Final Project Report

Soft Engineering

Captcha Hunter

Submission Date:

7th January 2017

Group Members:

Name	Student ID
Name 1 – Lingkun Kong	5140219016
Name 2 – Zhenchao Zhou	5140219075
Name 3 – Ruiji Wang	5140219030

Table of Contents

Table of Contents .....Preii
Group Members .....Preiii
1. Introduction i
   1.1. Purpose of the project ii
   1.2. Scope of the project ii
   1.3. Overview of the document ii
2. Requirements Specification iii
   2.1. Functional requirements iii
   2.2. Non-functional requirements iii
   2.3. Domain requirements iii
   2.4. Extra details of system requirements iv
3. Software Design viii
   3.1. Software model viii
   3.2. Software deveopment tools vii
   3.3. Archetectural design ix
   3.4.Object identification ix
   3.5. Extra details of software architecture ix
4. Testing xiv
   4.1. Test plan xiv
   4.2. Test-design specifiction xiv
   4.3. Test-case specification xiv
   4.4. Test-procedure specification xv
   4.5. Accuracy final result xvi
5. Conclusion xvii
6. References xviii
7. Use-Case Specification xix
   7.1. Use-Case Specification Error xix
   7.2 Use-Case Specification Image Adjust xx
   7.3 Use-Case Specification Image Browse xxi
   7.4 Use-Case Specification Image Estimate xxii
   7.5 Use-Case Specification Image Input xxiii
   7.6 Use-Case Specification Image Output xxiv
   7.7 Use-Case Specification Parameter Estimate xxv
   7.8 Use-Case Specification Parameter Input xxvi

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	Understanding the given Research Paper Coding of the proposed idea Implementation 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

