

Automatic Test Grading Using Image Processing and Machine Learning Techniques



Andries Petrus Smit
18183085

Report submitted in partial fulfilment of the requirements of the module
Project (E) 448 for the degree Baccalaureus in Engineering in the
Department of Electrical and Electronic Engineering at the University of
Stellenbosch

STUDY LEADER: Prof. J.A. du Preez

DATE: October 2017

Acknowledgements

I would like to acknowledge my study leader, Prof. J.A. du Preez, my family and the engineering class of 2017 for their kind contribution to this project. I would especially like to give thanks to the Applied Mathematics Department of Stellenbosch University for providing the opportunity and data needed to complete this project.



Plagiaatverklaring / Plagiarism Declaration

- 1 Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
- 2 Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.
- 3 Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.
- 4 Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelike aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
- 5 Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Studentenommer / Student number	Handtekening / Signature
Voorletters en van / Initials and surname	Datum / Date

Abstract

The aim of this research project is to develop software which automatically grades tests, which are written by students on a special template. The standard method used in these Optical Mark Recognition (OMR) software, is to allow a learner to specify answers by colouring in bubble grids. To correct a mistake, the incorrect answer's bubbles must first be erased. This is time consuming and increases the probability that a learner might colour in the bubbles incorrectly. This research project tries to solve this problem by implementing additional software that eases the use of such a template. Using computer vision and two machine learning techniques, namely Neural Networks (NN) and Probabilistic Graphical Models (PGM), a student can now erase answers by crossing bubbles out and write answers using characters. The software developed in this project thus improves ease of use, compared to other OMR software, while maintaining a high level of grading accuracy.

Uittreksel

Die doel van hierdie navorsingsprojek is om sagteware te ontwikkel wat automaties toetse, wat deur studente geskryf is, na te sien. Die standaardmetode wat in hierdie Optiesemerk-leser sagteware gebruik word, is om van inkleur roosterborrels gebruik te maak. Om 'n fout wat die student gemaak het te korrigeer, moet die ou ingekleurde borrels eers uitgevee word. Hierdie proses is tydrowend en verhoog die kans dat 'n student die borrels verkeerd invul. Hierdie navorsingsprojek poog om dié probleem op te los deur addisionele sagteware te implementeer, wat die gebruik van so 'n templaar vergemaklik. Met behulp van rekenaarvisie en twee masjienleertegnieke, naamlik Neurale Netwerke en Probabilistiese Grafiese Modelle, kan 'n student nou vroe beantwoord deur ingekleurde borrels uit te veer, asook antwoorde in karakters te skryf. Die sagteware wat in die projek ontwikkel word, maak die tipe toetse makliker om te gebruik, terwyl 'n hoër nasien akuraatheid in toetsresultate verkry word.

Contents

Abstract	iii
Uittreksel	iv
List of Figures	x
List of Tables	xi
Nomenclature	xiv
1 Introduction	1
1.1 Problem background	1
1.2 Problem statement	2
1.3 Project scope and assumptions	2
1.4 Project objectives	3
1.5 Research methodology	4
1.6 Graphical overview of system	5
2 Literature study	6
2.1 Existing Optical Marker Recognition techniques	6
2.1.1 Standard Optical Marker Recognition systems	6
2.1.1.1 Finding the template	8
2.1.1.2 Processing a bubble	8
2.2 Optical character recognition	9
2.2.1 Probabilistic approach	10
2.3 Conclusion: System requirements	10

3	Image processing	11
3.1	Orientation detection	11
3.1.1	Initial filtering and orientation detection	12
3.1.2	Radon transform	14
3.1.3	Finding the template	15
3.2	Bubble detection and processing	16
3.3	Data processing and grading	17
3.4	Conclusion	18
 4	 Machine learning approach	 19
4.1	Character recognition using a neural network	19
4.1.1	Preprocessing and creating digit images	20
4.1.2	Classification of digits	23
4.1.2.1	The Neural Network basics	24
4.1.2.2	The artificial neuron	24
4.1.2.3	Generating an output from the network	25
4.1.2.4	Deep Convolutional Neural Network	25
4.1.2.5	Training of the neural network	26
4.2	Probabilistic Graphical Models	26
4.2.1	Overview of the system	27
4.2.2	Estimating the intended digit	27
4.2.3	Estimating the student answer	28
4.2.4	Estimating the student number	29
4.2.5	Training of a Probabilistic Graphical Model	30
4.3	Conclusion	30
 5	 Analysis of results	 31
5.1	Results of 25 test cases	31
5.1.1	Basic system	33
5.1.1.1	Clash list	33
5.1.1.2	Incorrect automatic graded results	34
5.1.2	Complete system	34
5.1.2.1	Clash list	34
5.1.2.2	Incorrect automatic graded results	35

5.1.3	Analysis of results	35
5.2	Grading of tutorial tests	36
5.2.1	Marking statistics	36
5.2.2	Clash list	36
5.2.3	Incorrect automatic graded results	36
5.2.4	Conclusion	36
6	Summary and conclusions	39
6.1	Project summary	39
6.2	How this final year project benefits society	39
6.3	What the student learned	40
6.4	Future improvements	40
6.5	Summary and conclusions	40
	References	41
	A Project plan	42
	B Outcome compliance	44
	C Mathematical and graphical description of system	47
C.1	High-level overview	47
C.2	The student answer	48
C.3	The intended digit	49
C.4	The student number	51
	D Systems diagrams and software	54
D.1	Software	54
D.2	Interface	54
D.3	Templates	56
	E Validation and results	60
E.1	All tutorial results	60
E.1.1	Overview	60
E.2	Deep Convolutional Neural Network results	63

E.2.1	Trained on generated database	63
E.2.1.1	Accuracy of network	63
E.2.1.2	Conclusion on accuracy	63
E.2.2	Trained on MNIST database	64
E.2.2.1	Accuracy of network	64
E.2.2.2	Conclusion on accuracy	64
E.2.3	Trained on mixed database	64
E.2.3.1	Accuracy of network	64
E.2.3.2	Conclusion on accuracy	65

List of Figures

1.1	Automatic test grading template layout.	3
1.2	Graphical overview of the system.	5
2.1	Standard OMR template with reference blocks on the left (VijayaForm, 2017).	7
2.2	Contours found around corrected answer.	9
3.1	Four markers found from applying Radon transforms.	12
3.2	Reduced contours in image.	13
3.3	Radon transform applied on function f , adapted from Edoras (2017) . . .	14
3.4	Result in rotation after applying radon transform.	15
3.5	Detection of contours in image and estimation of bubble locations.	16
4.1	Image showing found contours for boxes used for character recognition. . .	20
4.2	The box contour found is normalized to form a rectangular shape.	21
4.3	Box after black lines are filtered out, found using a Radon transform. . .	21
4.4	Custom segmentation algorithm used to find the main cluster in the remaining image.	22
4.5	Area block drawn around the segment most probable to belong to the digit. . .	22
4.6	Final image after translation and normalization are applied.	22
4.7	Example image used as input to the neural network, from Tensorflow (2017) . . .	23
4.8	Basic structure of a neural network, from Karpathy (2017)	24
4.9	Graphical setup for determining the intended digit written by a student. . .	27
4.10	Graphical setup of student answer.	28
4.11	Graphical setup of student number.	29

LIST OF FIGURES

5.1	Image showing answer with crossed-out answers that the system misinterpreted.	33
5.2	Filled-in answer with only character information.	34
5.3	Crossed-out character that confused the grading system.	35
5.4	Incorrectly identified answer as 95.	38
A.1	Project plan for the final year project.	43
C.1	System overview.	47
C.2	Graphical setup of determining student answer.	48
C.3	Column with evidence that gets considered for the calculation of an intended digit.	49
C.4	Graphical setup of determining intended digit.	50
C.5	Graphical setup of determining student number.	52
D.1	Main interface of the test grader.	55
D.2	Clash list interface of the test grader.	55
D.3	Original template focussed on numbered answered questions.	57
D.4	Template allowing for numbered and multiple choice answers.	58
D.5	Template focussed solely on multiple choice type questions.	59

List of Tables

- 5.1 Description of 25 evaluation tests. 32
- 5.2 Description and quantity of clashes in the different categories. 37

- B.1 Description of exit level outcomes and how this project adhere to them. 45
- B.2 Description of exit level outcomes and how this project adhere to them. 46

- E.1 Description of tutorial results. 61
- E.2 Description of tutorial results. 62
- E.3 Test results for neural network trained on generated data. 63
- E.4 Test results for neural network trained on MNIST dataset. 64
- E.5 Test results for neural network trained on combined data. 65

Nomenclature

Abbreviations

<i>CT</i>	Computed tomography
<i>DAG</i>	Directed acyclic graph
<i>DCNN</i>	Deep convolutional neural network
<i>GAN</i>	Generative Adversarial Networks
<i>MNIST</i>	Modified national institute of standards and technology
<i>NN</i>	Neural network
<i>OCR</i>	Optical character recognition
<i>OMR</i>	Optical mark recognition
<i>PGM</i>	Probabilistic graphical model

Symbols

δ	Dirac delta function maps any function to its value at zero
$\sigma(z)$	Normalization function in a neural network
θ	Angle of summation line in the Radon transform
A	Random variable representing an answer for a specific question

b	Bias variable added to allow a neuron to have an offset in its output
BE_i	Random variable representing bubble evidence obtained from image processing of the digit at index i
BI_i	Random variable representing the bubbles the student to colour in for digit i
c	Number of inputs to a neuron
CE_i	Random variable representing a character evidence obtained from image processing at an arbitrary index i
D_i	Random variable representing a digit in column i for an answer or student number block
DE	Random variables representing a digit evidence obtained from image processing
DI	Random variables all the digit intended by the student
$f(x, y)$	Two dimensional function that a Radon transform is applied over
$G(r, \theta)$	Radon transform defined over r and θ
$G_\theta(r)$	The Radon transform's values at a given θ
I	Random variable representing the test image
k	Index value for a specific input neuron
n	Number of bubbles of template sheet
$p(i)$	Probability of digit at index i
r	Length of perpendicular offset of the line in a Radon transform

S	Random variable representing the possible student number
S_n	Random variable representing the sign of a specific answer
w_i	Weight value at index i
x	Horizontal coordinate value in two dimensional plane
x_i	Input value at index i
y	Vertical coordinate value in two dimensional plane
z	Weighted sum of a neuron's inputs and internal variables
z_k	Weighted sum of a neuron's inputs and internal variables at index k in the specific layer

Chapter 1

Introduction

As modern technology and machine learning techniques advances, it is important for the educational sector to continuously advance in their learning environment. This allows for an ever improving learning experience in and outside the classroom.

1.1 Problem background

In the recent years the Applied Mathematics Department of Stellenbosch Engineering, started observing a decrease in accuracy in the grading of tutorial tests done by teaching assistants and demies. Students complained on a regular basis about correct answers being marked incorrectly or even that their answers were totally ignored. Furthermore, the assistants took a long time to grade these tests with time and financial implications. To address this problem, the Applied Mathematics department proposed to automate the process of grading the tutorial tests.

The head of the department wanted a system that can analyse and grade tests written on a specific template. These answer sheets are handed out to the students to fill in their respective answers. The answer sheets are then scanned to create a digital copy. The system is tasked with automatically grading all these digital copies and transferring the graded results to a database.

The department has tried to use a basic version of this type of system in the past. Such a system only really becomes useful to the department if it can grade decimal valued answers instead of only being able to grade multiple choice answers. Thus a template needs to be designed that allows students to answer with decimal valued answers. Another

factor to consider is filling in those values must still be relatively easy. Therefore students need to have the freedom of quickly crossing out an answer instead of erasing it each time. In this research project the department sends weekly scanned answer sheets, which need to be graded and the results returned to them. The feedback from these weekly tutorial tests then acts as validation tests for the system. These tests are done in parallel with the development and expansion of the test grading software. For these reasons an agile development methodology is used to develop the software.

1.2 Problem statement

Given the problem background and stakeholders discussed in the previous section the problem to be solved can be formulated as follows:

Develop and implement an *automatic test grading system* that will *increase marking accuracy* and *decrease marking time* on the *grading* of Applied Mathematics tutorial tests written by students.

1.3 Project scope and assumptions

Initial discussions with the department revealed that a specific template can be used. This template allows the image processing software to more accurately determine what the student's intended answer is. The template consists of bubbles that can be filled in as well as blocks for handwritten digits, as shown in Figure 1.1. The focus of this project lies on processing the scanned answer sheet written on the specific template. To use the template the student must fill in his/her student number and question answers in the designated character blocks. They are also required to fill in the bubble underneath each digit, corresponding to that specific digit. Additionally, a bubble next to each question is provided if a negative sign is required.

Any additional assumptions are stated in the appropriate section throughout this project.

Stellenbosch University: Applied Mathematics B154 Test /...../20 ...

Surname : US Number →

Enter your student number, both as numbers and by filling in the ovals below the student number.

Give your answer for each question, both as numbers and by filling in the corresponding oval below each box.

Use only pen to colour the ovals.

.....
Student's signature

1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6
7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7
8	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8
9	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	9
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

Question 1:

(a) .
-

1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6
7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7
8	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8
9	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	9
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

(b) .
-

1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6
7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7
8	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8
9	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	9
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

Question 2:

(a) .
-

1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6
7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7
8	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8
9	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	9
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

(b) .
-

1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	3
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	4
5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	5
6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6
7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7
8	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	8
9	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	9
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

Figure 1.1: Automatic test grading template layout.

