# deCAPTCHA:Breaking IITK webmail / CSE-webmail captchas

**MLG - 4**
Bhangale Pratik Anil (14173)
Ashish Kumar Singh (14142)
Gaurav Kumar (14240)
Bhargav Ganguly (14177)
Shubham Jain (14676)

https://github.com/pratikab/deCAPTCHA

## Abstract

CAPTCHA stands for 'Completely Automated Public Turing test to tell Computers and Humans Apart'.It is generated by distorting an image with text/numbers in different ways. They are mainly used as a security check to ensure only human users can pass through. Generally, computers or bots are not capable of solving a captcha. Breaking of CAPTCHA is a very-well known machine learning problem and is being explored from a long time. In this project, we have attempted to break the IITK webmail/CSE-webmail text-based CAPTCHA images. We will try to design novel algorithms for automating captcha recognition.

## 1 Problem statement

Design and implementation of a learning algorithm that will correctly recognize and output the characters in any IITK webmail/CSE-webmail CAPTCHA image.

### 1.1 Input

A text-based image of a CAPTCHA is input having distorted character and noise in the background.Character can be of different size or may be rotated.Types of noise such as multiple shades dots or lines is present in the image to have more diffulty level of CAPTCHA.

### 1.2 Output

A string of correct sequentially recognized characters from the set of 26 letters and 10 digits have to be the output.

## 2 Problem motivation

CAPTCHAs are extensively used by websites so that they can restrict automatic data update and malicious attacks by bots. Generally, it has been difficult for bots to break CAPTCHAs when compared to humans.This difficulty is ensured by warping the shape of the characters, introducing clutter in the image etc. But, there has been several attempts towards captcha breaking and significant results have been observed using state-of-the-art machine learning techniques. So, there is indeed a need to design more robust text-based captchas.

# 3 Existing work

Conventionally, the processing of extracting the characters from a given natural image is done in three steps. Localizing the characters in the image; segmenting the image accordingly to obtain the characters; recognizing the characters.Chellapilla and Simard[1] have used this three-tier approach. However, modern captchas have overlapping characters which look similar to hand-written ones. Here, using the conventional technique to segment the characters would give bad results. LeCun *et al*[2] have suggested use of Convolutional Neural Networks(CNN) for recognition of hand-written digits. Goodfellow *et al*[3] proposed to merge the three steps using deep CNNs which required training on millions of images. Stark *et al*[4] have proposed a active deep learning based approach which uses a small initially labelled dataset and performs fine tuning on the network during the run-time.

# 4 Methodology

## 4.1 CSE-webmail CAPTCHA

Samples of CSE CAPTCHAs

### 4.1.1 Observations

- There is noise in the background.
- All noisy pixels have the same pixel value (colour).
- Number of characters is always 5.
- No characters are touching are overlapping
- Characters are not warped (distorted).
- No rotation in the characters.
- All character pixels have the same value (colour).
- Background is always white.

### 4.1.2 Segmentation

**Step1:** Noise removal
All noise pixels were modified to white pixels directly. This will completely remove the noise.

**Step2:** Grayscale and thresholding
The given image is converted into binary format with the background color being white.

**Step3:** Segmenting into 5 parts
We slide a vertical line across the captcha image, till we find a vertical line with all white pixels. Once we find such a vertical line, we store the coordinate corresponding to the horizontal axis temporarily. Let,it be $h_1$.We then look for a line which contains at-least one non-white pixel. If find such a line at $h_2$ within a distance $\leq 50$ from $h_1$. We store $0.5(h_1 + h_2)$. In the case of CSE webmail, this always holds. We store all these vertical lines.

**Step4:** Resizing every segment
Resize every image segment to 40x40 pixels. We resize the individual characters using cubic interpolation and store them separately for recognition.

