Breaking CAPTCHAs Using Deep Learning To Expose Vulnerability

A PROJECT REPORT

Submitted by

GAYATHRI S (715517104022)

KAVYA P (715517104034)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH, COIMBATORE 641 062

ANNA UNIVERSITY: CHENNAI - 600 025

AUGUST 2021

ANNA UNIVERSITY: CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**Breaking CAPTCHAs Using Deep Learning To Expose Vulnerability**" is the bonafide work of "**GAYATHRI S (715517104022), KAVYA P (715517104034)**" who carried out the project work under my supervision.

---------- ----------

SIGNATURE **SIGNATURE**

Dr. R. Manimegalai Ms. G. Niranjani

HEAD OF THE DEPARTMENT **SUPERVISOR**

Professor and Head Assistant Professor (Senior Grade)

Computer Science and Engineering Computer Science and Engineering

PSG Institute of Technology and Applied Research, PSG Institute of Technology and Applied Research,

Coimbatore – 641 062 Coimbatore – 641 062

Submitted for the project viva-voce Examination held on _____

---------- ----------

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We are very grateful to our Managing Trustee **Shri. L. Gopalakrishnan,** for providing us with an environment to complete our project successfully.

We would like to extend our heartfelt gratitude to our Principal **Dr. G. Chandramohan, B.E., M.Tech., Ph.D.,** in extending support to carry out our study.

We are very grateful to our Secretary **Dr. P. V. Mohanram, B.E., M.Tech., Ph.D.,** for the motivation and support to complete our project.

We are greatly indebted to our Head of the Department **Dr. R. Manimegalai**, **M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering, for providing us with all the required permission and support throughout this project.

We would like to thank our supervisor **Ms. G. Niranjani, M.Tech.,** Assistant Professor (Senior Grade), for the constant motivation which has helped us to complete the project as we determined to do, in the stipulated time.

We would like to thank our project coordinator **Mr. S. Thivaharan**, **M.Tech.,** Assistant Professor (Selection Grade), for the consistent support throughout the project.

Finally, we thank all the faculty of the Department of Computer Science and Engineering, our friends and family members who provided constant support and motivation to complete the project successfully.

Gayathri S

Kavya P

PLAGIARISM REPORT

VeriGuide - Originality Report

Individual Report

Background Information

File Name:	Breaking_CAPTCHAs_using_Deep_Learning_to_expose_vulnerability.docx
Report Generated On:	31/07/2021, 04:08:07 PM

Similarity Statistics Overview

Similar Sentence(s) Found By VeriGuide:	37 out of 552 sentences = 6.7%
Similar Sentence(s) Filtered by User:	37 out of 552 sentences = 6.70%
Sentence(s) Selected By User To Export:	0

ABSTRACT

Internet is vulnerable to malicious scripts or bots. Therefore, it becomes important task to identify a human from a bot. To overcome this, most of the services use CAPTCHAs. CAPTCHA is a short term for Completely Automated Public Turing test to tell Computers and Humans Apart. A CAPTCHA has series of alphabets or numbers lined one after another in a sequence. This image is distorted with random lines, blocks, grids, rotations and various such types of noise. CAPTCHAs have become the important methods to distinguish humans from bots or malicious programs. Most of the CAPTCHAs deployed on the websites are text-based. They are based on the assumption that bots cannot read text from an image. However, in recent times, people have been able to come up with programs that are able to recognize CAPTCHAs with significant accuracy. This project helps to break security using Convolutional Neural Networks (CNNs) which can independently recognize text-based CAPTCHAs. The procedure can be divided into 3 tasks: pre-processing, segmentation, and classification. The pre-processing of the image is performed to remove all the background noise of the image. The noise in the CAPTCHA are unwanted pixels in the background. The segmentation is performed by scanning the image for on pixels. The classification is done using the CNN. The results have shown that the model has a good recognition effect on CAPTCHA with background noise. Thus, our model shows significant improvement to the previously proposed models and has an accuracy of over 91%.

Keywords: CAPTCHA; security; deep learning; image; confidentiality; Convolutional Neural Network;

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
RNN	Recurrent Neural Network

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The term CAPTCHA is an acronym for Completely Automated Public Turing test to tell Computers and Humans Apart. This is a computer program designed to distinguish between a human user and a machine or a bot, typically as a security measure to prevent spam and data misuse. CAPTCHA requires a user to recognize the letters and numbers in an image present with background noises. This test is done in the hope that humans would easily be able to distinguish the characters in the distorted image, while an automated program or bot will not be able to distinguish them.

1.2 HISTORY OF CAPTCHAs

The concept of CAPTCHA was introduced as early as 1997, when the internet search company AltaVista was trying to block automatic URL submissions to the platform that were skewing their search engine algorithms. To tackle this problem, AltaVista's chief scientist, Andrei Broder, came up with an algorithm to randomly generate images of text that could easily be identified by humans, but not by bots. Later, in 2003, Luis von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford perfected this technology and called it CAPTCHA.

As of recently, CAPTCHA has started to serve a much bigger role than just preventing bot frauds. For example, Google used Captcha and one of its variants reCAPTCHA, when they digitized the archives of New York Times and some books in Google Books. This is typically done by asking the user to correctly enter the characters of more than one CAPTCHA. Only one of the CAPTCHAs is actually labelled and used to validate the user is human. The rest of the

CAPTCHAs are labelled by the user. Currently, Google uses Image-based CAPTCHAs.

1.3 PROBLEMS FACED IN CAPTCHA RECOGNITION

- Use of complex methods made it difficult to implement.
- Methods used for CAPTCHA generation were very slow.
- The accuracy of the CAPTCHA Recognition was relatively lower.

1.4 TEXT-BASED CAPTCHAs

CAPTCHA can be classified into three types: text, image and sound according to its different carrier and contents. Text CAPTCHA is one of the types of CAPTCHA, which is researched most and used all over the world. Some large networks, including Microsoft, Yahoo, Google choose CAPTCHA as the best safe-guarding system. Text-based CAPTCHAs as shown in Figure 1.1 are the simplest to implement where a sequence of digits and letters are presented to the user with some modifications added to the characters such as rotating, scattering, noise or making characters in 3D form, these modifications are added to make any robot programs unable to read the presented characters.

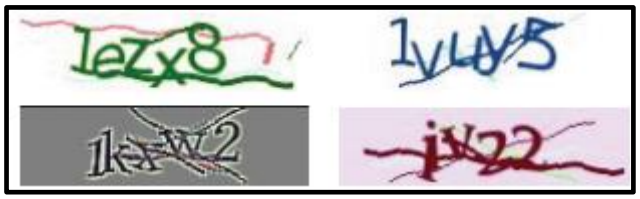


Figure 1.1 Text-based CAPTCHAs

1.5 PROBLEM STATEMENT

This project is about Breaking CAPTCHAs using Convolutional Neural Networks (CNNs) and to explore the vulnerability of CAPTCHAs being automatically detected using bots.

1.6 PROJECT OVERVIEW

This project is about Breaking CAPTCHAs using Deep Learning to expose vulnerability. With the recent success of Conventional Neural Networks (CNNs) in computer vision tasks, breaking CAPTCHAs in a few minutes is a relatively easy task. Therefore, CAPTCHAs need to be much more evolved than they have been in the past. We will explore the vulnerability of basic CAPTCHAs being automatically detected using bots with a deep learning framework.

1.7 ORGANISATION OF THE THESIS

Chapter 2 explains the literature review of the wide variety of papers in the CAPTCHA Recognition domain. Chapter 3 describes the system description of the project and gives an overview of the project. Chapter 4 explains about the system design and the various sources of data collection. Chapter 5 elaborates on the system implementation. Chapter 6 explains about the analysis of the result. Chapter 7 elaborates on the future work of the project.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEMS

The number of papers dealing with CAPTCHA Recognition in literature is growing exponentially. Several researchers have played a significant role in the development of CAPTCHA Recognition.

P. Wang et al. [10] in their paper titled "Simple and Easy: Transfer Learning-Based Attacks to Text CAPTCHA" have proposed a simple, generic and efficient transfer learning-based method that can greatly decrease the complexity in breaking text CAPTCHAs. They have used randomly generated synthetic images to pre-train the model and also used a few real labelled CAPTCHAs to fine-tune the model. Their recognition engine is a combination of a Residual Network (ResNet), a Recurrent Neural Network (RNN), and an attention mechanism.