1.4 Project objectives

The problem as stated in Section 1.2, is addressed by pursuing the following objectives:

1. Develop a *software application* to allow a *user* to grade a large number (approximately 1000) scanned student tests automatically.
2. The software should provide precise and useful feedback. In order to achieve this, every graded result will also include feedback on what questions the student answered incorrectly.
3. Upgrade the software to allow students to cross out answers instead of having to erase them.

4. Perform a weekly *validation experiment* with the software, by grading tutorial test for the department. The results are then used as the students' grades for that test.
5. Use an *agile development methodology* to improve the software in parallel with the grading of weekly tests.
6. Add additional software, using machine learning, to improve the accuracy of the system in grading these tests.
7. Add additional software that allows students to specify their student number only using characters, thereby simplifying the use of the template.

The objectives are covered in separated chapters in this report. Note that each objective (1 to 8) build on the objectives previously listed.

1.5 Research methodology

An agile development methodology is used to complete the objectives listed in Section 1.4. The methodology consists of six different phases:

1. Identify a new feature or update that needs to be implemented into the software package.
2. Perform a study on existing methods to implement this new software.
3. Implement and integrate the new software with the current knowledge of the solution.
4. Test the software and observe if it's performance is satisfactory.
5. If the software is not working as planned, revisit steps 2 to 4 until the new software is working.
6. Use version control, in this case Git, to save the latest version of the software.

The structure and graphical overview of the software is presented next.

1.6 Graphical overview of system

The process of writing an answer down on a paper can be represented using 6 information nodes. These nodes are illustrated in Figure 1.2. The unnamed blocks indicate that information processing occurs in these steps. The student has certain information he/she wants to portray on the paper, namely the 4 answers and student number he/she wants to write down. Thus those 5 nodes give rise to the image, representing the last node. These nodes have to be represented in a probabilistic manner as the process of writing answers down and scanning the test sheet, may produce a different image every time a test is written, even though the same answers are intended. Thus the system is fundamentally tasked with inferring the probability of each answer and the student's number given the particular image as evidence. This is done by processing the image evidence to produce and estimate the intended answers. These processes are described in later sections. In the next section a literature study on current systems is given. For a more detailed mathematical overview of the system, refer to Appendix C.

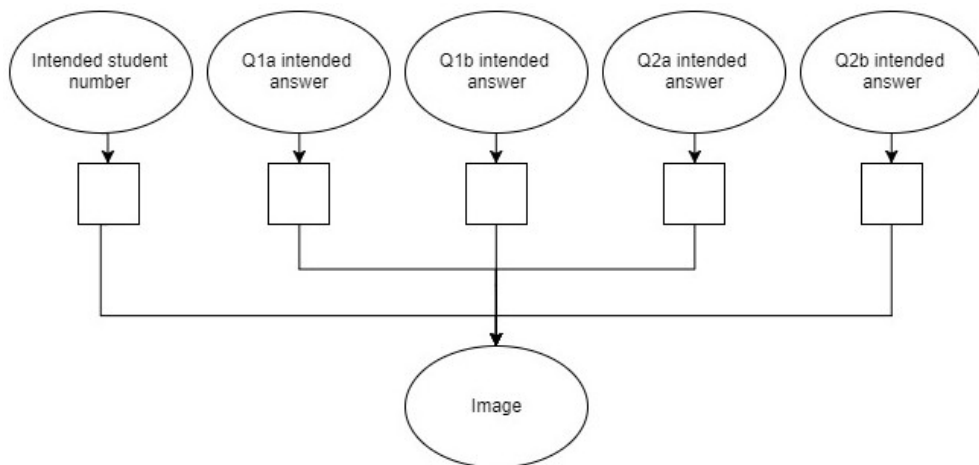


Figure 1.2: Graphical overview of the system.

Chapter 2

Literature study

In the previous chapter the problem statement and objectives of this project was laid out. Further, a brief system overview was provided. This chapter will provide information on already existing Optical Mark Recognition (OMR) systems and the techniques they use for conducting image processing. Research on additional machine learning approaches used in character recognition and probabilistic modelling is also given.


2.1 Existing Optical Marker Recognition techniques

An OMR system is software that is used to extracted handwritten information from a filled-in form. Each system normally has a specific template that it can extract information from. An example template is shown in Figure 2.1. These systems are generally used when fast and accurate grading of tests are required. The black boxes in the figure are used to locate the template. The main drawback of these systems is that information can only be portrayed in a limited manner, due to the bubbles. On an OMR template there is a grid of bubbles that allows a user to choose between different options to answer. OMR systems are thus excellent for the grading of multiple choice type questions.

2.1.1 Standard Optical Marker Recognition systems

As can be seen in Figure 2.1, there is normally specific reference blocks on an OMR template. These blocks are included to allow the computer vision and image processing

2.1 Existing Optical Marker Recognition techniques



MCQ ANSWER SHEET

REGISTER NUMBER

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

Date of Examination

Date: / /

INSTRUCTIONS TO CANDIDATE

1. Use HB Pencil only for shading inside the Bubbles.
2. Use Ball Point Pen only to write in the Rectangle Boxes Provided.
3. Write and shade your Register Number in the space provided.

Correct Method

0 1 2 3 4

Wrong Method

✓ ✗ ⊕ ⊖

Course : _____

Subject : _____

ANSWERS

1	(A) ● (C) ○ (D) ○	<input style="width: 30px;" type="text"/>	16	(A) ● (B) ● (C) ○ (D) ○	<input style="width: 30px;" type="text"/>
2	● (B) (C) (D)	<input style="width: 30px;" type="text"/>	17	(A) (B) ● (C) ○ (D) ○	<input style="width: 30px;" type="text"/>
3	(A) (B) (C) ● (D) ○	<input style="width: 30px;" type="text"/>	18	(A) (B) (C) ● (D) ○	<input style="width: 30px;" type="text"/>
4	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>	19	● (B) (C) (D)	<input style="width: 30px;" type="text"/>
5	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>	20	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>
6	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>	21	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>
7	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>	22	(A) (B) (C) ● (D) ○	<input style="width: 30px;" type="text"/>
8	● (B) (C) (D)	<input style="width: 30px;" type="text"/>	23	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>
9	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>	24	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>
10	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>	25	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>
11	● (B) (C) (D)	<input style="width: 30px;" type="text"/>	26	● (B) (C) (D)	<input style="width: 30px;" type="text"/>
12	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>	27	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>
13	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>	28	(A) (B) (C) ● (D) ○	<input style="width: 30px;" type="text"/>
14	(A) (B) (C) ● (D) ○	<input style="width: 30px;" type="text"/>	29	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>
15	(A) (B) ● (D)	<input style="width: 30px;" type="text"/>	30	(A) ● (C) (D)	<input style="width: 30px;" type="text"/>

Signature of the Candidate

Signature of the Invigilator

Figure 2.1: Standard OMR template with reference blocks on the left (VijayaForm, 2017).

2.1 Existing Optical Marker Recognition techniques

algorithms to find the orientation of the image more easily. Other templates include lines that can be used to locate the template.

In an OMR system there are normally two phases in the grading of a test (Ivetic & Dragan, 2003). The first step is to determine the grid locations where the answers are located in the image. In this process the system finds the orientation of the template in the image and therefore can approximate the locations of the bubbles. Normally, some preprocessing on a blank template is done beforehand to aid in locating the bubbles. Once the bubbles are found, their estimated locations gets stored. The second step is then to estimate the value of each bubble and use these values as the estimated answers. These steps are described in more detail next.

2.1.1.1 Finding the template

The first process performed by OMR software is to locate a template grid inside the test image. This step is necessary for the software to identify which bubbles corresponds with each answer. One method of locating the template is identifying lines in the template. The border normally contains long lines that can be extracted using a Hough transform (Patel & Prajapati, 2003). A Hough transform is used to locate instances of an imperfect object within a certain shape range. A specific form of a Hough transform can be implemented to detect lines. This form is called a Radon transform (MathWorks, 2017). A Radon transform provides a way of representing an image as a summation of different line integrations.

Once at least two line references on a page have been found, the template orientation can be determined. Those two lines are then used to find two reference points on a page. These points allow the system to estimate the locations of every bubble on the template sheet. In the next section a method to process these bubble is described.

2.1.1.2 Processing a bubble

Once an estimated location for each bubble is known, the next step for an OMR system is to process these bubbles. A basic image processing method to classify bubbles is by adding the number of coloured-in pixels (Patel & Prajapati, 2003). If this value is above a specific threshold value, the bubble is classified as filled-in, otherwise not. An additional algorithm is needed to detect if a bubble is crossed out .

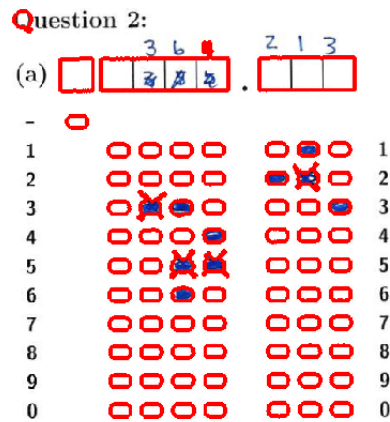


Figure 2.2: Contours found around corrected answer.

In this project additional contour detection is used to determine if an answer in a bubble is truly coloured in and not just crossed out. This means that the contour around the bubble needs to be detected and used in the analysis. In Python (or C++) this can be implemented using the freely available OpenCV library (Rosebrock, 2016). OpenCV is an image processing library that has highly optimized techniques to find contours in a given image. An example of this is shown in Figure 2.2. Once the contour information is known, the bubbles can be assessed by the pixels inside it, as well as its shape. Thus by evaluating the shape of a contour it can be determined if a bubble is crossed out or not.

2.2 Optical character recognition

Optical Character Recognition (OCR) software is also needed and applied on the characters blocks present on the templates, to further increase the accuracy of the system. It is found that one preferred way of performing OCR is using TensorFlow. This method is described by Google (2017). TensorFlow is a Python library, but allows the building of instructions to be implemented in efficient C++ code. For this test grader, TensorFlow is used to construct a deep convolutional neural network (DCNN). Deep convolutional neural networks are powerful machine learning techniques generally used to process images. One effective application of a DCNN is that it is excellent at processing an image containing a digit and estimating that digit.

2.2.1 Probabilistic approach

A last piece of information that needs to be investigate is how the bubble and character evidence can be brought together in an effective way to produce the best estimate of the intended student entries. Once a bubble and character are analysed, a probabilistic value is assigned to it. Each bubble has a probability of being filled in, while each character has a probability distribution over 10 digits. A probabilistic model is needed to predict these student entries. In [Ankan & Panda \(2015\)](#), a machine learning approach is used to infer a probability of random variables being in a certain state given evidence. This method is known as a Probabilistic Graphical Model (PGM). A *Probabilistic Graphical Model* (PGM) is a probabilistic framework that consists of a graph that specifies the probabilistic relationship between a number of random variables. These types of models work well in some aspects of the medical field. If a patient has certain symptoms, a PGM can be used to predict what the underlying illness behind the decease is. In this project a PGM is used to estimate the underlying student entries given the evidence presented. The library used by [Ankan & Panda \(2015\)](#) is called pgmpy. This library allows for a PGM to be constructed in Python.

2.3 Conclusion: System requirements

In conclusion it is seen that a combination of image processing and machine learning techniques is needed to successfully grade a student test paper. A good method in locating a template is using a Radon transform to find reference lines on an image. Once this is done, the bubbles can be estimated in the image. It is found that a DCNN can be implemented to classify digits in the TensorFlow library environment. Once all these evidence are acquired, a PGM was found to be a preferred method in predicting the estimated entry answer of the student. In the following chapter, a more detailed overview on the image processing techniques used in this project is given.

Chapter 3

Image processing

The previous chapter focused on existing methods of grading tests automatically. It was found that most systems only use image processing, without a machine learning component, to grade these tests. In this chapter the core techniques behind processing these answer sheets, using image processing, are described. By using only these image processing techniques a reasonably accurate system can already be constructed. For further improvements in accuracy, two machine learning approaches can be implemented.

3.1 Orientation detection

As mentioned in Section 2.1.1, there are two main steps in OMR grading. The first challenge with grading a scanned answer sheet, is finding the orientation of the template in the image. This can be done by finding two or more reference points on the page. These reference points then allow for the calculation of the template's rotation, offset and size inside the image. In Chapter 2 it was found that the traditional way to find these reference points was to include them on the page, in the form of black blocks or lines. Including black blocks is an effective method of finding the template, but might look a bit less attractive, due to the black blocks on the page.