(a) Original Image (b) Step 1 and 2 (c) Step 3 (d) Step 4

#### 4.1.3 Recognition

The segmented character images will be grayscaled which are sent to a CNN for recognition. The network will have have 1600 input nodes because of 40 X 40 pixels input character image. We have used 2 convolutional layers each to learn local features and to reduce the number of edges across subsequent hidden layers. The convolutional window size for these layers was tuned to 5 X 5. Each convolutional layer is followed by a max pooling operation and ReLU activation function. Max Pool operation has a stride window of size 2 X 2. Max Pooling is done to aggregate info learned in local feature and to use them in structure detection for higher level features. 3 fully connected layers are deployed after these convolution operations. Output layer has nodes equal to number of classes i.e.36 classes - 26 alphabets and 10 numerals. Loss function used is Cross Entropy Loss which is state-of-art loss function for multi-class classification in CNN.

#### 4.1.4 Experimental Results

| | |
|---|---|
| Training Dataset Size | 30,000 |
| Test Dataset Size | 10,000 |
| Training accuracy | 100% |
| Testing accuracy | 100% |

### 4.2 Webmail1 CAPTCHA

Samples of Webmail1 CAPTCHAs

#### 4.2.1 Observations

- No noise in background
- Always 6 characters are present
- Background is always lighter than character border
- Minor overlapping has been observed
- Characters are not warped (distorted)
- Image dimensions (40 x 200 ) pixels

#### 4.2.2 Segmentation

**Step1:** Gray scaling and Thresholding Convert the image into binary format and remove the background by thresholding

**Step2:** Segmenting image into 6 parts After thresholding we start scanning the image in the horizontal direction to find a vertical line with all white pixels. Once we find such a vertical line,we store the coordinate corresponding to the horizontal axis temporarily.Let,it be $h_1$. We then look for a line which contains at-least one non-white pixel. If find such a line at $h_2$ within a distance $\leq 50$ from $h_1$.We store $0.5(h_1 + h_2)$. Otherwise,we divide the region between $h_1$ and $h_2$ equally such that two adjacent vertical lines are 29 pixels apart.We store all these vertical lines. We do this until we scan the image completely.

**Step3:** Re-sizing every image segment Resize every segment to 40x40 pixels. Resize all the obtained characters using cubic interpolation and store them for segmentation.

(a) Original Image

(b) Step 1

(c) Step 2

(d) Step 3

#### 4.2.3 Recognition

The segmented character images will be grayscaled which are sent to a similar CNN for recognition.

#### 4.2.4 Experimental Results

| | |
|---|---|
| Training Dataset Size | 1,000 |
| Test Dataset Size | 200 |
| Training accuracy | 100% |
| Testing accuracy | 97% |

### 4.3 Webmail CAPTCHA

Samples of Webmail CAPTCHAs

#### 4.3.1 Observations

- No generalized pattern of background noise
- Number of characters is either 3 or 4.
- Characters are not warped (distorted).
- No rotation in the characters.
- Characters are visually almost equally spaced

#### 4.3.2 Binary Classification to get number of characters

The CAPTCHA image is sent into a CNN for binary classification. The network has 3 convolutional layers to learn local features and to greatly reduce the number of edges across hidden layers. Each convolutional layer is followed by a max pooling operation and ReLU activation function. Max Pool operation has a stride window of size $2 \times 2$. 3 fully connected layers are deployed after these convolution operations. Output layer has 2 nodes where each node represents the probability of a particular class. Loss function used is Cross Entropy Loss.

#### 4.3.3 Segmentation

**Approach 1** : Direct Method

- Binary Classifier gives the number of characters in the CAPTCHA image.
- In this method depending on the number of characters in the image, it is divided into that many equal parts to give the sub-images having only single character in all parts.
- After this, each sub-image will be directly passed in the CNN to recognize the single character.

(a) Original Image

(b) Segmented Images

**Approach 2** : Boundary Segmentation

- The CAPTCHA is first converted into gray-scale and then thresholded
- Connected components of image are first identified and those with less than 25 points are removed.
- Morphological operations such as erosion, dilation are applied and their parameters are suitably tuned
- Apply this resultant mask on coloured image to get the denoised image.