T. Zhang et al. [15] in their paper titled "Verification CAPTCHA Based on Deep Learning" have analysed the characteristics of simple segmentation CAPTCHA and indivisible verification code, and have put forward the CAPTCHA recognition algorithm based on CNN. They have designed the training network structure for different CAPTCHA and have made full use of the self-learning characteristics of CNN and have simplified the traditional CAPTCHA recognition.

Y. Zi et al. [18] in their paper titled "An End-to-End Attack on Text CAPTCHAs" have proposed an end-to-end method that is simple, generic and effective in breaking text CAPTCHAs. They have utilized an attention-based model that consists of a CNN and a Recurrent Neural Network (RNN) to break

CAPTCHAs. They have also analysed the security of a series of uncommon CAPTCHAs.

Y. S. Aljarbou [19] in his paper titled, “Improving of Current CAPTCHA Systems” has presented the problems with CAPTCHAs and provided a hypothesis about them. He has also discussed alternative solutions to solve these problems. He has described about the current existing CAPTCHA systems and why they are important and has also compared the different types of CAPTCHAs. He has also provided a brief description about new CAPTCHA idea that may solve all the mentioned problems. The new CAPTCHA system is based on Face and Heat scanning techniques. The face scanning technique is needed to make sure that the shape is for human face and the heat scanning technique is needed to make sure that the face shape belongs to a living human and not for a static picture or image.

Y. Zhang et al. [20] in their paper titled “A Survey of Research on CAPTCHA Designing and Breaking Techniques” have discussed the CAPTCHAs usability and robustness and have analysed its weaknesses and strengths. They have also summarized technical progress in attacking text-based and image-based CAPTCHAs. They have also compared the CAPTCHA categories as well as their attack methods.

2.2 DRAWBACKS IN THE EXISTING SYSTEMS

- The accuracy of the previously proposed models was relatively lower.
- The methods proposed were complicated to implement.
- The models proposed were not applicable for all types of text-based CAPTCHAs.
- The methods used for CAPTCHA generation were slower.

2.3 PROPOSED SYSTEM

This project helps to break security using CNNs which can independently recognize text-based CAPTCHAs. With the recent success of CNNs in computer vision tasks, breaking basic CAPTCHAs in a few minutes is a relatively easy task. Therefore, CAPTCHAs need to be much more evolved than they have in the past. We will explore the vulnerability of basic CAPTCHAs being automatically detected using bots with a deep learning framework.

2.4 BENEFITS OF THE PROPOSED SYSTEM

- CAPTCHA recognition studies can find security breaches in CAPTCHAs.
- To prevent automated processes, such as the actions of a malicious botnet.
- To analyze stability and shortcomings of CAPTCHA schemes.
- Open new gateways for providing security against automated scripts.
- Evolution of face-based CAPTCHAs.
- It can also promote the technologies of license plate recognition and handwriting recognition.

CHAPTER 3

SYSTEM DESCRIPTION

3.1 PROBLEM STATEMENT

The main idea is to break the CAPTCHAs using Convolutional Neural Networks and to explore the vulnerability of CAPTCHAs being automatically detected by bots.

3.2 PROJECT DESCRIPTION

This project helps to break security using Convolutional Neural Networks (CNNs) which can independently recognize text-based CAPTCHAs. With the recent success of CNNs in computer vision tasks, breaking basic CAPTCHAs in a few minutes is a relatively easy task. Therefore, CAPTCHAs need to be much more evolved than they have in the past. We will explore the vulnerability of basic CAPTCHAs being automatically detected using bots with a deep learning framework.

3.3 PROJECT GOALS

The major goals of the project are:

- Generating CAPTCHAs using Claptcha tool.
- Building a Convolutional Neural Network with two pairs of convolution and pooling layers before the dense layer.
- Pre-processing the CAPTCHA images.
- Training and measuring the accuracy of the CAPTCHA Recognition.

CHAPTER 4

SYSTEM DESIGN

4.1 COLLECTION OF DATA

CAPTCHAs can be generated using the Claptcha tool in Python. We use CAPTCHA images consisting of four characters consisting of characters and numbers as the input dataset. Each character can be any one of 26 letters and 10 digits. Along with the text, the Claptcha tool requires the font in which to print the text as input. Figure 4.1 shown below shows the CAPTCHAs generated using Claptcha tool.

Figure 4.1 CAPTCHAs generated using Claptcha tool

4.2 CNN ARCHITECTURE

A Convolutional Neural Network (CNN) are deep neural networks, mostly applied to image domains. Convolutional Neural Networks are state of the art models for Image Classification, Segmentation, Object Detection and many other image processing tasks. CNNs can recognize and classify particular features from images and are widely used for analysing images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

The term Convolution denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other.

The convolution layer uses filters that perform convolution operations as it is scanning the input with respect to its dimensions. Its hyperparameters include the filter size and stride. The resulting output is called feature map or activation map.

Figure 4.2 shown in page 10 depicts the CNN Architecture with the different Convolution layers such as the Convolutional layer, Pooling layer and the Dense layer.

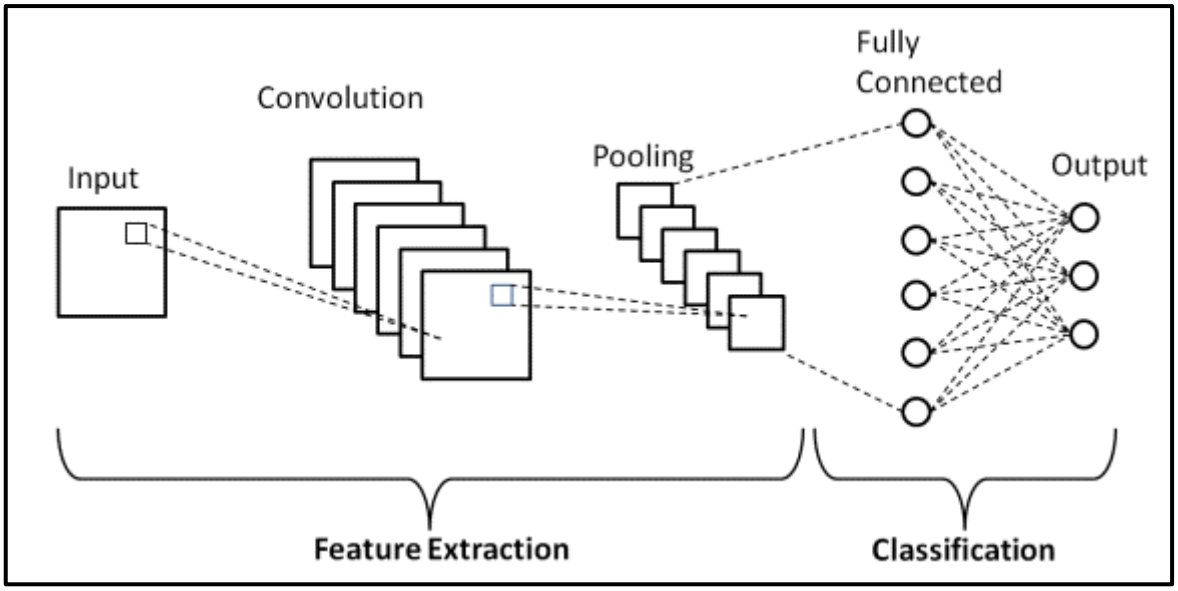


Figure 4.2 Convolution Neural Network

4.2.1 Convolution Layers

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

4.2.1.1 Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter.

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

4.2.1.2 Pooling Layer

A Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon the method used, there are several types of Pooling operations.

The pooling layer is a down sampling operation, applied after a convolution layer, which does some spatial invariance. Max and average pooling are special kinds of pooling where the maximum and average value is taken, respectively.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the Fully Connected Layer.

4.2.1.3 Fully Connected Layer

The Fully Connected layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers is flattened and fed to the Fully Connected layer. The flattened vector then undergoes few more Fully Connected layers where the mathematical function operations usually take place. In this stage, the classification process begins to take place.

4.2.1.4 Dropout

When all the features are connected to the Fully Connected layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model.

4.2.1.5 Activation functions

One of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. It decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU (Rectified Linear Unit), Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred and for a multi-class classification, generally softmax is used.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 SYSTEM FLOW

The CAPTCHAs are generated using the Claptcha tool which acts as the input dataset to the CNN model. CNN architecture is used to identify the characters in the CAPTCHA. The CNN model has two pairs of convolution and pooling layers before the dense layer. Instead of feeding the whole CAPTCHA to the network, we are going to break the CAPTCHA into four characters and feed them individually to the model.

