

# Solving CAPTCHAS using Deep Learning

Sukhdeep Singh, Macquarie University

Student Id: 44442467

## Abstract

Captchas are based on the semantic gap between the abilities of a human vs machines to understand multimedia content, this has significant implications when it comes to IT security. In this paper I will compare three different techniques for solving captchas namely Optical character recognition, K-nearest neighbour classification with Image segmentation and Modern deep learning-based approach with noise filtration. This paper looks at some of the challenges of different types of captchas, effect of preprocessing on the results and performance comparison for the models.

## 1 Introduction

Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) is a form of reverse turing test administered by computers. Because computers cannot process distorted images and text as humans, due to this semantic gap CAPTCHAs proved effective at thwarting most automated attacks. CAPTCHAs as a security measure have been present in the industry since 1997, many service providers such as Gmail, YouTube, MSN and others use CAPTCHAs to optimize their service and prevent automated bots from misusing their technology.

However, the pain caused by illegible strings of letters and numbers is deemed as unnecessary and obtrusive for users, who are already using these services. According to W3C consortium *"asking users who are blind, visually impaired or dyslexic to identify textual characters in a distorted graphic is asking them to perform a task they are intrinsically least able to accomplish"*

ML algorithms (specifically deep learning-based methods) have proved to be successful in

the past in solving these kinds of problems with significant success (measured as detection accuracy), this is one of the motivations behind this project, to train a neural network on different features of a captcha image and test its accuracy & time required to solve different kinds of Captcha challenges.

## 2 Related work

Various research efforts to solve captcha challenges [2,4] show that, some techniques show up quite frequently and form a strong basis of preprocessing, techniques such as image segmentation to single out characters, noise removal and choice of classifier for character recognition. Text based captcha generators are usually cheap to implement and widely used for the same reason, they are built by understanding the inner workings of popular OCR software(s) and are designed to defeat them by distorting the images so that it becomes difficult to recognize (adding noise, lines, background blur, shadows, rotation of characters etc.). All authors unanimously agree on the fact that the detection of standard OCR can be improved by using custom trained classifier and applying preprocessing techniques.

In the paper [1,2] authors reveal several of the captchas security measures and counter measures which are recommended for designing an attack (solver), the efficiency of these techniques has been tested thoroughly by comparing the recognition accuracy of the solver against respective countermeasures, paving way for increased success in recognition of such challenges and a generalized approach. Deep learning-based captcha solvers are able to generalize the solution to some extent and have proved successful against variety of security measures [2].

### 3. Description of the data

Common features	
Number of total images	20K
Number of classes	4

Figure 1

#### 3.1 Security features of the classes:

- Class 1: It contains clear captcha with no occluding lines and noise and contains only capital alpha-numeric characters with white background. The images<sup>1</sup> are in PNG format with resolution of 72\*24 pixels.
- Class 2: It contains clear captcha with no occluding lines and noise and contains only small alpha-numeric characters with non-white background. The images<sup>2</sup> are in PNG format with resolution of 120\*48 pixels.
- Class 3: It contains noisy captcha with occluding lines and hollow characters with background noise and contains only capital alpha-numeric characters. The images<sup>3</sup> are in PNG format with resolution of 100\*20 pixels.
- Class 4: It contains noisy captcha with occluding lines and noise and contains only small alpha-numeric characters. The images<sup>4</sup> are in PNG format with resolution of 100\*40 pixels.

Image name corresponds to the solution of the captcha for Classes 1 and 4. For classes 2 and 3 we have a label file which maps to the solution i.e filename-solution.

The estimated accuracy of the labels is 98-99% as mentioned in the individual sources.