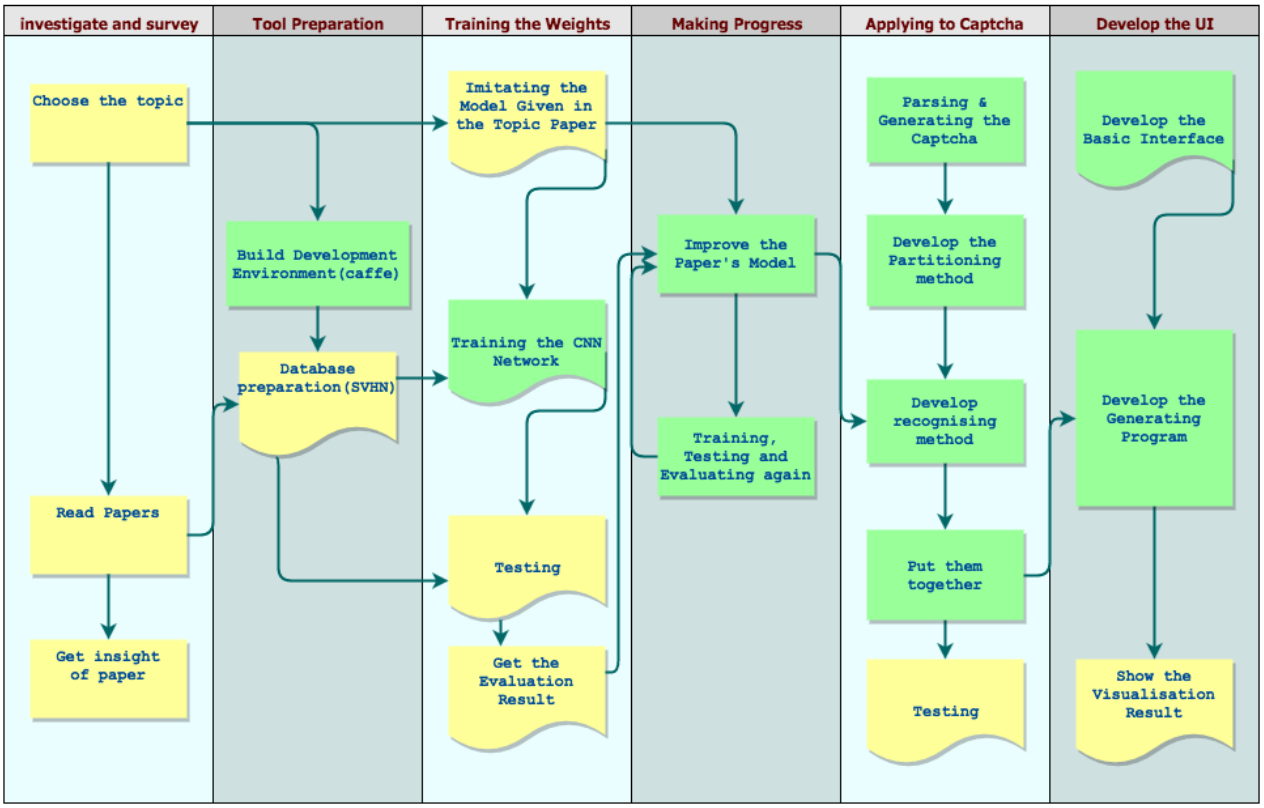


Fig1. Our scope of the project. The diagram in green means this part needs coding implement.

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 Crawler and Generator
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 Partitioner
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 Recognizer
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

2.4. Extra details of system requirements

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Finish the whole document	Whole team

Key Word

CNN
Functional
Non-functional

Abstract

This document is important for our Software Engineering project, since it will define most of the requirement for this system. We describe not only functional but non-functional requirement as well. And give a specific hardware and software environment, and other necessary information for our system.

2.4.1 Introduction

2.4.1.1 Purpose

This is a requirement specification document. In this document, we will define most of the system requirement, so that all the develop team member can have a clear picture of the whole system. We define not only the functional requirement, but the non-functional requirement as well. This is the main document of this define phase, in this phase, we also have glossary document, use case specification document and we offer a prototype.

2.4.1.2 Definition

CNN: Convolutional Neural Networks
Bom: bill of material
Functional: some requirement that need to be realized
Non-functional: some requirement that cannot be realized but is indispensable to our system

2.4.1.3 Reference

CS231n: Convolutional Neural Networks for Visual Recognition— Stanford Vision Lab
Convolutional Neural Networks Applied to House Numbers Digit Classification—Pierre Sermanet, Soumith Chintala and Yann LeCun

2.4.2 System Overvie

Main features of CNN:

a) Lp-looping
b) Multi-stage features

2.4.3 Detailed System Requirements

2.4.3.1 Functional Requirements

2.4.3.1.1 Functional/Behavioral

There are several functional requirements of our project:

1. captcha crawler

It can crawl enough captcha pictures from the Internet, as you know, our project requires a big dataset consisting of at least 5000 members.

2. captcha partitioner

After getting the dataset, we need to partition each digit or literal from the other respectively, in order to make it easier for our program to conduct the final recognition stage.

3. captcha recognizer

In this part, the main method is CNN network.

2.4.3.1.2 Non-Functional

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

2.4.3.2 Integration Requirements

Null.

2.4.3.3 Architecture Requirements

2.4.3.3.1 Computing Platforms

CPU: i7 2.3Ghz or better
Main memory:
Minimize: 512MB recommendation: 2GB
GPU：optional

2.4.3.3.2 Environment Requirements

Ubuntu 14.04 + caffe + Anaconda + python2.7 + matlab r2016b

2.4.3.4 User Interface Requirements

Details in Readme.