The final output layer of the CNN will predict one of the 36 classes among the 26 letters and 10 digits. Raw pixels of the images don’t work well with the CNN architecture. So, we have to normalize the images for the CNN to converge faster. Our scope lies in the grayscale images of the CAPTCHA, which means that we would be dealing with only one colour channel.

Loading all the data in memory before the start of training can lead to data storage issues. So, we will read the CAPTCHAs and build batches dynamically during training time. This leads to the optimal use of resources. The data generator will store the CAPTCHA file location during initialization and dynamically build batches in each epoch.

The order of the files is randomly shuffled after each so that the CAPTCHA images are not traversed in the same order in each epoch. We will measure accuracy from the overall CAPTCHA perspective and not on the character level of the CAPTCHA. When all four characters of the CAPTCHA target matches the prediction, we can say that the CAPTCHA is correctly identified by the CNN. Figure 5.1 shows the block diagram of the system.

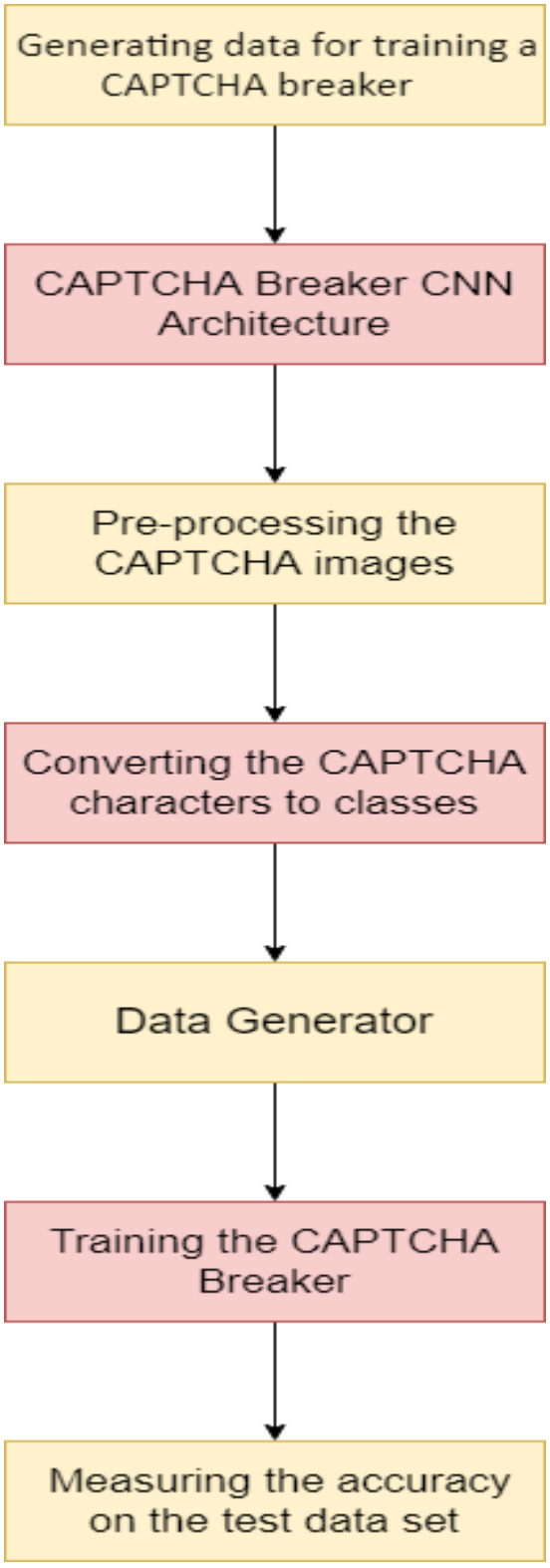


Figure 5.1 Block diagram depicting the system flow

5.2 SETTING UP ENVIRONMENT

Python is downloaded and installed. The necessary libraries like Keras, TensorFlow, OpenCV are installed. The CAPTCHAs are generated using the Claptcha tool.

5.2.1 Python

Python is a general-purpose programming language with the emphasis on indentation. It is used in large projects and the application of Python is enormous. It is used in applications like Machine Learning, Deep Learning, embedded systems, automation tasks etc. It is an interpreted high-level general-purpose programming language. It's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

It is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming. It is often described as a batteries included language due to its comprehensive standard library.

It is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. It's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are

available in source or binary form without charge for all major platforms, and can be freely distributed.

There is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy since a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

5.2.2 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use

of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

5.2.3 Tensorflow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming.

TensorFlow is an open-source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

It is an open-source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

5.2.4 OpenCV

OpenCV (Open Source Computer Vision) Library is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and then by Itseez.

The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface.

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. It can be integrated with various other libraries, such as NumPy which is a highly optimized library for numerical operations.

5.3 SYSTEM IMPLEMENTATION

Our project consists of 7 modules as shown in Table 5.1.

S. No	Module Name
1	Generating data for training a CAPTCHA Breaker
2	CAPTCHA Breaker CNN Architecture
3	Pre-processing the CAPTCHA Images
4	Converting the CAPTCHA characters to classes
5	Data generator
6	Training the CAPTCHA Breaker
7	Measuring the accuracy on the test data set

Table 5.1 Different Modules used

5.3.1 Generating data for training a CAPTCHA Breaker

CAPTCHAs are generated using the Claptcha tool for training a CNN model. The CaptchaGenerator.py script is coded to generate CAPTCHAs. CAPTCHA generators use a ttf file to get a font pattern for the CAPTCHAs. We have generated training, validation and the test set CAPTCHAs of size 16000,4000 and 4000 respectively.

python CaptchaGenerator.py --outdir_train 'C:/Users/gayu2/OneDrive/ Desktop/ captcha/captcha_train/' --num_captchas_train 16000 --outdir_val 'C:/Users/gayu2/ OneDrive/Desktop/captcha/captcha_val/' --num_captchas_val 4000 --outdir_test 'C:/Users/gayu2/OneDrive/Desktop/ captcha/captcha_test/' --num_captchas_test 4000 --font "C:/Users/gayu2/OneDrive /Desktop/captcha /DancingScript.ttf" is the command used to generate CAPTCHAs.

It took 11.984 mins to generate 16000 training CAPTCHAs, 4000 validation CAPTCHAs and 4000 test CAPTCHAs. Figure 5.6 shown below depicts the command to generate CAPTCHAs using the Claptcha tool. Figure 5.7 shown below shows the CAPTCHAs generated using the Claptcha tool.

