

# 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: J.A. du Preez

DATE: October 2017

## Acknowledgements

I would like to acknowledge my study leader, JA du Preez, parents and the engineering class of 2017 for their kind contributing 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.

## Declaration

I, the undersigned, hereby declare that the work contained in this final year project is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

.....

Signature

.....

Date

# Abstract

The aim of this research project is to develop software that can automatically grade tests, written by students on a special template. This template allows the software to extract the students intended answers, after it has been scanned into a digital form. 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 previous answer bubbles must first be erased and new ones coloured in. This takes time and increases the probability that a learner might colour in 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 answer questions using characters and bubbles. In the case of a student number the student does not need to colour in the bubbles at all. The software implemented in this project allows for a decreased time in filling in these templates and thus decreases the number of mistakes made. The methods proposed here thus simplifies the incorporation of OMR templates into the traditional education system.

## Uittreksel

Skryf dalk oor asebief:)

Die doel van hierdie navorsings projek is om sagteware te ontwikkel wat automaties toetse, wat deur studente geskryf is, na te kan sien. Hierdie toetse word elkeen op 'n spesiale templaar geskryf. Die templaar laat die sagteware toe om die student se antwoord te vind nadat dit digitaal gekopieer is. Die standaard metode wat gebruik word in hierdie Optiesemerk-leser sagteware is om van inkleur rooster borrels gebruik te maak. Om 'n fout wat die student gemaak het regtemaak, moet die ou borrels eers uitgevee word and nuwe borrels ingekleur word. Hierdie proses vat baie tyd en vermeerder die kans dat 'n student die borrels verkeerd in kan vul. Die navorsingsprojek poog om hierdie 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 Kunsmatige Neurale Netwerke en Probabilistiese Grafiese Modelle (PGM), kan 'n student nou vroe beantwoord deur karakters en borrels te gebruik. In die geval van 'n studentenommer hoef die student glad nie die borrels eers in te kleur nie. Die sagteware wat in hierdie projek gimplementeer word, veroorsaak 'n verminderde tyd in antwoorde inskryf en verminder dus die aantal foute wat gemaak word. Nuwe metodes soos hierdie kan dus help dat meer OMR sagteware in die tradisionele onderwysstelsels ingebring kan word.

# Contents

<b>Abstract</b>	<b>iii</b>
<b>Uittreksel</b>	<b>iv</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xi</b>
<b>Nomenclature</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
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
<b>2 Literature Study</b>	<b>6</b>
2.1 Existing OMR techniques . . . . .	6
2.1.1 Standard OMR systems . . . . .	6
2.1.2 Finding the template . . . . .	8
2.1.3 Processing a bubble . . . . .	8
2.2 Optical character recognition . . . . .	9
2.2.1 Probabilistic approach . . . . .	9
2.3 Conclusion: System requirements . . . . .	10

<b>3</b>	<b>Image Processing</b>	<b>11</b>
3.1	Orientation Detection . . . . .	11
3.1.1	Initial filtering and orientation detection . . . . .	12
3.1.2	Radon Transform . . . . .	13
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
<b>4</b>	<b>Machine learning approach</b>	<b>19</b>
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 . . . . .	27
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
<b>5</b>	<b>Analysis of results</b>	<b>31</b>
5.1	Results of 25 test cases . . . . .	31
5.1.1	Basic system . . . . .	32
5.1.1.1	Marking statistics . . . . .	32
5.1.1.2	Clash list . . . . .	33
5.1.1.3	Incorrect automatic graded results . . . . .	33
5.1.2	Complete system . . . . .	34
5.1.2.1	Marking statistics . . . . .	34

5.1.2.2	Clash list . . . . .	34
5.1.2.3	Incorrect automatic graded results . . . . .	34
5.1.3	Analysis or results . . . . .	35
5.2	Grading of tutorial tests . . . . .	35
5.2.1	Marking statistics . . . . .	36
5.2.2	Clash list . . . . .	36
5.2.3	Incorrect automatic graded results . . . . .	36
5.2.4	Analysis or results . . . . .	38
<b>6</b>	<b>Summary and conclusions</b>	<b>39</b>
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
	<b>References</b>	<b>41</b>
	<b>A Project plan</b>	<b>42</b>
	<b>B Outcome compliance</b>	<b>44</b>
	<b>C Overview of system</b>	<b>47</b>
C.1	System as a whole . . . . .	47
C.2	Student answer . . . . .	48
C.3	The intended digit . . . . .	50
C.4	Student number . . . . .	52
	<b>D Systems diagrams</b>	<b>55</b>
D.1	Interface . . . . .	55
D.2	Templates . . . . .	56
D.3	DCNN TensorFlow setup . . . . .	60



<b>E</b>	<b>Validation and results</b>	<b>62</b>
E.1	All tutorial results . . . . .	62
E.1.1	Overview . . . . .	62
E.2	Deep Convolutional Neural Network results . . . . .	64
E.2.1	Trained on generated database . . . . .	65
E.2.1.1	Accuracy of network . . . . .	65
E.2.1.2	Conclusion on accuracy . . . . .	65
E.2.2	Trained on MNIST database . . . . .	65
E.2.2.1	Accuracy of network . . . . .	65
E.2.2.2	Conclusion on accuracy . . . . .	65
E.2.3	Trained on mixed database . . . . .	66
E.2.3.1	Accuracy of network . . . . .	66
E.2.3.2	Conclusion on accuracy . . . . .	66

