Final Project Report

Soft Engineering

Captcha Hunter

Submission Date:

7th January 2017

Group Members:

Name	Student ID
Group Leader – Lingkun Kong	5140219016
Zhenchao Zhou	5140219075
Ruiji Wang	5140219030

Written by: Lingkun Kong, Zhenchao Zhou, Ruiji Wang

Table of Contents

Group Members:

Name	Student ID	Tasks Assigned **
Lingkun Kong – Group Leader	5140219016	Understanding the given Research Paper Coding of the proposed idea Implementation Report Documentation Presentation
Zhenchao Zhou	5140219075	Coding of the proposed idea Report Documentation
Ruiji Wang	5140219030	Understanding the given Research Paper Coding of the proposed idea Implementation Report Documentation

****Note:**
Task assigned can be:

- Understanding the given Research Paper
- Coding of the proposed idea
- Implementation
- Report Documentation
- Presentation

1. Introduction

1.1. Purpose of the project

A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is an automated test to identify whether the user is a human or not. Captchas are often used on the internet to prevent automated programs from abusing online services. Nowadays, most service providers such as email or online shopping sites require users to solve captchas, which most often consist of some distorted text that must be read and typed in correctly. For humans this is a comparably simple task, but computers still have difficulties here. Useful chaptchas should be solvable by humans at least 80% of the times while programs using reasonable resources should succeed in less than 0.01% of the cases.

Recently, researchers have started investigating automated methods to solve capthas. The paper *Convolutional Neural Networks Applied to House Numbers Digit Classification* writen by *Pierre Sermanet, Soumith Chintala and Yann LeCun*, provides a method to classify digits of real-world house numbers using convolutional neural networks (ConvNets). ConvNets are hierarchical feature learning neural networks whose structure is biologically inspired. But to ricognize captchas containing both digits and characters, this method will failed.

Our ultimate goal is to implement an application by which we can recognize a certain sequence of digits and literals. We propse in the paper an approach that is based on ConvNets including Captcha Crawler and Generator to provide dataset of auto-generated captchas, Captcha Partitioner to do the segmetation and Captcha Recognizer finish the ultimate goal. In general, we propose a captcha solving technique which has a good classification accuracy.