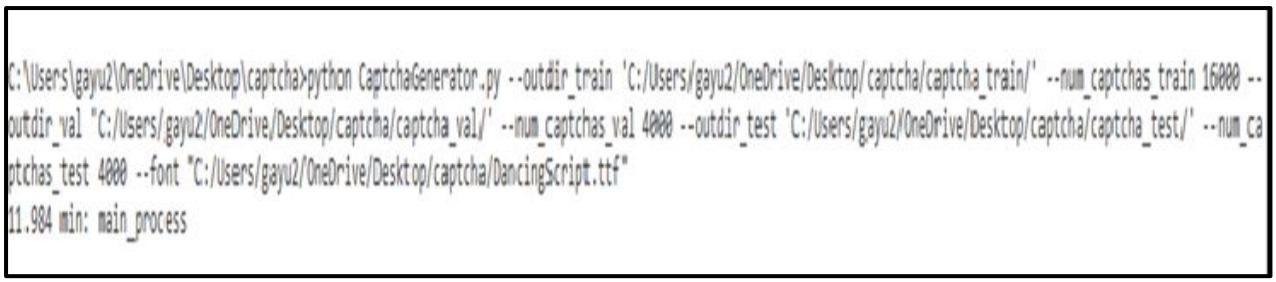


Figure 5.6 Generating data for training a CAPTCHA Breaker

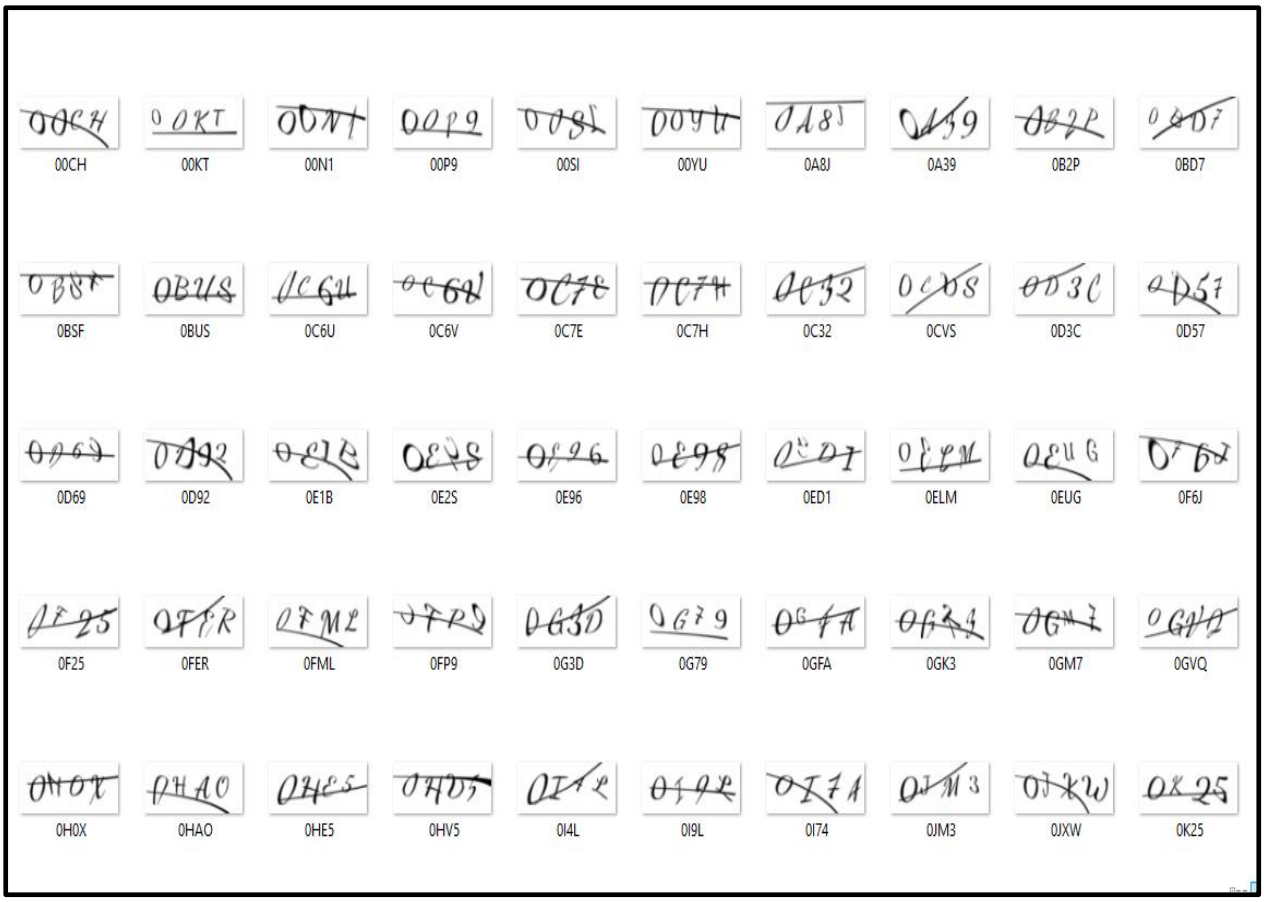


Figure 5.7 CAPTCHAs generated using Claptcha tool

5.3.2 CAPTCHA Breaker CNN Architecture

CNN architecture is used to identify the characters in the CAPTCHA. The CNN would have two pairs of convolution and pooling layers before the dense layer. Instead of feeding the whole CAPTCHA to the network, we are going to break the CAPTCHA into four characters and feed them individually to the model. The final output layer of the CNN will predict one of the 36 classes among the 26 letters and 10 digits.

The CAPTCHA breaker CNN model can be pictorially depicted as shown in the following Figure 5.8.

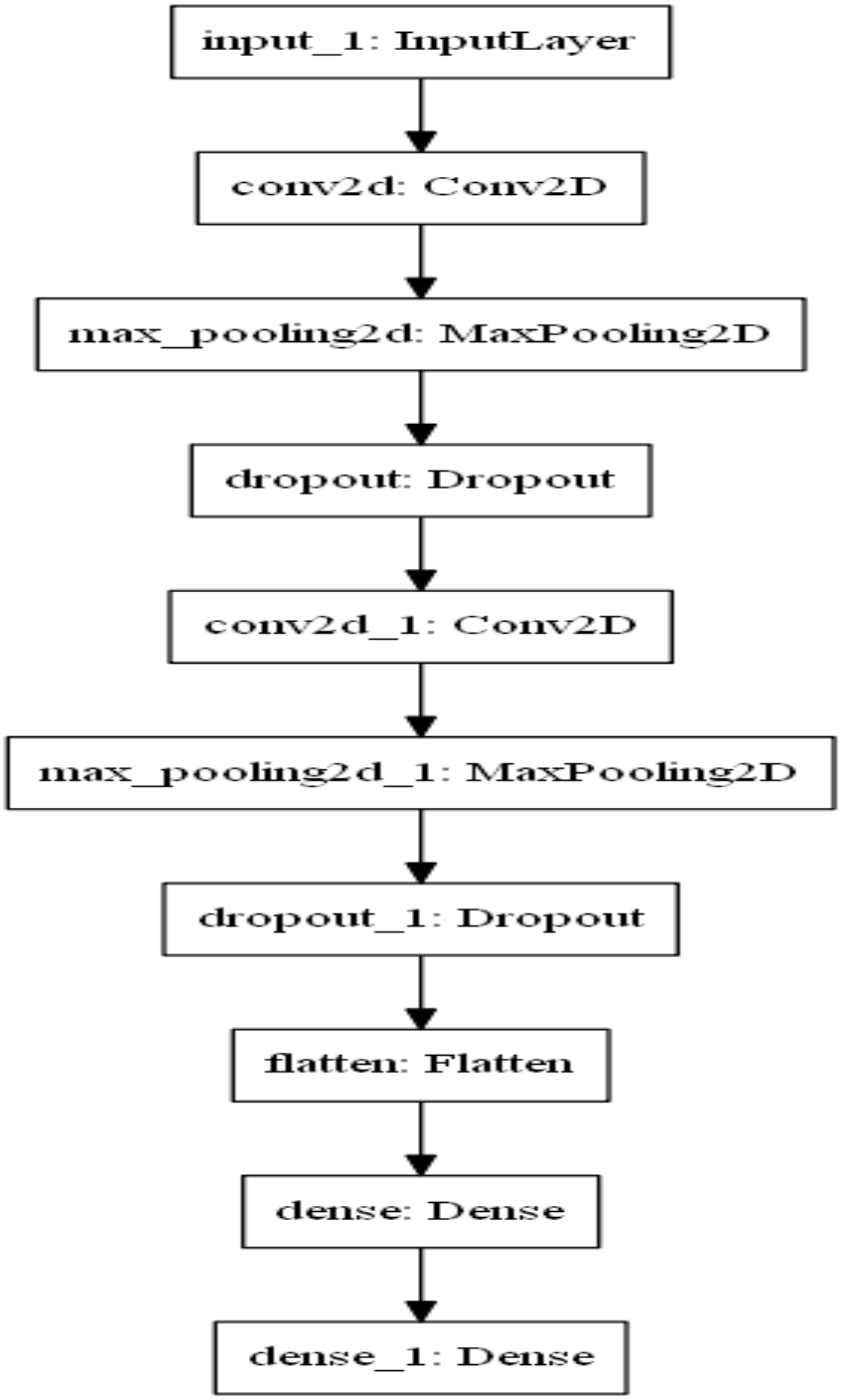


Figure 5.8 CAPTCHA Breaker CNN Architecture

5.3.3 Pre-processing the CAPTCHA images

Raw pixels of the images don't work well with the CNN architecture. So, we have to normalize the images for the CNN to converge faster. We can

normalize the images to lie in the range of [0,1] by dividing the pixel values by 255. Our scope lies in the grayscale images of the CAPTCHA, which means that we would be dealing with only one colour channel.

5.3.4 Converting the CAPTCHA characters to classes

The characters of the CAPTCHA need to be converted to numerical classes for training purposes.