# List of Figures

1.1	Automatic test grading template layout. . . . .	3
1.2	Graphical overview of the system as a whole. . . . .	5
2.1	Standard OMR template with reference blocks on the left, from <a href="#">VijayaForm (2017)</a> . . . . .	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. . . . .	14
3.3	Radon transform applied on a two dimensional space, adapted from <a href="#">Edoras (2017)</a> . . . . .	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 gets 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	Image after final translation and normalization is applied. . . . .	22
4.7	Example image used as input to the neural network, from <a href="#">Tensorflow (2017)</a> . . . . .	23
4.8	Basic structure of a neural network, from <a href="#">Karpathy (2017)</a> . . . . .	24
4.9	Graphical setup for determining the intended digit written by a student. . . . .	28
4.10	Graphical setup of student answer. . . . .	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. . . . .	36
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 the evidence that gets considered for the calculation of an intended digit. . . . .	50
C.4	Graphical setup of determining intended digit. . . . .	51
C.5	Graphical setup of determining student number. . . . .	53
D.1	Main interface of test grader. . . . .	55
D.2	Clash list interface of test grader. . . . .	56
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 just on multiple choice type questions. . . . .	59
D.6	DCNN structural setup diagram, from <a href="#">Google (2017)</a> . . . . .	61

# List of Tables

5.1	Description of 25 evaluation tests. . . . .	32
5.2	Description and quantity of clashes in the different catagories. . . . .	37
B.1	Disreption of Exit level outcomes and how this project adhere to them. .	45
B.2	Disreption of Exit level outcomes and how this project adhere to them. .	46
E.1	Description of tutorial results. . . . .	63
E.2	Description of tutorial results. . . . .	64
E.3	Test results for neural network trained on generated data. . . . .	65
E.4	Test results for neural network trained on MNIST dataset. . . . .	66
E.5	Test results for neural network trained on generated data. . . . .	66

# Nomenclature

## Acronyms

<i>CT</i>	Computed Tomography
<i>DCNN</i>	Deep convolutional neural network
<i>DCNN</i>	Deep convolutional neural network
<i>NN</i>	Neural network
<i>OCR</i>	Optical character recognition
<i>OMR</i>	Optical mark recognition

<i>PGM</i>	Probabilistic graphical model
------------	-------------------------------

## Symbols

$\sigma(z)$	Normalization function in a neural network
$A$	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 a bubble evidence obtained from image processing at an arbitrary 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 $\theta$
$I$	Random variable representing the test image
$n$	Number of bubbles of template sheet
$p(i)$	Probability of digit at index $i$
$S$	Random variable representing the possible student number
$Sn$	Random variable representing the sign of a specific answer
$w_i$	Weight value at index $i$
$x_i$	Input value at index $i$
$z$	Weighted sum of a neuron's inputs and internal variables

# Chapter 1

## Introduction

As modern technology and machine learning techniques advances, it is important for the educational sector to continuously advance 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 a teaching assistants and demies. Students complain 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 proposes 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 in 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 values 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 in answer sheets, which needs 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:

**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-in 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. (Kry die templaar wat nie scrubbles op het nie.)

Any additional assumptions are stated in the appropriate section throughout this project. In the next section the project objectives are laid out.



[illegible]

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. Do a *literature study* on the topics of *Image Processing*, *Computer Vision* and *Character Recognition*. Further a *literature study* on *Neural Networks* and *Probabilistic Graphical models* is also done.
2. Develop a *software application* to allow a *user* to grade a large number (approximately 1000) scanned in student tests automatically.
3. 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.
4. Upgrade the software to allow students to cross out answers instead of having to erase them.
5. Do 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.

6. Use an *agile development methodology* to improve the software in parallel with the grading of weekly tests.
7. Add additional software, using machine learning, to improve the accuracy of the system in grading these test.
8. Add additional software that allows students to specify their student number using 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. Do 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 is working as planned.
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 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 in, is going to 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. These processes are described in later sections. In the next section a literature study on current systems is discussed. For a more detailed mathematical overview of the system, refer to Appendix C.

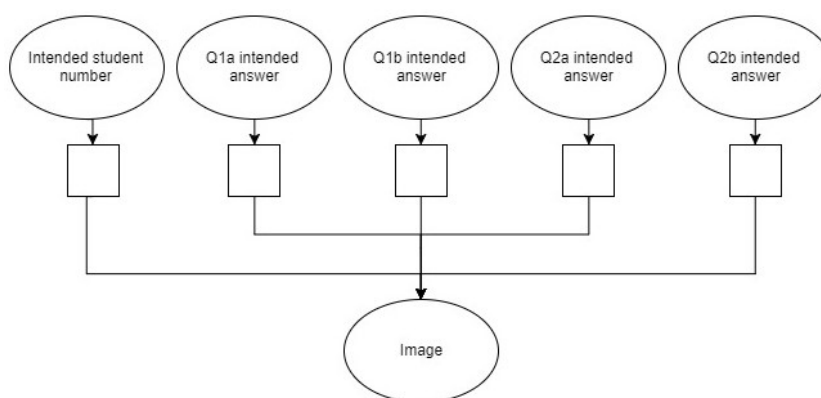


Figure 1.2: Graphical overview of the system as a whole.

