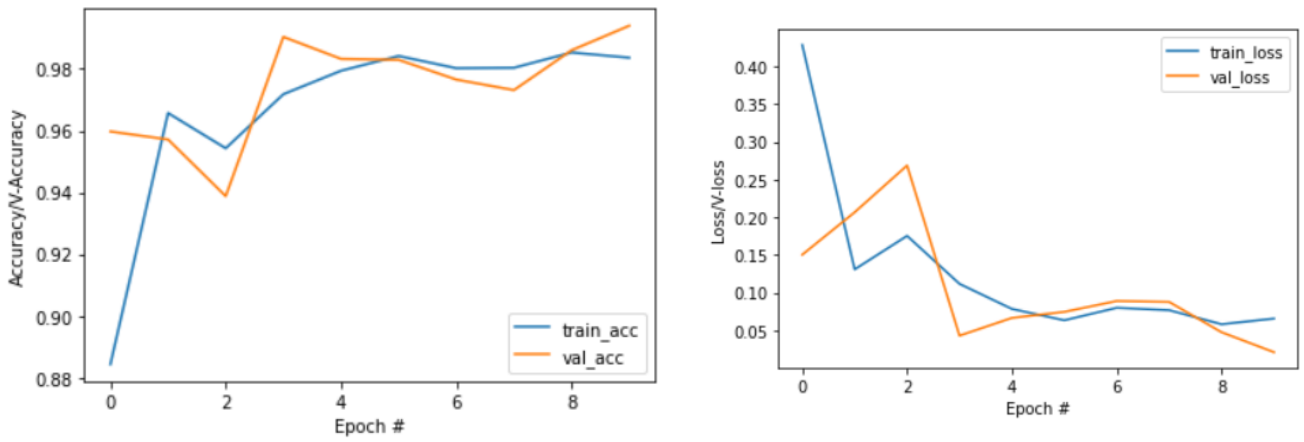

```
_________________________________________________________________
batch_normalization_106 (Bat (None, 1, 1, 1024)        4096
_________________________________________________________________
re_lu_106 (ReLU)             (None, 1, 1, 1024)        0
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 1, 1, 1024)        1049600
_________________________________________________________________
batch_normalization_107 (Bat (None, 1, 1, 1024)        4096
_________________________________________________________________
re_lu_107 (ReLU)             (None, 1, 1, 1024)        0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 32)                32800
=================================================================
Total params: 3,272,032
Trainable params: 3,250,144
Non-trainable params: 21,888
_________________________________________________________________
```

Figure 8. The summary of the MobileNets architecture on the CAPTCHA dataset.

Figure 9a,b shows the accuracies and losses based on the fine-tuned MobileNets, respectively. The training time lasted for 10 s and 11 ms/step on the GPU.

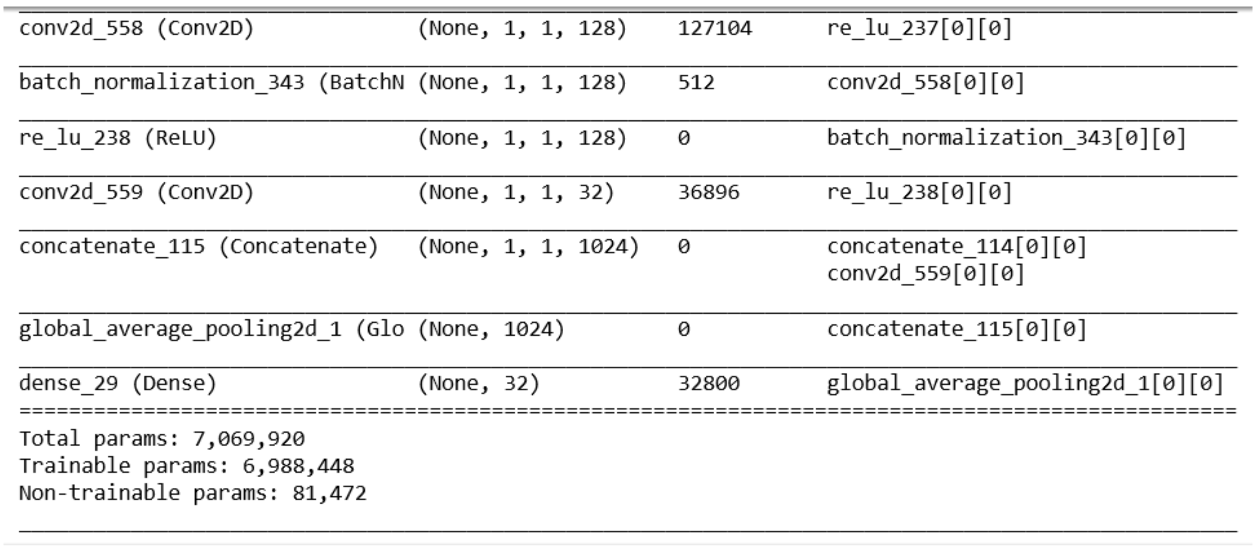


(**a**) (**b**)

Figure 9. (**a**) The training and validation accuracies after training MobileNets on the CAPTCHA dataset; (**b**) the training and validation losses after training MobileNets on the CAPTCHA.

