



**Captcha Recognition using  
Generative Adversarial Network Implementation**

A Project Report of Capstone Project-2

*Submitted By*

**Nishant Rai**

**(1613101458/16SCSE101835)**

*in partial fulfillment for the award of the degree  
of*

**Bachelor of Technology  
In  
Computer Science and Engineering**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**Under the Supervision of  
Mr. V. GOKUL RAJAN, M.Tech.  
Asst. Professor**

**JUNE-2020**



**SCHOOL OF COMPUTING AND SCIENCE AND  
ENGINEERING**

**BONAFIDE CERTIFICATE**

Certified that this project report “ Captcha Recognition using Generative Adversarial Network Implementation” is the bonafide work of “NISHANT RAI(1613101458)” who carried out the project work under my supervision.

SIGNATURE OF HEAD  
**Dr. MUNISH SHABARWAL,**  
PhD (Management), PhD (CS)  
**Professor & Dean,**  
**School of Computing Science &  
Engineering**

SIGNATURE OF SUPERVISOR  
**MR. V. GOKUL RAJAN**  
M.Tech.,  
**Professor**  
**School of Computing Science &  
Engineering**

# TABLE OF CONTENTS

---

<b>Chapter No.</b>	<b>TITLE</b>	<b>Page No.</b>
	<b>ABSTRACT</b>	<b>4</b>
	<b>LIST OF FIGURES</b>	<b>32</b>
	<b>LIST OF TABLES</b>	<b>33</b>
1	Introduction	5
	1.1 Overall Description-5	
	1.2 Purpose-6	
	1.3 Motivation And Scope-8	
2	Existing System	9
3	Proposed Model	12
4	Implementation Details	16
5	Experimentation Results	26
6	Conclusion/Future Enhancement	31
7	References	34

## Abstract

---

Captchas are widely implemented as stop gap measures in web services against automated attack vectors like spamming and bruteforcing. In many scenarios they are a single point of failure against such attack vectors till date despite several methods having been demonstrated to break captchas and circumvent the security that they offer by making use of advancement in image processing and machine learning technology. In Spite of such efforts, they continue to be widely deployed due to prior attacks being rigid in their approach and extensively form fitted for a particular captcha type requiring intensive manual effort to reconfigure to the ever changing captcha schemes and their associated security mechanisms such as fuzzing and distortions and, requirement of large training dataset that in absence of the captcha scheme source code becomes hard to collect. The project aims to develop a generic captcha solver using the GAN network approach to create a method where in minimal real captcha is required to train the solver by augmenting the training and testing set with synthesized captchas. This is achieved by first learning a captcha synthesizer to automatically generate synthetic captchas to learn a base solver, and then fine-tuning the base solver on a small set of real captchas using transfer learning. The beauty of this method allows us to turn the workflow into a procedural one allowing quick refitting to suit the needs of a different captcha than it was previously trained on. The process also benefits from increased accuracy and performance factors.

# 1.Introduction

## 1.1 | Overall Description

Text-based captcha(Completely Automated Public Turing Test to Tell Computers and Human Apart) are extensively deployed as a major step in the security mechanism across the entire web.Introduced in the year 1997 they were developed as a way to differentiate between humans and bots so as to prevent automated security intrusion techniques.Early captchas were simple text based images and proved sufficient in doing their tasks but with the advent of modern machine learning algorithms various security features were introduced to make them more robust as shown in below representation.

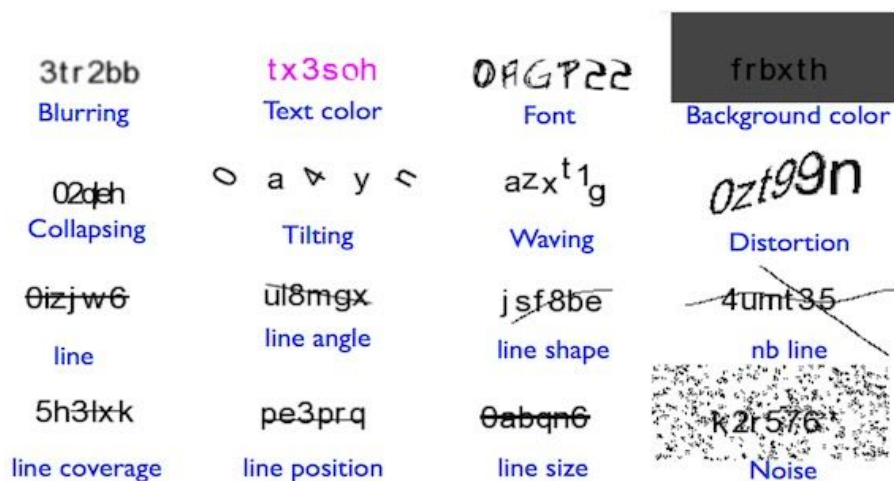


Figure 1:Sample Captcha Security Features

Newer advancements in deep learning such as Convolutional Neural Networks breakthroughs have presented methods to circumvent these security features but despite the various attacks demonstrated against captchas they still continue to be deployed extensively as the previous

attack were very rigid in their approach by either hard coding them for a specific captcha scheme or requiring extensive human intervention and manual labour to gather data and relabel it. The captcha scheme security features are rapidly changed and can easily outpace the attack techniques before they become big enough to cause significant damage or at the earliest sign of detection. Prior methods rely on image segmentation techniques and advanced security mechanisms or any alteration in their processing breaks the prior solvers and requires heavy expert involvement. These shortcomings of the previous attack techniques can be circumvented by introducing Generative Adversarial Network for image translation tasks in solvers that removes the need for massive labeled dataset of real captcha by augmenting the training dataset with synthesized ones so as to increase the operational accuracy and performance. The beauty of this process is that it is generic in its approach thus allowing quick adaptation to new captcha scheme in a procedural fashion with minimal human effort or may be automated altogether to adjust to a new captcha scheme by leveraging transfer learning.