# Chapter 2

## Literature Study

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

### 2.1 Existing OMR techniques

An OMR system is a piece of software that is used to extracted hand written 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 needed. The biggest drawback of these systems is that information can only be portrayed in a limited manner, due to the bubbles. On OMR templates there is a grid of bubbles that allows a user to choose between different options to answer. Optical Mark Recognition systems are thus excellent for the grading of multiple choice type questions.

#### 2.1.1 Standard OMR 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

Figure 2.1: Standard OMR template with reference blocks on the left, from [VijayaForm \(2017\)](#)

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 an test, as stated in [Ivetic & Dragan \(2003\)](#). The first step is to determine the grid within 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 location of the use of colour in bubbles. Normally some preprocessing on a blank template is done beforehand to aid in locating the bubbles. Once the bubbles were 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 is described in more detail next.

### 2.1.2 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 bubble 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, as described in [MathWorks \(2017\)](#). A Radon transform provides a way of representing an image as a summation of different line integrations, as is discussed in [Section 3.1.2](#).

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 bubbles is described.

### 2.1.3 Processing a bubble

Once an estimated location for each bubble is known, the next step for an OMR system is to process these bubbles. As stated in [Patel & Prajapati \(2003\)](#), a basic image processing method to classify bubbles is to simply add the number of coloured in pixels. 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.

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, as described in [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 looking at the shape of a contour it can be determined if a bubble is crossed out or not. (Verander die image na een wat 'n contour om het)

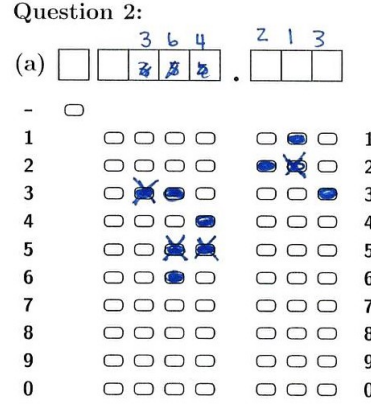


Figure 2.2: Contours found around corrected answer.

## 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 doing OCR is using TensorFlow. This method is described in [Google \(2017\)](#). TensorFlow is a Python library, but allows the build 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 a effective way to produced the best estimate of the intended student entries. Once a bubble and character is 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 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 underlining illness behind the decease is. In this project a PGM can be used to estimate the underling student entries given the evidence presented. The library used in [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 classified digits in the TensorFlow library space. Once all these evidence are acquired a Probabilistic Graphical Model was found to be a perfected method in predicting the estimated entry answer of the student. In Chapter 3, 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 test. In this chapter the core techniques behind processing these answer sheets, using image processing, is 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 is implemented. These approaches are discussed in [Chapter 4](#).

### 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 in answer sheet, is finding the orientation of the template in the image. This can be done by finding two or more reference point 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.

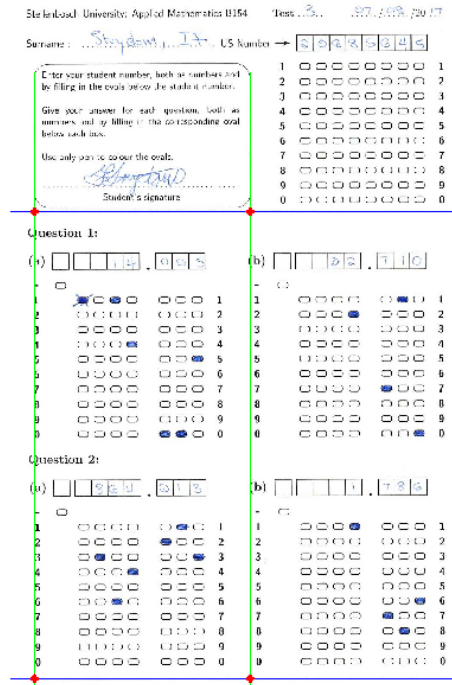


Figure 3.1: Four markers found from applying Radon transforms.

The reason these lines are chosen as references, is because a Radon transform can easily be applied to locate them, as discussed in Section 3.1.2. Therefore the software can successfully find the template even though one of the four lines are 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, it is a good idea to quickly check if the image might be upside down. This is done to make it easier to find the orientation afterwards. This requires some initial image filtering is firstly required and is 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 five steps:

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 analysis 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 size and aspect ratios desired.
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.

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 now be to determine the coordinates of the answers the student wrote down. The first step to achieve this is to locate the template in the image as discussed in the following section.

#### 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 is 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  $\theta$  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  $\theta$ .