Again, in the DenseNet research paper, the input tensor in every dense block passes through a series of convolutional operations with a fixed number of filters (k) and the outcome of each is then concatenated to the original tensor. The initial Convolution Layer comprises 2 k convolutions of size 7 × 7 with stride 2. In their experiments, they let each 1 × 1 convolution produce 4 k feature-maps.

This present study evaluated the performance of the proposed Depth-wise Separable CNN model by adopting a transfer learning approach. In addition, the DenseNet architecture was fine-tuned and then trained on the CAPTCHA dataset. The Dense-121 (k = 32) version of the model was implemented since it was very easy to modify and train. The old classifier of the DenseNet architecture was removed, and a new classifier was replaced with 32 nodes. As shown in Figure 10, the result network summary of the fine-tuned DenseNet showed a total parameter of 7,069,920. The trainable parameters and non-trainable parameters were 6,988,448 and 81,472, respectively. Figure 11a,b shows the accuracies and losses based on the fine-tuned DenseNet, respectively.



```
conv2d_558 (Conv2D)             (None, 1, 1, 128)    127104      re_lu_237[0][0]
__________________________________________________________________________________________________
batch_normalization_343 (BatchN (None, 1, 1, 128)    512         conv2d_558[0][0]
__________________________________________________________________________________________________
re_lu_238 (ReLU)                (None, 1, 1, 128)    0           batch_normalization_343[0][0]
__________________________________________________________________________________________________
conv2d_559 (Conv2D)             (None, 1, 1, 32)     36896       re_lu_238[0][0]
__________________________________________________________________________________________________
concatenate_115 (Concatenate)   (None, 1, 1, 1024)   0           concatenate_114[0][0]
                                                                 conv2d_559[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 1024)         0           concatenate_115[0][0]
__________________________________________________________________________________________________
dense_29 (Dense)                (None, 32)           32800       global_average_pooling2d_1[0][0]
==================================================================================================
Total params: 7,069,920
Trainable params: 6,988,448
Non-trainable params: 81,472
__________________________________________________________________________________________________
```

Figure 10. The summary of the DenseNet architecture on the CAPTCHA dataset.

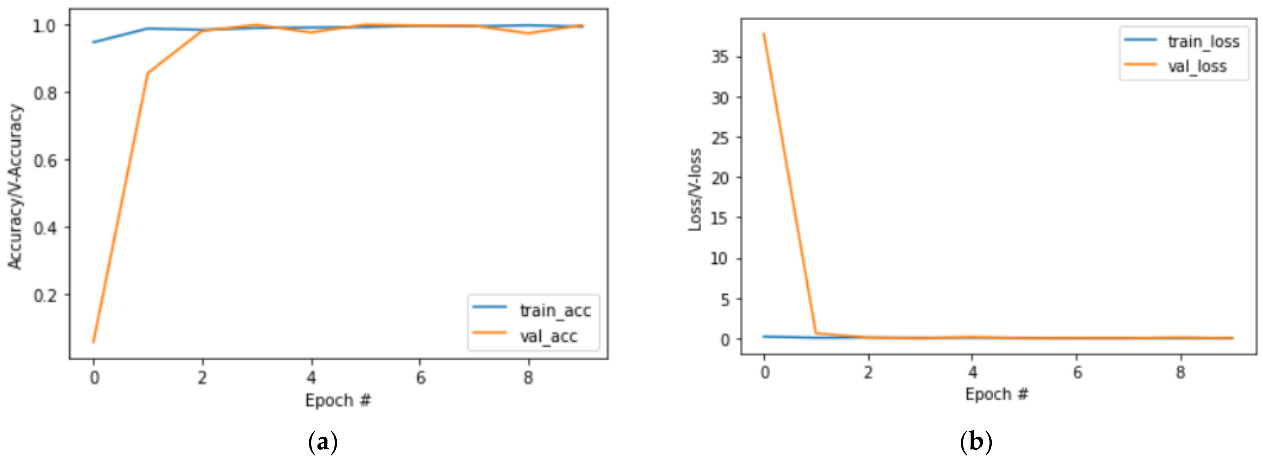
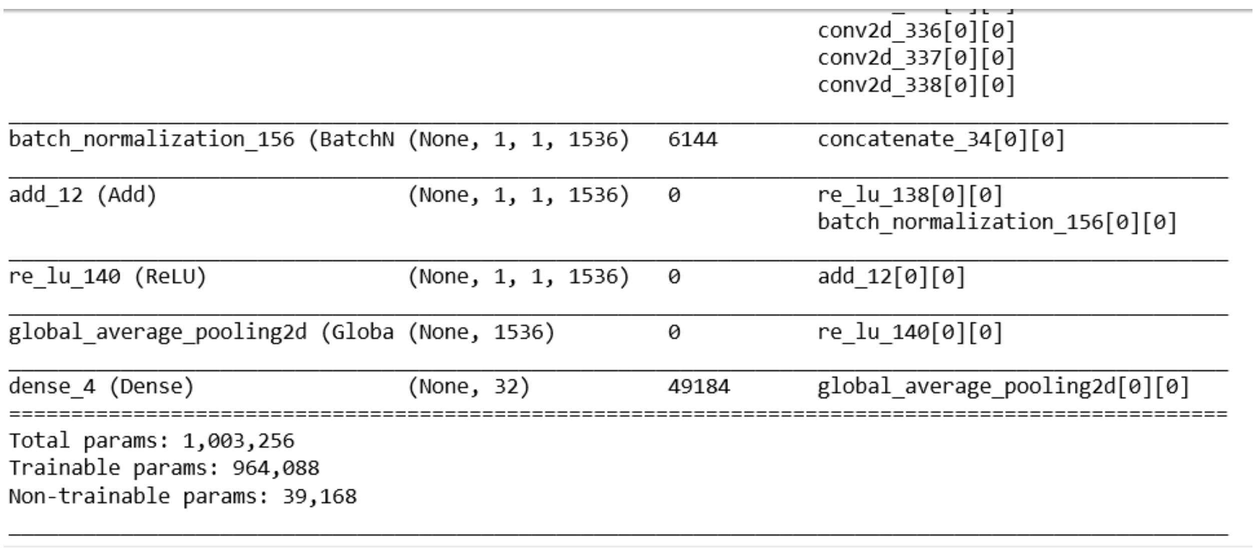