We achieve this by implementing a Generative Adversarial Network for generating synthetic captchas that are a close match to the ones in question allowing us to rapidly train a base solver on these synthetic dataset and then fine tuning the model utilising real captchas effectively reducing the need for large data collection and labelling overhead which is quite a challenge when the source code for the captcha schemes is unavailable.

## 1.2 | Purpose

The main aim/ goal of this project is to demonstrate the use of recently introduced Generative Adversarial Network, a novel technique for image translation tasks to create a generic text-based

captcha solver that can easily solve them with minimal demands on training data and reliable performance as well as accuracy rate. The knowledge learned can also be extrapolated for usage with different captcha schemes easily with minimal human intervention. Through this project i hope to demonstrate the shortcomings of using captcha techniques as security measures without proper consideration into the ease with which they may be circumvented. These shortcomings may result in widespread spam and direct denial of service attacks that may cause losses to the establishment.

### 1.3 | Motivation and Scope

The main motivation for this project came after trying to use websites wherein captchas had been extensively employed as check stops against bots and as anti crawling mechanisms. The captchas had been deployed repeatedly so as to prevent spamming from bot agents and automated attack mechanisms but in actuality they hindered smooth user experience and added an extra hoop that people had to jump through. Until there were not many reliable methods to automatically circumvent the captchas in a reliable manner their use case may have been justified but with the advent of modern machine learning techniques there is a need to rethink their purpose as to what the captchas were trying to solve in the first place. As a result I decided to create a project demonstrating a method to solve captchas automatically in hopes that people would rethink the need and design of captchas to ensure they are relevant in the current times instead of a hindrance in an otherwise smooth user experience.

The scope of this project is as follows:-

1. Text based Captchas
2. Single word captchas
3. Captchas with security mechanisms such as random lines and and fuzziness as well as noise or text distortions



## 2.Existing Systems

---

Over the years there have been many demonstrations compromising the captcha system by implementing OCR based solvers that implement standard character recognition but were easily circumvented by adding confusion in the final output image.

With the advent of newer machine learning models there have been demonstrations utilising deep learning based models for analysing captchas

Important demonstrations in this have been in the form of (a) Model Based Approach (b) Deep Learning Based Approach

### (a) Model Based Approach-

The system works by employing a machine learning model utilising segmentation technique to separate individual letters of the associated captcha and then use simple character recognition models to deduce individual letters from the segmented images.

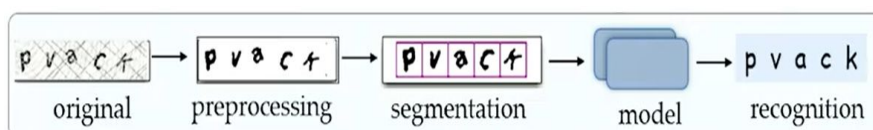


Figure 2:Model Based Approach

The technique heavily relies on segmentation process to ease the model difficulty. Segmentation tasks are usually accomplished by OpenCV blob or pixel level manipulation functions and hence very form fitted to a particular captcha scheme.

The preprocessing task is also very important for successful segmentation and can easily be defeated by complex security features. An example of a complex security feature which degrades the original captcha while undergoing pre-processing for data extraction is as follows.

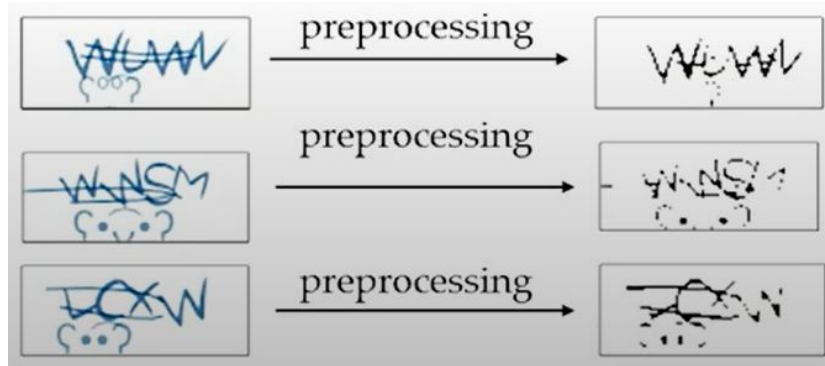


Figure 3: Degradation in image due to Pre-Processing

Any alteration in the captcha scheme causes the solver to be reconfigured and involves massive human intervention.

On the flipside this method is relatively robust and easy to use with good accuracy against captchas with rudimentary security features.

### **(b) Deep Learning Based Approach -**

With the advent of powerful deep learning based algorithms there have been multiple demonstrations utilising Convolutional Neural Network that have been used to essentially create solvers that are more generic in nature and can be refactored to changing needs.

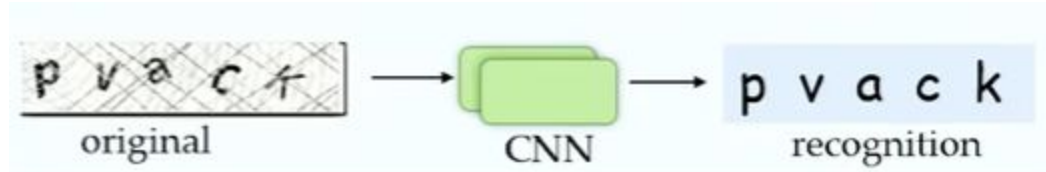


Figure 4:Deep Learning based model using CNN

While this method offers a real generic solver for breaking captchas it suffers from a major flaw as in the need for a huge dataset for training the convolutional neural network that can be easily exploited in situations wherein the source code for the captcha generating mechanism is unavailable. The two major methods for preventing third party entity from collecting large dataset of captcha images from a webservice that may be used for training neural network models are as follows-

1. Anti-Crawling Mechanisms
2. Constantly Changing Security Mechanisms in captcha

Another service currently in the market are CAPTCHA solving services that hire human labour in 3rd world countries like India, Bangladesh and Pakistan to solve the captcha and upload the solution with per captcha solving rate of 20 paise per captcha but this system is slow and in the current generation of Deep Learning a highly un-efficient method.