### 3.1 Orientation Detection

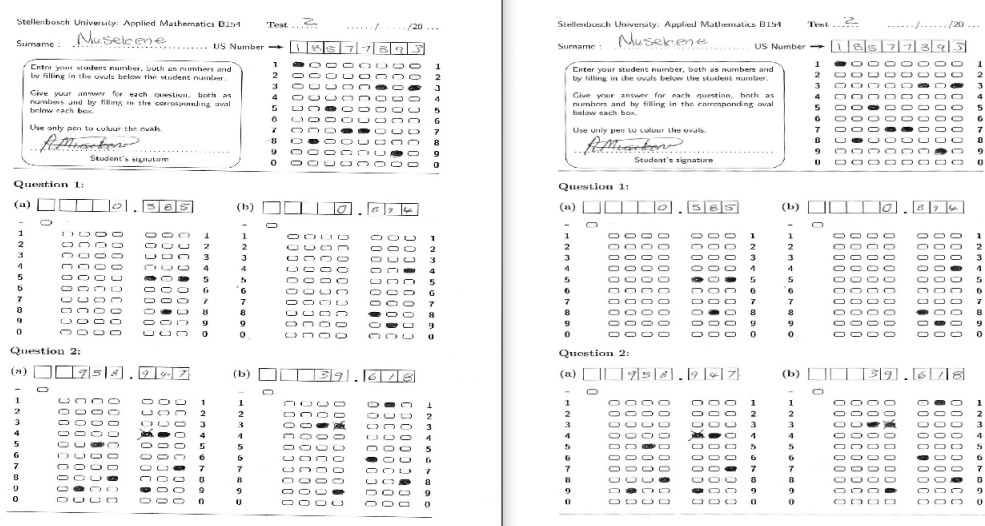


Figure 3.4: Result in rotation after applying radon transform.

#### 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 determine where the 2 horizontal and 2 vertical lines are located, as mentioned in Section 3.1.

The Radon transform's values are calculated between the angles  $85^\circ$  and  $95^\circ$  to find the first two horizontal lines. It is assumed that the image will not be rotated by more than  $5^\circ$  in either direction. Black lines will cause a spike to appear on the Radon transform for the angle where it is summing parallel with that perpendicular black line. By finding the transform angle that has this maximum value, the rotation of the template can be found. The two maximum values that is recorded at this angle is then recorded 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 a angle  $90^\circ$  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

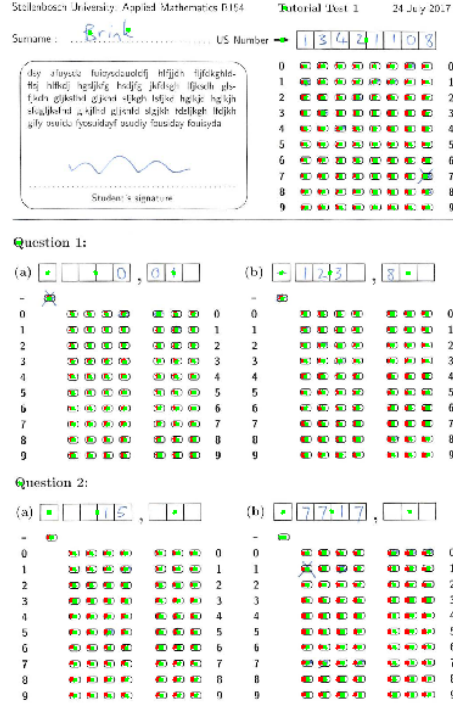


Figure 3.5: Detection of contours in image and estimation of bubble locations.

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. This is discussed in the next section.

## 3.2 Bubble detection and processing

The location of each bubble in the image is found by simply taking the contour closest to that bubble's estimated location. This can be 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 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 an estimated area is calculated where the bubble is most likely to be located. This becomes problematic if the system needs to know if the bubble has been crossed out, as only data about pixel intensities are present.

Using the contour information also provides information on the shape of the bubble entry. 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 an 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 and crossed out. An additional category of partially filled in will also be introduced, as it aids grading of tests where students write lightly. An algorithm to determine what bubble was chosen can now be described as follows:

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 are 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 gets 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 one critical piece of evidence that has not yet been observed in this basic system. This information is 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. Next 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 down in the designated boxes, as shown in Figure 4.1. A machine learning approach, called a neural network, 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 grey scaled image must first be found for each digit inside the test sheet. Image processing is required to achieve this, as described in the next section.

## 4.1 Character recognition using a neural network

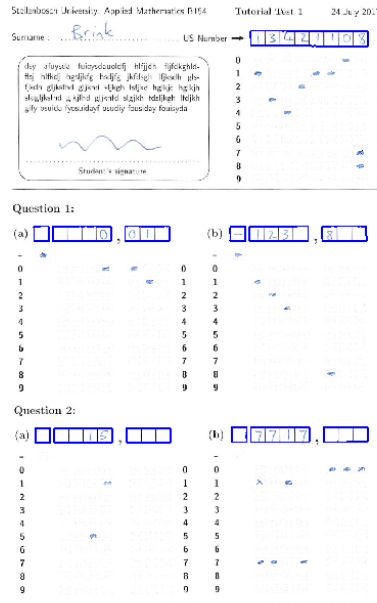


Figure 4.1: Image showing found contours for boxes used for character recognition.

### 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 locations of each digit. Once this is done, the digits is processed and a corresponding 28 by 28 image, best representing that digit, can be created. This is done in the following 6 steps as seen below:

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 the image 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  $0^\circ$  and  $90^\circ$  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

---

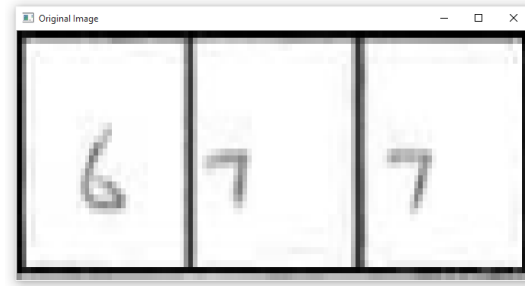


Figure 4.2: The box contour found is normalized to form a rectangular shape.