2.4.4 Requirements Analysis Models

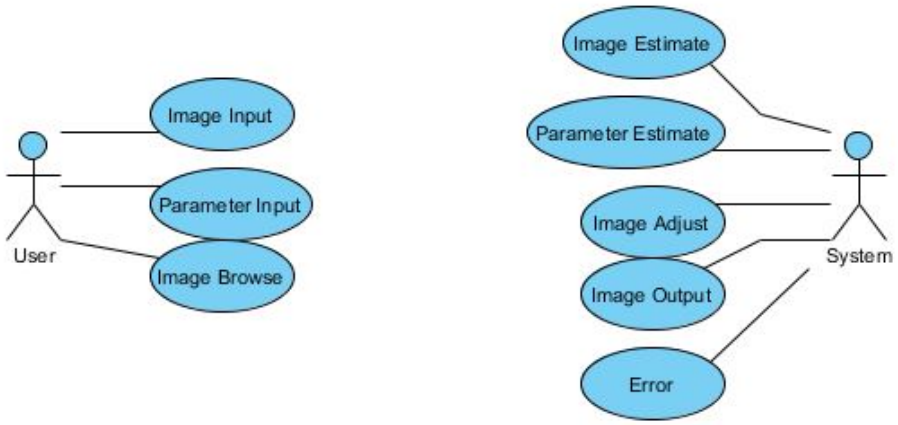
2.4.4.1 Functionality (behavioral)

2.4.4.1.1 Use Cases

This system is divided into eight use cases: Image Input, Image Estimate, Parameter Input, Parameter Estimate, Image Adjust, Image Output, Image Browse, Error.
Each use case is specified in the use case specification document. Please refer to each use case specification document to get detail information to know more about each use case.

2.4.4.1.2 Functional Analysis Model

We use Rational Rose to create the use case model. Here is the view of use case model.



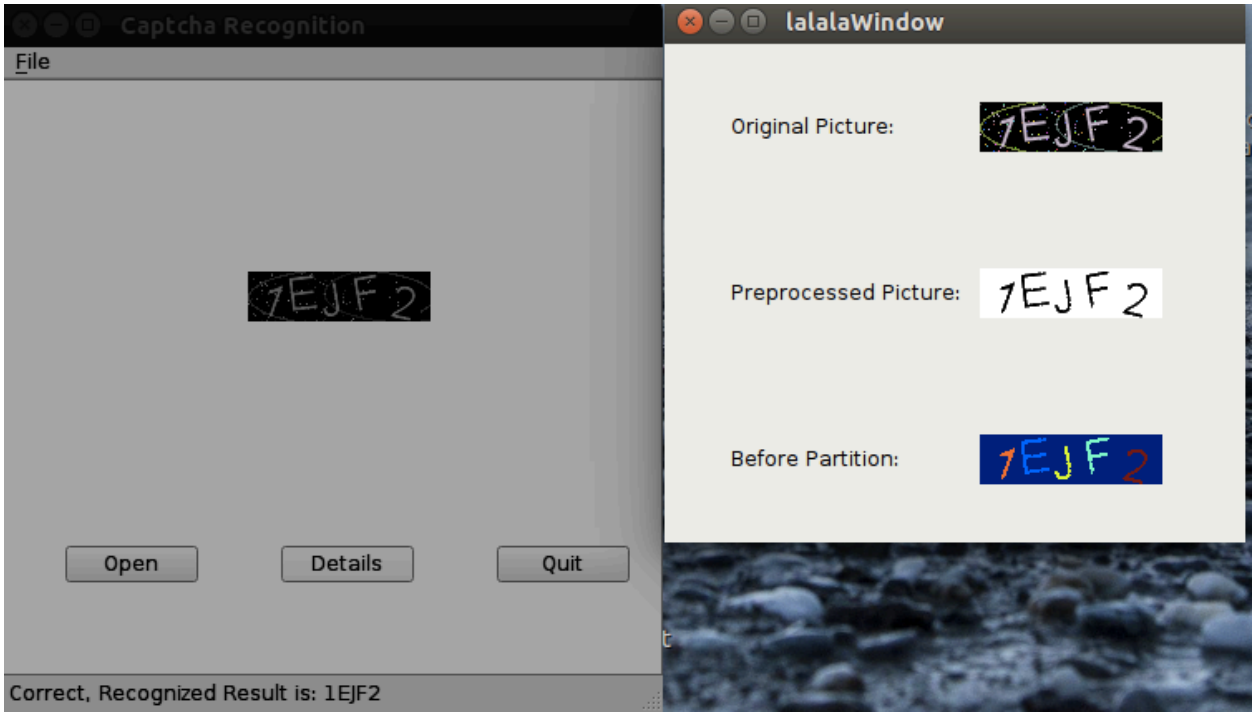
2.4.4.2 Architecture & Bill of Materials