**We will not be able to identify any kind of type II error (False Negatives – Predictions are correct, but labels are wrong).**

In our case probability of type II error is given by:

$$P(\text{Type II error}) = P(\bar{L}) * P(M)$$

where,

$$\bar{L} = P(\text{incorrect label})$$

$$M = P(\text{correct prediction by model})$$

Data sources:

1. <https://bit.ly/2ZAcZLT>
2. <https://bit.ly/2L3Gkuq>
3. <https://bit.ly/2MX3WDv>
4. <https://bit.ly/2MX47yF>

#### 3.2 Data pre-processing

The four classes have been divided into two main categories:

- Clear Captcha – 10K
- Noisy Captcha – 10K

We have two main categories of clear captcha and noisy captcha where noisy captcha contains high levels of salt & pepper noise, occluding lines and hollow characters which can be removed by applying various filters to captcha. Following is a flow diagram which displays various steps involved in removing noise:

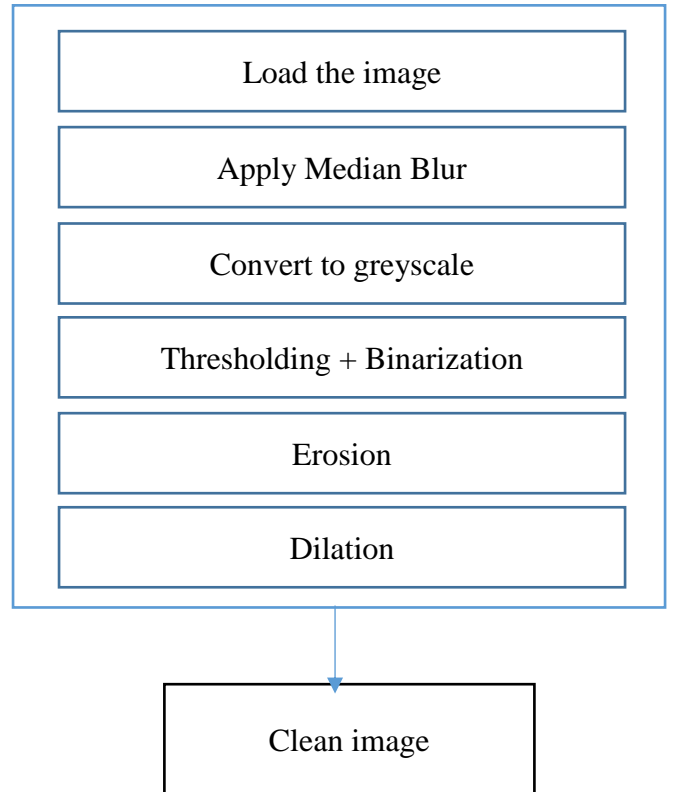


Figure 2

**Median Blur:** This filter is used to remove the background noise of the image (if present), this function replaces each pixel in the image by taking the median of the nearby pixels, for this project I have selected a window size (number of nearby neighbors = 3)

**Convert to greyscale:** As the name suggests this function is used to convert a 3 channel (Red, Green, Blue) image to 2 channels. This is an essential step because this allows us to express an image as a 2-dimensional array for processing purposes.

**Erosion & Dilation:** These are morphological image processing techniques, mainly used for removing salt and pepper noise for images.

**Thresholding:** This is essentially used to convert grayscale image to black and white, I have compared the outputs of various thresholding algorithms where I have chosen a local method.

The last part of pre-processing is line removal from the image. I have attempted to grab the start and end position of the lines, however due to the curved nature of the line, this method proved to be marginally successful, since I was unable to get rid of the lines completely. However, this technique is successful in cases if lines are straight.

Here's a visual comparison of various preprocessing steps, I used one image from each class and compiled a table of result to get a better understanding of the all the preprocessing steps.

Original captchas	
2 A 2 X	8384
XDEY	FAIRLY

**Pre-processed Class 1 captchas using various Thresholding techniques:**

Original	Isodata
2 A 2 X	2 A 2 X
Li	Mean
2 A 2 X	2 A 2 X
Minimum	Otsu
2 A 2 X	2 A 2 X
Triangle	Yen
2 A 2 X	2 A 2 X

**Pre-processed Class 2 captchas using different algorithms of Threshold:**

Original	Isodata
8384	8384
Li	Mean
8384	8384
Minimum	Otsu
8384	8384
Triangle	Yen
8384	8384

**Pre-processed Class 3 captchas using different algorithms of Threshold:**




Original	Isodata
XDEY	XDEY
Li	Mean
XDEY	XDEY
Minimum	Otsu
XDEY	XDEY
Triangle	Yen
XDEY	XDEY