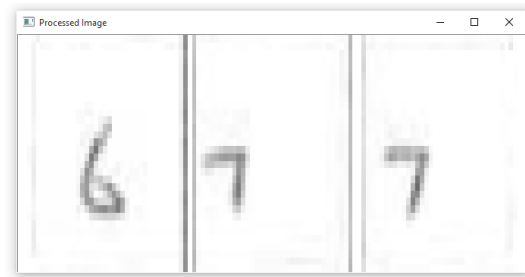


Figure 4.3: Box after black lines gets filtered out, found using a Radon transform.

4. Use the indexes received from the Radon transform, of the positions of 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. This algorithm is developed and implemented using a breadth first search technique to cluster the image into different segments. The algorithm works by first searching 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 gets assigned to a segment. A segment is thus classified as a local region that does not connect to any other segment through pixels higher than a threshold. The segment that most likely represents the digit is then extracted, as shown in Figure 4.4.
6. The area that the segment occupies is then calculated, as shown in Figure 4.5
7. Next the image 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

---

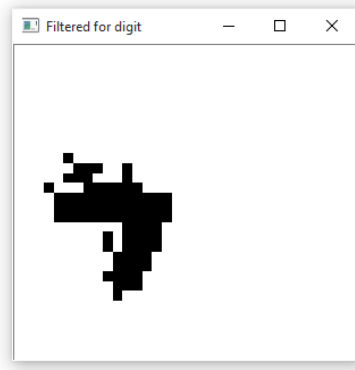


Figure 4.4: Custom segmentation algorithm used to find the main cluster in the remaining image.

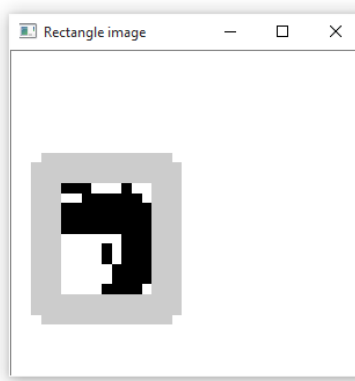


Figure 4.5: Area block drawn around the segment most probable to belong to the digit.

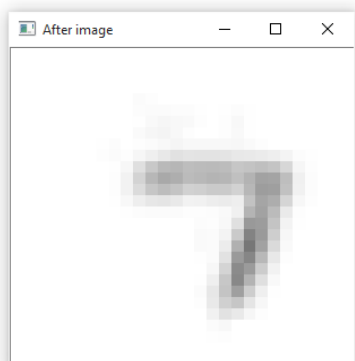


Figure 4.6: Image after final translation and normalization is applied.



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

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 is total of 784 input values. Each digit image is thus 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 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 neural network lies in that fact that these neurons have internal values that can be tweaked depending on the characteristics the user wants the network to approximate. This therefore allows complex functions to be trained onto such networks.

For this project, a neural network 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 will take a two dimensional array, representing a greyscaled image, as input to the network. Figure 4.7 illustrates an example input of a neural network 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 neural networks are discussed.

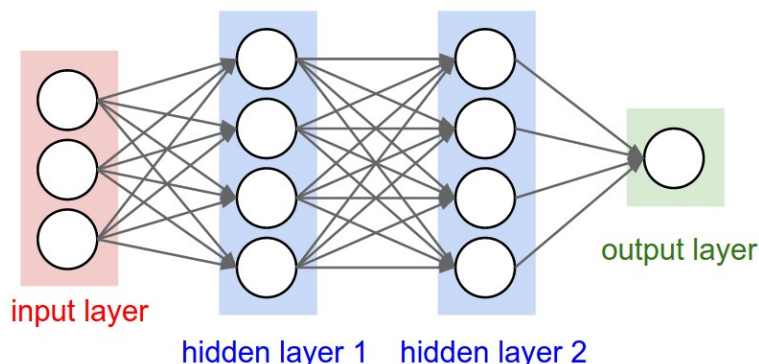


Figure 4.8: Basic structure of a neural network, from [Karpathy \(2017\)](#)

### 4.1.2.1 The Neural Network Basics

The neuron unit is a small processing unit and 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 digit 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 given by the output values of the neurons in the previous layer, illustrated in Figure 4.8. These values are then multiplied by internal weights of the neuron and a final weight, named the bias, is then added. These internal weights are therefore fixed to a specific value, depending on what function that neuron tries to approximate. A bias variable is also added to expand the range of functions a neuron can approximate. The weighted sum of these inputs are then added with the bias variable to give,

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

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. The

summed value then gets normalized using an sigmoid function,

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

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 the 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. This process is repeated until all the layers is calculated. After this step 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 is 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 and  $i$  being an index for each digit. The value  $\sigma(z_i)$  represent 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 neural networks such as Deep Convolutional Neural Network (DCNN) has been introduced. A DCNN uses the basic principles described above, but has extra optimized features that allows a neural network with many layers to be constructed and trained. The exact mathematics behind these Deep Neural Networks

## 4.1 Character recognition using a neural network

---

are beyond the scope of this project report. This neural network is implemented in TensorFlow, [Google \(2017\)](#), and allows a neural network to achieve highly accurate results on classifying digits. For this project a neural network optimized for digit classification, as described in [Google \(2017\)](#). The neural network was only adjusted to increase its accuracy for this specific grading system. These results are described in [5](#).

