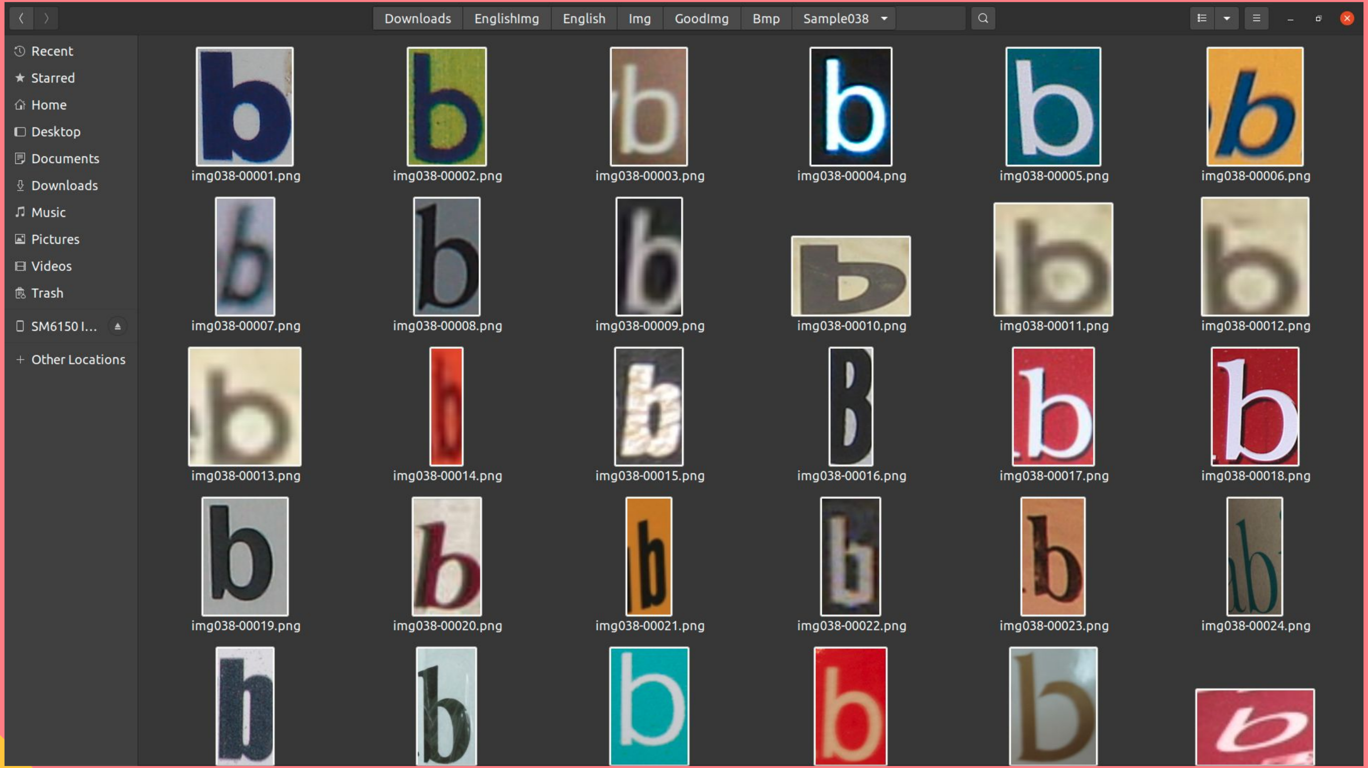
Solving CAPTCHAs using Optical Character Recognition

ES654 Machine Learning Project Presentation

- Preet Patel (17110112)
- Vivek Modi (18110190)
- Arun Shakya (17110025)
- Anupam Kumar (18110022)

Chars74K Dataset

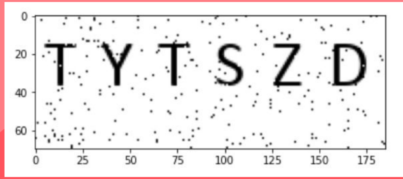


Datasets

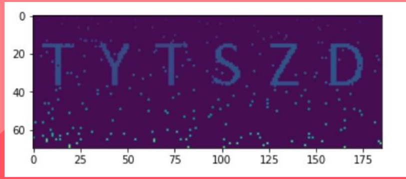
1. Single character dataset

2. Salt and pepper dataset

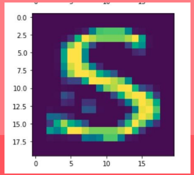
Basic Pipeline



Reading the images from dataset



Pre-processing (Thresholding)