(a) Original Image

(b) Segmented Images

**Approach 3** : Dominant Color Segmentation

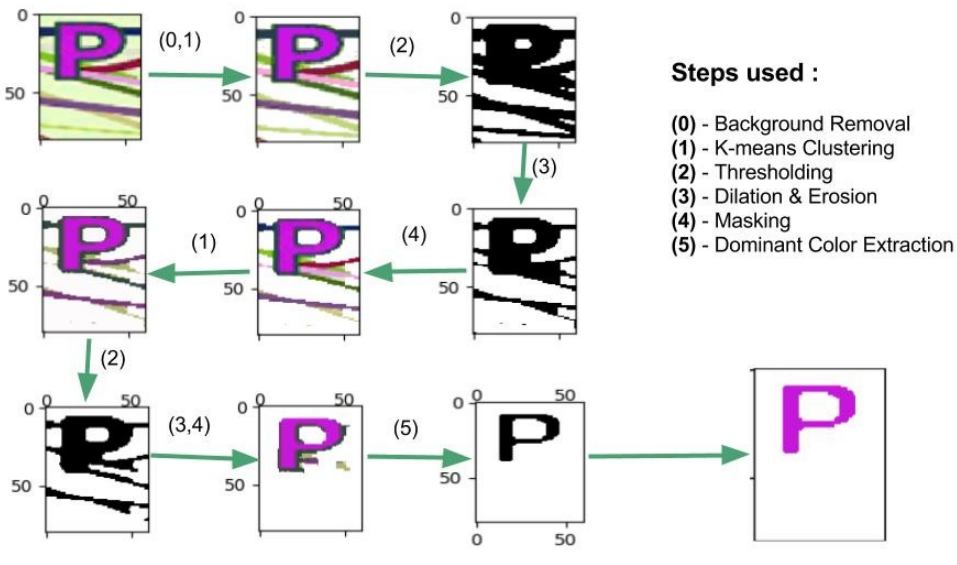
Sub-images are created by equally dividing the whole image depending on the no. of the characters in the CAPCTHA image provided by binary classifier(3 or 4). Segmentation steps for this approach are as follows:

- Background color is set to all white pixel value in the image. Pixel value of background is all set to 255 value to have better boundary distinction in the letter and noise.
- K-means Clustering with sufficient no. of clusters is done on image for color quantization.This is necessary to remove the irregularities in visually similar pixel values.
- Thresholding is done to apply morphological filters on the image. Dilation followed by erosion is applied to remove thin lines.
- Filtered mask is applied on the colored image.
- Further K-means clustering is used with 5 clusters on the masked image. This is followed by same morphological filters used above to remove noisy line as much as possible.
- Now all pixel values frequency is calculated in the image.
- Dominant color is filtered out on the basis of largest pixel value frequency among the 5 cluster center values.
- Finally we get image with only letter having similar pixel value and with almost all noisy lines removed.

**Limitations:** (a) When background and image character is of same colour, the whole character will be removed. (b) Also, in some cases it can happen that noise of some colour have more area than the character, which will cause the character to be removed.

(a) Same Color

(b) Extreme Noise




Dominant Color Segmentation

### 4.4 Training

- We have trained on each segmented character using a feed-forward CNN.
- The network has 2 convolutional layers where each convolutional layer is followed by a max pooling operation and ReLU activation function.
- Max Pool operation has a stride window of size 2 X 2.
- 3 fully connected layers are deployed after these convolution operations where the final layer is output layer.
- Output layer has nodes equal to number of classes and is used for prediction.
- Cross Entropy Loss function has been used.