### 4.1.2.5 Training of the neural network

The 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 training set, there is an accompanied label that specifies the digit value. The neural network 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 as follows:

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. Calculate the value with which each weight should be changed to reduce the error function. One method of doing this is using gradient decent with back propagation.
4. Repeat steps 1 to 3 until a time or accuracy criteria 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 a accurate estimate of the intended student entries. A method to achieve this is discussed in the next section.



## 4.2 Probabilistic Graphical Models

The final problem the system needs to solve, is to probabilistically determining the most likely student entries given the bubble and character evidence. This is achieved by implementing two probabilistic graphical models (PGM).

### 4.2.1 Overview of the system

A probabilistic graphical model 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). The directions of the arrows represent what information causes other information to be created, thus given a parent to offspring interpretation. An example of such a graph is shown in Figure 4.9. 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 system is models are then tasked with inferring what 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 'User intended 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 them. 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 also gets introduced, which then produces the bubble evidence, as seen 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)

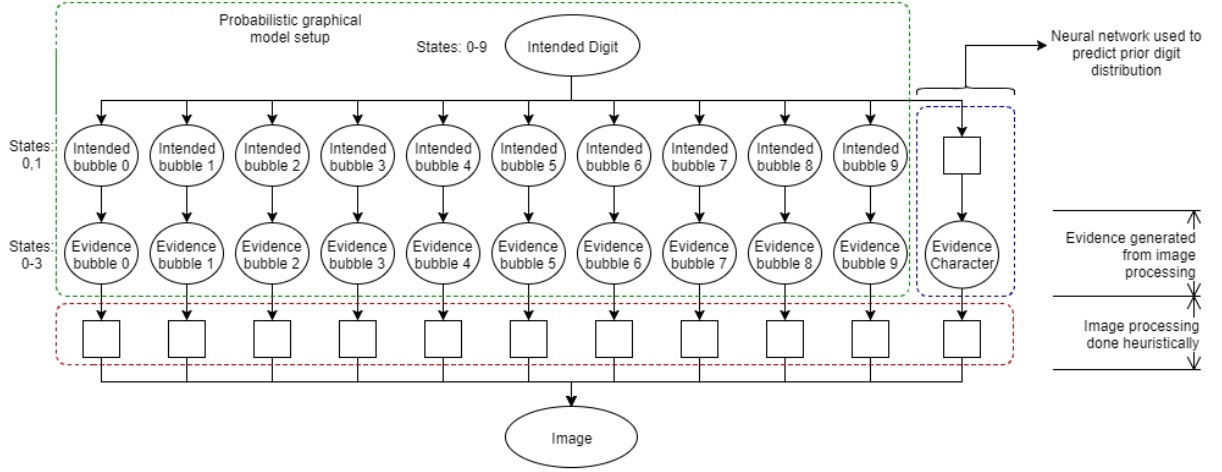


Figure 4.9: Graphical setup for determining the intended digit written by a student.

upwards to the intended digit. The first step is to process the image to produce more tractable evidence. Producing bubble evidence from a 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 neural network 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 Probabilistic Graphical Model. Using this digit PGM, the intended digit can be inferred.

### 4.2.3 Estimating the student answer

An answers 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. Thus there are 10 possible values for each of these 7 columns. This gives a possible number of values that an answer can take to be  $2 * 10^7$  equalling 20 000 000. Calculating each of these states are computationally intractable. Thus a 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. Thus 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  $2 + 10 * 7$  equalling 72, because each column's states can now be calculated

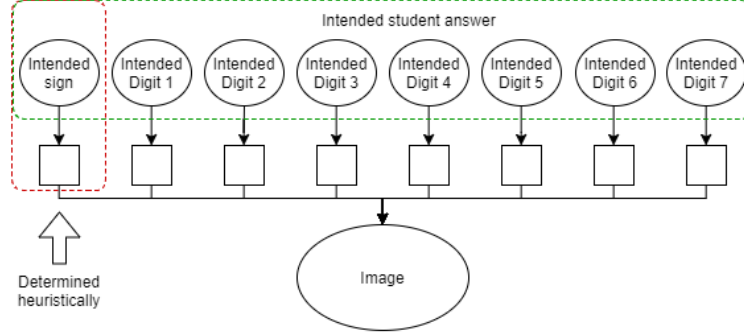


Figure 4.10: Graphical setup of student answer.

independently. Knowing this independent property, the student 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, there are only a limit number of student numbers to consider. Thus 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 C.5. 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 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 which student number is most likely. It is found that if the student provides only character information with no bubbles coloured in, a highly probable 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 an 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 check. Thus each training label now contained bubble evidence and character proirs from the neural network. 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 neural network, 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. This 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

In the following chapter a study is described on 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 systems is assessed using 3 categories. In category 1, marking statistics are given on the grading of these test. The second category describes all the tests the system transferred to the user for manually marking. The reason for this is, because the system had a certainty level below a specific threshold for that 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 is also displayed. The user is then tasked with scanning through each of the clashed 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 guessing the answers correctly. In the last category, all the answers that the system decided to grade automatically, but identified incorrectly, will be 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 evaluates different

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
Random page with no template on them	2
Page with tilted template inside image	2

aspects and limitations of the systems. The categories and the number of test for the particular category, are shown in Table 5.1.