Seq	tem	Qty	Functional Description	Vendor/ Manufacturer	Product Name	Version	Part No.	Purchase Reuse Upgrade	Standard (Y/N)	CR #*	Remark
1	System	High	Provide database service, response all the request from the clients	ASUS, China	Laptop	1.0	10001	R	N	2	
2	Operation System	High	Platform to develop system	Microsoft, America	Windows 7	7 Professional	10002	U	N	3	
3	UML tools	Middle	Use to design system	IBM Rational, America	Rational Rose	2003	10003	R	Y	2	

4	Develop Environment	Middle	Integrated develop environment to develop system.	MathWorks, America	Matlab	9.0	10004	R	Y	4	
5	Office Tools	Middle	To edit documents, table, etc.	Kingsoft, China	WPS	2013 Edition	10005	R	Y	1	

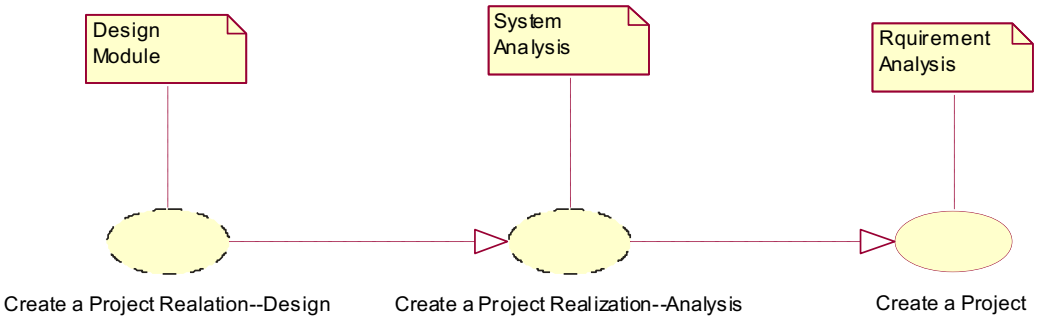
* If not a standard, indicates Standards Change Request #

2.4.4.3 User Interface



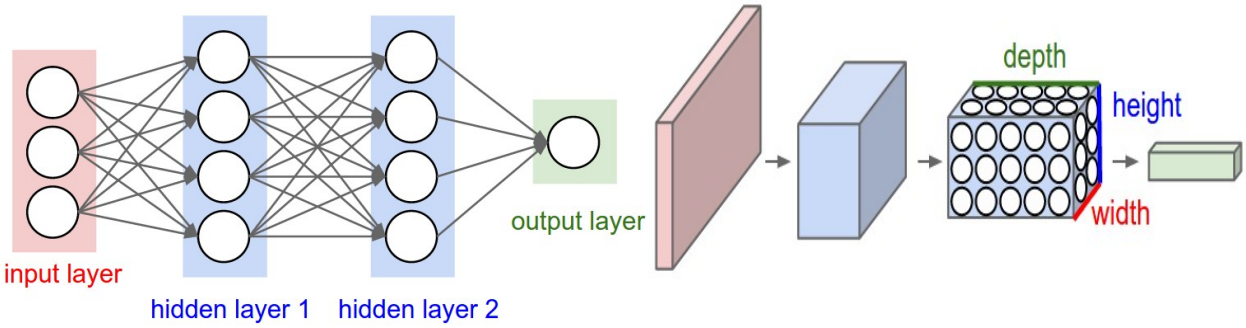
2.4.5 Requirements Traceability

In phase of BTM developing, we will make three modules: Use-case Module, Analysis Module, and Design Module, so there exists the traceability among those modules. In UML, we describe this in the following ways:



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 √

3.5. Extra details of software architecture

Revision History

Date	Version	Description	Author
2016-12-23	<1.0>	Final edition	The whole team

Key Word

CNN
Architecture
Design

Digest

This document is to describe the software architecture of CNN. It is the guideline of our CNN developing. In this document , we describe the detailed design idea and reader can have a clear picture with referring to the rational rose design model.