For this project, the markers or reference points that the software uses are already present on the template paper. These are the two longest horizontal lines as well as the two vertical lines on the comment box, as can be seen in Figure 3.1. Together, these lines have enough information to determine 4 reference points, shown in red in the figure. These lines are chosen as references, since a Radon transform can easily be applied to

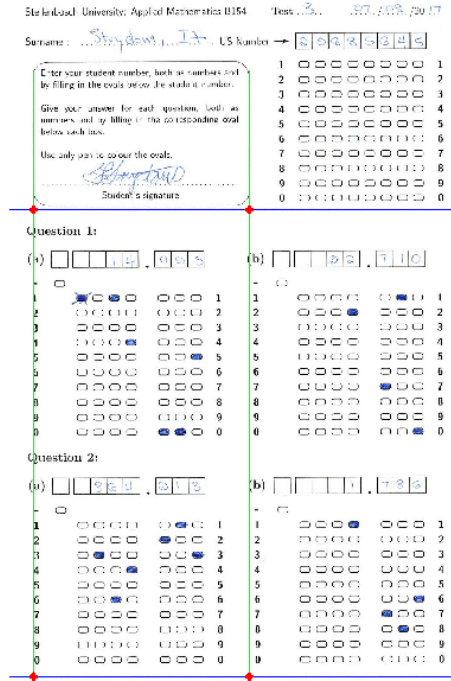


Figure 3.1: Four markers found from applying Radon transforms.

locate them, as discussed in Section 3.1.2. The system only needs three of those four lines to find the template and thus can successfully find the template even though one of the four lines is identified incorrectly. Using the reference points the rotation, offset and size of the template is determined. However, before the orientation of the image can be determined, a check must be made to determine if the image might be upside down. This is done to make it easier to find the orientation afterwards. To determine if an image is upside down, initial image filtering is required as discussed in the next section.

3.1.1 Initial filtering and orientation detection

In order to check if an image is upside down, the software first needs to find relevant contours on the page. The contours are then filtered out if it does not loosely match the characteristics of a bubble or character block. This process can be described in Algorithm 1.

Algorithm 1 : Filter contours and check image rotation.

1. Threshold the image by making all the pixel values either white (lower than the mean) or black (higher than the mean).
2. Conduct contour analyses on the image to find all the contours, using the Python library OpenCV.
3. Filter through the contour array to filter out all contours that are not approximately the desired size and aspect ratios.
4. Save these contours for later use.
5. Determine if more contours lie above the middle of the image. This is true if the image is orientated upright. Rotate the image by 180° otherwise.

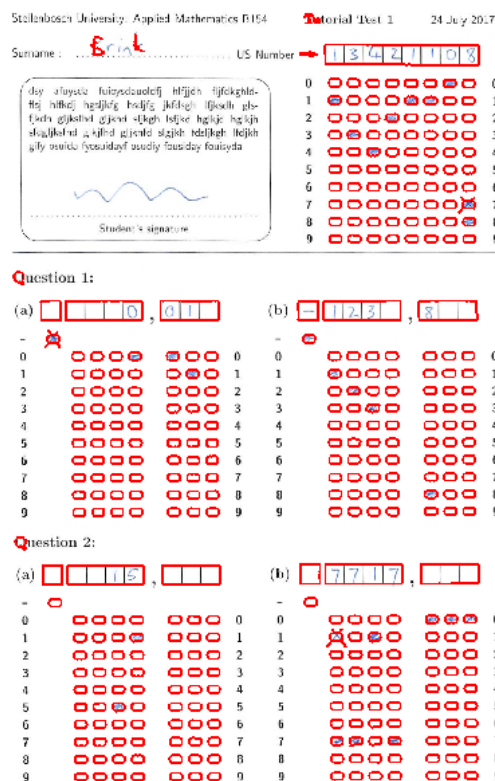


Figure 3.2: Reduced contours in image.

It is important to note that there are still unwanted contours in the list, but for now this reduced list is sufficient. Once the list is found, the software counts the number of contour centrepoints below and above the image center. Figure 3.2 shows the resulting contours found in the image. As can be seen in the figure, more bubbles should be below the horizontal center line, for the image to be the right side up. The next step will be to determine the coordinates of the answers the student wrote down, by first locating the template in the image. In the next section a mathematical transform is define, which is used to find the template.

3.1.2 Radon transform

The Radon transform is an integral transform that can be represented by a series of line integrals over a function $f(x, y)$. These transforms are commonly used in Computed Tomography (CT) scans where cross-sectional images of the body are needed. Mathematically this transform is defined as

$$G(r, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos(\theta) + y \sin(\theta) - r) dx dy, \quad (3.1)$$

where r represents the perpendicular offset of the line and θ the angle of the line. The variables x and y specify coordinates in a two-dimensional space within which the function $f(x, y)$ is defined. A visual interpretation of this transform is shown in Figure 3.3. In the figure $G_{\theta}(r)$ is the Radon transform's values at a given θ .

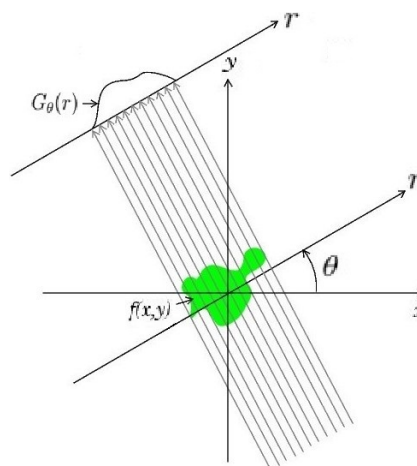


Figure 3.3: Radon transform applied on function f , adapted from [Edoras \(2017\)](#).

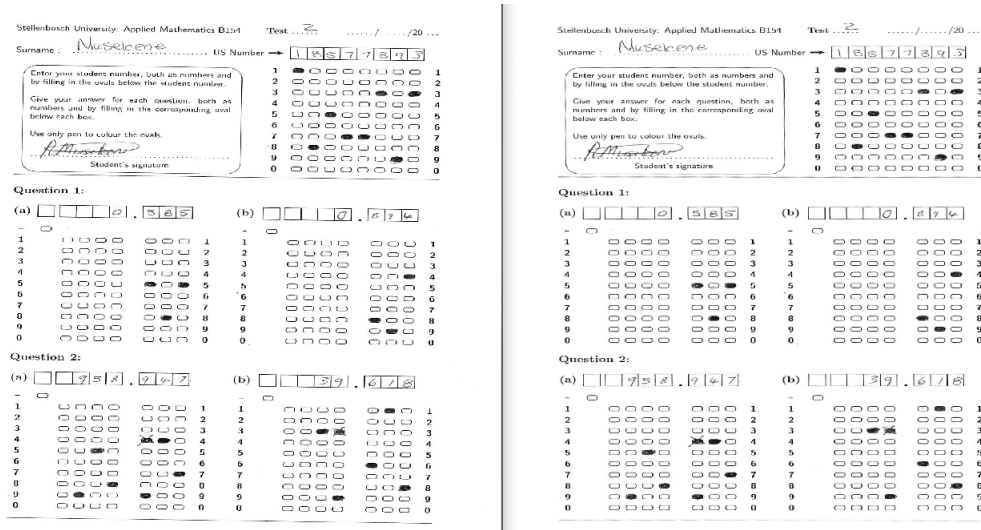


Figure 3.4: Result in rotation after applying radon transform.

3.1.3 Finding the template

In this project, the image's Radon transform is calculated for specific intervals of the gradient. Each gradient interval will thus generate a one dimensional array of values corresponding with the pixel intensities along the lines that are being summed, as shown in Figure 3.3. This method is used to determine where the 2 horizontal and 2 vertical lines are located, as previously mentioned.

The Radon transform's values are calculated between the angles 85° and 95° to find the first two horizontal lines. It is assumed that the image will not be rotated by more than 5° in either direction. Black lines will cause a spike in value to appear on the Radon transform for the angle where it is summing parallel with that particular black line. By finding the transform angle that has this spike in value, the rotation of the template can be found. The two maximum values that are recorded at this angle are then taken as the relative locations of the two horizontal lines. After the correct angle is found, the two vertical lines can be found by applying a Radon transform at an angle 90° clockwise from the previously calculated angle. The two peak values at that angle then provides the relative locations of the two vertical lines. The four reference points can now be calculated, as seen in Figure 3.1. Once the reference points are found, the image is rescaled and orientated to the original template size for further processing. In Figure 3.4 the corrected image is shown.

3.2 Bubble detection and processing

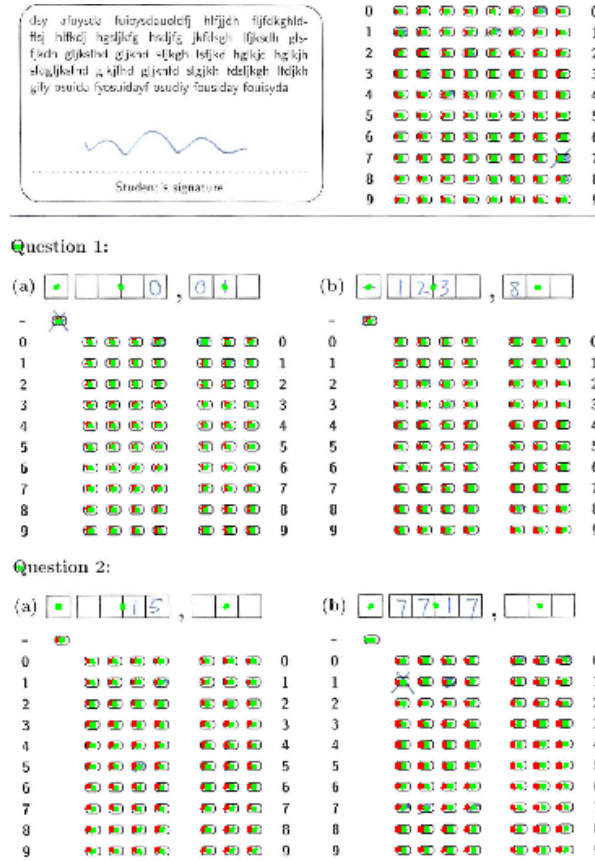


Figure 3.5: Detection of contours in image and estimation of bubble locations.

Once the template is located the bubble values and digit blocks can be determined, using reference location. These reference locations were calculated in preprocessing conducted on an empty template. Figure 3.5 illustrates the final estimation of all the bubbles in the template. The estimated bubble centres are coloured red, while the green points represent the centres of all the remaining contours.

Now that a list of possible bubble contours are found with the estimate bubble centres, one contour needs to be assigned to each bubble.

3.2 Bubble detection and processing

The location of each bubble in the image is found by simply taking the contour closest to that particular bubble's estimated location. This is done in an efficient manner by

sorting the contours by their locations. Searching through the contours now becomes linear and of complexity $O(n)$, where n is the number of bubbles. This means that the processing time to find these bubbles is linearly related to the number of bubbles in the template image.

Next, the data in each contour needs to be processed and stored. The first type of evidence is obtained from calculating the average pixel intensity inside the contours. If this value is high, the bubble is most likely coloured in or crossed out. The advantage of using the closest contour as the bubble's estimated location, over conventional methods, now becomes apparent. In conventional methods, only pixels at the bubble's estimated location are used to determine if the bubble is coloured-in. This becomes problematic if the system needs to know if the bubble has been crossed out, as the only data available is about pixel intensities.

Using the contours information about the shape of the bubble entry is also known. Thus by drawing the smallest possible block around the contour, that still covers every value inside the contour, an area can be calculated. This area becomes large when a answer is crossed out, due to the lines stretching outside the initial bubble. By inspecting the area value, the system can successfully determine if the bubble is filled in or crossed out.

3.3 Data processing and grading

The previous section now allows each bubble to be classified into three categories namely, empty, completely filled-in or crossed out. An additional category of partially filled in is also introduced, as it aids grading of tests where students write lightly. An algorithm to determine what bubble was chosen is described in Algorithm 2.

Algorithm 2 : Calculate the student answer from bubble grid.

1. Start with column 0.
 2. Count the number of completely filled-in answers in this column. Store the position of that entry for later use.
 3. If there are no completely filled-in answers, count the amount of partially filled-in answers and override the previous values.
 4. If the previous result is 0, set the output value for that column to 0.
 5. If step 2 or 3 presents a value greater than 1, save the answer sheet to a clash list to be evaluated manually once the automatic grading of the test is completed.
 6. Repeat steps 2 to 5 for each column in the bubble grid.
-

This algorithm therefore checks if there are more than one entry in any of the columns. If this is true, the results are sent to a clash list to be marked manually. If one bubble was found to be coloured in, the value gets set to the index of that bubble. Finally, if no bubbles were coloured in the result for that column are set to 0.

3.4 Conclusion

This chapter provided an overview of a basic automatic test grading system using image processing and computer vision. The system can achieve acceptable results using only these techniques.

The following chapter will focus on applying additional machine learning techniques to further improve the accuracy of grading these tests.

Chapter 4

Machine learning approach

The previous chapter briefly described the basic workings of the optical mark recognition system inside the automatic test grader. There is still a critical piece of evidence that has not yet been utilized in this basic system, namely the characters that the student writes in the designated boxes.

This next chapter will provide two machine learning approaches to significantly improve the accuracy in identifying digits over the previous standalone basic system discussed in Chapter 3. An approach to locate and classify handwritten characters, provided by the students, using a Deep Convolutional Neural Network (DCNN), is described. A more accurate method in estimating the student answers and student number, using Probabilistic Graphical Models (PGM), is also discussed.

4.1 Character recognition using a neural network

This section focusses on processing the characters that the students write in the designated boxes, as shown in Figure 4.1. A machine learning approach, called a Neural Network (NN), is implemented to process these digits. This neural network can take an input of a 28 by 28 dimensional array of floating point numbers. These numbers represent a 28 by 28 greyscaled image that includes the digit being classified. The neural network is then tasked with calculating the probability of each digit being present in the particular image. Before each digit can be classified using a neural network, the individual 28 by 28 greyscaled image must first be found for each digit inside the test sheet. Image processing is required to achieve this, as described next.