Figure 11. (**a**) The training and validation accuracies after training DenseNet on the CAPTCHA dataset; (**b**) the training and validation losses after training DenseNet on the CAPTCHA dataset.

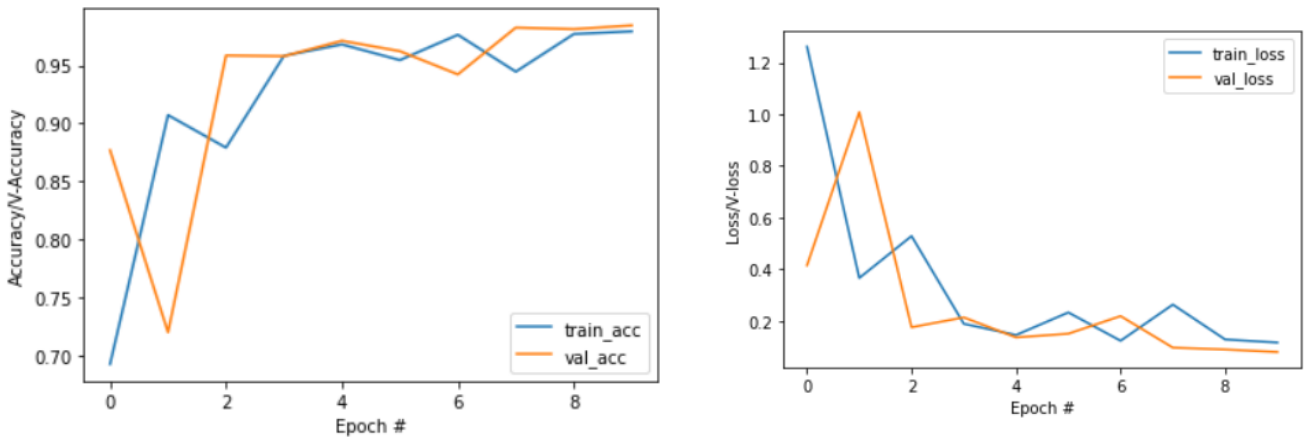
Thirdly, in the notable ShuffleNet architecture research paper, the first building block in each stage is applied with stride 2, and other hyper-parameters within a stage stay the same, and for the next stage the output channels are doubled. They set the number of bottleneck channels to 1/4 of the output channels for each ShuffleNet unit. They added a Batch Normalization layer after each of the convolutions to make the end-to-end training easier. For stage 2, they did not apply the group convolution on the first pointwise layer since the number of inputs is relatively small.