5.3.5 Data Generator

Loading all the data in memory before the start of training can lead to data storage issues. Therefore, we will read the CAPTCHAs and build batches dynamically during training time. This leads to the optimal usage of resources. The generator will store the CAPTCHA file location during initialization and dynamically build batches in each epoch. The order of the files is randomly shuffled after each epoch so that the CAPTCHA images are not traversed in the same order in each epoch.

5.3.6 Training the CAPTCHA Breaker

For the CAPTCHAs in the batch, all four characters are considered for training. The training can be invoked by running the captcha_solver.py script with the train argument as follows. python captcha_solver.py train --dest_train 'C:/Users/gayu2/OneDrive/Desktop/captcha/captcha_train/' --dest_val 'C:/Users/gayu2/OneDrive/Desktop/captcha/captcha_val/' --outdir 'C:/Users/gayu2/OneDrive/Desktop/captcha/model/' --batch_size 16 --lr 1e-3 --epochs 20 --n_classes 36 --shuffle True --dim (40,100,1)

In just 20 epochs of training, the model achieves a validation accuracy of around 96.71% per character level of the CAPTCHA, as seen from the following output log as follows. The training time for 20 epochs with roughly 16000(64000 CAPTCHA characters) is around 1.56 hrs as shown in the figures 5.9 and 5.10.

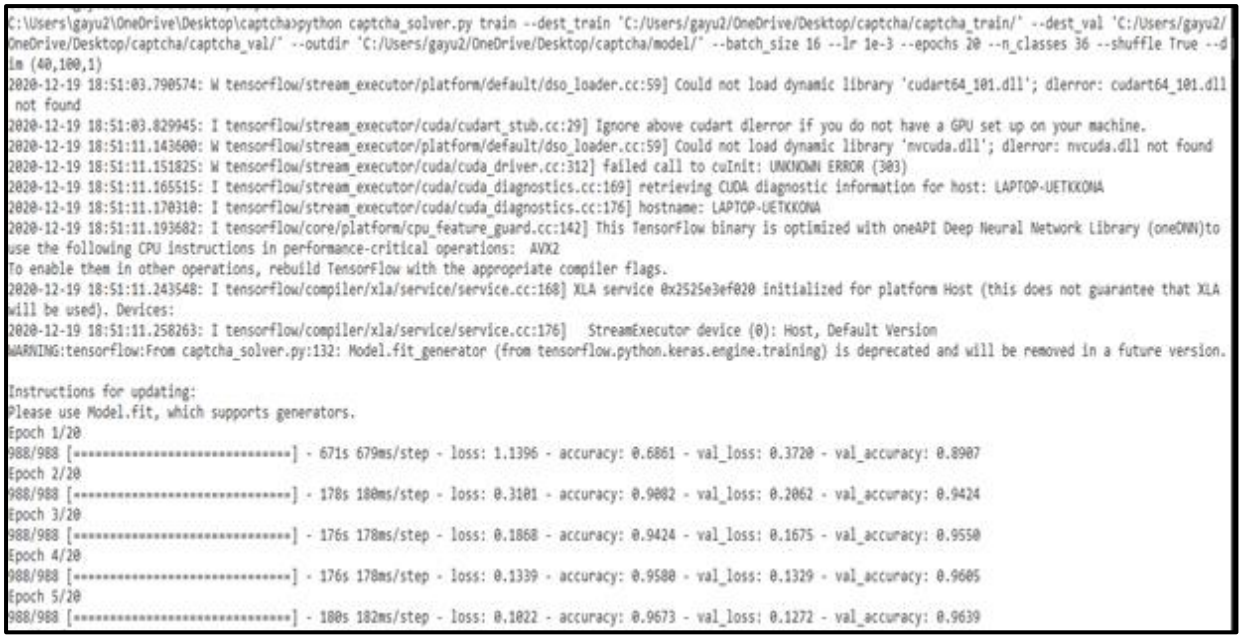


Figure 5.9 Accuracy on the validation data set

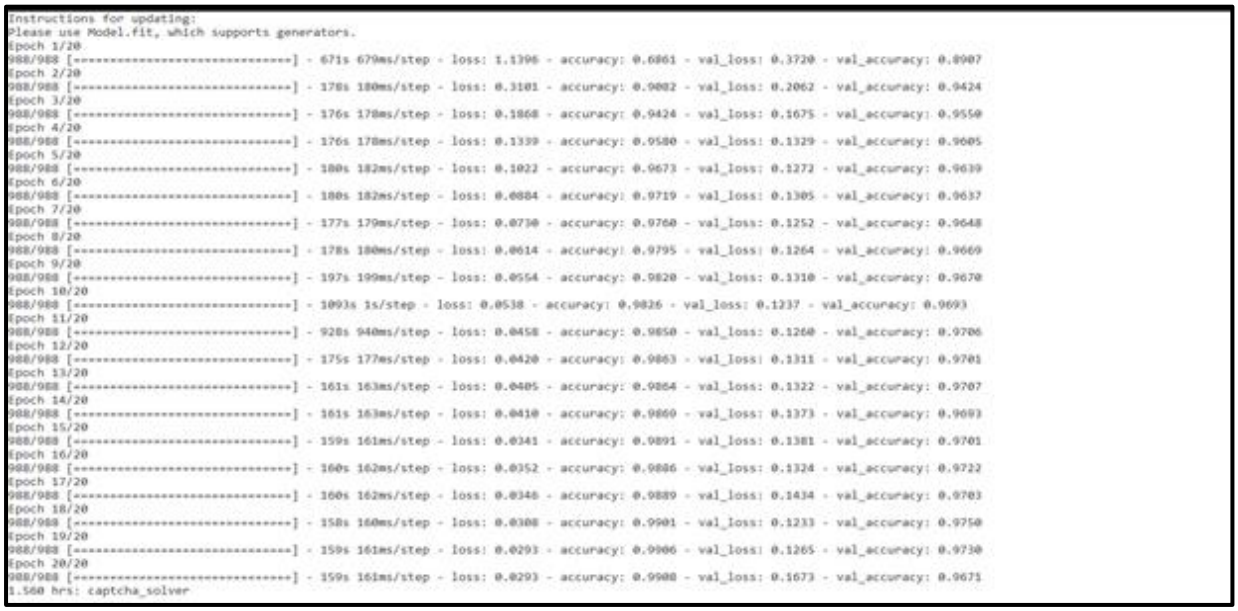


Figure 5.10 Accuracy on the validation data set

5.3.7 Measuring the accuracy on the test data set

We have to measure accuracy from the overall CAPTCHA perspective and not on the character level of the CAPTCHA. When all four characters of the CAPTCHA target matches the prediction, we can say that the CAPTCHA is correctly identified by the CNN.

python captcha_solver.py evaluate --model_path C:/Users/gayu2/ OneDrive/Desktop/captcha/model/captcha_breaker.h5 --eval_dest 'C:/Users/ gayu2/OneDrive/Desktop/captcha/captcha_test/' --outdir C:/Users/gayu2/ OneDrive/Desktop/ captcha/ --fetch_target True is the command used to measure accuracy on the test data set.

The accuracy achieved on the test dataset of 4000 CAPTCHAs is around 91%. Evaluation file is written at C:/Users/gayu2/OneDrive/Desktop/captcha/ evaluation.csv. It took around 19.930 minutes for testing 4000 CAPTCHAs as shown in Figures 5.11 and 5.12.