4.1 Character recognition using a neural network

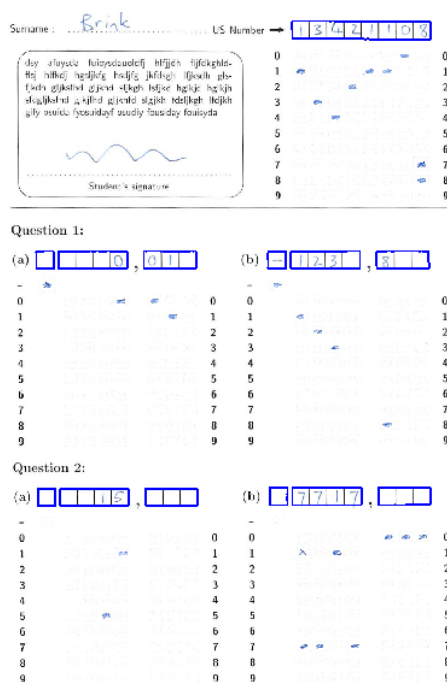


Figure 4.1: Image showing found contours for boxes used for character recognition.

4.1.1 Preprocessing and creating digit images

In order to generate a 28 by 28 pixel image for each digit, the system must first find the location of each digit. Once this is done, the digits are processed and a corresponding 28 by 28 image, best representing the particular digit, can be created. This is done in the following 6 steps:

1. Find the contour closest to the expected location of the block, calculated in Section 3.1.3. This is illustrated in Figure 4.1. It can be seen that the bubbles are already processed.
2. Transform each box to become fully rectangular using OpenCV's *four_point_transform* method. This method applies a four point perspective transform on the image to reshape it into a rectangular form. An example of the final product can be seen in Figure 4.2.
3. Perform a Radon transform on the image, at an angle of 0° and 90° to find the dark lines in the blocks. These lines are then removed from the image, as shown in Figure 4.3.

4.1 Character recognition using a neural network

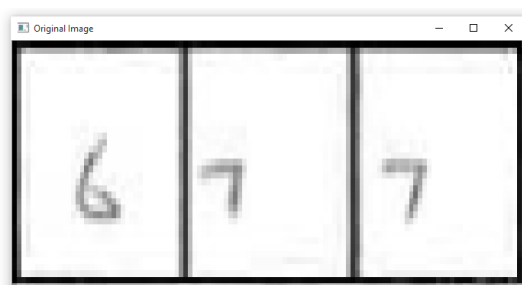


Figure 4.2: The box contour found is normalized to form a rectangular shape.

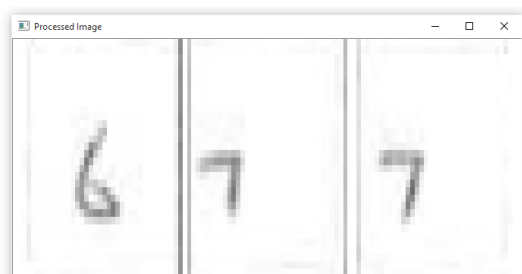


Figure 4.3: Box after black lines are filtered out, found using a Radon transform.

4. Use the locations received from the Radon transform, of the positions of the original black lines, to segment the image into the different boxes.
5. A custom segmentation algorithm is used to find the pixels most likely to belong to the digit, as shown in Figure 4.4. This algorithm is developed and implemented using a breadth first search technique to cluster the image into different segments. The algorithm first searches the image for pixels higher than a specific threshold value. This value specifies if a pixel is background or belongs to an object. Then all the pixels higher than the threshold value are assigned to a segment. A segment is classified as a local region that does not connect to any other segment through pixels higher than the threshold value. The segment that most likely represents the digit is then extracted by looking at it's area and centrepoint.
6. The area that the segment occupies is then calculated, as shown in Figure 4.5.
7. Next the segment is centred and normalized using the image area as reference. This is shown in Figure 4.6.

4.1 Character recognition using a neural network

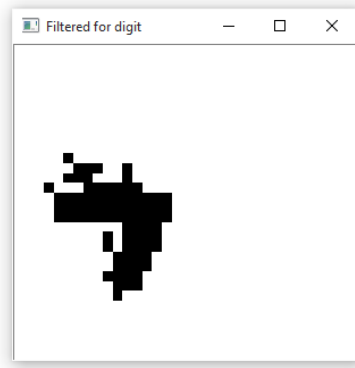


Figure 4.4: Custom segmentation algorithm used to find the main cluster in the remaining image.

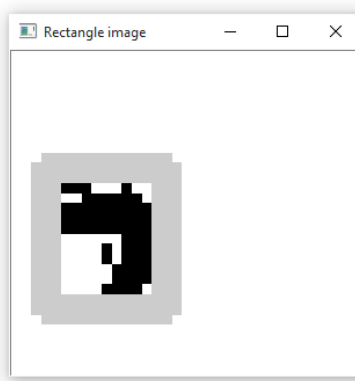


Figure 4.5: Area block drawn around the segment most probable to belong to the digit.

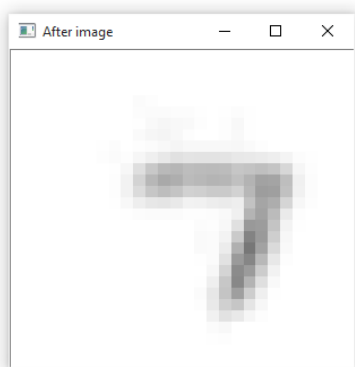


Figure 4.6: Final image after translation and normalization are applied.

8. The image is then reshaped into a 28 by 28 greyscaled image to be processed by the neural network.

As mentioned, a 28 by 28 greyscaled image is used as input. Thus if each pixel represents one value, ranging from 0.0 to 1.0, there are a total of 784 input values. Each digit image is therefore represented by one of these 28 by 28 greyscaled images. These 784 numbers are now used as input to a neural network to predict the digit. An overview of this neural network will be given next.

4.1.2 Classification of digits

A neural network (NN) is a powerful machine learning tool for approximating complex functions. The basic architecture of a neural network is illustrated in Figure 4.8. These networks are simplified approximations of how neurons in the brain work. Each neuron in the network acts as a small processing unit that can take an input and produces an output. The power of a NN lies in the ability of these neurons to be able to approximate different functions by adjusting their internal values, depending on the characteristics the user wants the network to approximate. By adjusting these internal weights, complex functions can be trained onto these networks.

For this project, a NN is trained to estimate the probability of which of the 10 digits, from 0 to 9, are most likely present in the input image. The neural network inputs a two dimensional array, representing a greyscaled image, to the network. Figure 4.7 illustrates an example input of a NN using a 14 by 14 example image. For this project a 28 by 28 image is used. In the next section the basics of these NN are discussed.

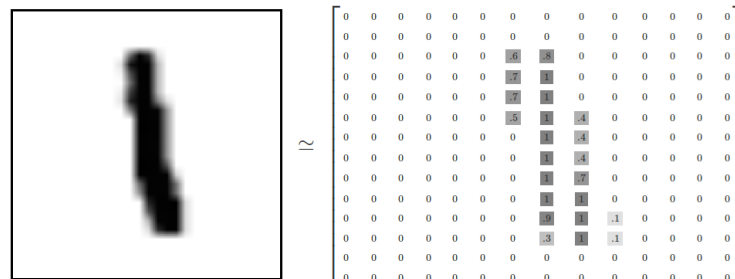


Figure 4.7: Example image used as input to the neural network, from [Tensorflow \(2017\)](#).

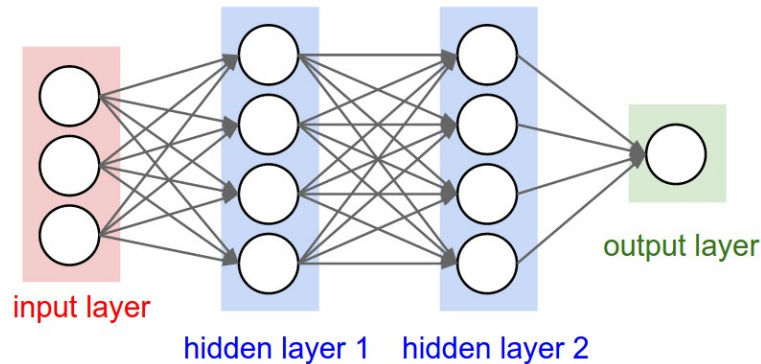


Figure 4.8: Basic structure of a neural network, from [Karpathy \(2017\)](#).

4.1.2.1 The Neural Network basics

The neuron unit is a small processing unit. By including many of these neurons a neural network can be built. Neural networks consist of input, hidden and output layers ([Nielsen, 2015](#)). This network works by first assigning values to the input layer of the network. For this project those values are the 784 values received from the digitized image. The network then sends signals through the network and generates an output. A breakdown of a neural network will be given next.

4.1.2.2 The artificial neuron

Each neuron in the network takes in a list of input values. These input values are the output values of the neurons in the previous layer, again illustrated in Figure 4.8. The weighted sum of these inputs are then added with the bias variable to give

$$z = \sum_{i=0}^c x_i \cdot w_i + b. \quad (4.1)$$

In Equation 4.1, c is the number of inputs and x_i and w_i are the input and weight values at index i . The bias term, b , enables an offset in the output, z , of the neuron. These bias term and weighted values are determined in the network's training process. The value, z , is then normalized using the sigmoid function,

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \quad (4.2)$$

4.1 Character recognition using a neural network

The artificial neuron therefore takes in a weighted input with a bias and produces a normalized output $\sigma(z)$. By adjusting the weights and bias variable, each neuron can learn to exhibit certain characteristics. This process allows different functions to be approximated by adjusting these variables. If a network of these neurons is used together, as illustrated in Figure 4.8, complex functions can be trained onto the network. In this project, these weights and biases need to be set to specific values, to allow the network to accurately categorize digits from an image.

4.1.2.3 Generating an output from the network

After the 748 input values have been assigned to the network inputs, each of the network's layers can be calculated consecutively. This is done using Equations 4.1 and 4.2 on each layer consecutively, starting at the first hidden layer. Once the first layer's outputs are calculated, the next layer can be calculated using those values as input. This process is repeated until all the layers are calculated. Once all these values are calculated, the final values can be read from the output neuron that represents the result. In this project ten output neurons are used to represent the likelihood of each of 10 digits, given the input data. The output neurons are then turned into probabilities by normalizing these outputs using

$$p(i) = \frac{\sigma(z_i)}{\sum_{k=0}^{10} \sigma(z_k)}, \quad (4.3)$$

where $p(i)$ is the probability of digit i being in the image. The value $\sigma(z_i)$ represents the values of the output neuron at index i .

4.1.2.4 Deep Convolutional Neural Network

The previous section gave an overview of a basic neural network implementation. In recent years much more powerful NN, such as Deep Convolutional Neural Network (DCNN), have been introduced. A DCNN uses the basic principles described above, but has extra optimized features that allow a neural network with many layers to be constructed and trained. A DCNN is implemented in TensorFlow (Google, 2017) and allows a NN to achieve highly accurate results on classifying digits. For this project a neural network

optimized for digit classification, as implemented by [Google \(2017\)](#), is used. The neural network is customized to increase its accuracy for this specific grading system.

4.1.2.5 Training of the neural network

The Modified National Institute of Standards and Technology (MNIST) dataset is used to train the DCNN neural network ([Yann LeCun & Burges, 1998](#)). This is a database that has a labelled training set of 60 000 images and a labelled test set of 10 000 images. For each image in the database, there is an accompanied label that specifies the digit value. The NN is then trained to model this training set. This is done by adjusting the network's internal weights so that the network's output better represent the labelled training data, given the input images. The basic idea behind the training method used in a neural network is described in [Algorithm 3](#).

Algorithm 3 : Overview on training a neural network.

1. Calculate the network's output for each of the training images used in the training round.
 2. Obtain the error function of the network, using a formula that compares the true labels of the training image with the estimated labels generated by the network.
 3. Use a method, such as gradient decent with back propagation, which allows the system to adjust the weights in a direction that minimize a specific error function.
 4. Repeat steps 1 to 3 until a time or accuracy criterion is met.
-

Once this process is completed, the network is ready to classify handwritten digits. This produces a probabilistic output of each intended digit in the test sheet. The next step is to incorporate this character evidence with the bubble evidence to produce an accurate estimate of the intended student entries.

4.2 Probabilistic Graphical Models

The final problem the system needs to solve, is to probabilistically determine the most likely student entries, given the bubble and character evidence. This is achieved by

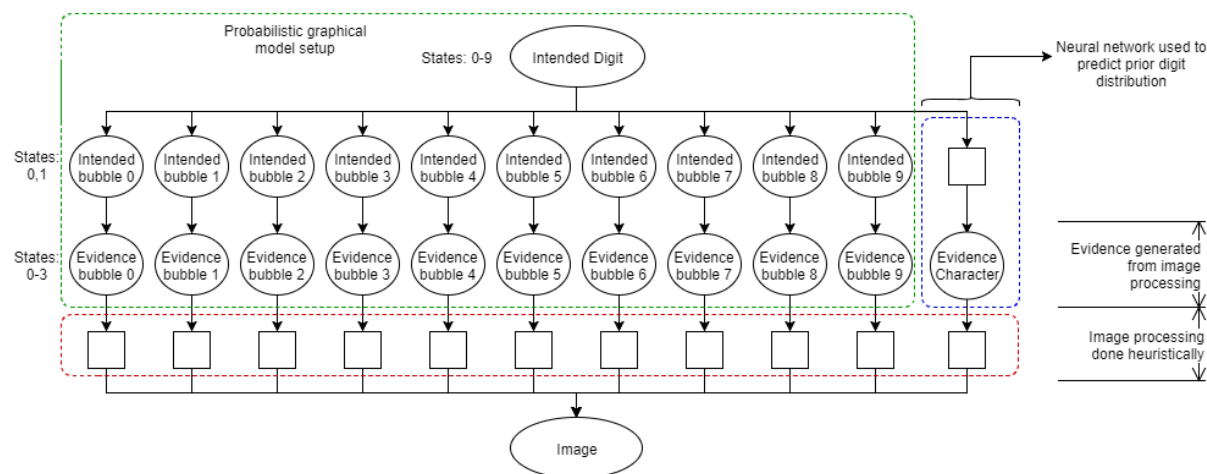


Figure 4.9: Graphical setup for determining the intended digit written by a student.

implementing two Probabilistic Graphical Models (PGMs).