These tests are specifically chosen, because combined they approximates all the types of tests the system has to asses. Most of the tests are extreme cases of what students have filled in on test forms. These test provides a good test-bench to see how well each system performs in challenging situations.

### 5.1.1 Basic system

#### 5.1.1.1 Marking statistics

Using this basic system, an average time for each test is calculated to be 0.305 seconds.

Question 2:

(a)  $\square \square \square \square \square \square \cdot \square \square \square \square$

-  $\square$

1	<input type="checkbox"/>	<input checked="" 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 checked="" 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 checked="" 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 checked="" 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 checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6
7	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	7
8	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" 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 5.1: Image showing answer with crossed out answers that the system misinterpreted.

#### 5.1.1.2 Clash list

A total of 6 out of the 25 test were reported as clashes. The two images without a template were both reported in the clash list. Furthermore the 1 test which only has the student number written in characters with no bubble information is also reported to the list. The other 3 clashes were reported due to the crossed out bubbles interpreted as still filled in, causing the system to think that two answer were filled in. An example of this is shown in Figure 5.1.

#### 5.1.1.3 Incorrect automatic graded results

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

(b)      2 .  7  1  0

-

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

Figure 5.2: Filled in answer with only character information.

## 5.1.2 Complete system

### 5.1.2.1 Marking statistics

Using this complete system, an average time for each test is calculated to be 2.011 seconds.

### 5.1.2.2 Clash list

A total of 6 out of the 25 test were reported as clashes. The two images with no template in were both again reported in the clash list. Two cases was reported to the clash list due to the character recognition determining a crossed out character as the intended character. An example of this is shown in Figure 5.3. One test was reported, because it only has characters in with no bubbles coloured in. Thus, even though the system identified every character correctly, it had a to low of a percentage confidence in its answer and reported it to the clash list.

### 5.1.2.3 Incorrect automatic graded results

There were no automatically graded answers that were done so incorrectly.



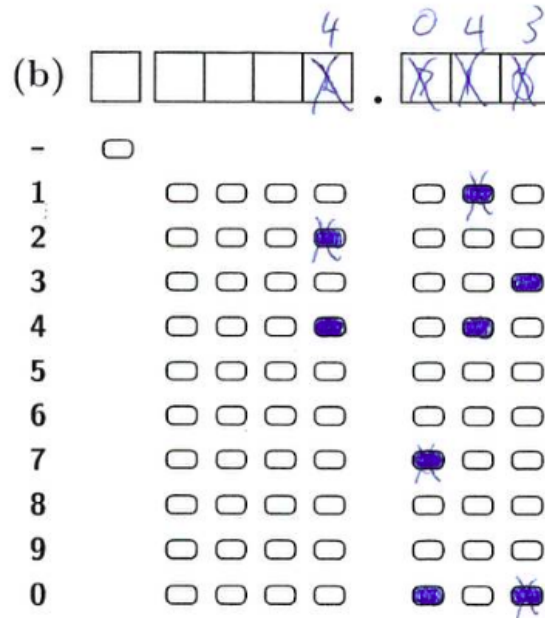


Figure 5.3: Crossed out character that confused the grading system.

### 5.1.3 Analysis or results

The complete system does not reduce the number of clashes significantly for these 25 images. The completed system is that it takes on average 2.011 seconds to grade a test. The student number PGM takes 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 gets 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 written by students.

## 5.2 Grading of tutorial tests

In this section the complete system is tested on the final 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

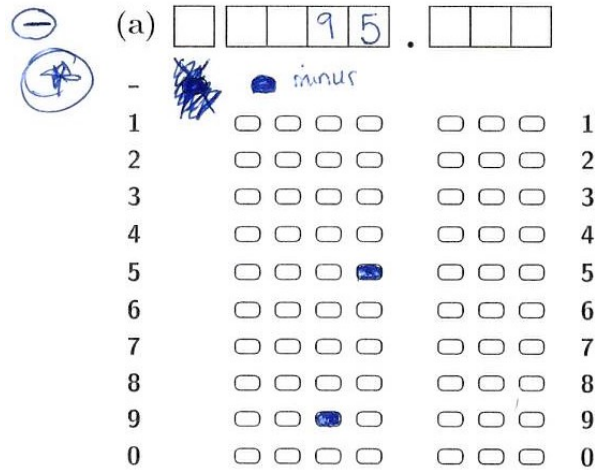


Figure 5.4: Incorrectly identified answer as 95.

and used as the grader's results.

### 5.2.1 Marking statistics

For these tutorial test an average marking time per test of 2.3 seconds was recorded.

### 5.2.2 Clash list

There were in total 67 clashes in the 888 tests. These 67 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 wrong answer to a test. 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 test(s) were not reported. In comparison there were 15 known mistakes made in the first tutorial using the basic system.

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

Number of tests in category	Category description
41	In these tests the system guessed the right values, but was unsure about its answer. Some of the cases was when the student number was only filled in the character box. The software always identified the correct student number, but was still unsure about that 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 tried 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.