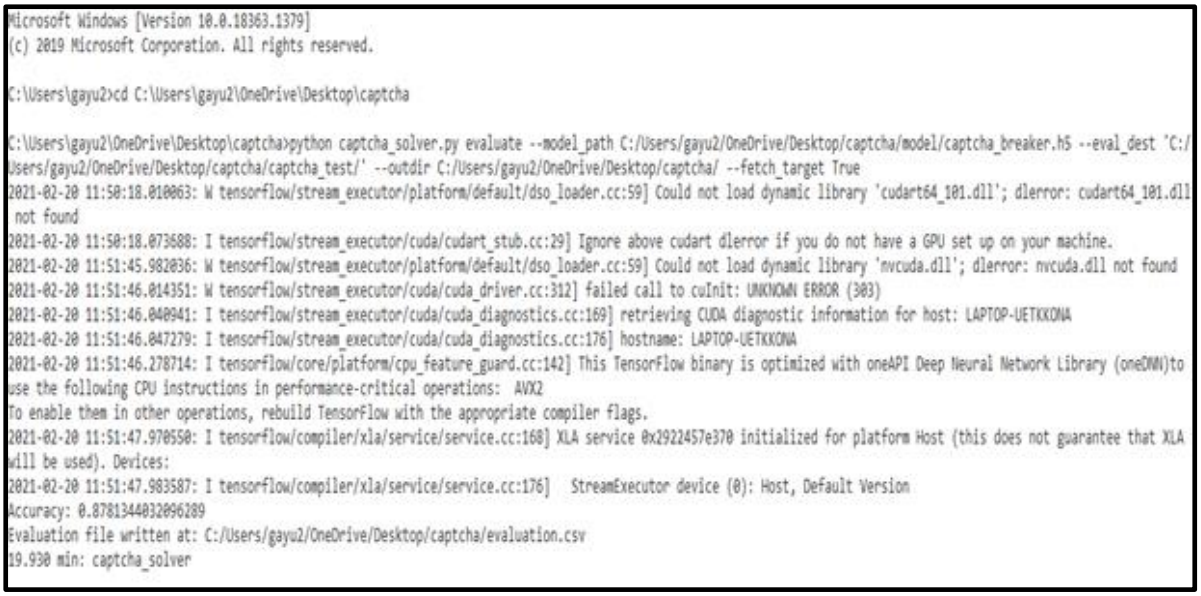
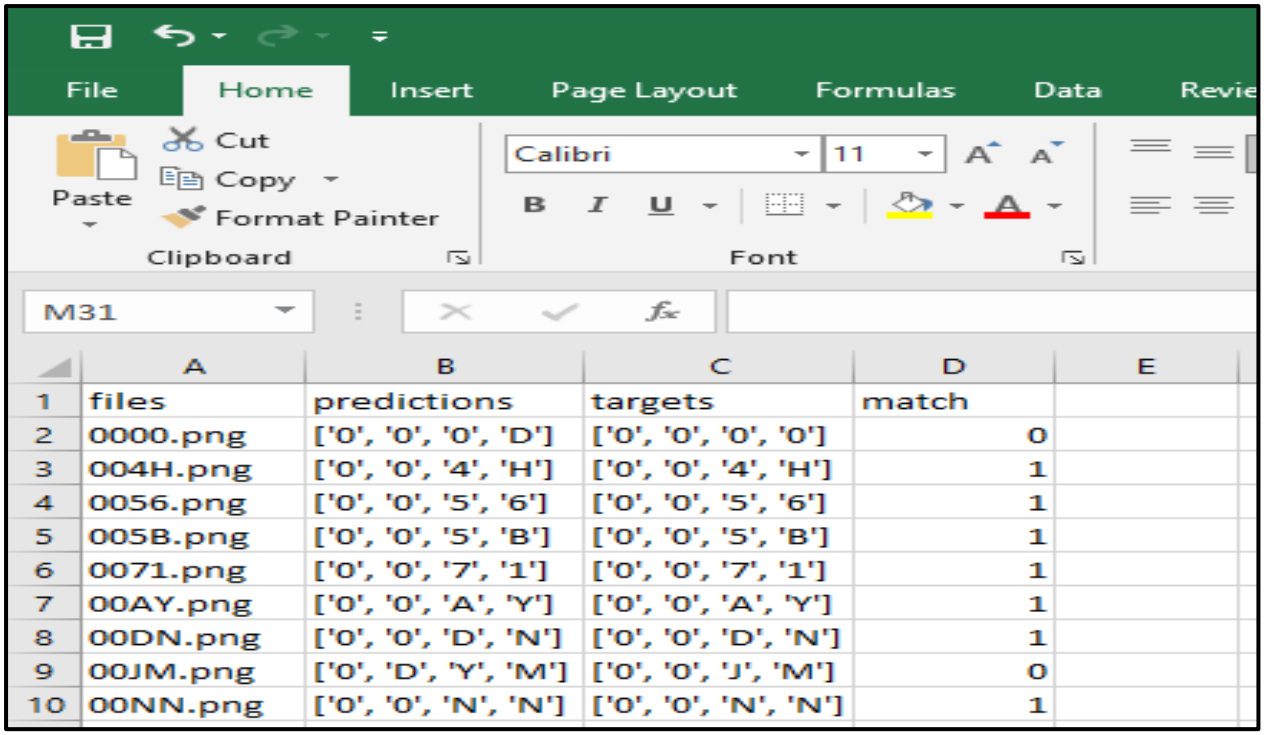


Figure 5.11 Accuracy on the test data set



	A	B	C	D	E
1	files	predictions	targets	match	
2	0000.png	['0', '0', '0', 'D']	['0', '0', '0', '0']	0	
3	004H.png	['0', '0', '4', 'H']	['0', '0', '4', 'H']	1	
4	0056.png	['0', '0', '5', '6']	['0', '0', '5', '6']	1	
5	005B.png	['0', '0', '5', 'B']	['0', '0', '5', 'B']	1	
6	0071.png	['0', '0', '7', '1']	['0', '0', '7', '1']	1	
7	00AY.png	['0', '0', 'A', 'Y']	['0', '0', 'A', 'Y']	1	
8	00DN.png	['0', '0', 'D', 'N']	['0', '0', 'D', 'N']	1	
9	00JM.png	['0', 'D', 'Y', 'M']	['0', '0', 'J', 'M']	0	
10	00NN.png	['0', '0', 'N', 'N']	['0', '0', 'N', 'N']	1	

Figure 5.12 evaluation.csv file

CHAPTER 6

RESULT AND ANALYSIS

6.1 TESTING THE PROJECT

System testing presents an interesting anomaly for the software engineer. The engineer creates a series of test cases that are intended to demolish the software that has been built. Testing requires that the developer discards the preconceived notions of the correctness of the model just developed and overcome a conflict of interest that occurs when errors are uncovered.

There are several rules that can serve well as testing objectives. Testing is a process of executing a program with the intent of finding an error. A good testcase is one that has a high probability of finding an as-yet undiscovered error. A successful test is one that uncovers an as-yet undiscovered error.

Initially the environment checking is done to see if all the necessary requirements are installed or not. Installation of Python and the necessary libraries are confirmed. The model is then tested. The results are obtained and the accuracy is checked. The model is able to recognize the CAPTCHAs and hence the model is tested successfully.

Figure 6.1 depicts the accuracy of the CAPTCHA Breaker and Figure 6.2 depicts the evaluation.csv file which shows the file name, predicted CAPTCHA value, target CAPTCHA value and also denotes the match.