Character Segmentation (Bounding boxes)

Training and Loses

Character Recognition and Prediction

Character Classification and Sorting

SVM Implementation

- ★ We have generated characters for inputs from captcha's.
- ★ We have currently used scikit learn to classify our model.
- ★ Implementation uses one to one coding design.
- ★ We will make two hyperplanes which will separate positive points with negative points and the Hinge loss between the planes will be
  - ○ Hingeloss =max(0,(1 −y_i * (w^T*x_i + b))/2
- ★ Then we will compute the gradient of cost function.

$$\nabla_w J(\mathbf{w}) = \frac{1}{N}\sum_i^n \begin{cases} \mathbf{w} & \text{if } \max(0,\ 1 - y_i * (\mathbf{w}\cdot x_i)) = 0 \\ \mathbf{w} - C y_i x_i & \text{otherwise} \end{cases}$$

- ★ We will then find the weights and bias for the given input images and target images.
- ★ We have also tried to implement SVM from scratch.

Fully-Connected NN Implementation

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 100)               40100
_________________________________________________________________
dense_1 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 26)                2626
=================================================================
Total params: 62,926
Trainable params: 62,926
Non-trainable params: 0
```

```
Epoch 10/10
21/21 [==============================] - 0s 12ms/step - loss: 0.0204 - accuracy: 0.9967
```

CNN Implementation

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_54 (Conv2D)           (None, 32, 32, 1)         26
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 16, 16, 1)         0
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 16, 16, 1)         2
_________________________________________________________________
up_sampling2d_18 (UpSampling (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_56 (Conv2D)           (None, 32, 32, 1)         2
_________________________________________________________________
flatten_18 (Flatten)         (None, 1024)              0
_________________________________________________________________
dense_108 (Dense)            (None, 1024)              1049600
_________________________________________________________________
dropout_90 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_109 (Dense)            (None, 512)               524800
_________________________________________________________________
dropout_91 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_110 (Dense)            (None, 256)               131328
_________________________________________________________________
dropout_92 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_111 (Dense)            (None, 128)               32896
_________________________________________________________________
dropout_93 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_112 (Dense)            (None, 64)                8256
_________________________________________________________________
dropout_94 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_113 (Dense)            (None, 26)                1690
=================================================================
Total params: 1,748,600
Trainable params: 1,748,600
Non-trainable params: 0
```

```
Epoch 10/10
42/42 [==============================] - 12s 297ms/step - loss: 0.0266 - accuracy: 0.9931
```

Results: SVM sklearn model

Salt-n-pepper

```
Total images processed: 10000
Accuracy:  60.08
WCITEB  :  WCTEB  :  False
```

Without noise

```
Total images processed: 10000
Accuracy:  73.99
PRQVTU  :  PRQVTU  :  True
```

Results: Fully-connected

Chars74k

	Pos	Neg	Acc
Char	43548	13407	76.460363
Captcha	3152	6848	31.520000

```
Current file:  10000
FXYQIR : PXYQR
```

Coloured & Noisy

	Pos	Neg	Acc
Char	48972	8368	85.406348
Captcha	6383	3617	63.830000

```
Current file:  10000
ZJPVCH : ZJPVCH
```

Results: CNN

	Pos	Neg	Acc
Char	4869	855	85.062893
Captcha	576	424	57.600000

```
Current file:  1000
RJDFMI : RJDFMI
```

	Pos	Neg	Acc
Char	48459	8881	84.511685
Captcha	5849	4151	58.490000

```
Current file:  10000
REWBDL : REWBDL
```

Insights & Inferences

- CNN requires more dissimilar Input data and computational power.
- Our simple dataset (without any noise) had poor performance efficiency as it is reaches its accuracy very quickly. Hence, we added colour and noise.
- Training dataset has variating:
  - Text & background color, rotation, transformation
  - Blur, noise, font
- Bias-Variance trade off helped us decide the variations required for generating our dataset.

Limitations & Future Prospects

- ★ Using our boxed segmentation method we could not recognize the anti-segmented captchas (captcha with connected characters) with good accuracy.
- ★ An End-to-end CNN based model could be developed to give better results.
  - Con: Computationally heavy

Thank you

👏🤝👏