3.5.1 Introduction

3.5.1.1 Purpose

This document provides a comprehensive architectural overview of the Convolutional Neural Networks, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions, which have been made on the system.

3.5.1.2 Scope

This document should be an overview of the architecture and how it should be modeled. Decisions in this document affect how the system is modeled.

3.5.1.3 References

a) Convolutional Neural Networks Applied to House Numbers Digit Classification, Written by Pierre Sermanet, Soumith Chintala and Yann LeCun
b) UseCaseSpecification_Image Input.doc
c) UseCaseSpecification_Image Estimate.doc
d) UseCaseSpecification_Parameter Input.doc
e) UseCaseSpecification_Parameter Estimate.doc
f) UseCaseSpecification_Image Output.doc
g) UseCaseSpecification_Image Browse.doc
h) UseCaseSpecification_Error.doc
i) SystemRequirementsDocument.doc

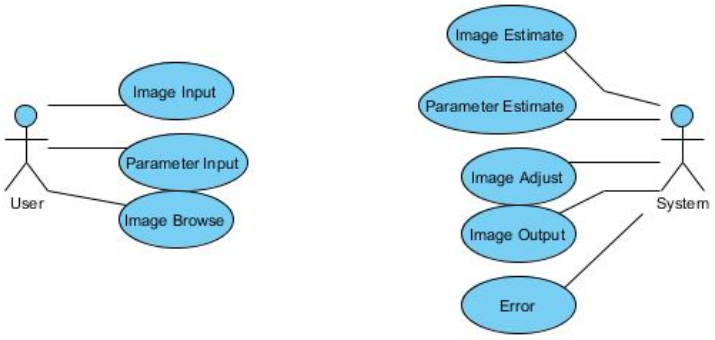
3.5.2 Architectural Representation

The architecture of the application is represented following the recommendations of the Rational Unified Process and the Rational Architecture Practice guidelines. And this document presents the architectural as a series of views:

a) Use Case View
b) Logical View
c) Process View
d) Implement View
e) Deploy View

3.5.3 Use-Case View

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. The functionality of CNN is captured in the use-case diagram below:



All implemented use cases have an associated Use Case Specification document. References to these documents can be found in the same directory.

3.5.3.1 Architecturally-Significant Use Cases

The architecturally-significant use cases are those, that "exercise" the most critical parts of the system architecture and demonstrate the core system functionality. In this system they are:
Image Input: This use case allows user to input an image. When a develop group apply for using this system to get Loop stage images, this use case starts.
Basic Scenarios:
a) User uploads an Loop stage image.
b) System receives the image and estimates if the image is an image.

Parameter Input: This use case allows user to input parameters for the input image.
Basic Scenarios:
a) The user inputs the parameters.
b) The software receives the parameters.
Image Output: This use case allows the software to generate LDR images and show on the output device.
Basic Scenarios:
a) User uploads an image.
b) Software estimates if the image is Loop stage.
c) User input the parameter(parameters).
d) Software output an final image.
Actually, you can reference to the sequence diagrams and collaboration diagrams of these very important use cases in the analysis model.

3.5.4 Logical View

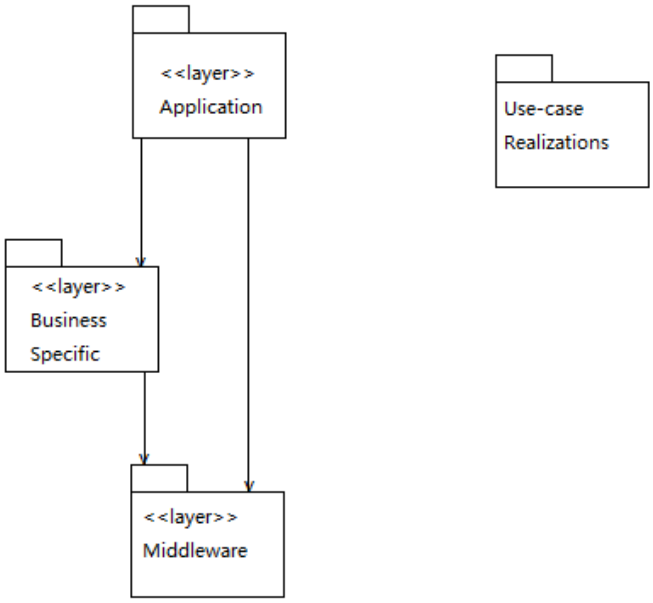
This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. It describes the logical structure of the system. It starts from the overview of the architecture and then presents its key structure, behavioral elements and mechanisms.