### 3. Proposed Model

The proposed model consists of four distinct segments each fulfilling a major step in the overall system creating a modular system.

The four segments are as follows:-

#### 1. Captcha Synthesis

Generate captchas that are visually similar to the target captchas. Generative Adversarial Network-based captcha generator consists of two parts: a captcha generator that tries to produce captchas which are as similar as possible to the target captchas, and a discriminator that tries to identify the synthetic captchas from the real ones. The process is continued till the discriminator fails to identify a certain major portion of the generated captchas from the real ones. Later the generator (referred as captcha synthesizer) is used to generate captchas as per demand with no upper limit. The synthesizer can thus create unlimited number of labelled captchas visually similar to the real ones without knowledge of the captcha schemes mode of operation.

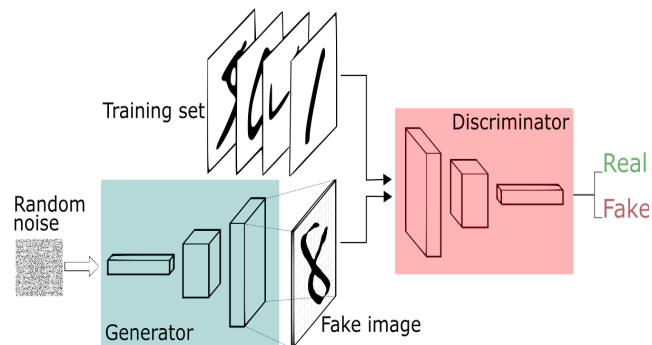


Figure 5: Generative Adversarial Network Model

## 2. Preprocessing

Removal of captcha security features such as lines and letter rotations as well as hollow fonts to create a systematically consistent data for ease of usage in further steps. The pre-processing model consists of 2 major operation

1. OpenCV based thresholding operation to reduce the bit depth of input image data and optimise resolution

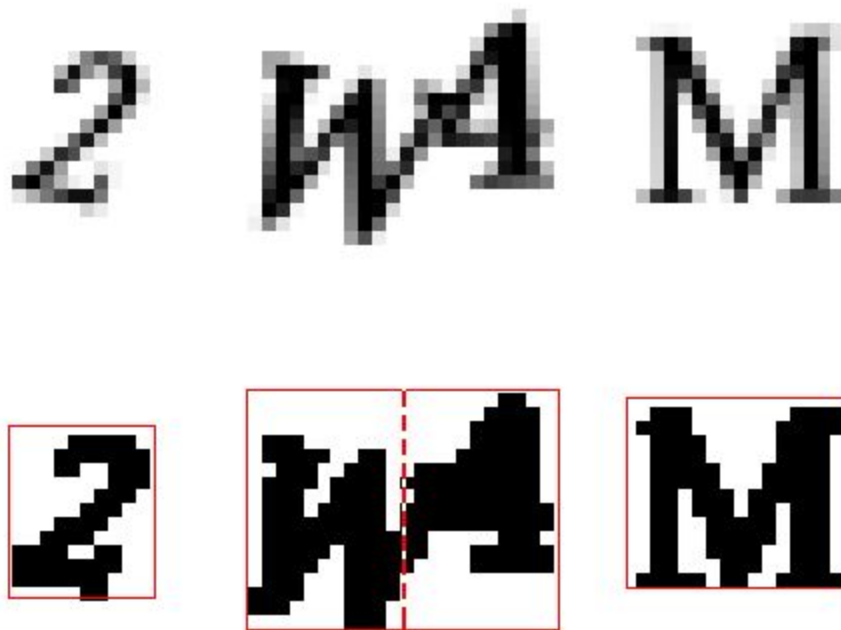


Figure 6: Sample Captcha Image Thresholding and Resolution Reduction

2. Removal of security features such as occluding lines rotations and transformations to generate clean images for usage with the next layers. This utilises a specialised GAN called Pix2Pix which is excellent for image translation tasks. The Pix2Pix model is trained with the synthetic captchas generated by the synthesiser and their corresponding

clean versions without security features. The trained model can now be used for any unseen captcha image of the target scheme.

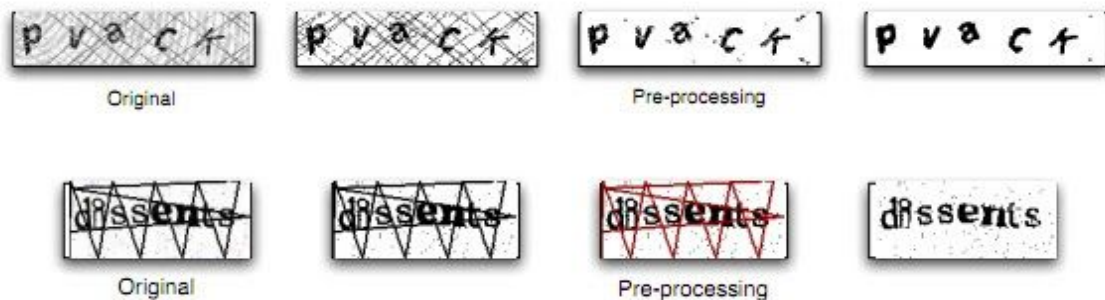


Figure 7: Example of captcha Pre-Processing Operation

### 3. Training the Base Solver

With the captcha synthesizer and the pre-processing model in place, generate a large number of synthetic captchas together with their labels (i.e., corresponding characters) and use this dataset to learn a base solver for a target captcha scheme. Captcha solver is a convolutional neural network (CNN). The captcha solver utilises pre-processed captcha images to generate the corresponding labels.