In this current study, the performance of the proposed CNN model was evaluated through the transfer learning approach. In addition, the ShuffleNet architecture was fine-tuned and then trained on the CAPTCHA dataset. The old classifier part of the ShuffleNet architecture was removed and replaced with a new classifier with 32 nodes or classes. As shown in Figure 12, which is a network summary of the fine-tuned ShuffleNet, there was a total parameter of 1,003,256. The trainable and non-trainable parameters were 964,088 and 39,168, respectively. Figure 13a,b shows the accuracies and losses based on the fine-tuned ShuffleNet, respectively. The training time lasted for 57 s and 63 ms/step on the GPU.



```
                                                                    conv2d_336[0][0]
                                                                    conv2d_337[0][0]
                                                                    conv2d_338[0][0]
____________________________________________________________________________________________________
batch_normalization_156 (BatchN (None, 1, 1, 1536)   6144           concatenate_34[0][0]
____________________________________________________________________________________________________
add_12 (Add)                    (None, 1, 1, 1536)   0              re_lu_138[0][0]
                                                                    batch_normalization_156[0][0]
____________________________________________________________________________________________________
re_lu_140 (ReLU)                (None, 1, 1, 1536)   0              add_12[0][0]
____________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 1536)         0              re_lu_140[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                 (None, 32)           49184          global_average_pooling2d[0][0]
====================================================================================================
Total params: 1,003,256
Trainable params: 964,088
Non-trainable params: 39,168
____________________________________________________________________________________________________
```

Figure 12. The summary of the ShuffleNet architecture on the CAPTCHA dataset.

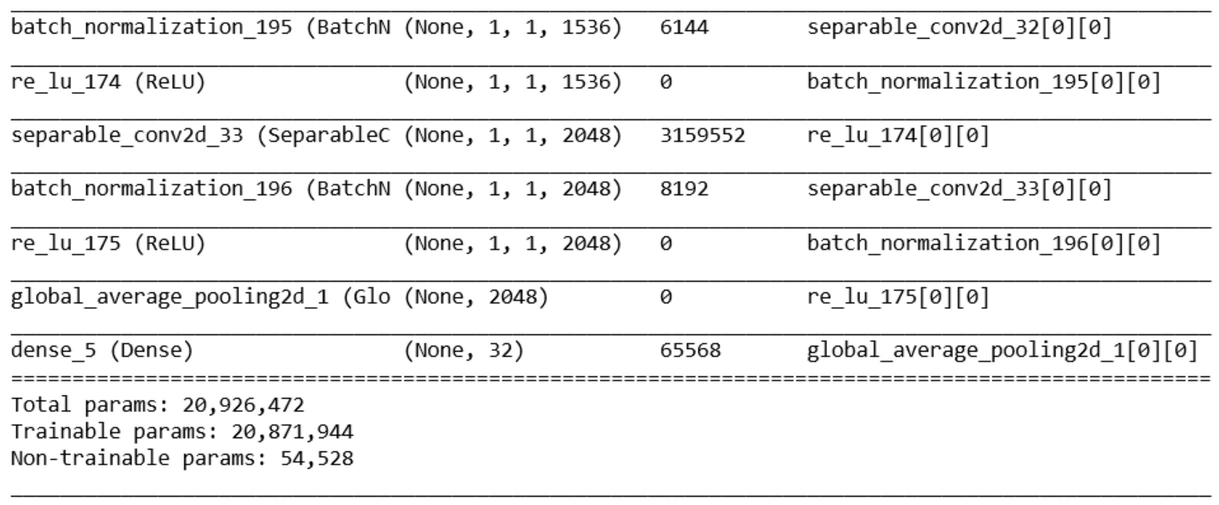


(**a**) (**b**)

Figure 13. (**a**) The training and validation accuracies after training ShuffleNet on the CAPTCHA dataset; (**b**) the training and validation losses after training ShuffleNet on the CAPTCHA dataset.

Lastly, in the Xception architecture paper, all the Convolution and Separable Convolution layers are followed by Batch Normalization, and all the Separable Convolution layers contain a depth multiplier of 1. The Xception network architecture is separated in three flows, which are the entry flow, middle flow with eight repetitions of the same block, and the exit flow.

In this current study, in order to evaluate the performance of the proposed CNN model, the transfer learning approach was adopted. In addition, the Xception architecture was fine-tuned and then trained on the CAPTCHA dataset. The classifier part of the Xception network architecture was removed and replaced with a new one with 32 nodes or classes. As shown in Figure 14, which is the network summary of the fine-tuned Xception, there was a total parameter of 20,926,472. The trainable and non-trainable parameters were 20,871,944 and 54,528, respectively.



```
__________________________________________________________________________________________________
batch_normalization_195 (BatchN (None, 1, 1, 1536)  6144        separable_conv2d_32[0][0]
__________________________________________________________________________________________________
re_lu_174 (ReLU)                (None, 1, 1, 1536)  0           batch_normalization_195[0][0]
__________________________________________________________________________________________________
separable_conv2d_33 (SeparableC (None, 1, 1, 2048)  3159552     re_lu_174[0][0]
__________________________________________________________________________________________________
batch_normalization_196 (BatchN (None, 1, 1, 2048)  8192        separable_conv2d_33[0][0]
__________________________________________________________________________________________________
re_lu_175 (ReLU)                (None, 1, 1, 2048)  0           batch_normalization_196[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 2048)        0           re_lu_175[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 32)          65568       global_average_pooling2d_1[0][0]
==================================================================================================
Total params: 20,926,472
Trainable params: 20,871,944
Non-trainable params: 54,528
__________________________________________________________________________________________________
```

Figure 14. The summary of the Xception architecture on the CAPTCHA dataset.