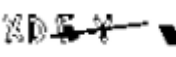

**Pre-processed Class 4 captchas using different algorithms of Threshold:**

Original	Isodata
FAIRLY	FAIRLY
Li	Mean
FAIRLY	FAIRLY
Minimum	Otsu
FAIRLY	FAIRLY
Triangle	Yen
FAIRLY	FAIRLY

**Choosing optimum filtration parameters:** In this part, I would like to demonstrate the effects of poor choice of parameters selected during filtration processes applied to the data, if we set a specific parameter too high or too low e.g. Median Blur, we observe a blurred picture, similarly the resulting image may be either too black, too white, blurry, too noisy etc. if the choice of the parameters is not good, I used trial and error to see what works, and found class 2 images do not respond well to the prepro-

cessing pipeline, so we used just the thresholding and binarization for this subset.

Results with 'poor choice' of filtration parameters		
		

Final cleaned captchas	
	
2 A 2 X	8384

## 4. Methods

Model is trained on 80% of the data and 20% is used for testing the prediction accuracies. Images are randomly shuffled to ensure unbiased.

The validation set consists of 100 images, completely independent of the training and testing datasets. To test for generality of captcha recognition technique, each model is trained on a single class of captcha and attempt to recognize the other classes, this will help see the performance of the recognition techniques.

The recognition algorithms used are:

- Optical character reader with PyTesseract (OCR)
- KNN with Image segmentation
- Neural network with Noise filtering (Median blur)


### 4.1 Optical Character Reader

Optical Character Recognition (OCR) is the technique used by computers for recognizing printed or written characters. It enables conversion of different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data.

In this conventional technique I use an OCR library developed by google, known as Tesseract, the python wrapper for this OCR library is called as PyTesseract which allows the data to be read in a way which can be interpreted by python.

Initial approach is simple, attempt to read the Captcha text from an Image using the image\_to\_string function without any pre-processing. I observed that

results were not very accurate even for the clear captchas and noisy ones were way off the actual text.

<code>Image.open("2abnp.png")</code>	<code>Image.open("2A2X.png")</code>
	
<code>recognize_captcha("2abnp.png")</code>	<code>recognize_captcha("2A2X.png")</code>
'ZafdsiyR'	'2A2X%'

Later we revisit the same captchas with some image pre-processing and see marginally better results.

<code>Image.open("test2.jpg")</code>	<code>Image.open("2A2X.png")</code>
	
<code>recognize_captcha_threshold("test2.jpg")</code>	<code>recognize_captcha_threshold("2A2X.png")</code>
'3558'	'2A2X'

### 4.2 KNN with Image segmentation

K-nearest neighbours is a well-known ML algorithm for classification, in this technique the output of the model consists of a class membership (predicted label) which is computed by calculating its distance from the ground truth labels from feature space.

For segmentation of the individual characters I have defined the manual pixel ranges of the letters and divided it into four regions. Each segment of the image is passed on the KNN model for prediction