4.2.1 Overview of the system

A *Probabilistic Graphical Model* (PGM) is a probabilistic graph containing random variables, where the graph expresses the conditional independence structure between these variables. The type of PGM used for this project is a Bayesian network. A Bayesian network models a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). A graphical model in essence allows a problem to be represented as information (nodes or circles) and relationships (directed arrows). An example of such a graph is shown in Figure 4.9. The directions of the arrows represent what information causes other information to be created, thus given a parent to offspring interpretation. These graphs allow for intuitive reasoning about how the system should operate. For this project an observation is made in the form of character and bubble evidence. The models are then tasked with inferring the values the student most likely wrote down, given the evidence.

4.2.2 Estimating the intended digit

Figure 4.9 should be interpreted in the direction which information flows. Originally a student has a certain digit that he/she wants to portray on the page. This is given by

the 'Intended Digit' node. There are 10 possible digits to consider and thus the node has 10 possible states. The intended digit then gives rise to character evidence as an image written in a block. Bubble evidence is also produced from the intended digit, but a variable is first introduced in between these two. The student might sometimes mistakenly think that the first bubble represents 0 and thus even if the intended digit is 0, the intended bubble might be 1. Thus the intended bubble category is also introduced, which then produces the bubble evidence, again illustrated in Figure 4.9.

After the model is constructed, the intended digit needs to be estimated, with the image as evidence. This can be done by reasoning from the bottom (image evidence) upwards to the intended digit. The first step is to process the image to produce more tractable evidence. Producing bubble evidence from an image is described in Chapter 3. In Section 4.1.1, the process to extract the character evidence from the image is also described. Using a NN the prior probability of each digit can be determined from the character box evidence. The next step is to assign the prior intended digit probability and bubble evidence to a PGM. Using this digit PGM, the intended digit can be inferred.

4.2.3 Estimating the student answer

An answer is represented by 8 columns, shown in Figure 4.10. The first column represents the sign of an answer. Thus two signs are possible. For each of the remaining 7 columns a number from 0 to 9 can be represented in each column. Thus there are 10 possible values for each of these 7 columns. This gives a possible number of values that an

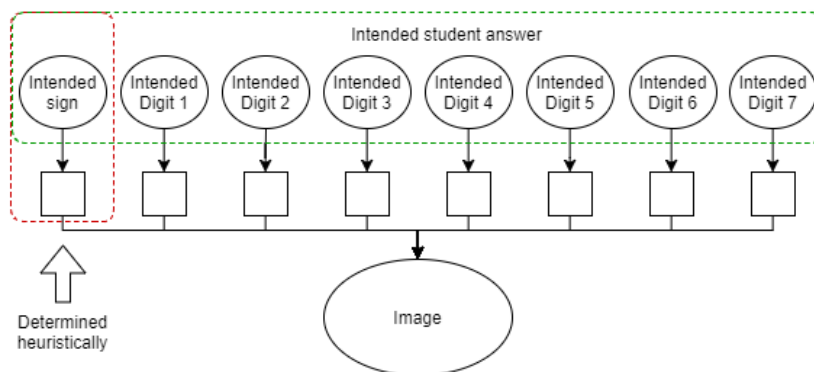


Figure 4.10: Graphical setup of student answer.

answer can take to be $2 \cdot 10^7$, equalling 20 000 000. Calculating each of these states is computationally intractable. Thus an assumption is needed to reduce the number of possible states. A fair assumption to make, is that all 20 000 000 possible values are equally likely to be written down. Consequently each column digit becomes independent of the other columns' values. This means that if a value in one column is known, it does not influence the probabilities of the other columns having a certain value. Thus the number of states to calculate now becomes $2 + 10 \cdot 7$, equalling 72, since each column's states can now be calculated independently. Knowing this independent property, the student's answer can be calculated by using 7 digit models and a heuristic calculation of the intended sign.

4.2.4 Estimating the student number

For the student numbers, knowing one digit value of a column influences the probabilities of the other columns having a certain number. The reason for this is that there are only a limited number of student numbers to consider. For example if the first digit is a 2, only student numbers starting with 2 still have to be considered. To account for this, an additional node is added above the individual digit probabilities, as seen in Figure 4.11. This node represents the probability of each student number being present in the image. The number of states of this node is in the order of 900, depending on the number of student numbers. Once the graph has been set up, the model can be used to infer the most probable student number. By setting all the bubble evidence and character priors,

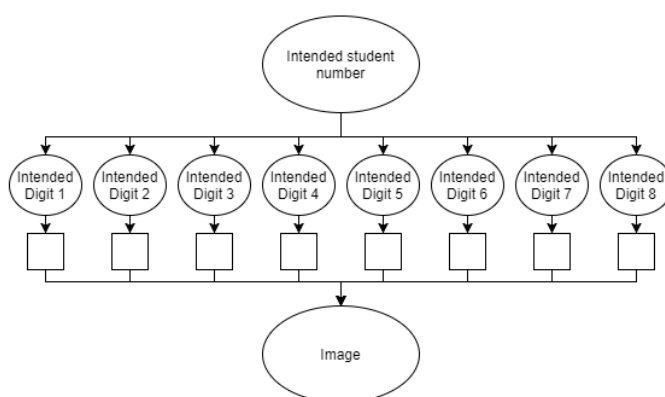


Figure 4.11: Graphical setup of student number.

the student number probabilities can be inferred. This model allows for an accurate result, because student numbers normally differ in more than one digit. The system can thus strongly differentiate between student numbers and determine which student number is most likely. It is found that if the student provides only character information with no bubbles coloured in, an accurate estimate can still be made.

4.2.5 Training of a Probabilistic Graphical Model

In a PGM the conditional probabilities distributions between each random variable that has a relationship (arrow) is needed. In order to calculate these conditional probability distributions, training data must be gathered. The previous basic model with the character recognition software is used to estimate the answers for 100 test sheets. Once these sheets were graded, the results were manually checked. Thus each training label now contained bubble evidence and character priors from the NN. With this information, each training set is used to infer the conditional probability needed to train both the digit and student number models. For a mathematically approach to these two PGM models, refer to Appendix C.

4.3 Conclusion

This chapter discusses two machine learning techniques to improve the accuracy with which the system infers the answers written on each scanned test sheet. A method is shown, using a NN, to estimate the probability of each digit given only the character box as input. Additionally, an approach is discussed, using a PGM, to allow the system to make a final prediction of what the student intended to write down given the image evidence. The PGM method is found to be accurate enough to determine the student number by only using the character recognition information provided. The following chapter will cover the validation and results of the system from weekly grading done for the Applied Mathematics Department.

Chapter 5

Analysis of results

This chapter discusses the accuracy of the test grading system. In the first two sections of this chapter, the basic and complete system are compared using the same datasets. The first test grading system includes only image processing techniques described in Chapter 3. The second system contains the complete project software with all the machine learning techniques described in Chapter 4.

Each system is assessed using two categories. The first category describes all the tests the system transferred to the user for manual marking. The reason for this is that the system calculated a certainty level below a specific threshold for that particular test. In this category the tests are send to a clash list. After all the tests are graded, the system displays an interface with values it estimated the student wrote down. An additional image of the test is also displayed. The user is then tasked with looking through each of the clash list tests, using the interface, to see if any tests is graded incorrectly by the system. This process is normally fast, as the system has a high accuracy in predicting the answers correctly. In the last category, all the answers that the system decided to grade automatically, but identified incorrectly, is described.

5.1 Results of 25 test cases

In this section the results of grading 25 hand picked tests are compared between the two systems. Among the 25 tests, there are different categories that evaluate different aspects and limitations on both systems. The categories and the number of tests for the particular category, are listed in Table 5.1.

Table 5.1: Description of 25 evaluation tests.

Description of test type	Number of tests
Test with crossed-out answers	7
Test with lightly coloured entries or partially coloured-in entries	4
Test with negative signed answers	1 (also included under other categories)
Test with no bubbles and only characters filled in for a specific entry field	5
Test with no characters filled in and only bubbles for a specific entry field	2
Test with data filled in correctly	2
Page with no template on it (Possible scanned upside down)	2
Page with rotated template inside image	2

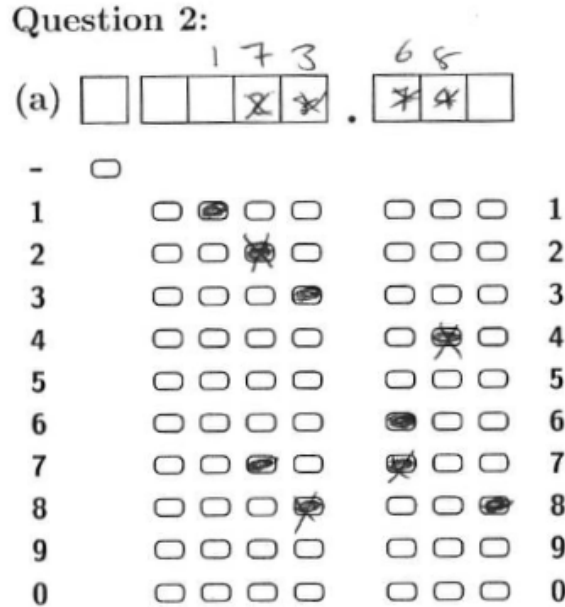


Figure 5.1: Image showing answer with crossed-out answers that the system misinterpreted.

These tests are specifically chosen, because combined, they approximate all the types of tests the system have to assess. Most of the tests are extreme cases of what students have filled in on test forms. These tests provide a good benchmark to determine how well each system performs in challenging situations.

5.1.1 Basic system

Using the basic system, an average time evaluating each test is calculated to be 0.305 seconds on the grading computer. An overall grading accuracy of 84.6% was recorded for this system.

5.1.1.1 Clash list

A total of 6 out of the 25 tests were reported as clashes. The two images without a template were reported in the clash list. Furthermore, 1 test which only has the student number written in characters with no bubble information was also reported to the list. The other 3 clashes were reported due to the crossed-out bubbles being interpreted as

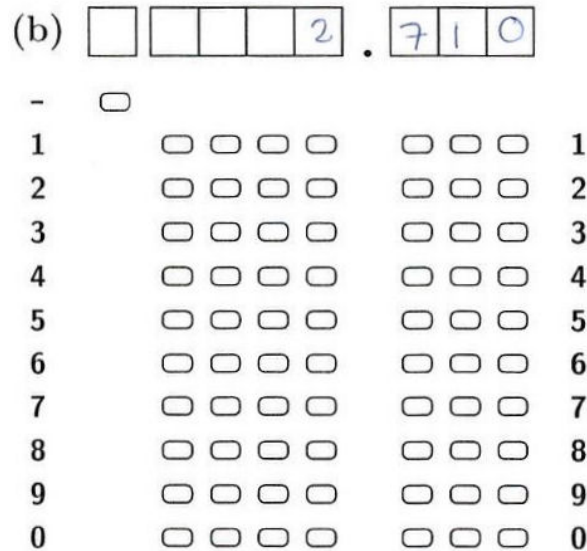


Figure 5.2: Filled-in answer with only character information.

still filled in, causing the system to think that two answer bubbles were filled in. An example of this is shown in Figure 5.1.

5.1.1.2 Incorrect automatic graded results

There were 4 tests that were graded automatically, but incorrectly. All of these tests had only character information in at least one of the answers. An example of this can be seen in Figure 5.2.

5.1.2 Complete system

Using the complete system, an average time evaluating each test is calculated to be 2.011 seconds. An overall grading accuracy of 100.0% was recorded for this system.

5.1.2.1 Clash list

A total of 6 out of the 25 tests were reported as clashes. The two images with no template in were again reported in the clash list. Two cases were reported to the clash list due to the character recognition determining a crossed-out character as the intended character, but with low certainty. An example of this is shown in Figure 5.3. One test was reported,

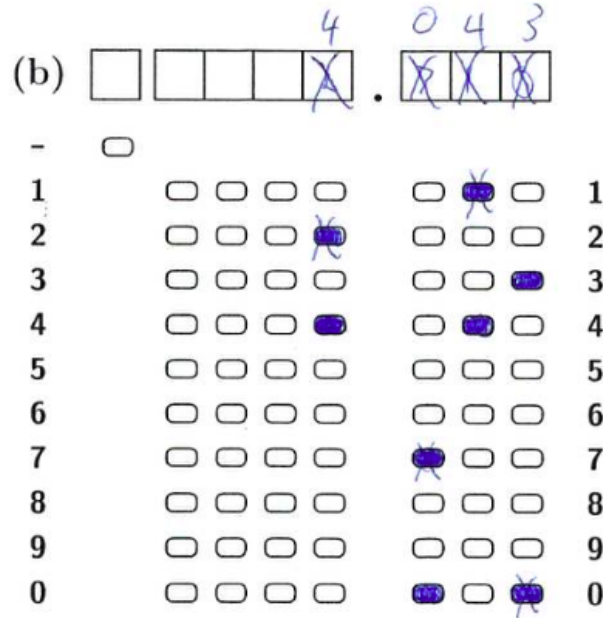


Figure 5.3: Crossed-out character that confused the grading system.

because it only had characters in with no bubbles coloured in. Thus, even though the system identified every character correctly, it had a too low percentage confidence in its answer and reported it to the clash list.