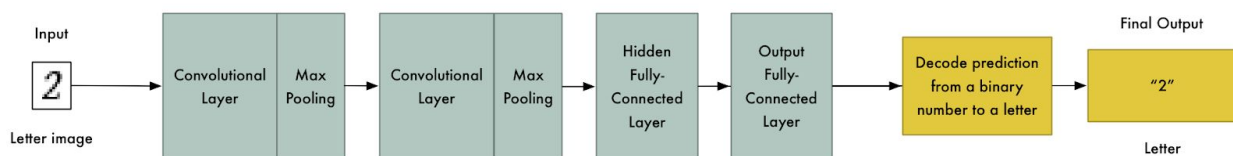


Figure 8: Example of a CNN model for image recognition

#### 4. Fine Tuning the Base Solver

Apply transfer learning to refine the base solver by using a small set of manually labeled captchas that are collected from the target website. Transfer learning allows us to leverage knowledge learned from synthetic captchas to reduce the cost of collecting and labeling captchas, and to further improve performance of the base model.

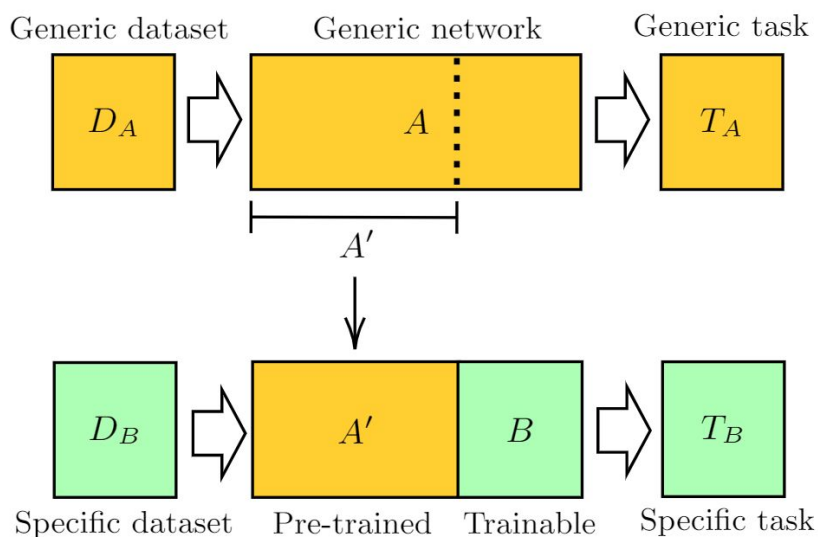


Figure 9: Transfer Learning Process to Specialise model to Particular Use Case

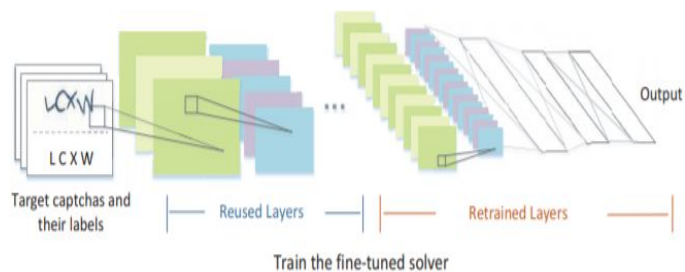


Figure 10: Fine Tuning the fully connected end layers

## 4.Implementation

---

Prior to detailing the implementation detail it is necessary to establish certain background details regarding the software libraries and implementation platform on which the system will be trained and deployed on.

Essential Softwares/Libraries -

- 1.Tensorflow 2 (GPU Variant)-Open Source Machine Learning Library for various creating and training the models.Background library for all machine learning models
2. Keras 2.1.2 - Abstraction for the Tensorflow Library for ease of deployment of machine learning library. Used for building the Solver and GAN network
- 3.OpenCV 4 - Image processing and computer vision library for image manipulation.Used for initial captcha image processing for normalisation process
- 4.Pix2Pix Library- Used in the process of image pre-processing for removal of security features for subsequent usage as means to simplify complexity of later networks operation.
- 5.Python 3 - Programming language in which the project will be developed.

Hardware Platform -

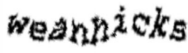



Initial testing and development was done on workstation with 2.4GHz Core i3 CPU , Nvidia 920MX GPU and 8GB Ram running Windows 10.The training and testing process was done on a cloud server running 2.4GHz Intel Xeon CPU, One NVIDIA Tesla P40 GPU and 32GB of RAM



## Targeted Captcha Schemes and Their Collection -

For the purposes of this project the schemes detailed in the below will be utilised.

While for some the source code is available and will be mainly used during the development phase. All captcha schemes will have a sample size of 1500 real captchas of which 500 would be used during training and 1000 for testing. Due to limitations in manual labour required to label the collected captchas for the schemes that do not have available source code thus hindering labeling operation they cannot be completely tested as yet.