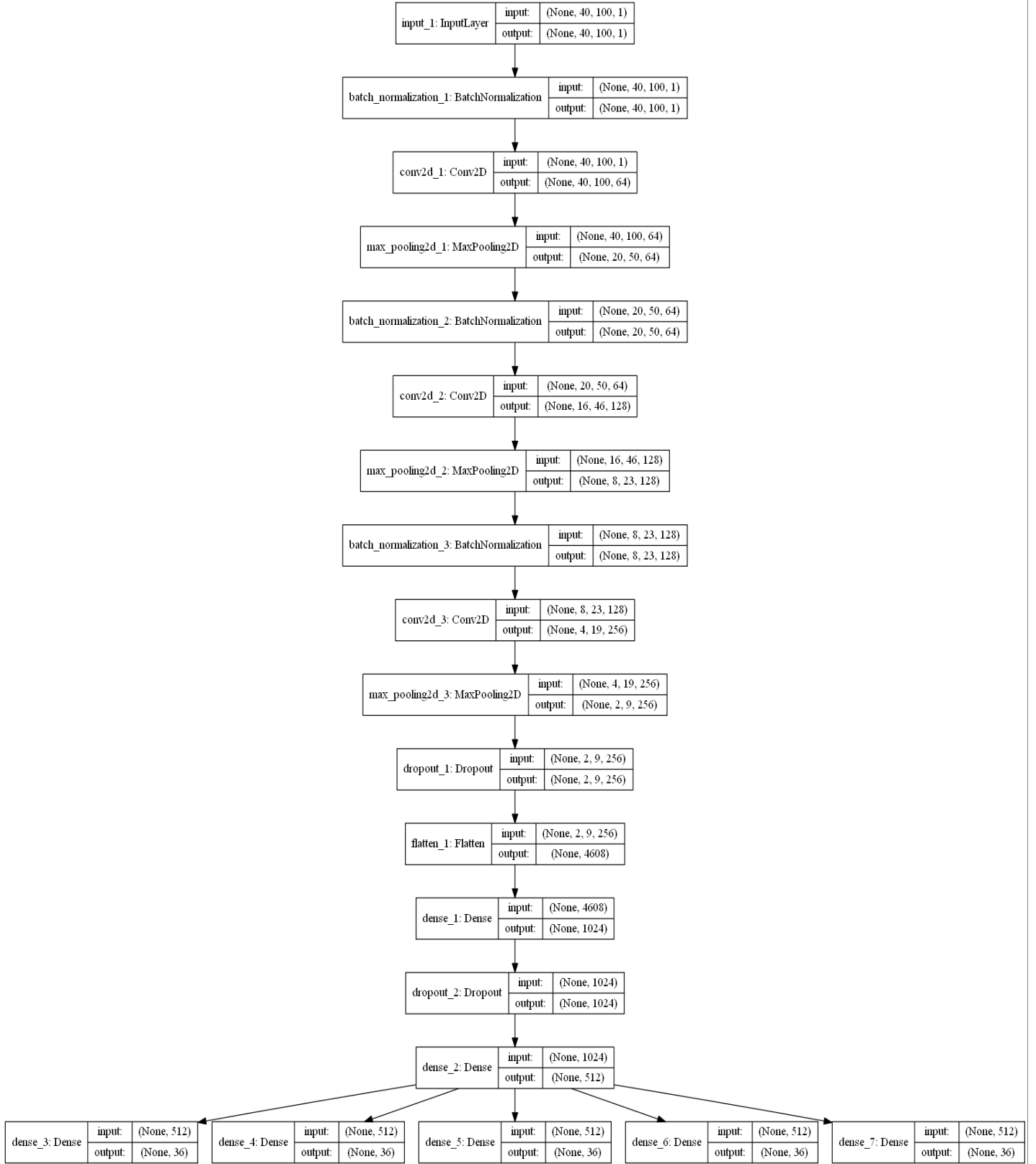
The ground truth library consists known segments of characters i.e. 0, 1...,9 and A – Z which are used to calculate the Euclidean distance for classifying segmented characters.

In case the characters overlap we take the mean distance as boundary.