5.1.2.2 Incorrect automatic graded results

There were no automatically graded answers that were done incorrectly.

5.1.3 Analysis of results

The systems both had the same number of clash list tests from these 25 images. The complete system took on average 2.011 seconds to grade a test. The student number PGM consumes most of that time, 1.5 seconds, to infer the correct student number. The complete system had no incorrectly automatically graded answers, in contrast to the 4 graded incorrectly using the basic system. A reason to this is attributed to more evidence that are considered in the complete system. Therefore only if the evidence match up will the system be certain enough to accept its answer as the correct one. In the next section the complete system is used to grade a tutorial test written by all the students.

5.2 Grading of tutorial tests

In this section the complete system is tested on a tutorial written by all the students in the class. The final version of the complete system was used in grading the students' tests. Student feedback was recorded to find tests that were graded incorrectly and used as the grader's results.

5.2.1 Marking statistics

For these tutorial test an average marking time per test of 2.01 seconds was recorded.

5.2.2 Clash list

There were in total 57 clashes in the 888 tests. These 57 clashes are categorised in [Table 5.2](#).

5.2.3 Incorrect automatic graded results

The students were asked to report any results that were marked incorrectly by the system for this particular test. These results are all the results that the system decided to accept the automatically graded answer, and in doing so, graded the test incorrectly.

Only 1 result was reported where the software automatically marked the answer in an incorrect manner. The correct answer was -95.0 and the system marked the answer as 95.0, as shown in [Figure 5.4](#). There may still be tests that were marked incorrectly, but these tests were not reported. In comparison there were 15 known mistakes made in the first tutorial using the basic system.

For a more detailed description of the results from grading all 4 tutorial tests, refer to [Appendix E](#).

5.2.4 Conclusion

In conclusion, it is noted that the complete system, with its machine learning capabilities, reduces the number of tests that the user must mark manually, in comparison with the previous basic system. For this complete system, about 7.5% of the tests in the tutorial had to be marked manually, due to the system being unsure of the answers. The system

Table 5.2: Description and quantity of clashes in the different categories.

Number of tests in category	Category description
31	In these tests the system determined the right values, but was unsure about its answer. Some of the cases were when the student number was only filled in the character box. The software always identified the correct student number, but was still uncertain about the answer.
15	In these tests the system could not distinguish between a crossed-out answer and correct answer. This is due to the crossed-out answer being interpreted as a filled in answer.
8	These tests have an answer with only character information in them. The system attempted to identify each answer, but made a mistake in at least one of the digits.
2	These images contained blank papers that did not include test templates.
1	In this test the grid of the test paper could not be found and thus test could not be marked.

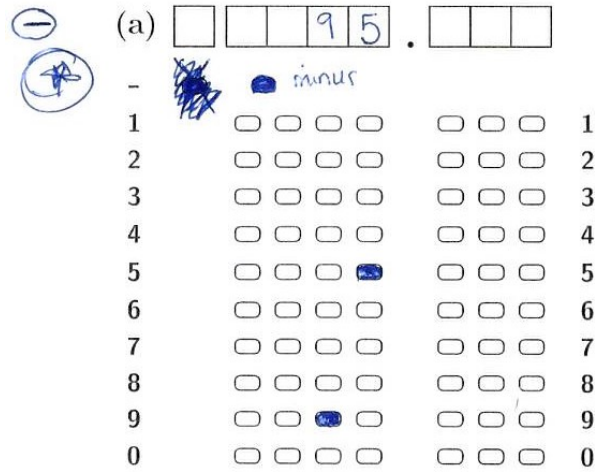


Figure 5.4: Incorrectly identified answer as 95.

does however grade 99.9% of the tests correctly, when the system decides to grade them automatically. The reason for this can be attributed to the fact that the system correlates two pieces of evidence to predict the correct answer. Thus, both the character and bubble information had to be interpreted incorrectly for the system to automatically grade an answer incorrectly.

The system could grade 92.5% of all the test correctly and automatically. From the remaining 7.5% only 1 test or 0.1% of the tests was found to be graded incorrectly and the other 7.4% of the tests were sent to a clash list. The system classified most of the clash list tests correctly. If the clash list threshold level of the system is set to 0 the system could possibly have graded 97.1% of the tests correctly.

Chapter 6

Summary and conclusions

6.1 Project summary

In this project an automatic test grading system was developed with the aim of grading student tests using a special template. Initially, research was done on existing methods for grading tests automatically. It was found that traditionally only image processing methods are used to grade bubbles on a paper. For this project additional machine learning capabilities was built into the system. This allows for a better estimation of what the student intended to write down on the paper.

6.2 How this final year project benefits society

In the African society there is a great number of individuals who do not have access to quality educational opportunities. The educational systems these individuals do have are normally not up to standard and have limited teaching assistance. Educators who are available are not always accessible to learners to provide quality education. Automatic Mark Recognition software such as the one developed in this project allows for a large number of tests to be assessed automatically and accurately in a short time span. This assist educators in handling bigger classes and thus provide more learners the opportunity for a better education.

6.3 What the student learned

During the execution of this project, the student learned that time management is important to complete a project of this scale. Time management also allows an individual to continuously assess how he/she is doing with respect to a schedule. This not only increases performance, but also self-confidence in the final product. Finally, the student learned how to develop a software package in a professional environment. This project also allowed the student to gain a basic knowledge on a broad range of fields including image processing, neural networks and probabilistic graphical models.

6.4 Future improvements

To increase the speed of grading tests it should be considered to use Stellenbosch's custom PGM library, implemented in C++. By continuously updating the estimated orientation of the template as more bubble contours are classified, accuracy in finding these bubbles can be increased. Further increases in test grading speed can be achieved by only doing image processing on the expected locations of the bubbles. This will bring some extra technical hurdles, but can increase the software's speed. Lastly the accuracy of the character recognition neural network can be increased by making use of Generative Adversarial Networks (GAN) to train the network on actually classified test results.

6.5 Summary and conclusions

For 890 tests the system takes approximately 30 minutes to grade the tests. This time is acceptable for the department, since the test format allows for more flexible answers, compared to traditional multiple choice tests. Additionally a student number can be identified by only referring to the characters written in the student number box. The system has an overall accuracy of 97.1% with only automatic grading. An additional feature is implemented that transfers tests which the system is uncertain about, to a user to manually grade. Combined with the human operator, only 1 test in every tutorial session are graded incorrectly. In combination with the manually checked tests, the system thus obtains a 99.9% accuracy on grading tests correctly, while still allowing students greater flexible in the methods of answering these tests.

References

- ANKAN, A. & PANDA, A. (2015). pgmpy: Probabilistic graphical models using python. Proc. of the 14th Python in science conf. [10](#)
- EDORAS (2017). The inverse radon transform. Image Processing Toolbox. [ix](#), [14](#)
- GOOGLE (2017). Deep mnist for experts. [9](#), [25](#), [26](#)
- IVETIC, D. & DRAGAN, D. (2003). Projections based omr algorithm. [8](#)
- KARPATHY (2017). Cs231n convolutional neural networks for visual recognition. [ix](#), [24](#)
- MATHWORKS (2017). Detect lines using the radon transform. [8](#)
- NIELSEN, M.A. (2015). *Neural Networks and Deep Learning*, vol. 1. Determination Press. [24](#)
- PATEL, N.V. & PRAJAPATI, G.I. (2003). Various techniques for assessment of omr sheets through ordinary 2d scanner: A survey. *International Journal of Engineering Research & Technology (IJERT)*, **4**. [8](#)
- ROSEBROCK, A. (2016). Bubble sheet multiple choice scanner and test grader using omr, python and opencv. [9](#)
- TENSORFLOW (2017). Mnist for ml beginners. [ix](#), [23](#)
- VIJAYAFORM (2017). Omr sheet. [ix](#), [7](#)
- YANN LECUN, C.C. & BURGESS, C.J. (1998). The mnist database of handwritten digits. [26](#), [63](#)

Appendix A

Project plan

The work layout plan for this project is shown in Figure [A.1](#). This plan was last updated on 23 October 2017. The idea behind this plan was to allow the student to have a time frame to plan and measure the project's progress against.

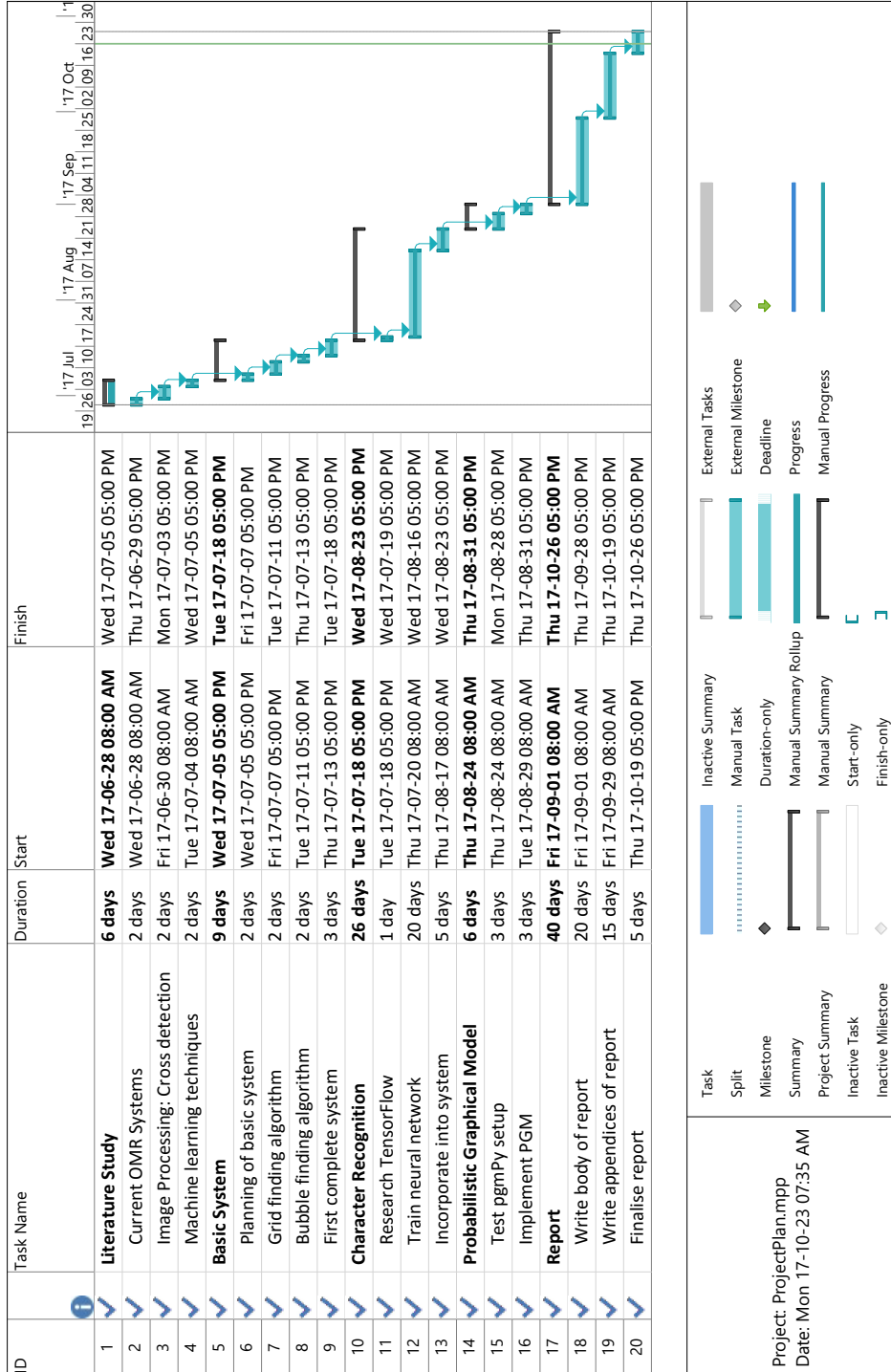


Figure A.1: Project plan for the final year project.

Appendix B

Outcome compliance

Tables [B.1](#) and [B.2](#) describes the required ECSA Exit Level Outcomes and how this project adheres to these outcomes.

Table B.1: Description of exit level outcomes and how this project adhere to them.

Outcome	Reference	Description
1. Problem solving: Identify, formulate, analyse and solve complex engineering problems creatively and innovatively.	<i>1,2,3,4</i>	In Chapter 1 the project was analysed and formulated into discrete problems. This allowed the problem to be solved by solving small sub-problems. Further improvements using two machine learning techniques were also introduced above the normal OMR systems.
2. Application of scientific and engineering knowledge: Apply knowledge of mathematics, natural sciences, engineering fundamentals and an engineering speciality to solve complex engineering problems.	<i>1, 3 & 4</i>	Engineering knowledge obtained in the degree allowed for the understanding and implementation of mathematical concepts such as Neural Networks, Radon transforms and Probabilistic Graphical Models. Further computer programming knowledge was also used to implement the software in an efficient manner.
3. Engineering Design: Perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.	<i>1, 3 & 4</i>	In the design process of implementing Image Processing and Neural Networks a procedural process is demonstrated. Implementing the two Probabilistic Graphical Models demonstrates a more creative approach to this problem.
5. Engineering methods, skills and tools, including Information Technology: Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.	<i>1.4, 1.5 & 5</i>	Engineering methods used in this project include agile development, software version control, through Git, and project scheduling to efficiently manage time.