1.2. Scope of the project

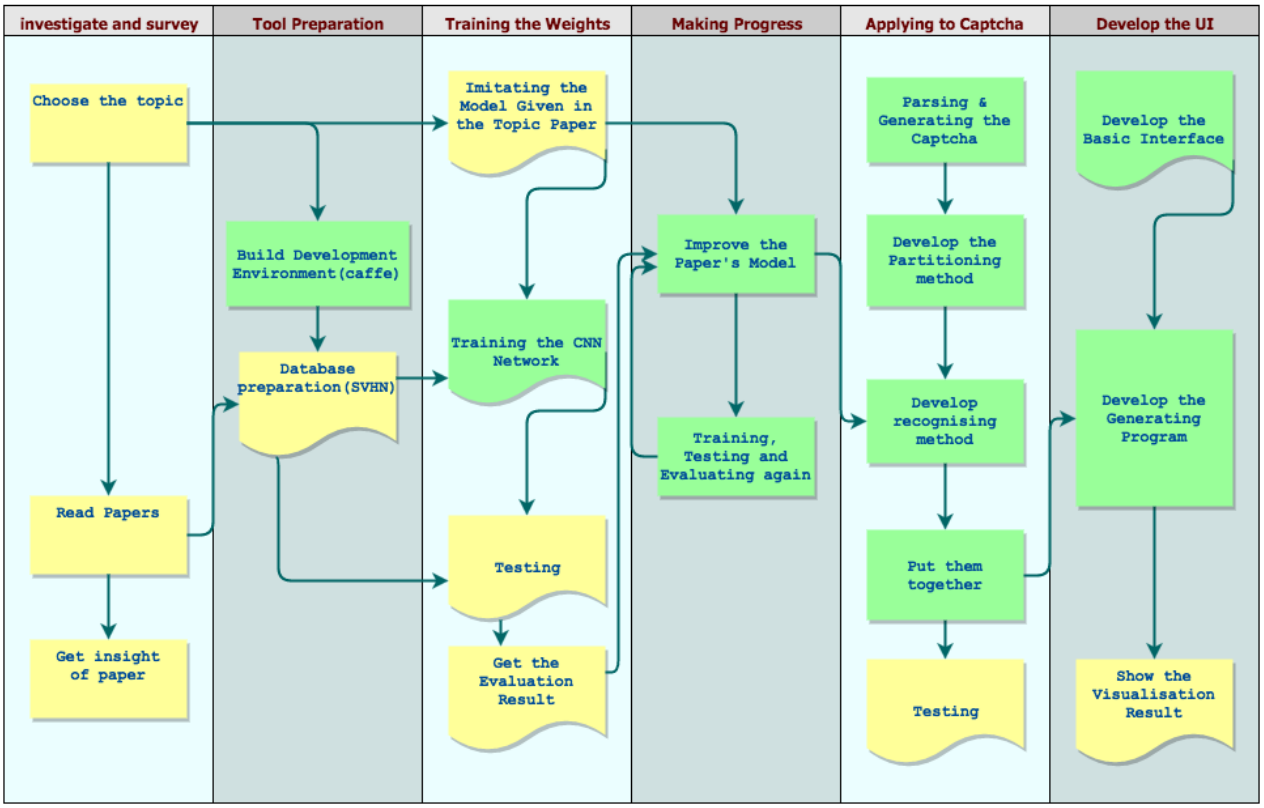


Fig1. Our scope of the project. The diagram in green means this part needs coding implement.

The Fig 1 shows the scope of our project.
Investigate and survey: Searched for relative materials and get insight of project at.
Tool Preparation: Build Development Environment & Prepared the dataset.
Training the Weights: Imitating the Model, Training, Testing and Evaluation
Making Progress: Training, Testing and Evaluation
Applying to Captcha: Generation & Partition & Recognition
Develop the UI: Development of UI and Combination

1.3. Overview of the document

In overview, we briefly introduced the structure of our final report by which we also present whole scope of our project.

First of all, it comes to 'requirements' part which is about our project's requirements. There are three main components in this part – functional requirements, non-functional requirements and domain requirements.

Secondly, we introduce software design, which including software model, software development tools, architectural design and object identification shows how we design our project's software.

After that, we illustrate our 'testing' part of our project. It covering test plan, test-design specification, test-case specification, test-procedure specification, largely strengthens our final products' robustness.

Finally, we draw our conclusion list related references at the end of our report.

2. Requirements Specification

2.1. Functional requirements

There are several functional requirements of our project:

a. Captcha Dataset Generating
Using CNN structure to solve classification problems requires large number of data. Thus, we use both crawling and generating approaches to build our own dataset which we call as Captcha dataset, including thousands of hundreds of captcha pictures.

b. Captcha Partitioning
In partitioning, we first denoise the input picture by specific trained parameters by filters of Gaussian distribution. And then, we use morphological methods offered by python's scipy library to partition the image and use the number of final separating parts as one of the important indicators which tells whether the partitioning success or not.

c. Captcha Recognizing
In this part, we update our CNN model from only able recognize digits to can handle sequence of digit and literals, which means we have to extend model's outputs numbers and re-train weights. We implement these part by using python's theano library which is developed for deep learning.

2.2. Non-functional requirements

The main non-function requirement is the UI design of our application. It is implemented in python.

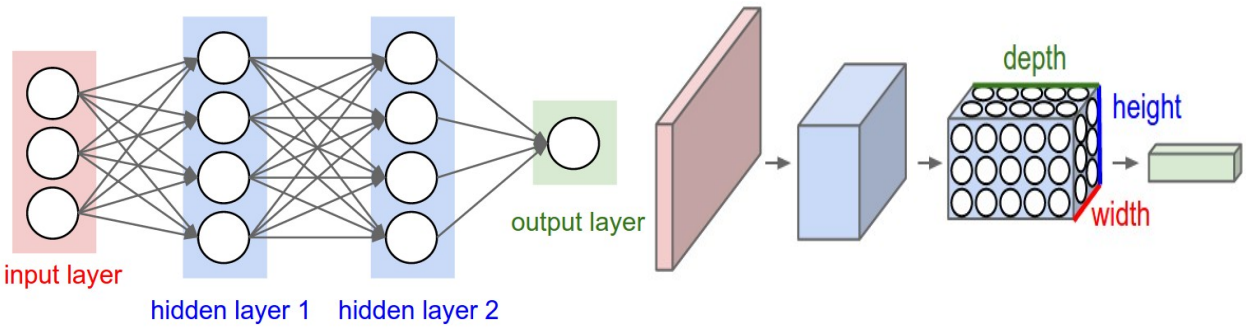
Besides, the visualization part about how our work works is also needed, since the presentation is also important.