Figure 15a,b shows the accuracies and losses based on the fine-tuned Xception, respectively. The training time lasted for 28 s and 31 ms/step on the GPU.

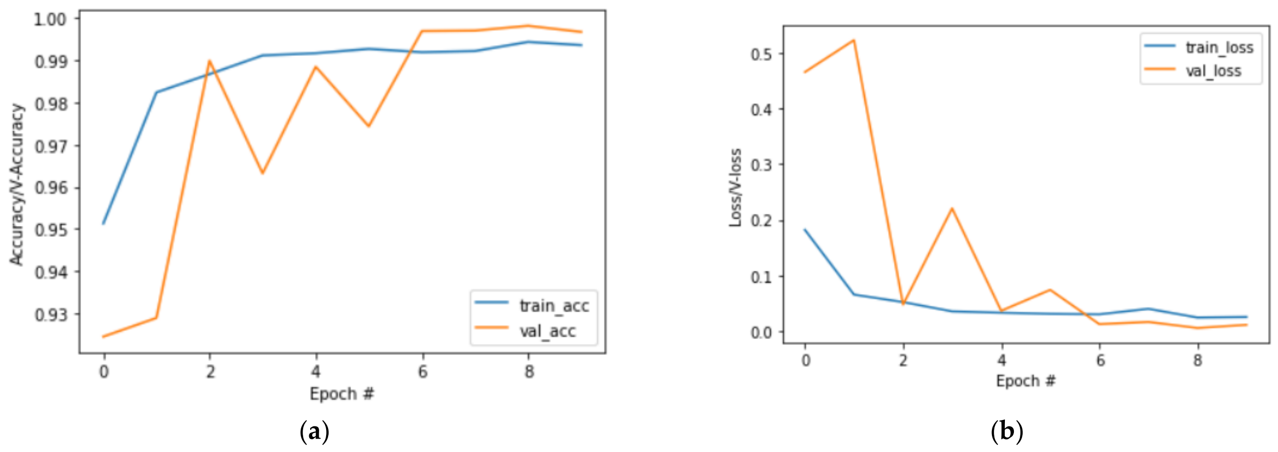


Figure 15. (**a**) The training and validation accuracies after training Xception on the CAPTCHA dataset; (**b**) the training and validation losses after training Xception on the CAPTCHA dataset.

4. Discussion

CAPTCHA breaking is an ethical and essential form of the vulnerability assessment technique to evaluate the strength of security in text based CAPTCHAs before being deployed on web applications. Text-based CAPTCHA breaking algorithms can be grouped into two main divisions, segmentation-based and segmentation-free. The segmentation-based algorithms are categories of segmentation [24] and character recognition [25]. When it comes to CAPTCHA breaking research, it may not always matter which kind of algorithm category it may fall under. However, what matters most is the speed and high accuracy which can be achieved by the proposed algorithm to break the CAPTCHA.

Based on previous works, Simard et al. [26] and Jaderberg et al. [27] used a traditional technique such as image processing to locate a single number or character regions in an image, and then segment them in order to recognize the individual characters. Yan et al. [28] used the segmentation technique to segment Microsoft CAPTCHAs with a 60% rate of recognition [2].

Furthermore, CAPTCHA image segmentation has been implemented using the vertical projection [29–31] approach based on the work of Zhang et al. [32]. In their work, they enhanced the vertical projection to treat the characters with a combination of size features of the characters and their locations with a vertical projection histogram [3]. The connected component algorithm has been used to segment Yahoo and Google CAPTCHA schemes [5]. However, according to Thobhani et al. [3], the vertical projection and connected component algorithms require massive preprocessing procedures, which are computationally expensive and consume a lot of time.

In the work of Yu et al. [33], they adopted a low-cost approach based on open source python libraries to generate CAPTCHA images, and then proposed a peak segmentation algorithm together with the Convolutional Neural Network (CNN) to recognize their generated CAPTCHAs. They defined their peak segmentation as one which relies on the writing order of CAPTCHA from left to right. The whole CAPTCHA is compressed into the x-axis by summing up the values in the y-axis. In their character recognition of the CNN architecture, they constructed a convolutional input layer using the ReLU activation layer of 28×34 in input size. Their CNN model consisted of standard CNN layers, max-pooling layer, 20% dropout layer, a flatten layer, and an output layer with Softmax activation. Their model obtained a 99.32% training accuracy and 92.37% validation accuracy.