3.5.4.1 Overview

There are three dominant structures in the application design model:

a) Logical decomposition of the system into three layers.
b) The structure of the use case realizations derived from model templates of architectural mechanisms.
c) The design mechanisms package contains the a pre-designed solution to a common problem.

The high-level diagram of above is showed below:



The three layers are introduced below:

a) Middleware: The Caffe environment we need to use in this system.
b) Business Specific: Including business components and one common elements and services component. The components in this layer have a high reusability.
c) Application: This layer aim at special logic. It has a close relation to the presentation logic. The boundary classes are contained in this layer.

3.5.4.2 Architecturally-Significant Model Elements:

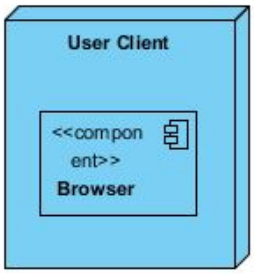
Application:
Captcha Recognition User Interface
Business Specific:
Ubuntu 14.04
Anaconda
python2.7
matlab r2016b
Middleware:
Caffe environment

3.5.5 Process View

There is only one process in our software, so there is no need for additional deployment.

3.5.6 Deployment View

The deployment view of a system shows the physical nodes on which the system executes and the assignment of the system processed to the nodes. The diagram below shows the most typical deployment configuration used by the development team.



3.5.7 Implementation View

Organized by IDE , in this system, we use Ubuntu 14.04 + caffe + Anaconda + python2.7 + matlab r2016b.

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.

7. Use-case Specification

7.1. Use-Case Specification Error

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Error	ZhenChao Zhou

7.1.1 Definition

This is the requirement description for the Error use case. Error use case is for system to throw an error message.

7.1.2 Preconditions

a)The image is not CNN.
b)The parameters are illegal.
c)The software is not successfully executed.

7.1.3 Post Conditions

The software goes to homepage.

7.1.4 Scenarios

1) The system throws an error message.
2) The system goes to homepage.

7.1.5 Exceptions or Branches

The system will give a warning and the user should try again.

7.1.6 Note

Null.

7.2. Use-Case Specification Image Adjust

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Image Adjust	ZhenChao Zhou

7.2.1 Definition

This is the requirement description for the Image Adjust use case. Image Adjust use case is for system to adjust the Lp-looping through CNN.

7.2.2 Preconditions

The system outputs an digit after user inputs an image.

7.2.3 Post Conditions

The Lp-looping is adjusted by CNN.

7.2.4 Scenarios

1) An Lp-looping produced by CNN be delivered.
2) The Lp-looping is adjusted by CNN.
3) The adjusted image is displayed on the screen.

7.2.5 Exceptions or Branches

Null.

7.2.6 Note

Null.

7.3. Use-Case Specification Image Browse

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Image Browse	ZhenChao Zhou

7.3.1 Definition

This is the requirement description for the Image Browse use case. Image Input use case is for user to browse Lp-looping images generated from the input CNN image.

7.3.2 Preconditions

The system has output an adjusted Lp-looping image.

7.3.3 Post Conditions

User returns to homepage and chooses the next action.

7.3.4 Scenarios

1)The user gets the Lp-looping images from system.
2)System returns to homepage and wait for user's next command.

7.3.5 Exceptions or Branches

When the Lp-looping image does not satisfy the user, user can choose to change the parameter and try again.