Scheme	Example	Anti-Segmentation	Anti-Recognition	Excluded Characters	Source Code Available
Wikipedia		Overlapping Characters, English Letters	Rotation, Distortion, Waving	-	No
Microsoft		Overlapping Characters, Solid Background	Different font size, rotation, distortion, waving	0,1,5,D,G,I,Q,U	No
Really Simple Captcha		Overlapping	Rotation, Distortion	-	Yes
SimpleCaptcha		Solid Background	Occluding Lines, Rotation	-	Yes

**Table 1: Captchas and their associated properties**

### Common Captcha Security Features -

The main security features that this project will deal with are as below

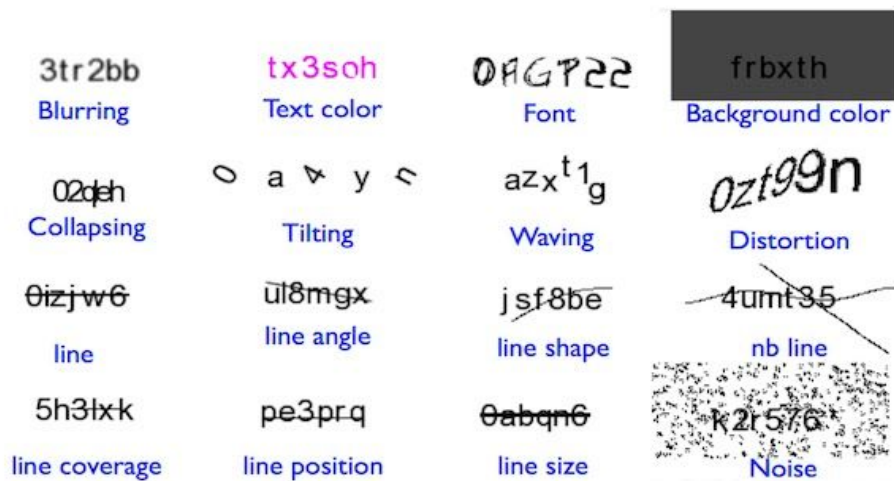


Figure 11:Security Features

### Normalisation for real captchas -

Prior to using the collected real captchas for any purpose they need to be first normalised and their bit depth as well as resolution reduced to make them suitable for operation.

This is accomplished by means of OpenCV thresholding function to more manageable size.



Figure 12: Captcha Pre Processing

```

filename = os.path.basename(captcha_image_file)
captcha_correct_text = os.path.splitext(filename)[0]
image = cv2.imread(captcha_image_file)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.copyMakeBorder(gray, 8, 8, 8, 8, cv2.BORDER_REPLICATE)
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
contours = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours = contours[1] if imutils.is_cv3() else contours[0]

```

### Captcha Synthesizer -

Using a captcha synthesizer, we can populate the training data with an unbounded number of synthetic captchas (that are similar to the real captchas). This allows the training data to cover the problem space far more finely than what could be achieved by exclusively using human-labelled training data. The training process is largely automatic except that a user needs to provide a small set of real captchas (500 in this work) of the target captcha scheme, and to define the set of security features. The security feature definition is achieved by configuring a set of pre-defined parameters as previously mentioned. These parameters can be easily extended and adjusted to target other captcha schemes. Captcha synthesizer consists of two components, a generator and a discriminator. The generator,  $G$ , is trained to produce outputs that cannot be distinguished from real captchas by an adversarially trained discriminator,  $D$ , which is trained to do as well as possible at detecting the synthetic captchas.

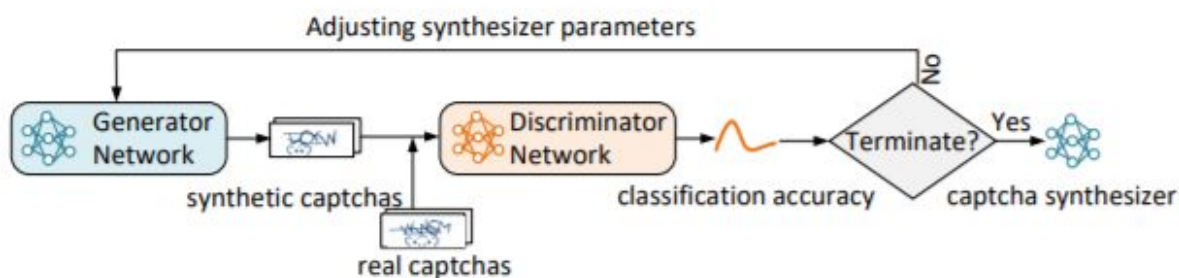


Figure 13:Synthesizer Model

## Captcha Generator -

The captcha generator is made of two main components as depicted below.

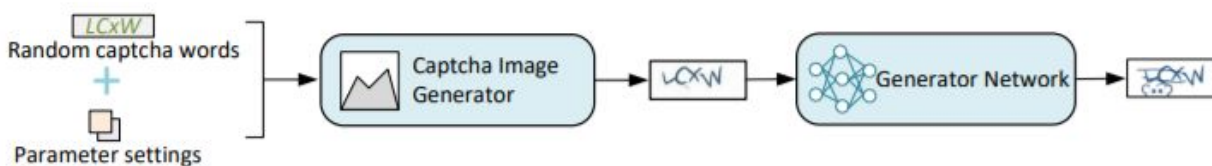


Figure 14: Captcha Generator Model

The first component captcha image generator takes in random captcha words/letters and generates images according to the set of security parameter settings and the captcha words.

The generator network uses a CNN then learns to modify the image at pixel level. The training process employs readjusting the parameter settings and the readjusting of CNN model to fool the discriminator from distinguishing amongst real and synthetic captchas.

The code for the generator is as follows -

```
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256) # Note: None is the batch size

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_bias=False, activation=
    assert model.output_shape == (None, 28, 28, 1)

    return model
```

## Captcha Discriminator -

The discriminator is also a convolutional network with the last layer outputting the probability of a captcha being real or fake.

For real captchas as  $y_j$  and synthetic captcha as  $x_i$  with target labels 0 and 1 for each respectively the loss function that the discriminator tries to minimise is as follows-

$$LD = -\sum_i \log D(x_i) - \sum_j \log(1 - D(y_j))$$

```
def make_discriminator_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                            input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model
```

### **Training -**

Training Done using Gradient Descent Method and utilises Adam Solver with a learning rate of 0.0002. The standard epoch for training process is currently set at 200.