Hu et al. [2] proposed a technique based on the Convolutional Neural Network to recognize CAPTCHA and avoided the traditional image processing technology including location and segmentation [2]. Stark et al. [34] also used the segmentation-free algorithm based on the Convolutional Neural Network to recognize CAPTCHAs. Another CAPTCHA recognition study based on the combination of Convolutional Neural Network and attention-based Recurrent Neural Network has been achieved under the segmentation-free algorithm. In this kind of network architecture, the CNN serves as the feature extractor to obtain meaningful information from the CAPTCHA image such as feature vectors, and a variant of RNN, such as the Long- Short Term Memory (LSTM) network, which is used to transform the feature vectors into a text sequence. Even though this kind of model has a high recognition rate, according to Thobhani et al. [3], the architecture of the CNN-RNN model is relatively complicated and could also result in increased memory and storage size.

In the work of Kwon et al. [4], they generated their CAPTCHA images using a two-step style-transfer learning in deep neural networks. Then, they tried to break the generated CAPTCHAs using a fine-tuned VGGNet Convolutional Neural Network based on the transfer learning approach.

In the work of Thobhani et al. [3], they proposed an attached binary image algorithm. In their ABI algorithm, they made a specific number of copies of the input CAPTCHA image, which is equal to the number of characters in the input CAPTCHA image. Then, they attached unique binary images to each copy. Thereafter, they built a CNN architecture to use their ABI algorithm to recognize CAPTCHAs with a white background and CAPTCHAs with a noisy background. Their CNN architecture consisted of 17 standard convolutional layers, five max-pooling layers, one flatten layer, one dropout layer, and one output Softmax layer. After training their model on the CAPTCHA dataset of four characters with a white background, which is from the Weibo website for 120 epochs with a 128 batch size, their model obtained accuracies for training, validating, and testing of 98.45%, 93.26%, and 92.68%, respectively.

This current study is inspired by the above observed assumptions, and is motivated to contribute to CAPTCHA breaking research in order to detect the loopholes in cybersecurity based on text-based CAPTCHAs which are loosely generated. The significance of our research will assist website owners or companies to think twice and generate CAPTCHAs which are hard to break by machines in order to protect the private information of their end-users.

Therefore, in this current work, rather than using the whole text-based CAPTCHA image as the training input, the original CAPTCHA image is split for the individual characters based on the SCE algorithm. As a result, a single character of 20×20 pixels in input shape is used as a training input.

The competitive results of the proposed text-based CAPTCHA breaking Depth-wise Separable CNN model have been compared with notable image recognition CNN architectures such as MobileNets, DenseNet, ShuffleNet, and Xception through transfer learning. This study has made the problem simple, and for that matter, a network with a complex number of parameters may not be necessary.

After training the fine-tuned mobileNets architecture for 10 epochs, the model obtained training and validation accuracies of 98.3% and 99.4%, respectively. The training and validation losses were 0.0657 and 0.0211, respectively. The size of the fine-tuned MobileNets model obtained was 37.76 MB. Based on the fine-tuned DenseNet, after 10 epochs, the model obtained training and validation accuracies of 99.1% and 99.6%, respectively. The training and validation losses were 0.0487 and 0.0122, respectively. The size of the DenseNet model after training on the custom CAPTCHA dataset was 82.43 MB. For the same 10 epochs, the fine-tuned ShuffleNet model obtained training and validation accuracies of 97.9% and 98.4%, respectively. The training and validation losses were 0.1156 and 0.0790, respectively. The size of the ShuffleNet model after training on the custom CAPTCHA dataset was 14.25 MB. After training the fine-tuned Xception for 10 epochs, the model obtained training and validation accuracies of 99.3% and 99.6%, respectively. The training and validation losses were 0.0256 and 0.0115, respectively. The size of the Xception model after training on the custom CAPTCHA dataset was 239.79 MB.

Based on the transfer learning models observed above, all the models performed good on the training and validation sets. However, we also noticed a lot of parameters in the fine-tuned CNN architectures and that may demand unnecessary storage and memory sizes relative to the current problem at hand.

Therefore, this study proposed an efficient and accurate CAPTCHA breaking CNN architecture which is flexible, less complex, and above all, has a high breaking accuracy. Our proposed CNN architecture consisted of six Depth-wise Separable Convolutional layers, seven Batch Normalization layers, seven ReLU activation function layers, four Dropout layers, three Max-pooling layers, one flattened layer, one fully connected layer, and one output Softmax layer. The number of parameters in our proposed model is far less than the fine-tuned CNN architectures, as shown in Table 2. Our proposed CNN model is the first CNN architecture research work to utilize a Depth-wise Separable Convolution to perform text-based CAPTCHA breaking. After 10 epochs, our proposed model achieved training and validation accuracies of 99.5% and 99.8%, respectively. The training and validation losses were 0.0164 and 0.0090, respectively. The size of our proposed model after training on the custom CAPTCHA dataset was 2.36 MB.

As shown in Table 1, we observed that, our proposed model has lesser number of trainable and non-trainable parameters as compared with the fine-tuned CNN models. The training and validation accuracies of our proposed model were higher as compared with the fine-tuned CNN models. It took less time to train our proposed model for 7 s and 7 ms/step for 10 epochs on the GPU. And lastly, after training, the model size of our proposed CNN model was lesser as compared with the fine-tuned CNN models.