### 4.5 Experimental Results:-

- Webmail($1^{st}$, Direct Method approach)

| | |
|---|---|
| Training Dataset Size | 4,000 |
| Test Dataset Size | 1,000 |
| Training accuracy | 96% |
| Testing accuracy | 94% |
| Entire Captcha Accuracy | 80.7% |

- Webmail($2^{nd}$, Boundary Segmentation approach)

| | |
|---|---|
| Training Dataset Size | 4,000 |
| Test Dataset Size | 1,000 |
| Training accuracy | 97% |
| Testing accuracy | 94% |

- Webmail($3^{rd}$, Dominant Color Segmentation approach )

| | |
|---|---|
| Training Dataset Size | 4,000 |
| Test Dataset Size | 1000 |
| Training accuracy | 96% |
| Testing accuracy | 91% |

# 5 Experimental details

## 5.1 Dataset generation and labelling

Images for the training as well as testing dataset were directly grabbed from the website. Initially, Labelling of images were done manually , but efforts were made to reduce manual labour and human involvement. Initially, a small set of images was generated and labelled manually. Those small set of images were used for training and learning a character recognition model. Further a larger set of data was generated and they were tested on previously generated model. This way labels of new set of data was obtained. But, since our previously learnt model was not accurate, there were lot of errors in new labels. It was corrected by manually correcting the imperfect data points. This process of bootstrapping was implemented for 2 stages. It is important to stress the fact that manual labor was used to correct mistakes in further labels but it was very less compared to labelling all the datasets manually. Since the model is trained on larger datasets in further stages, it generates pretty accurate models which can be used to generate larger datasets with minimal error in their labels.

## 5.2 Tuning

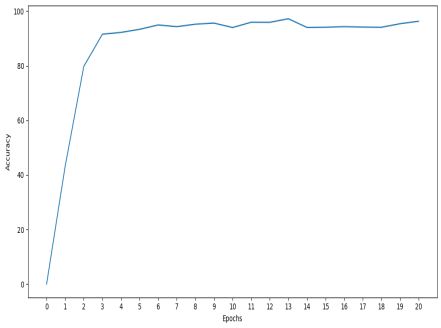
Tuning was done using 3000 training images and 1000 validation images.

### 5.2.1 Epochs

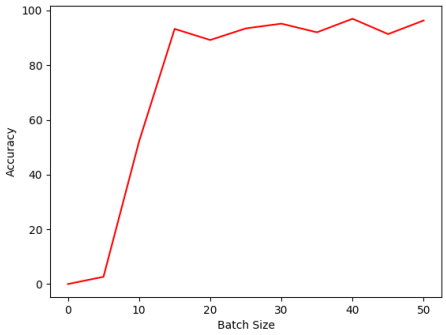
Epochs is the number of times dataset was repeated for training for better accuracy. The optimum value found was 13 epochs.

### 5.2.2 Batch Size

Batch Size is used in Stochastic Gradient Descent for optimizing the weights of CNN. Optimum Batch Size was set to be 25.




(a) Accuracy vs Batch Size




(b) Accuracy vs Batch Size

# 6 Novel contributions

For the case of webmail,we could have obtained higher accuracy using simple template matching approach since there is almost no visible rotation in the characters.
However,our approach for noise removal after segmentation doesn't assume anything regarding the orientation of the characters and hence is robust to rotations.

# 7 Future work

- Find a workaround for labelling dataset.
- Find a workaround for dealing with Captchas which may have one or more characters colored same as the background color.

# 8 References

[1] Chellapilla, K.,Simard, P.Y.,Mozer, M.C. (2004) Using machine learning to break visual human interaction proofs

[2] Lecun, Y.,Bottou, L.,Bengio, Y.,Haffner, P. (2004) Gradient-based learning applied to document recognition.Proceedings of the IEEE

[3] Goodfellow, I.J.,Bulatov, Y.,Ibarz, J.,Arnoud, S.,Shet, V. (2014) Multi-digit number recognition from street view imagery using deep convolutional neural networks.ICLR

[4] Stark, F.,Hazirbas, C.,Triebel, R.,Cremers, D. (2015)CAPTCHA Recognition with Active Deep Learning