The training objective is the standard GAN approach using the L1 norm with the regularization term set to 0.0001.

$$G^* = \arg \min G, \max D \text{ LcGAN } (G,D) + \lambda \text{L1}(G)$$

### **Captcha Preprocessing -**

Due to prior successful attempts to break captcha schemes have led to complex security features to circumvent which requires extensive effort in create a tailor fit strategy and this may still result in degradation of the captcha quality making it unusable for following steps or resort to lesser pre-processing and instead rely on complex deep learning models to circumvent the noise and distortion introduced due to the security features.

The key to a great captcha solving mechanism lies in being able to clean the captchas and provide enough separation amongst the letters for ease in training and recognition tasks.

Since we have an unbounded set of captchas both with security features and their associated version without them we can leverage Pix2Pix Gan for image cleanup and removal of security features.

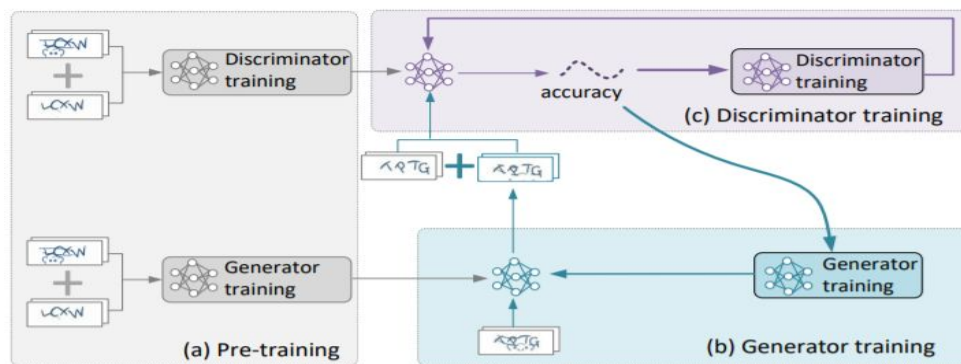


Figure 15: Generator Training

The Pix2Pix model also consists of a generator and discriminator which are pre-trained on captcha pairs with and without security features. Since the synthetic captcha are generated in the previous step it is trivial to generate these captcha pair as any security feature can simply be turned off thus allowing to find optimal balance of captcha pre-processing complexity such as occluding lines and noisy backgrounds.

## Captcha Solvers -

The preprocessed captcha is segmented into individual characters using the OpenCV

findContours() since the pre-processing step helps to separate the individual letters satisfactorily

helping to reduce complexity in the CNN model and not have to classify entire captcha problem

altogether

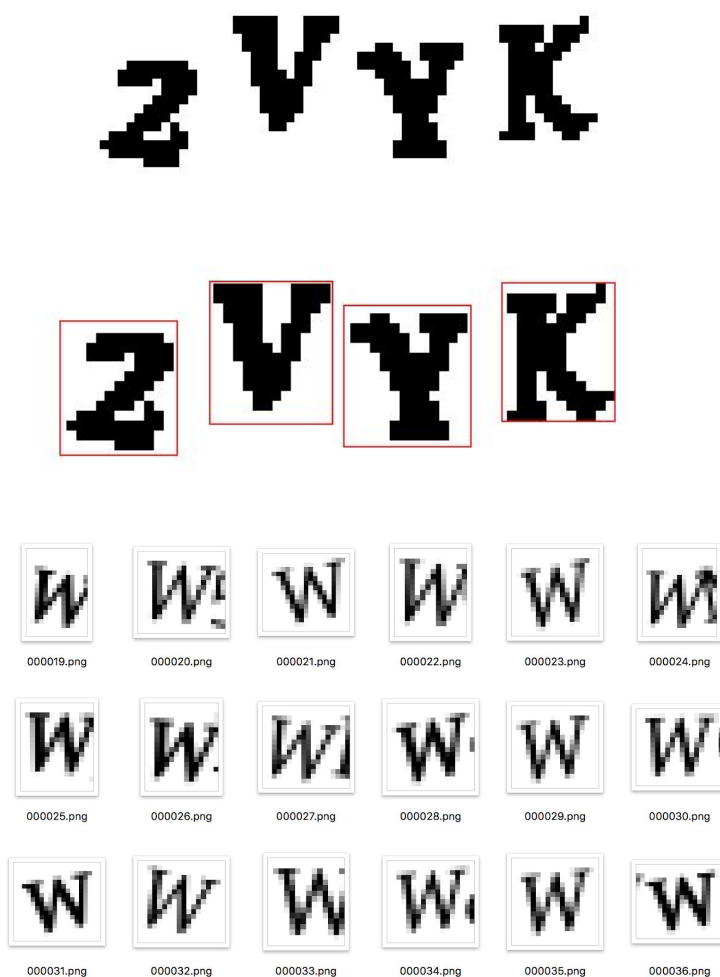


Figure 16:Character Segmentation



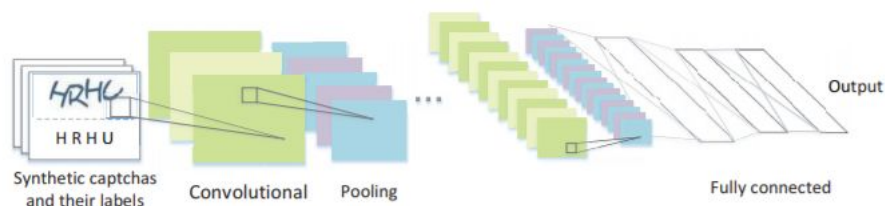
Since the labels for each captcha was already known the captcha can be further broken into individual labelled letters. The networks are trained on these individual labelled letters.