Table 1. Our proposed CNN model performance compared with the fine-tuned CNN models based on the training and validation sets.

Model	Trainable Parameters	Non-Trainable	Training Accuracy	Validation Accuracy	Epoch	Time/Step	Model Size
Fine-tuned MobileNets	3,250,144	21,888	98.3	99.4	10	10 s and 11 ms/step	37.76 M
Fine-tuned DenseNet	6,988,448	81,472	99.1	99.6	10	46 s and 51 ms/step	82.43 M
Fine-tuned ShuffleNet	964,088	39,168	97.9	98.4	10	57 s and 63 ms/step	14.25 M
Fine-tuned Xception	20,871,944	54,528	99.3	99.6	10	28 s and 31 ms/step	239.79 M
CNN (Proposed)	191,607	622	99.5	99.8	10	7 s and 7 ms/step	2.36 M

By performing inference on the testing datasets, the results are shown in Table 2. We observed that, all the models obtained accuracies of more than 90% on the testing dataset. However, in terms of the breaking time, our proposed model is simple, flexible, efficient, and used lesser time to break the text-based CAPTCHAs as compared with the fine-tuned CNN models. Examples of text-based CAPTCHA breaking by our proposed model are shown in Figure 16. The limitation of our proposed CAPTCHA breaking model is that it performs better on CAPTCHA images with rotated, overlapping characters, and with a white background. It has not been tested on CAPTCHA images with distortion, strikethrough, and with a complex noisy background.

Table 2. Our proposed CNN model performance compared with the fine-tuned CNN models based on the testing set.

Model	Model Size	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Breaking Time (ms)
Fine-tuned MobileNets	37.76 M	99	99	99	99	15 ms
Fine-tuned DenseNet	82.43 M	100	100	100	100	19 ms
Fine-tuned ShuffleNet	14.25 M	98	98	98	98	21 ms
Fine-tuned Xception	239.79 M	100	100	100	100	20 ms
(Proposed)	2.36 M	100	100	100	100	10 ms

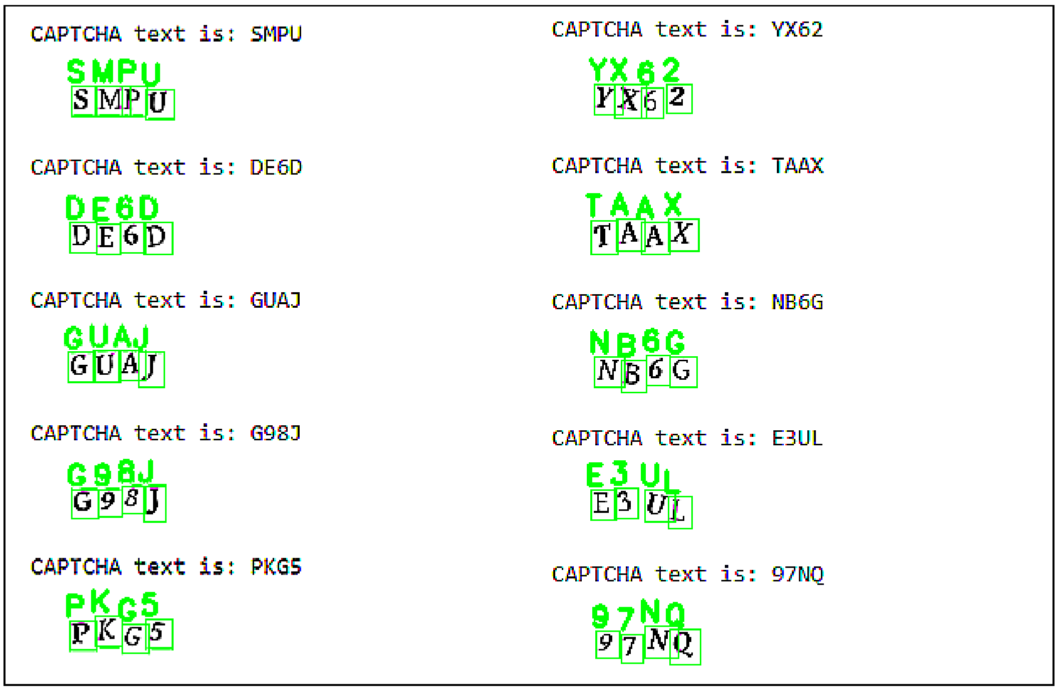


Figure 16. Examples of text-based CAPTCHAs broken by our proposed CNN model.

5. Conclusions

Cybersecurity practitioners sometimes generate text-based CAPTCHAs as a form of standard security mechanism in websites, through which end-users are connected to the internet to perform their daily life activities. In this current study, we have assessed a vulnerability in cybersecurity through CAPTCHA breaking. We achieved this objective by developing a simple, flexible, and computation-efficient Depth-wise Convolutional Neural Network model to accelerate the CAPTCHA breaking time as compared with fine-tuned models with a high accuracy. This current research work will assist cybersecurity practitioners develop and generate robust text-based CAPTCHAs with their security mechanisms, which are capable of resisting malicious attacks.