### 4.3 Neural network with Noise filtering (Median blur)

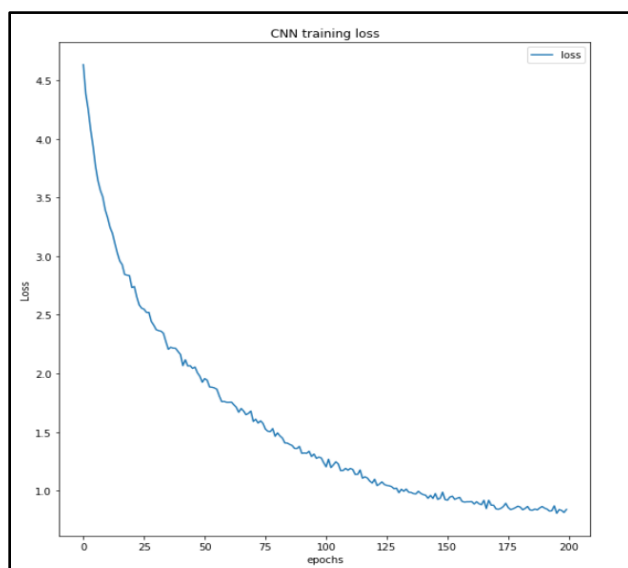
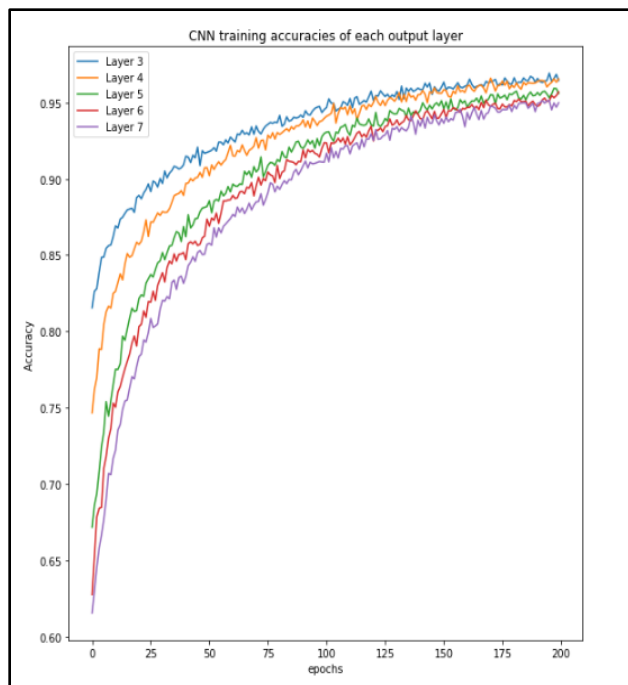
Neural Network with Noise filtering works on the principle of **Convolutional Neural Network**. CNN is a deep learning algorithm which is used to train and test, each input image and pass it further through a series of convolution layers with filters (Kernels), pooling, fully connected layers (FC) and apply **softmax function to classify an object with probabilistic values between 0 and 1**. This CNN is specific to 5-character images and uses Keras as frontend to train the model

### 4.3.1. Model Architecture



## 5. Results

Technique	Validation set accuracy			
	Class 1	Class 2	Class 3	Class 4
OCR	50%	39%	0%	1%
KNN	-	98%	-	-
CNN	-	-	-	13%



OCR is the only model which can currently accept all different resolutions of image out of the box, as the goal was to train the models on a one class of image only and test on the remaining ones, the heterogenous sizes and resolution proved to be a deterrent.

KNN model achieves a maximum of 98% accuracy for class 2.

CNN achieves an accuracy of 13% when testing on 5 letter noisy captchas (class 4) page border, and if trained on clear captcha it manages performance close to KNN. Training for the model was done on Intel CPU and took more than 10hrs.

## 6. Further scope

Scope of this project experienced a few bottlenecks during the testing phase as mentioned due to the heterogenous data, therefore the first step would be to generate the captcha programmatically or choose a single data set. Nevertheless, the ability of models to crack the captcha well exceeds the minimum threshold level of acceptance which is 0.01% [according to authors in 2 “the captcha design goal is that “automatic scripts should not be more successful than 1 in 10,000” attempts (i.e. a precision of 0.01%)]. This shows that computers equipped with smart algorithms will soon be able to close the semantic gap and achieve human like performance.

We have also overlooked many other vital security features of other popular captchas

1. Skewed and randomly rotated characters
2. Animated captchas
3. Variable font size and charset

Although a more sophisticated approach such as [1] shows that these can also be solved easily with the appropriate pre-processing and segmentation techniques.

## 7. References

1. Ye, G., Tang, Z., Fang, D., Zhu, Z., Feng, Y., Xu, P., ... & Wang, Z. (2018, October). Yet another text captcha solver: A generative adversarial network-based approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 332-348). ACM.
2. Bursztein, E., Martin, M., & Mitchell, J. (2011, October). Text-based CAPTCHA strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 125-138). ACM.
3. Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., & Yan, J. (2013, November). The robustness of hollow CAPTCHAs. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 1075-1086).
4. ACM.Kapoor, D., Bangar, H., & Sethi, A. (2012, December). An ingenious technique for symbol identification from high noise CAPTCHA images. In *2012 Annual IEEE India Conference (INDICON)* (pp. 098-103). IEEE.