Table B.2: Description of exit level outcomes and how this project adhere to them.

Outcome	Reference	Description
6. Professional and technical communication: Demonstrate competence to communicate effectively, both orally and in writing, with engineering audiences and the community at large.	<i>All</i>	This outcome is demonstrated in the report and oral presentation.
9. Independent Learning Ability: Demonstrate competence to engage in independent learning through well-developed learning skills.	<i>2,3 & 4</i>	Independent learning was continuously required in understanding new concepts in each stage of the project's design process. These concepts includes Neural Networks, Radon transforms, Probabilistic Graphical Models and Image Processing, which were not taught in the engineering degree.

Appendix C

Mathematical and graphical description of system

In this appendix a graphical and mathematical derivation for the system is given. It is shown through mathematical derivation how the pgmpy software can predict the student number and answer, given the probability distributions and evidence. It should be noted that the software exploits additional methods in calculating these values in an efficient manner.

C.1 High-level overview

As described in Chapter 1, the system can fundamentally be represented with 6 information nodes. These nodes are shown in Figure C.1. The student has 5 pieces of information

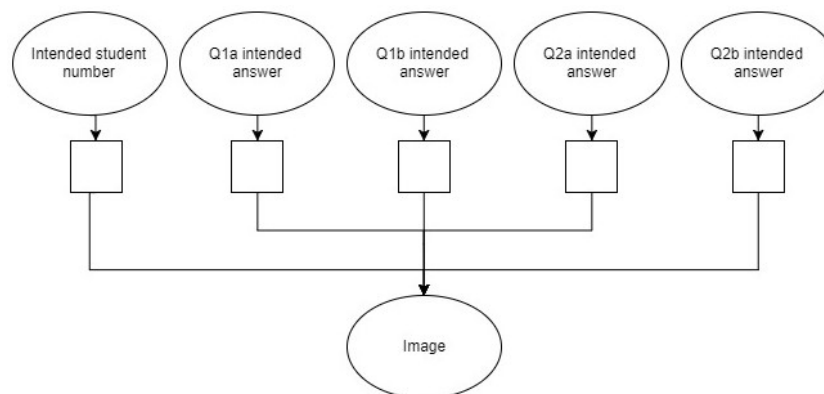


Figure C.1: System overview.

that he/she wants to portray, signifying the first 5 nodes. Those 5 nodes give rise to the image, representing the last node. At its core the system is tasked with inferring two types of conditional probabilities namely $P(S/I)$ and $P(A/I)$. The random variables S and A represent all the possible values that the student number and answers can possibly have. The image, I , is also a random variable representing the total number of possible states the image can take. Each image has a width and length of 1 240 by 1 754 pixels. For every pixel there are 256 possible values, ranging from 0.0 to 1.0. Thus the number of possible images are in the range of $1\,240 \times 1\,754 \times 256$. To practically represent this, more detailed derivations and assumptions is needed. These derivations are described in the next sections.

C.2 The student answer

In Section 4.2.3, it was determined that the student’s answer can be calculated by combining the intended sign and digits of each column, which is calculated separately. This is attributed to the fact that these digits are independent of one another, as illustrated in Figure C.2.

The intended digit in a certain column is not influenced by what the values in the other columns are. This independence property is thus described by

$$P(A/I) = P(S_i/I)P(D_1/I)...P(D_7/I), \tag{C.1}$$

where A and I again represent an answer for a specific question and the image. The random variable S_i represents the sign of the answer. $P(D_i/I)$ represents the probability of each of the 10 digits being written down in column i .

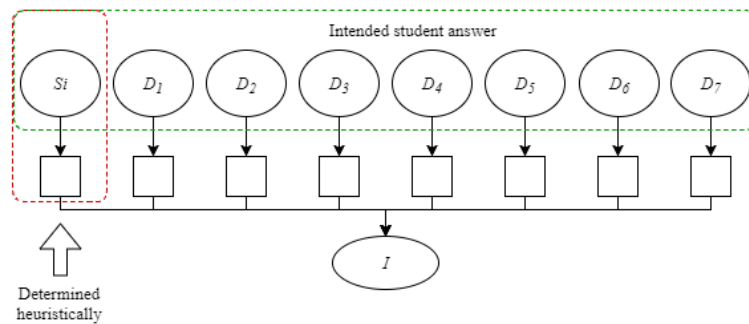


Figure C.2: Graphical setup of determining student answer.

To find the most probable answer, only $P(Sn/I)$ and $P(D_{1-7}/I)$ need to be calculated. Using image processing techniques described in Section 3, $P(Si/I)$ can be determined heuristically by determining the probability of the bubble being coloured in, underneath the sign. Thus the only values that still need calculate is $P(D_{1-7}/I)$, as derived in the next section.

C.3 The intended digit

In determining an intended digit there are 11 information nodes to use as evidence. These nodes represent the 10 bubbles and character block, as shown in Figure C.3. Extra nodes are also added to symbolise the intended bubbles as described in Section 4.2.2. The first bubble might sometimes incorrectly be associated with digit 0. Thus even if the student intended the digit 0 as an answer, the intended bubble of that student was actually the bubble for digit 1.

The probability of each column's digit is describe by $P(D_i/I)$. This value again represents the probability of each of the 10 digits being written down in column i . We know that the digit evidence is calculated heuristically using image processing. Each bubble thus has an evidence value with 4 possible states, namely not filled-in, crossed out, partially filled-in and completely filled-in. Thus the estimate of the intended digit is now represented by $P(D_i/I, BE_i, CE_i)$. BE_i and CE_i represents all the bubble and character evidence for that digit. As seen in Figure C.4, D_i and I becomes independent from one another given the values of BE_i and CE_i . Thus the probability of the intended

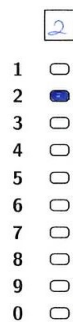


Figure C.3: Column with evidence that gets considered for the calculation of an intended digit.

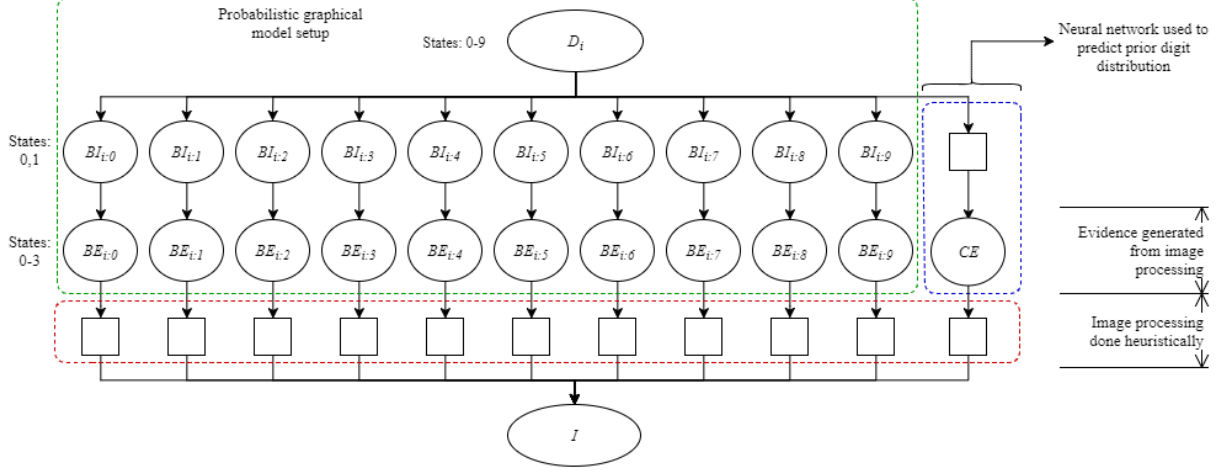


Figure C.4: Graphical setup of determining intended digit.

digit is now given by $P(D_i/BE_i, CE_i)$. BE_i is further described by

$$P(BE_i) = P(BE_{i:0}, BE_{i:1}, \dots, BE_{i:9}), \quad (\text{C.2})$$

where $P(BE_{i:j})$ represents the probability of the bubble evidence at bubble j in digit i . BI_i is then represented by

$$P(BI_i) = P(BI_{i:0}, BI_{i:1}, \dots, BI_{i:9}), \quad (\text{C.3})$$

where $P(BI_{i:j})$ represents probability of the intended bubble at bubble j in digit i .

$P(D_i/BE_i, CE_i)$ can now be represented by

$$P(D_i/BE_i, CE_i) = \sum_{\forall BI_i} P(D_i, BI_i/BE_i, CE_i) \quad (\text{C.4})$$

$$= \sum_{\forall BI_i} \frac{P(BE_i, CE_i/D_i, BI_i)P(D_i, BI_i)}{P(BE_i, CE_i)}. \quad (\text{C.5})$$

In Equation C.5, the intended bubble term is brought in by making use of the sum rule. Bayes' rule is then applied. In Figure C.4, BE_i and CE_i is seen to be independent when D_i or BI_i is known. Further it is observed that BE_i is only dependent on BI_i and CE_i only dependent on D_i . Thus by applying an additional product rule it is determined that

$$P(D_i/BE_i, CE_i) = \sum_{\forall BI_i} \frac{P(BE_i/BI_i)P(CE_i/D_i)P(BI_i/D_i)P(D_i)}{P(BE_i, CE_i)}. \quad (\text{C.6})$$

Bayes' rule also provides us with,

$$P(CE_i/D_i) = \frac{P(D_i/CE_i)P(CE_i)}{P(D_i)}. \quad (C.7)$$

By factoring out all the constant terms out of the summation the equation reduces to,

$$P(D_i/BE_i, CE_i) = \frac{P(CE_i)}{P(BE_i, CE, i)} \sum_{\forall BI_i} P(BE_i/BI_i)P(BI_i/D_i)P(D_i/CE_i). \quad (C.8)$$

The constant terms gets ignored in the pgmpy package due to them only being normalizing terms. Once the summation has been calculated the software simply normalizes the resulting values without needing those terms. Thus only $P(BE_i/BI_i)$, $P(BI_i/D_i)$ and $P(D_i/CE_i)$ are needed.

$P(BE_i/BI_i)$ and $P(BI_i/D_i)$ are both terms that is deduced from training. The final term that is needed is $P(D_i/CE_i)$. This term is efficiently represented through the use of a neural network. Thus the PGM system can successfully infer the digit probabilities and thus the student answer with these 3 distributions specified. A derivation on the student number probability is discussed next.

C.4 The student number

As state in Section 4.2.4, the assumption of independence between digits does not hold in the case of a student number. The reason for this is, because every 8 digit number is not equally likely to be a student number. Only student numbers that are valid needs to be considered as a possible state that the student number node can take. This node has approximately 900 states, depending on the number of student numbers. The student number graph can be seen in Figure C.5.

A derivation for $P(S/I)$ is needed next. As stated previously, S represents a random variable over all the possible student numbers. I again represents the image. We know that the digit evidence is calculated heuristically using image processing. Thus the estimate of the student number is now represented as $P(S/I, DE)$. DE now represents all the bubble and character evidence. As seen in Figure C.5, S and I becomes independent from one another given the values of DE . Thus the probability of the intended digit is now given by $P(S/DE)$.

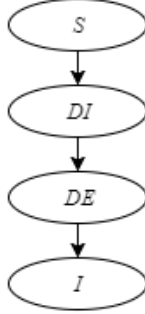


Figure C.5: Graphical setup of determining student number.

DE is further described by

$$P(DE) = P(BE_1, CE_1, BE_2, CE_2, \dots, BE_8, CE_8). \quad (C.9)$$

DI is described by

$$P(DI) = P(D_1, D_2, \dots, D_8). \quad (C.10)$$

$P(S/DE)$ can now be represented by

$$P(S/DE) = \sum_{\forall DI} P(S, DI/DE) \quad (C.11)$$

$$= \sum_{\forall DI} \frac{P(DE/S, DI)P(S, DI)}{P(DE)}. \quad (C.12)$$

In Equation C.12, the digit intended term, DI , is brought in by making use of the sum rule. Bayes' rule is then applied as shown.

In Figure C.5, DE is shown to be independent of S if DI is known and thus,

$$P(S/DE) = \sum_{\forall DI} \frac{P(DE/DI)P(DI/S)P(S)}{P(DE)}. \quad (C.13)$$

Finally from the digit and student number graph structure the following independence properties are also known,

$$P(DE/ID) = P(BE_1/BI_1)P(CE_1/D_1)...P(BE_8/BI_8)P(CE_8/D_8) \quad (C.14)$$

$P(S)$ can be initialized as an equal distribution, because every student number has the same likelihood of being in a given test. $P(DI/S)$ are values that are trained from data using the independence property,

$$P(DI/S) = P(D_1/S)...P(D_8/S). \quad (\text{C.15})$$

This value symbolizes the probability that the user intended to write down a digit given that student number. If the first digit of the student number is 1, the first intended digit will have a high probability of being 1. Thus these two random variables are strongly correlated. The only values that still need to be calculated are thus $P(DE/ID)$. Using the independency property of

$$P(DE/ID) = P(BE_1/BI_1)P(CE/D_1)...P(BE_8/BI_8)P(CE/D_8), \quad (\text{C.16})$$

the intended digit model's conditional distributions can be used. The two PGM models can now be fully defined and used to infer the intended student entries from an image.