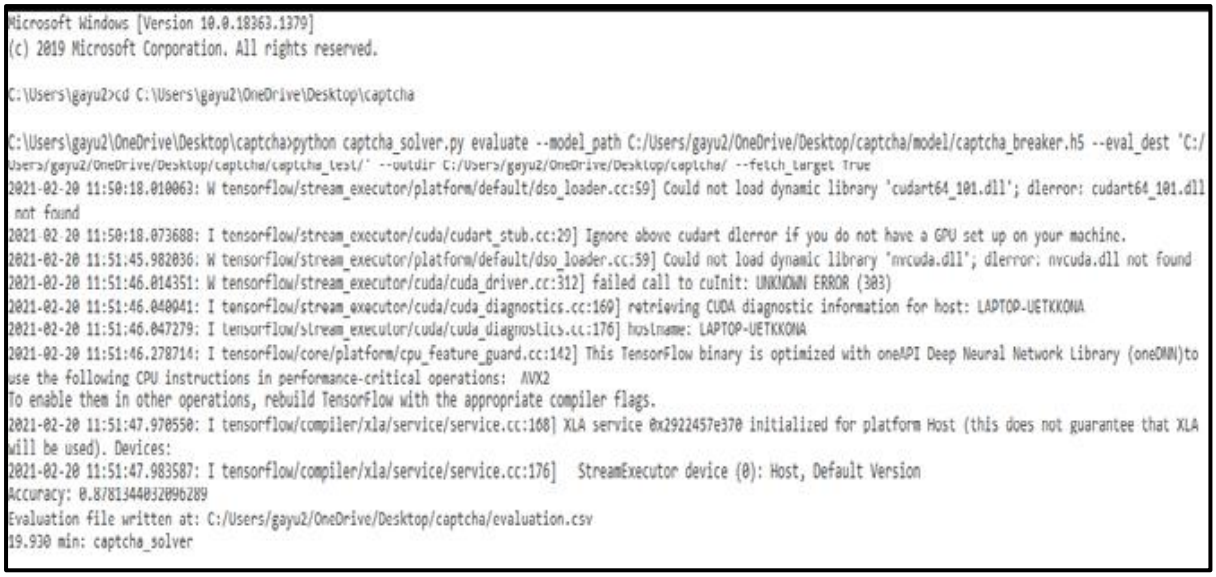
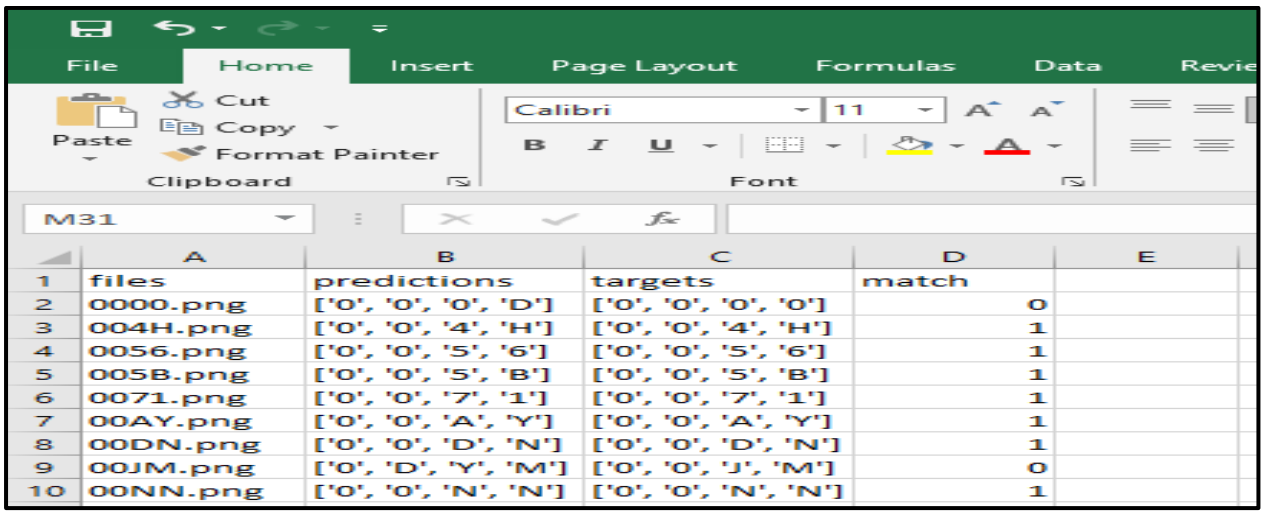


Figure 6.1 Accuracy of the CAPTCHA Breaker

6.2 RESULT ANALYSIS

The working of the model has been tested for various CAPTCHA values and was found to be accurate. The accuracy of the CAPTCHA Breaker is found to be **91%.** Whenever there is a deviation in the target CAPTCHA value and predicted CAPTCHA value, the match is set to 0 and whenever there is a match between the target and predicted CAPTCHA value, the match is set to 1, as shown in the Figure 6.2.



	A	B	C	D	E
1	files	predictions	targets	match	
2	0000.png	['0', '0', '0', 'D']	['0', '0', '0', '0']	0	
3	004H.png	['0', '0', '4', 'H']	['0', '0', '4', 'H']	1	
4	0056.png	['0', '0', '5', '6']	['0', '0', '5', '6']	1	
5	005B.png	['0', '0', '5', 'B']	['0', '0', '5', 'B']	1	
6	0071.png	['0', '0', '7', '1']	['0', '0', '7', '1']	1	
7	00AY.png	['0', '0', 'A', 'Y']	['0', '0', 'A', 'Y']	1	
8	00DN.png	['0', '0', 'D', 'N']	['0', '0', 'D', 'N']	1	
9	00JM.png	['0', 'D', 'Y', 'M']	['0', '0', 'J', 'M']	0	
10	00NN.png	['0', '0', 'N', 'N']	['0', '0', 'N', 'N']	1	

Figure 6.2 evaluation.csv file

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The CAPTCHAs are used to distinguish between humans and bots. The studies on CAPTCHA recognition can better detect vulnerabilities in the security of the CAPTCHA, thereby preventing some malicious intrusion in the network.

With the wide use of information system, CAPTCHA has been an essential component in login form in terms of system security. We have seen how Deep learning can influence CAPTCHAs and how they can be solved by bots with deep learning AI applications in them.

7.2 FUTURE WORK

Our further works involve creating CAPTCHAs that are difficult to be break by bots and deep learning frameworks. One idea is to generate CAPTCHAs using Generative Adversarial Networks (GANs) that are difficult to break, more secure and more usable.

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. So, using GAN we will be able to generate more secure CAPTCHAs.

APPENDICES

APPENDIX 1:

Code of CaptchaGenerator.py:

```
from claptcha import Claptcha
import os
import numpy as np
import cv2
import fire
from elapsedtimer import ElapsedTimer


def generate_captcha(outdir,font,num_captchas=20000):
    alphabets = 'abcdefghijklmnopqrstuvwxyz'
    alphabets = alphabets.upper()
    try:
        os.mkdir(outdir)
    except:
        'Directory already present,writing captchas to the same'
    for i in range(num_captchas):
        char_num_ind = list(np.random.randint(0,2,4))
        text = ''
        for ind in char_num_ind:
            if ind == 1:
                loc = np. random.randint(0,26,1)
                text = text + alphabets[np.random.randint(0,26,1)[0]]
            else:
                text = text + str(np.random.randint(0,10,1)[0])
        c = Claptcha(text,font)
        text,image = c.image
```

```
        image.save(outdir + text + '.png')


def
main_process(outdir_train,num_captchas_train,outdir_val,num_captchas_val,
outdir_test,num_captchas_test,font):


    generate_captcha(outdir_train,font,num_captchas_train)

    generate_captcha(outdir_val,font,num_captchas_val)

    generate_captcha(outdir_test,font,num_captchas_test)


if __name__ == '__main__':

    with ElapsedTimer('main_process'):

        fire.Fire(main_process)
```

APPENDIX 2:

Code of captcha_solver.py:

```
from claptcha import Claptcha
import pandas as pd
import os
import numpy as np
import cv2
import keras
from keras.models import Sequential,Model
from keras.layers import Dropout,Activation,Convolution2D,
GlobalAveragePooling2D,merge,MaxPooling2D,Conv2D,Flatten,Dense,Input
from keras import backend as K
from keras.optimizers import Adam
import fire
from elapsedtimer import ElapsedTimer
from keras.utils import plot_model

def create_dict_char_to_index():
    chars = 'abcdefghijklmnopqrstuvwxyz0123456789'.upper()
    chars = list(chars)
    index = np.arange(len(chars))
    char_to_index_dict,index_to_char_dict = {},{}
    for v,k in zip(index,chars):
        char_to_index_dict[k] = v
        index_to_char_dict[v] = k

    return char_to_index_dict,index_to_char_dict
def load_img(path,dim=(100,40)):
```

```
    img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img,dim)
    img = img.reshape((dim[1],dim[0],1))
    return img/255.


class DataGenerator(keras.utils.Sequence):
def __init__(self,dest,char_to_index_dict,batch_size=32,n_classes=36,dim=(40,
100,1) ,shuffle=True):
        self.dest = dest
        self.files = os.listdir(self.dest)
        self.char_to_index_dict = char_to_index_dict
self.batch_size = batch_size
        self.n_classes = n_classes
        self.dim = dim
        self.shuffle = shuffle
        self.on_epoch_end()


    def __len__(self):
        return int(np.floor(len(self.files) / self.batch_size))


    def __getitem__(self, index):
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]
        list_files = [self.files[k] for k in indexes]
        X, y = self.__data_generation(list_files)
        return X, y


    def on_epoch_end(self):
        self.indexes = np.arange(len(self.files))
```

```
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    def __data_generation(self,list_files):
        dim_h = self.dim[0]
        dim_w = self.dim[1]//4
        channels = self.dim[2]
X = np.empty((4*len(list_files),dim_h,dim_w,channels))
        y = np.empty((4*len(list_files)),dtype=int)
        k = -1
        for f in list_files:
            target = list(f.split('.')[0])
            target = [self.char_to_index_dict[c] for c in target]
            img = load_img(self.dest + f)
            img_h,img_w = img.shape[0],img.shape[1]
            crop_w = img.shape[1]//4
            for i in range(4):
                img_crop = img[:,i*crop_w:(i+1)*crop_w]
                k+=1
                X[k,] = img_crop
                y[k] = int(target[i])

        return X,y

def _model_(n_classes):
    input_ = Input(shape=(40,25,1))
    x = Conv2D(20, (5, 5), padding="same",activation="relu")(input_)
```

```
    x = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)
    x = Dropout(0.2)(x)
    x = Conv2D(50, (5, 5), padding="same", activation="relu")(x)
    x = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)
    x = Dropout(0.2)(x)
    x = Flatten()(x)
    x = Dense(1024, activation="relu")(x)
    out = Dense(n_classes,activation='softmax')(x)
    model = Model(inputs=[input_],outputs=out)
    model.compile(loss="sparse_categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])
    plot_model(model,show_shapes=True, to_file='model.png')
    return model


def train(dest_train,dest_val,outdir,batch_size,n_classes,dim,shuffle,epochs,lr):
    char_to_index_dict,index_to_char_dict = create_dict_char_to_index()
    model = _model_(n_classes)
    from keras.utils import plot_model
    plot_model(model, to_file=outdir + 'model.png')
    train_generator =
DataGenerator(dest_train,char_to_index_dict,batch_size,n_classes,dim,shuffle)
    val_generator =
DataGenerator(dest_val,char_to_index_dict,batch_size,n_classes,dim,shuffle)

model.fit_generator(train_generator,epochs=epochs,validation_data=val_genera
tor)
    model.save(outdir + 'captcha_breaker.h5')


def evaluate(model_path,eval_dest,outdir,fetch_target=True):
```

```
    char_to_index_dict,index_to_char_dict = create_dict_char_to_index()
    files = os.listdir(eval_dest)
    model = keras.models.load_model(model_path)
    predictions,targets = [],[]

    for f in files:
        if fetch_target == True:
            target = list(f.split('.')[0])
            targets.append(target)

        pred = []
        img = load_img(eval_dest + f)
        img_h,img_w = img.shape[0],img.shape[1]
crop_w = img.shape[1]//4
        for i in range(4):
            img_crop = img[:,i*crop_w:(i+1)*crop_w]
            img_crop = img_crop[np.newaxis,:]
            pred_index  = np.argmax(model.predict(img_crop),axis=1)
            pred_char   = index_to_char_dict[pred_index[0]]
            pred.append(pred_char)
        predictions.append(pred)

    df = pd.DataFrame()
    df['files'] = files
    df['predictions'] = predictions
    if fetch_target == True:
        match = []
```

```
        df['targets'] = targets
        accuracy_count = 0
        for i in range(len(files)):
            if targets[i] == predictions[i]:
                accuracy_count+= 1
                match.append(1)
            else:
                match.append(0)
        print(f'Accuracy: {accuracy_count/float(len(files))} ')
        eval_file = outdir + 'evaluation.csv'
        df['match'] = match
        df.to_csv(eval_file,index=False)
        print(f'Evaluation file written at: {eval_file} ')

if __name__ == '__main__':
    with ElapsedTimer('captcha_solver'):
        fire.Fire()
```

REFERENCES

1. D. George, W. Lehrach, K. Kansky, M. L´azaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang *et al.*, “A generative vision model that trains with high data efficiency and breaks text-based captchas,” *Science*, vol. 358, no. 6368, page. 2612, 2017.

2. G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, et al., "Yet another text captcha solver: A generative adversarial network based approach", Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 332-348, 2018.

3. H. Weng, B. Zhao, S. Ji, J. Chen, T. Wang, Q. He, et al., "Towards understanding the security of modern image captchas and underground captcha-solving services", *Big Data Mining and Analytics*, vol. 2, no. 2, pp. 118-144, 2019.

4. Li Wei, Xiang Li, TingRong Cao, Quan Zhang, LiangQi Zhou, and WenLi Wang. 2019. Research on Optimization of CAPTCHA Recognition Algorithm Based on SVM. In Proceedings of the 2019 11th International Conference on Machine Learning and Computing (ICMLC'19). ACM, New York, NY, USA, 236-240.

5. L. Zhang, Y. Xie, X. Luan and J. He, "Captcha automatic segmentation and recognition based on improved vertical projection," 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou, 2017, pp.1167-1172, doi: 10.1109/ICCSN.2017. 8230294.

6. M. Tang, H. Gao, Y. Zhang, Y. Liu, P. Zhang and P. Wang, "Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 10, pp. 2522-2537, Oct. 2018, doi: 10.1109/TIFS.2018.2821096.

7. M. Yadav and A. Kumar, "Feature extraction techniques for handwritten character recognition", International Journal of Advanced Research in Computer Science, vol. 9, no. 2, pp. 521, 2018.

8. Ning Zhang, Mohammadreza Ebrahimi, Weifeng Li, Hsinchun Chen, "A Generative Adversarial Learning Framework for Breaking Text-Based CAPTCHA in the Dark Web", Intelligence and Security Informatics (ISI) 2020 IEEE International Conference on, pp. 1-6, 2020.

9. Ondrej Bostik and Jan Klecka, “Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms,” IFAC-PapersOnLine, 2018, pp. 208-213.

10. P. Wang, H. Gao, Z. Shi, Z. Yuan and J. Hu, "Simple and Easy: Transfer Learning-Based Attacks to Text CAPTCHA," in IEEE Access, Vol. 8, pp. 59044-59058, 2020, doi: 10.1109/ACCESS.2020.2982945.

11. R. Chinchole, A. Kakad, S. Bhirud, A. Raghunath and M. S. Mumbaikar, "Online CAPTCHA Replacement through Face Detection," 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), Jaipur, India, 2019, pp. 32-35, doi: 10.1109/ICCT46177.2019.8969072.

12. R. Hussain, H. Gao and R. A. Shaikh, "Segmentation of connected characters in text-based captchas for intelligent character recognition", Multimedia Tools and Applications, vol. 76, no. 24, pp. 25547-25561, 2017.

13. S. Sachdev, "Breaking CAPTCHA characters using Multi-task Learning CNN and SVM," 2020 4th International Conference on Computational Intelligence and Networks (CINE), Kolkata, India, 2020, pp. 1-6, doi: 10.1109/CINE48825.2020.234400.

14. T. A. Le, A. G. Baydin, R. Zinkov, and F. Wood, “Using synthetic data to train neural networks is model-based reasoning,” in Neural Networks (IJCNN), 2017 International Joint Conference on. IEEE, 2017, pp. 3514–3521.

15. T. Zhang, H. Zheng and L. Zhang, "Verification CAPTCHA Based on Deep Learning," 2018 37th Chinese Control Conference (CCC), Wuhan, 2018, pp. 9056-9060, doi: 10.23919/ChiCC.2018.8482847.

16. Vaibhavi Deshmukh, Swarnima Deshmukh, Shivani Deosatwar, Reva Sarda, Lalit Kulkarni, "Versatile CAPTCHA Generation Using Machine Learning and Image Processing", Computing Communication and Automation (ICCCA) 2020 IEEE 5th International Conference on, pp. 385-389, 2020.

17. Y. Hu, L. Chen and J. Cheng, "A CAPTCHA recognition technology based on deep learning," 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, 2018, pp. 617-620, doi: 10.1109/ICIEA.2018.8397789.

18. Y. Zi, H. Gao, Z. Cheng and Y. Liu, "An End-to-End Attack on Text CAPTCHAs," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 753-766, 2020, doi: 10.1109/TIFS.2019.2928622.

19. Y. S. Aljarbou, "Improving of Current CAPTCHA Systems," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 2019, pp. 1-6, doi: 10.1109/CAIS.2019.8769466.

20. Y. Zhang, H. Gao, G. Pei, S. Luo, G. Chang and N. Cheng, "A Survey of Research on CAPTCHA Designing and Breaking Techniques," 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 2019, pp. 75-84, doi: 10.1109/TrustCom/BigDataSE.2019.00020.

21. Z. Fan, "Anti-cracking Technology for Captcha Recognition," 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Huangshan, China, 2018, pp. 1032-1035, doi: 10.1109/FSKD.2018.8686977.

22. https://en.wikipedia.org/wiki/OpenCV

23. https://en.wikipedia.org/wiki/TensorFlow

24. https://habr.com/en/post/496854/

25. https://keras.io/

26. https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697 target=”_blank” rel=”nofollow”

27. https://habr.com/en/post/496854/

28. https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697 target=”_blank” rel=”nofollow”