The captcha solver is a standard CNN based model(LeNet-5) consisting of four convolutional layers (5x5 filter) with max pooling followed by 500 Node hidden layer followed by a fully connected layer as output corresponding to each character and number.

```
with open(MODEL_LABELS_FILENAME, "wb") as f:
    pickle.dump(lb, f)
model = Sequential()
# First convolutional layer with max pooling
model.add(Conv2D(20, (5, 5), padding="same", input_shape=(20, 20, 1), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
# Second convolutional layer with max pooling
model.add(Conv2D(50, (5, 5), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
# Third convolutional layer with max pooling
model.add(Conv2D(50, (5, 5), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
# Fourth convolutional layer with max pooling
model.add(Conv2D(50, (5, 5), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
# Hidden layer with 500 nodes
model.add(Flatten())
model.add(Dense(500, activation="relu"))
# Output layer with 32 nodes (one for each possible letter/number we predict)
model.add(Dense(32, activation="softmax"))
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
# Train the neural network
model.fit(X_train, Y_train, validation_data=(X_test, Y_test), batch_size=32, epochs=10, verbose=1)
model.save(MODEL_FILENAME)
```

The training process is broken into two phases

1. Using the synthetic captcha the model is first trained as a generalized base solver





### Base Solver

- The base trained model is then fine tuned using the real captchas previously collected for the scheme in question and retraining the later layers using transfer learning process.

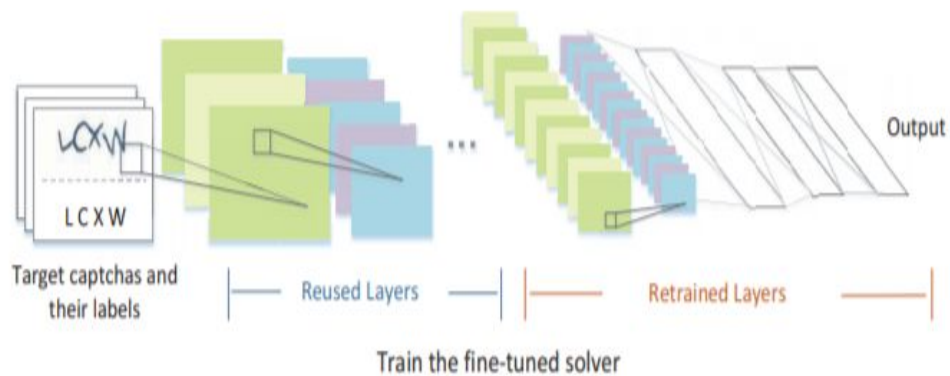


Figure 17: Fine Tuning

A complete flowchart diagram of the entire process is as follows

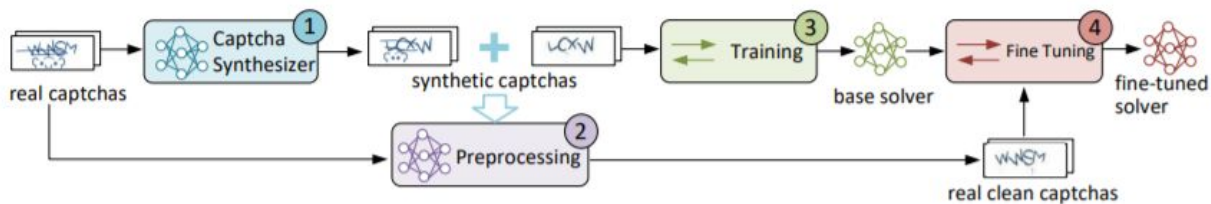


Figure 18: Overall Implementation Process

## 5. Experimentation Results

---

### 1. Synthetic Captcha Generation

mP6N      mP6N      mP6N      mP6N

(a) Overlapped characters

mP6N      mP6N      mP6N      mP6N

(b) Rotated characters

mP6N      mP6N      mP6N      mP6N

(c) Distorted characters

mP6N      mP6N      mP6N      mP6N

(e) Waved characters

Figure 19: Synthetic Captcha using Single Security Mechanism

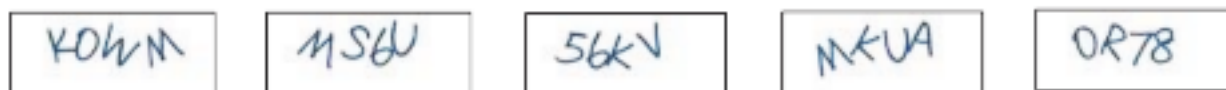


Figure 20: Synthetic Captcha w/o Security features(Occluding lines,Background Noise)

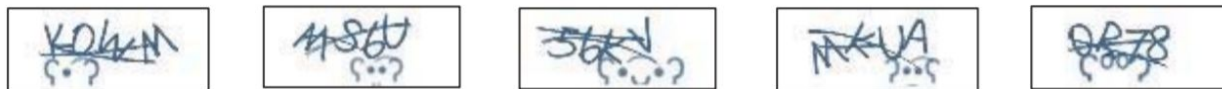


Figure 21: Synthetic Captcha with Security Features(Occluding Lines/Background Noise)

## 2.Pre-Processing Outcomes -



Figure 22: Security Feature Removal by Pix2Pix Pre-Processing(Occluding Line,Background Noise)



Figure 23: Example-2 Security Feature Removal by Pix2Pix Pre-Processing(Hollow Character,Occluding Line)

Important to note that the preprocessing method automatically standardizes the fonts and creates enough gap for easy segmentation prior to the training or testing operation as well as during captcha detection.




### 3.Captcha Recognition -



Figure 24: Labelling of Recognized Captcha Letters

#### 4.Success Rates

The method proposed shows better performance under varied comparison with respect to prior captcha solver methods[8,9,10,11].The success rates obtained during experimentation are detailed in Table 2.