Appendix D

Systems diagrams and software

D.1 Software

The Git repository for this complete system can be found at <https://github.com/DriesSmit/AutoTestMarker/tree/master/Software>.

D.2 Interface

The software's main interface and clash list are show in Figure [D.1](#) and Figure [D.2](#), respectively.

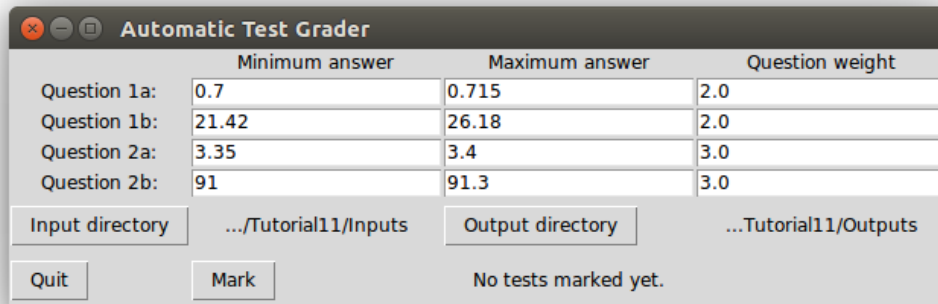


Figure D.1: Main interface of the test grader.

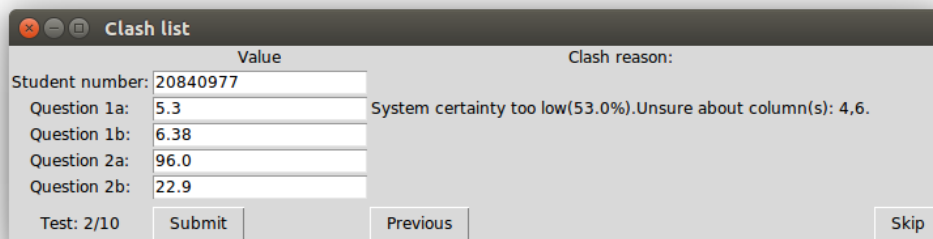


Figure D.2: Clash list interface of the test grader.

D.3 Templates

The original template is shown in Figure [D.3](#). Two additional templates have also been developed and implemented for the department. These templates provides the option of grading numbered answered questions as well as multiple choice questions. These templates are shown in Figure [D.4](#) and Figure [D.5](#).

Stellenbosch University: Applied Mathematics B154 Test /...../20 ...

Surname : US Number →

Enter your student number, both as numbers and by filling in the ovals below the student number.
 Give your answer for each question, both as numbers and by filling in the corresponding oval below each box.
 Use only pen to colour the ovals.

 Student's signature

- 1 1
- 2 2
- 3 3
- 4 4
- 5 5
- 6 6
- 7 7
- 8 8
- 9 9
- 0 0

Question 1:

- (a) .
 -
 1 1
 2 2
 3 3
 4 4
 5 5
 6 6
 7 7
 8 8
 9 9
 0 0
- (b) .
 -
 1 1
 2 2
 3 3
 4 4
 5 5
 6 6
 7 7
 8 8
 9 9
 0 0

Question 2:

- (a) .
 -
 1 1
 2 2
 3 3
 4 4
 5 5
 6 6
 7 7
 8 8
 9 9
 0 0
- (b) .
 -
 1 1
 2 2
 3 3
 4 4
 5 5
 6 6
 7 7
 8 8
 9 9
 0 0

Figure D.3: Original template focussed on numbered answered questions.

Stellenbosch University: Applied Mathematics B154 Test /...../20 ...

Surname : US Number →

Enter your student number, both as numbers and by filling in the ovals below the student number.

Question 1: Give your answer for each question, both as numbers and by filling in the corresponding oval below each box.

Question 2: Indicate your answer to each question by filling in the appropriate oval.

Use only pen to colour the ovals.

.....
Student's signature

1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1
2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	2
3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	3
4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	4
5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	5
6	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6
7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	7
8	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	8
9	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	9
0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	0

Question 1:

(a) .

-

1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1
2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	2
3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	3
4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	4
5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	5
6	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6
7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	7
8	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	8
9	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	9
0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	0

(b) .

-

1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1
2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	2
3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	3
4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	4
5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	5
6	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6
7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	7
8	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	8
9	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	9
0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	0

Question 2:

	A B C D E F		
(a)	<input type="radio"/>	(a)	<input type="radio"/>
(b)	<input type="radio"/>	(b)	<input type="radio"/>
(c)	<input type="radio"/>	(c)	<input type="radio"/>
(d)	<input type="radio"/>	(d)	<input type="radio"/>
(e)	<input type="radio"/>	(e)	<input type="radio"/>
(f)	<input type="radio"/>	(f)	<input type="radio"/>
(g)	<input type="radio"/>	(g)	<input type="radio"/>
(h)	<input type="radio"/>	(h)	<input type="radio"/>
(i)	<input type="radio"/>	(i)	<input type="radio"/>
(j)	<input type="radio"/>	(j)	<input type="radio"/>

	A B C D E F		
(k)	<input type="radio"/>	(k)	<input type="radio"/>
(l)	<input type="radio"/>	(l)	<input type="radio"/>
(m)	<input type="radio"/>	(m)	<input type="radio"/>
(n)	<input type="radio"/>	(n)	<input type="radio"/>
(o)	<input type="radio"/>	(o)	<input type="radio"/>
(p)	<input type="radio"/>	(p)	<input type="radio"/>
(q)	<input type="radio"/>	(q)	<input type="radio"/>
(r)	<input type="radio"/>	(r)	<input type="radio"/>
(s)	<input type="radio"/>	(s)	<input type="radio"/>
(t)	<input type="radio"/>	(t)	<input type="radio"/>

Figure D.4: Template allowing for numbered and multiple choice answers.

Stellenbosch University: Applied Mathematics B154 Tut 9 2 October 2017

Surname : US Number →

Enter your student number, both as numbers and by filling in the ovals below the student number.

Indicate your answer to each question by filling in the appropriate oval. **Only fill one oval for each question.**

Make dark marks that fill the oval completely.

.....
Student's signature

- | | | | | | | | | | | |
|---|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|---|
| 1 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 1 |
| 2 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 2 |
| 3 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 3 |
| 4 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 4 |
| 5 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 5 |
| 6 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 6 |
| 7 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 7 |
| 8 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 8 |
| 9 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 9 |
| 0 | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 0 |

Section A:

- | | A | B | C | D | E | | A | B | C | D | E | |
|------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|------|------|-----------------------|-----------------------|-----------------------|-----------------------|------|
| (1) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (1) | (26) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (26) |
| (2) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (2) | (27) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (27) |
| (3) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (3) | (28) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (28) |
| (4) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (4) | (29) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (29) |
| (5) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (5) | (30) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (30) |
| (6) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (6) | (31) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (31) |
| (7) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (7) | (32) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (32) |
| (8) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (8) | (33) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (33) |
| (9) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (9) | (34) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (34) |
| (10) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (10) | (35) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (35) |
| (11) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (11) | (36) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (36) |
| (12) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (12) | (37) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (37) |
| (13) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (13) | (38) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (38) |
| (14) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (14) | (39) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (39) |
| (15) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (15) | (40) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (40) |
| (16) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (16) | (41) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (41) |
| (17) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (17) | (42) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (42) |
| (18) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (18) | (43) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (43) |
| (19) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (19) | (44) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (44) |
| (20) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (20) | (45) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (45) |
| (21) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (21) | (46) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (46) |
| (22) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (22) | (47) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (47) |
| (23) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (23) | (48) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (48) |
| (24) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (24) | (49) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (49) |
| (25) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (25) | (50) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | (50) |

Figure D.5: Template focussed solely on multiple choice type questions.

Appendix E

Validation and results

This appendix described additional test results obtained from experiments done on the automatic test grading system.

E.1 All tutorial results

E.1.1 Overview

The automatic test grader developed in this study was successfully used to grade 11 tutorial test in 2017. The number of students per tutorial varies due to students having valid excuses. On average 889 tests were written per tutorial. The results of these tutorials can be seen in [Table E.1](#) and [Table E.2](#).

It is possible that there are tests with mistakes that was not reported. To calculate the probability of this happening the 6th tutorial test was manually checked for mistakes. None were found. Thus it is assumed that tests that have mistakes in, but is not reported, are unlikely and is not incorporated into the calculations.

In the 11 tutorials an average of 99.3% of tests are estimated to be graded correctly, as no corrections were made by students. This percentage is lower that expected, because the basic system's averages are also included. When only taking the complete grading system's result, an average of correctly grading tests are calculated to be 99.9%.

Table E.1: Description of tutorial results.

Tutorial number	Percentage tests graded correctly	Reason for results
The basic system is now implemented.		
Tutorial 1	98.4% (14 mistakes)	The system has a problem with identifying crossed-out answers.
Tutorial 2	98.8% (11 mistake)	The system still had a problem with crossed-out answers. This problem was subsequently resolved.
Tutorial 3	99.4% (5 mistakes)	The system made a few mistakes with answers with only character information.
Tutorial 4	98.5% (13 mistakes)	A rounding error in the software led to some answers being marked incorrectly.
Tutorial 5	99.3% (5 mistakes)	The system made a few mistakes with answers with only character information.
Tutorial 6	99.7% (3 mistakes)	Again the system made a few mistakes with answers with only character information.

Table E.2: Description of tutorial results.

Tutorial number	Number of tests graded incorrectly	Reason for results
		The complete system, with machine learning, is now implemented.
Tutorial 7	99.9% (1 mistake)	The system classified a crossed-out answer as being coloured in.
Tutorial 8	99.8% (2 mistakes)	The system classified 2 crossed-out answers as being coloured in. This problem was subsequently resolved.
Tutorial 9	100.0% (0 mistakes)	No mistakes where found.
Tutorial 10	99.9% (1 mistakes)	This tutorial was discussed in Chapter 5. The student wrote over the negative sign bubble, confusing the system. This mistake is attributed to the student.
Tutorial 11	100.0% (0 mistakes)	No mistakes where found.

E.2 Deep Convolutional Neural Network results

This section describes the results obtained on testing a trained neural network on a test dataset. Tests are conducted on 3 neural networks trained on different datasets and compared with each other. This testing process is conducted to find the neural network weights that classifies written digits most accurately. The neural networks are tested on a test dataset generated by grading 900 student tests and extracting the character images. The answers from these tests was used to create labels for each digit image. Thus each 28 by 28 pixel digit image has an accompanied label specifying the digit. The dataset contains 16 000 labelled images and is split into a training set of 11 000 digits and a test set of 5 000 digits. An additional dataset, called the MNIST dataset, (Yann LeCun & Burges, 1998), is also used in this process. This dataset contains 60 000 training set digits and a test set of 10 000 digits. Each neural network is tested on both datasets. Every network was trained for 8 hours on the same processor, before being tested. The results of these test is shown in the next section.

E.2.1 Trained on generated database

In a first attempt at training a neural network the generated 11 000 digit training set was used to train the network.

E.2.1.1 Accuracy of network

The test accuracy of the neural network on both test sets are given in Table E.3.

Table E.3: Test results for neural network trained on generated data.

Test dataset	Percentage accuracy
MNIST dataset	94.62%
Generated dataset	92.16%

E.2.1.2 Conclusion on accuracy

The results are promising, but the average on the MNIST dataset is still too low. A standard digit classifier has a MNIST testing accuracy of above 99%. A reason for this accuracy is attributed to the small training set size of the generated dataset. The deep

E.2 Deep Convolutional Neural Network results

neural network thus does not have enough data to accurately model each digit and starts to overfit on the data.

E.2.2 Trained on MNIST database

For the second neural network the MNIST dataset of 60 000 digit was used to train the network.

E.2.2.1 Accuracy of network

The test accuracy of the neural network on both test sets are given in Table [E.4](#).

Table E.4: Test results for neural network trained on MNIST dataset.

Test dataset	Percentage accuracy
MNIST dataset	99.35%
Generated dataset	83.23%

E.2.2.2 Conclusion on accuracy

The resulting 83.23% in classifying the generated data is low. The reason for this low accuracy is attributed to the MNIST dataset having only centred and normalized digits. In the test grader example digits are sometimes written off-centre and are unnormalized. The system mostly corrects for this, but there are cases where the digits cannot be centred. This causes a few off-centred digits that the neural network is not trained to classify.

E.2.3 Trained on mixed database

For the final neural network a combined dataset is used. The 11 000 training digits of the generated dataset and the 60 000 training digits of the MNIST is combined to train the model.

E.2.3.1 Accuracy of network

The test accuracy of the neural network on both test sets are given in Table [E.5](#).

E.2 Deep Convolutional Neural Network results

Table E.5: Test results for neural network trained on combined data.

Test dataset	Percentage accuracy
MNIST dataset	99.1%
Generated dataset	94.35%

E.2.3.2 Conclusion on accuracy

The best result is thus achieved by combining the two datasets. This 94.35% accuracy is high enough for the neural network to provide useful information to the test grading system. This neural network is used as the character recognition unit in the test grader.