2.3. Domain requirements

a. CNN model developing part: Ubuntu 14.04 + caffe + Anaconda + python2.7 + matlab r2016b
b. UI developing part: Ubuntu 14.04 + Anaconda + python3.5 + pyinstaller

3. Software Design

3.1. Software model



Left: A regular 3-layer Neural Network. **Right**: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

In machine learning, a convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex.

3.2. Software development tools

a. CNN model developing part: Ubuntu 14.04 + caffe + Anaconda + python2.7 + matlab r2016b
b. UI developing part: Ubuntu 14.04 + Anaconda + python3.5 + pyinstaller

3.3. Architectural Design

The ConvNet has 2 stages of feature extraction and a two-layer non-linear classifier:

1. The first convolution layer produces 16 features with 5x5 convolution filters.
2. The second convolution layer outputs 512 features with 7x7 filters.

The output to the classifier also includes inputs from the first layer, which provides local features/motifs to reinforce the global features.

The classifier is a 2-layer non-linear classifier with 20 hidden units. Hyper-parameters such as learning rate, regularization constant and learning rate decay were tuned on the validation set. We use stochastic gradient descent as our optimization method and shuffle our dataset after each training iteration.

3.4. Object Identification

1.Understand the paper and learn the related knowledge √

2.Apply ConvNets to house numbers digit classification and improve the accuracy √

3.Develop under Linux to make it more useful √

4. Testing

4.1. Test Plan

This project proposes a method for captcha classification which is modelled in convolutional neural networks(ConvNets). ConvNets are hierarchical feature learning neural networks whose structure is biologically inspired.

In our project, we will test the different part of Captcha Recognizer separately including CNN model, Captcha generation, Captcha partition, Captcha recognition.

Function	Input	Output
House Numbers Digit Classification	An image of digit house number	System will analyze the image and conduct the digit classification modelled in ConvNets which will output the digits in the picture.
Captcha Production	No Input	System will produce several captcha images including both digit numbers and characters. It will output these images.
Captcha Partition	An image of captcha	System will do the partition and then output several images. Each of these images contains one digits or character of the captcha.
Captcha Recognition	An image of digits or single characters	System will do the classification and then output correct results or error messages if fail to classify.

Table 1: Test plan and specification of project.

4.2. Test-design specification

Our test-design is vividly presented in Table 1.

4.3. Test-case Specification

4.3.1 House Numbers Digit Classification

Function	Human Action	Data	Expect Result	Actual Result
House Numbers Digit Classification	Input an image	JPEG image	System correctly classify and output the result	Classify successfully with the the rate of 98%

This function works **perfect.** Our correct rate reaches **98%** , even **better** than the rate (which is 95.1%) in the paper *Convolutional Neural Networks Applied to House Numbers Digit Classification*. And 98% almost equals to human performance.

4.3.2 Captcha Production

Function	Human Action	Data	Expect Result	Actual Result
Captcha Production	Produce captcha	JPEG	Images of captcha contains	Fit

	automatically	image	numbers and characters	expectation

This function aims to produce images of captcha which is not too complicated and difficult to classify. Each images contains several digits and single characters with interfering lines and noise.

4.3.3 Captcha Partition

Function	Human Action	Data	Expect Result	Actual Result
Captcha Partition	Input an image	JPEG image	Divide captcha into individual image. Each image contains individual digit or character	Fit expectation

This function is supposed to cut captcha into several pieces of which contains individual digit or character. This partition part has a correct rate above **85%**.

4.3.4 Captcha Classification

Function	Human Action	Data	Expect Result	Actual Result
Captcha Classification	Input an image	JPEG image	Excuting the image contains a digit and a character and output the result	Fit expectation

This function aims to classify the input image and output the result. If the partition part works well, correct rate of this part can reaches **99%!**

4.4. Test-procedure specification

4.4.1 Capability

The Captcha Classification System have follow functions: Produce captcha, conduct the partition and classification and then output the result.