Scheme	Example	Success Rates				
		Ref. [8]	Ref. [9]	Ref [10]	Ref. [11]	Our Approach
SimpleCaptcha(simple background)		93.2%	85.5%	90.1%	95.2%	98.3%
SimpleCaptcha(Image Background)		78.3%	54.6%	83.1%	87.1%	89.4%
SimpleCaptcha(Basic)		95.1%	90.6%	96.3%	98.3%	98.5%

**Table 2: Success rates compared to prior methods**

#### 5.Impact of security features on Success Rates

The ability of the captcha solver to succeed is highly dependent on the security feature implementations utilised in a captcha scheme and to understand the correlation between the different security features and their associated success rates in captcha solving can be seen in Table 3.Here we observe that as the difficulty of the captchas security mechanisms increases the success rates drops off thus suggesting using complex and multiple security features for better prevention against automated captcha solvers.

No.	Sample	Overlapping	Rotation	Distortion	Waving	Success Rate
1		✓	✓			74.85%
2		✓		✓		65.05%
3		✓			✓	58.8%
4			✓	✓		64.95%
5			✓		✓	82.35%
6				✓	✓	62.45%
7		✓	✓	✓		57.50%
8		✓	✓	✓	✓	52.50%
9		All security features				46.30%

Table 3: Impact of security features on success rates

## 6. Conclusion and Future Enhancement

---

The project demonstrates a novel approach to handling the problem of acquiring data for the training and testing purpose for making captcha solvers by moving away from traditional methods wherein large amounts of labelled dataset is required creating massive overhead in terms of manual labour in situations wherein a reliable method apart from crawling webpage for the captcha scheme is the only option. By only relying on a small captcha set to be collected from the target and then being able to generate labelled synthetic captchas in an unbounded manner opens up venues for advanced techniques usually impossible without having source code of the actual captcha mechanism. This allows the creation of a robust and highly adaptable solver hardened against the ever changing captcha security mechanisms and techniques. The mechanisms also reduces manual intervention in case of alterations in the target captcha mechanism for rapid adjustment.

Going further the key points that can be further improved upon in this project are as follows -

1. Comparative Analysis against prior techniques in terms of success rate and training efficiency.
2. Multi Word Captcha Adaptation
3. Implementation without segmentation to allow for true generic solving capabilities even in scenarios where the captcha word length may not be fixed.
4. Reduction in the training resource requirement to allow mobile deployment.

## List of Figures

- Figure 1:Sample Captcha Security Features
- Figure 2:Model Based Approach
- Figure 3:Degradation in image due to Pre-Processing
- Figure 4:Deep Learning based model using CNN
- Figure 5:Generative Adversarial Network Model
- Figure 6:Sample Captcha Image Thresholding and Resolution Reduction
- Figure 7:Example of captcha Pre-Processing Operation
- Figure 8:Example of a CNN model for image recognition
- Figure 9:Transfer Learning Process to Specialise model to Particular Use Case
- Figure 10:Fine Tuning the fully connected end layers
- Figure 11:Security Features
- Figure 12: Captcha Pre Processing
- Figure 13:Synthesizer Model
- Figure 14: Captcha Generator Model
- Figure 15: Generator Training
- Figure 16:Character Segmentation
- Figure 17:Fine Tuning
- Figure 18:Overall Implementation Process
- Figure 19:Synthetic Captcha using Single Security Mechanism
- Figure 20:Synthetic Captcha w/o Security features(Occluding lines,Background Noise)
- Figure 21:Synthetic Captcha with Security Features(Occluding Lines/Background Noise)
- Figure 22:Security Feature Removal by Pix2Pix Pre-Processing(Occluding Line,Background Noise)
- Figure 23:Security Feature Removal by Pix2Pix Pre-Processing(Hollow Character,Occluding Line)
- Figure 24:Labelling of Recognized Captcha Letters



## **List of Tables**

- Table 1: Captchas and their associated properties
- Table 2: Success rates compared to prior methods
- Table 3: Impact of security features on success rates

# References

---

1. Bursztein, E., Aigrain, J., Moscicki, A., and Mitchell, J. C. The end is nigh: generic solving of text-based captchas. In USENIX WOOT (2014).
2. Hyun KWON, Yongchul KIM, Hyunsoo YOON, CAPTCHA Image Generation Systems Using Generative Adversarial Networks
3. Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu . Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach
4. Audet, C., and Jr, J. E. D. Mesh adaptive direct search algorithms for constrained optimization. *Siam Journal on Optimization* 17, 1 (2006), 188–217.
5. Chellapilla, K., Larson, K., Simard, P. Y., and Czerwinski, M. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *Conference on Email & Anti-Spam* (2005).
6. George, D., Lehrach, W., Kansky, K., LÍczaro-Gredilla, M., Laan, C., Marthi, B., Lou, X., Meng, Z., Liu, Y., and Wang, H. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science* (2017), eaag2612
7. Sivakorn, S., Polakis, I., and Keromytis, A. D. I am robot: (deep) learning to break semantic image captchas. In *IEEE European Symposium on Security and Privacy* (2016), pp. 388–403.
8. Bursztein, E., Aigrain, J., Moscicki, A., and Mitchell, J. C. The end is nigh: generic solving of text-based captchas. In USENIX WOOT (2014)

9. Bursztein, E., Martin, M., and Mitchell, J. Text-based captcha strengths and weaknesses. In CCS (2011), pp. 125–138.
10. Gao, H., Yan, J., Cao, F., Zhang, Z., Lei, L., Tang, M., Zhang, P., Zhou, X., Wang, X., and Li, J. A simple generic attack on text captchas. In NDSS (2016).
11. George, D., Lehrach, W., Kansky, K., LÍczaro-Gredilla, M., Laan, C., Marthi, B., Lou, X., Meng, Z., Liu, Y., and Wang, H. A generative vision model that trains with high data efficiency and breaks text-based captchas. Science (2017), eaag2612.
12. Mohameda, M., Gaob, S., Sachdevac, N., Saxena, N., Zhangd, C., Kumaraguruc, P., and Oorschote, P. C. V. On the security and usability of dynamic cognitive game captchas. Journal of Computer Security (2017), 1–26.