7.3.6 Note

Null.

7.4. Use-Case Specification Image Estimate

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Image Estimate	ZhenChao Zhou

7.4.1 Definition

This is the requirement description for the Image Estimate use case. Image Estimate use case is for system to estimate if the image's format is correct.

7.4.2 Preconditions

The user press the Load Image button an upload something.

7.4.3 Post Conditions

The system will exam if it is an image.

7.4.4 Scenarios

a)User press the Load Image button.
b)User input a file.
c)The software judges if it is an image.

7.4.5 Exceptions or Branches

7.4.5.1 The input is an image

The Image Output use case is executed.

7.4.5.2 The input is not an image

The Error use case is executed.

7.4.6 Note

Refer to the Use Case Model.

7.5. Use-Case Specification Image Input

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Image Input	ZhenChao Zhou

7.5.1 Definition

This is the requirement description for the Image Input use case. Image Input use case is for user to load captcha images.

7.5.2 Preconditions

The user has opened this software.

7.5.3 Post Conditions

When the user gets the captcha image, this use case is over.

7.5.4 Scenarios

1) User press on the Load Image button.
2) User uploads an captcha image.
3) System receives the image.

7.5.5 Exceptions or Branches

System receives the image and estimates if the image is a captcha one. If the image is not a captcha image, system will return an error information and ask user to upload another image.

7.5.6 Note

Null.

7.6. Use-Case Specification Image Output

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Image Output	ZhenChao Zhou

7.6.1 Definition

This is the requirement description for the Image Output use case. Image Output use case is for system to generate captcha images.

7.6.2 Preconditions

7.6.2.1 The input must be an image

The user must upload an image then software will estimate it. If not, the software will report an error.

To see more details, refer to Use Case Image Estimate.

7.6.2.2 The user must have chosen the parameters

The user must choose or type in an parameter so that the software can start to work. If not, the software will report an error.

To see more details, refer to Use Case Parameter Estimate.

7.6.3 Post Conditions

Captcha images are generated and use case Image Adjust executes.

7.6.4 Scenarios

a)System generates a captcha image by Lp-looping.
b)The captcha waits for adjustment.

7.6.5 Exceptions or Branches

7.6.5.1 The input is not an image

When the input is estimated as not an image, an error message will be presented on the output device.

7.6.5.2 Wrong parameters

If the parameters are illegal, the software will report an error message.

7.6.5.3 The image is too large

If the image is too large, the software will report an error message.

7.6.6 Note

Null.

7.7. Use-Case Specification Parameter Estimate

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Parameter Estimate	ZhenChao Zhou

7.7.1 Definition

This is the requirement description for the Parameter Estimate use case. Parameter Estimate use case is for system to estimate if the parameters are illegal.

7.7.2 Preconditions

The user has input parameters.

7.7.3 Post Conditions

If the use case was successful, the software will start and output an LDR image.

7.7.4 Scenarios

This use case starts when a user has input the parameters.

a) If the parameters are illegal, the software throws an error message.
b) Else the use case Image Output is executed.

7.7.5 Exceptions or Branches

7.7.5.1 The parameters are illegal

a) The parameters are too large or too small, the software throws an error message.
b) The count of parameters are not correct, throw an error message.

7.7.5.2 The parameters are acceptable

c) The Image Output is executed.

7.7.6 Note

Refer to the use case module

7.8. Use-Case Specification Parameter Input

Revision History

Date	Version	Description	Author
2016-12-23	1.0	Use case specification of Parameter Input	ZhenChao Zhou

7.8.1 Definition

Parameter Input use case is for user to input parameter(parameters) for the captcha images.

7.8.2 Precondition:

If the image has not been uploaded, this use case can't start.

7.8.3 Post Condition:

Software will estimate if the parameters are correct.

7.8.4 Scenarios

a) The user inputs the parameters.
b) The software receives the parameters.

7.8.5 Exceptions or Branches

If the parameters are illegal,Error use case is executed and the software will and go to 3.a.

7.8.6 Note

Null.