4.4.2 Steps

- Log
- Set-up
- Start
- Input
- Measure
- Output
- Restart
- Stop
- Wrap-up

4.4.3 Flaws and Limitations

4.4.3.1 Some existing problems

a) Capability
The Captcha Classification System can't conduct the classification of captcha images that are too complicated.

b) Interface
The User Interface is too simple and crude. And it's only available on LINUX operating system.

c) The correct rate of partition
Owing to lack of time and experience, our parameters are not the optimal ones. As a consequence, the correct rate of partition is not optimal.

4.4.3.2 Some unrealized functions

a) Find the best parameter automatically.

This function is concerned about graphics and machine learning. Due to the complicated design of algorithms and limited time, we kindly make this function unchanged.

4.5. Accuracy Final Result

To make it easier to understand the different accuracy between several algorithms, we draw 2 tables to show the contrast between them and accuracy of different partition stages respectively.

Algorithm	Test Accuracy
Binary Features (WDCH)	63.3%
HOG	85.0%
Stacked Sparse Auto-Encoders	89.7 %
K-Means	90.6%
ConvNet / MS / Average Pooling	90.94%
ConvNet / MS / L2 / Smaller training	91.55%
ConvNet / SS / L2	94.46%
ConvNet / MS / L2	94.64%
ConvNet / MS / L12	94.89%
ConvNet / MS / L4	94.97%
ConvNet / MS / L4 / Padded	95.10%
ConvNet / CH / Smaller training	**93.40%**
ConvNet / CH / Fully training	**98.12%**
ConvNet / CH / Using captcha dataset	**99.07%**
Human Performance	98.0%

Table 1. Performance reported by using our own algorithm in the project of CaptchaHunter(CH) with the improvement of accuracy from originally 95.10% which is proposed by the original paper

to fully-trained 98.12% and finally 99.07% by using our captcha dataset, which is even superior to human's eyes! (The last three lines in bold represent our own algorithms.)

Stage	1st Partition	2nd Partition	3rd Partition
Test Accuracy	51.7%	70.4%	85.0%

Table 2. Performance is reported by different stages of partition in the project of CaptchaHunter(CH) with the improvement of accuracy from originally 51.7% to finally 85.0%. The 2nd partition modifies some details in noise processing. The 3rd partition changes an important part in cutting off images.

5. Conclusion

In conclusion, firstly, we advance the convolutional neural networks model based on the model in the proposed paper – significantly improving the accuracy of the real world housing digit prediction from 95% to 98%. Secondly, we develop approaches for more complicated recognition which targets the real world captcha picture, including both digits and letters. Finally, we construct proper User Interface for captcha recognition, and we also accomplish the visualization of our approach.

The accuracy of recognizing the separated captcha element is amazingly high – reaching 99%. However, there is still room for our partition method to progress as our successful partitioning rate is about 85%.

In the future, we hope to enhance our algorithms to make the accuracy up to 99.5% and what is more, we hope to modify our algorithm to let itself find the best parameter automatically.

6. References

[1] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in vision algorithms. In *Proc. International Conference on Machine learning*, 2010.
[2] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. A committee of neural networks for traffic sign classification. In *International Joint Conference on Neural Networks*, pages 1918–1921, 2011.
[3] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.
[4] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *Neural Networks, IEEE Transactions on*, 21(10):1610 –1623, 2010.
[5] A. Hyvrinen and U. Kster. Complex cell pooling and the statistics of natural images. In *Computation in Neural Systems,*, 2005.
[6] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *Proc. International Conference on Computer Vision and Pattern Recognition*. IEEE, 2009.
[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
[8] Y. Lecun and C. Cortes. The MNIST database of handwritten digits.
[9] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
[10] P. Sermanet, K. Kavukcuoglu, and Y. LeCun. Traffic signs and pedestrians vision with multi-scale convolutional networks. In *Snowbird Machine Learning Workshop, 2011*.
[11] P. Sermanet, K. Kavukcuoglu, and Y. LeCun. Eblearn: Open-source energy-based learning in c++. In *Proc. International Conference on Tools with Artificial Intelligence*. IEEE, 2009.
[12] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of International Joint Conference on Neural Networks*, 2011.
[13] E. P. Simoncelli and D. J. Heeger. A model of neuronal responses in visual area mt, 1997.
[14] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.
[15] T. Yamaguchi, Y. Nakano, M.Maruyama, H.Miyao, and T. Hananoi. Digit classification on signboards for telephone number recognition. In *ICDAR*, pages 359–363, 2003.
[16] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition*, 2009.