CS229 Autumn 2017

CAPTCHA Breaking with Deep Learning

Nathan Zhao, Yi Liu, Yijun Jiang

Introduction

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is used to determine whether or not the user is a human, consisting of texts or images with distortions and noise. In this project, we aim to recognize letter CAPTCHAs using deep learning. With our own CAPTCHA dataset, we analyze supervised and unsupervised techniques in distorted letter recognition and cracking multiple-letter CAPTCHAs.

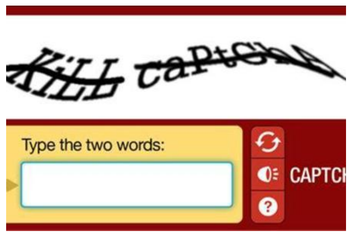

Fig.1 CAPTCHA application

Data

Fig.2 Single-letter CAPTCHAs of “A”

Fig.3 Four-letter CAPTCHA of “GCKD”

Single-letter dataset: 50,000 images generated from *PyCaptcha*

Multi-letter dataset: 50,000 two-letter images, 50,000 four-letter images generated from *PyCaptcha*

Generalization dataset: 1,580 images generated by different dictionaries. Used as a final metric to measure how well the convolutional neural net has learned each letter’s structural archetype.

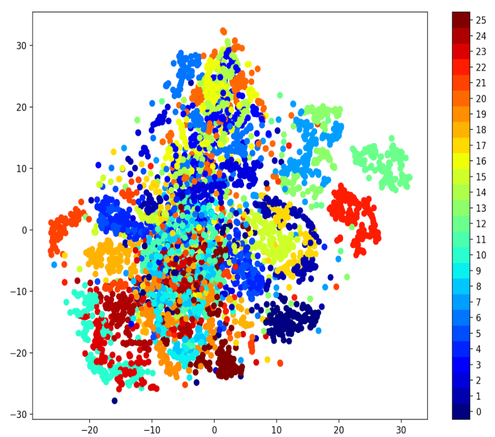

Fig. 5 Generalization dataset

Fig. 4: 2D visualization by *t-SNE* of how well labels can be separated in the captchas dataset. There are some classes that are clustered and well separated, but the majority are not.

Models

Single-letter CAPTCHA recognition:

- An SVM model was built using Python package *sklearn.svm*. L2 regularization and hinge loss function were used.
- A k-means algorithm with 26 centroids was implemented with Python package *sklearn.cluster*. As an unsupervised technique, k-means is a rough measure of how ‘difficult’ the captcha generator is to crack.
- A convolutional neural network (CNN) was built with the structure shown in Fig.6 using *Keras*.
- Also, VGG-19 pre-trained model was used for transfer learning.

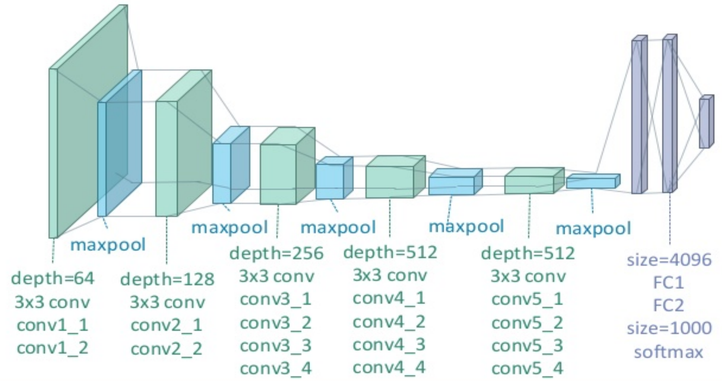

Fig.6 Visualization of the structure of CNN

Four-letter CAPTCHA recognition:

- Moving-windows algorithm: a CNN trained on single letter was used to scan the entire four-letter image by reading only a subset (a.k.a. the “window”) each time. This window was translated across the full image. From the predicted probabilities of each letter as a function of window position, we could extract the four-letter string.
- Multi-CNN algorithm: four single-letter CNNs were trained, each focusing on a single digit of the string (e.g. 1st CNN trained with 1st char as label). When testing, each CNN predicted a letter and all four letters were concatenated as the output string.

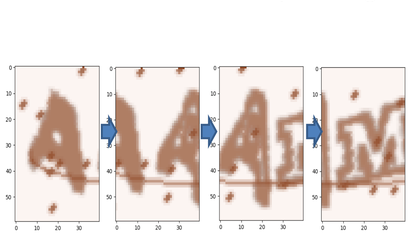

Fig. 7 Illustration of the “moving window” algorithm

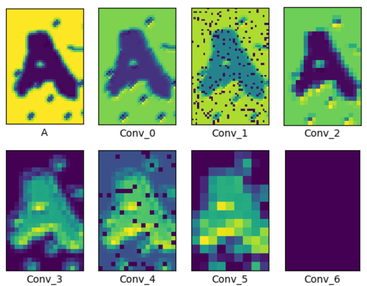

Fig.8 Filtered images by CNN

Results

Algorithm	Test Score	Prediction
Rbf-SVM (single letter)	69%	0.1%
26-means (single letter)	30%	--NA--
CNN (single letter)	99%	60%
CNN (VGG-19 Transfer, single letter)	95%	70%
CNN moving Window (4 letters)	38%	0.50%
Multi-CNN (4 letters)	76%	0.75%

CNN Errors: confusion matrix on the test data indicates that certain letters are susceptible to being misclassified as T, G or Z for example

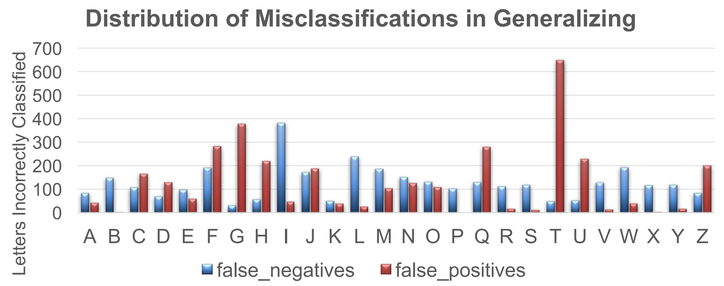

Fig.9 Confusion matrix of CNN

Discussion

For single-letter CAPTCHAs, CNN is superior than SVM and k-means since CNN is better at recognizing abstract features.
For multi-letter CAPTCHAs, the accuracy is no more than the fourth power of single-letter CNN accuracy. Besides, mistakes of judging the locations of each letter further decreases the accuracy by 50% per letter according to our estimation. These factors account for the low generalization accuracy.

Future Work

Multi-letter CAPTCHA recognition can be improved by:

- Object-in-scene recognition
- Recurrent neural networks

References

- *t-SNE* documentation: https://lvdmaaten.github.io/tsne/
- *sklearn* documentation: http://scikit-learn.org/stable/
- *Keras* documentation: https://keras.io