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

Funding: This reseach was partially supported by NSFC (No. 61871074).

Data Availability Statement: The data and codes presented in this current study are publicly available at https://github.com/sm-multimedia/Text-based-CAPTCHA-Breaking- (accessed on 13 February 2021).

Acknowledgments: First of all, we thank God for His strength and wisdom. We express our sincere thanks to the graduate school of University of Electronic Science and Technology of China for providing us the experimental resources. And lastly, we thank all the Lab mates in 905Lab of School of Automation Engineering for their encouragements.

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

References

1. Kolupaev, A.; Ogijenko, J. CAPTCHAs: Humans vs. Bots. *IEEE Secur. Priv.* **2008**, *6*, 68–70. [CrossRef]
2. 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]
3. 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]
4. 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]
5. Chellapilla, K.; Simard, P.Y. Using machine learning to break visual human interaction proofs (HIPs). In Proceedings of the 17th International Conference on Neural Information Processing Systems, NIPS 2004, Vancouver, BC, Canada, 13–18 December 2004; pp. 265–272.
6. 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]
7. Shao, X.; Zhang, X.; Tang, G.; Bao, B. Scene Recognition Based on Recurrent Memorized Attention Network. *Electronics* **2020**, *9*, 2038. [CrossRef]
8. 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. [CrossRef]
9. 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]
10. 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.
11. 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]
12. 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.
13. Yang, Y.; Deng, H. GC-YOLOv3: You Only Look Once with Global Context Block. *Electronics* **2020**, *9*, 1235. [CrossRef]
14. 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] [PubMed]
15. 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]
16. 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.
17. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
18. Andrew, G.H.; Menglong, L.; Bo, C.; Dmitry, K.; Weijun, W.; Tobias, W.; Marco, A.; Hartwig, A. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
19. Gao, H.; Zhuang, L.; Laurens, V.D.M.; Kilian, Q.W. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:1608.06993, 2018.
20. Xiangyu, Z.; Xinyu, Z.; Mengxiao, L.; Jian, S. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *arXiv* **2017**, arXiv:1707.01083.
21. Francois, C. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv* **2017**, arXiv:1610.02357.
22. 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.
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
24. Abdussalam, A.; Sun, S.; Fu, M.; Sun, H.; Khan, I. License Plate Segmentation Method Using Deep Learning Techniques. In Proceedings of the Signal and Information Processing, Networking and Computers, Guiyang, China, 13–16 August 2019; pp. 58–65. [CrossRef]
25. Abdussalam, A.; Sun, S.; Fu, M.; Ullah, Y.; Ali, S. Robust Model for Chinese License Plate Character Recognition Using Deep Learning Techniques. In Proceedings of the CSPS 2018: Communications, Signal Processing, and Systems, Dalian, China, 14–16 July 2018; Volume 517, pp. 121–127. [CrossRef]
26. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition, Edinburgh, UK, 6 August 2003; pp. 958–963.

27. Jaderberg, M.; Vedaldi, A.; Zisserman, A. Deep features for text spotting. In Proceedings of the European conference on computer vision, Zurich, Switzerland, 6–12 September 2014; pp. 512–528. [CrossRef]
28. Yan, J.; Ahmad, A.S.E. A low-cost attack on a Microsoft captcha. In Proceedings of the CCS '08: Proceedings of the 15th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 27–31 October 2008; pp. 543–554. [CrossRef]
29. Anagnostopoulos, C.E.; Anagnostopoulos, I.E.; Psoroulas, I.D.; Loumos, V.; Kayafas, E. License Plate Recognition from Still Images and Video Sequences: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2008**, *9*, 377–391. [CrossRef]
30. Chen, C.J.; Wang, Y.W.; Fang, W.P. A Study on Captcha Recognition. In Proceedings of the 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kitakyushu, Japan, 27–29 August 2014; pp. 395–398.
31. Wang, Q. License plate recognition via convolutional neural networks. In Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 24–26 November 2017; pp. 926–929.
32. Zhang, L.; Xie, Y.; Luan, X.; He, J. Captcha automatic segmentation and recognition based on improved vertical projection. In Proceedings of the 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou, China, 6–8 May 2017; pp. 1167–1172.
33. Yu, N.; Darling, K. A Low-Cost Approach to Crack Python CAPTCHAs Using AI-Based Chosen-Plaintext Attack. *Appl. Sci.* **2019**, *9*, 2010. [CrossRef]
34. Stark, F.; Hazırbaş, C.; Triebel, R.; Cremers, D. CAPTCHA Recognition with Active Deep Learning. In Proceedings of the German Conference on Pattern Recognition Workshop, Aachen, Germany, 7–10 October 2015.