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

### 5.2.4 Analysis or results

In conclusion, it is noted that the complete system, with its machine learning capabilities, slightly reduces the number of tests that the user must mark manually from the previous basic system. About 7.5% of the tests in the tutorial had to be marked manually, due to the system being unsure of its answers. The system does, however, have a high probability of grading tests correctly if they are done automatically. A 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 has 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. For the remaining 7.4% of the tests, the system was not certain enough to grade these test automatically. The system did classify most of the clash list test correctly. This means the system could possible have graded 97.1% of the tests correctly. This can be done by changing the threshold value, but allows the other 2.9% of the tests to be classified incorrectly.

# 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 test using a special template. Initially research was done into excising methods of grading these tests automatically. It was found that these software packages normally uses only image processing methods to grade bubbles on a paper. For this project additional machine learning capabilities were also 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 does not have access to quality educational opportunities. The educational systems these individuals do have are normally not up to standard with limited teaching assistance. Educators who are available are not always accessible to learners to provide quality education. Automatic Mark Recognition software like 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 assists 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++W. By continuously updating the estimated orientation of the template as more bubble contours get 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 if solved can significantly increase the software's speed. Further 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 a tutorial setting with around 890 tests the system takes  $\pm 30$  minutes to grade these tests. This grading time is longer than other OCR software, but allows for a broader range answers to be given by a student. An example of these answers are the system's capability to identify a student number by only referring to the characters written in the student number box. The system has an overall accuracy of 97.1%. An additional feature is implemented that transfers tests, which the system is uncertain about, to a user to manually grade. Cabined with the human operator, only 1 test in every tutorial session of an average of 890 tests gets graded incorrectly. In combination with the manually checked tests, the system thus obtains a 99.8% accuracy on grading tests correctly.

# References

- ANKAN, A. & PANDA, A. (2015). pgmpy: Probabilistic graphical models using python. PROC. OF THE 14th PYTHON IN SCIENCE CONF. [9](#), [10](#)
- EDORAS (2017). The inverse radon transform. Image Processing Toolbox. [ix](#), [14](#)
- GOOGLE (2017). Deep mnist for experts. [x](#), [9](#), [26](#), [61](#)
- IVETIC, D. & DRAGAN, D. (2003). Projections based omr algorithm. [7](#)
- 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. [8](#)
- 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](#), [64](#)

# Appendix A

## Project plan

The work layout plan for this project, is seen in Figure [A.1](#). This plan was last updated before the hand in date of 30 October 2017. The idea behind this plan was to allow the student to have a schedule to measure the project's progress against.



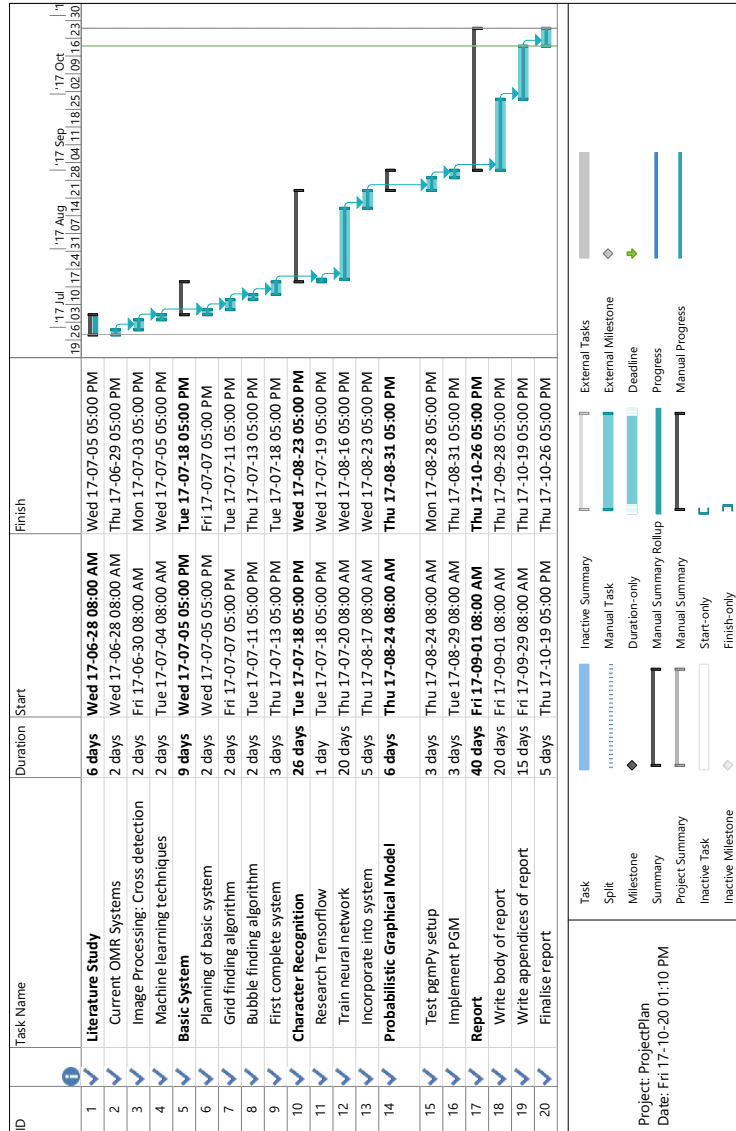


Figure A.1: Project plan for the final year project.

# 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: Discreption 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.	1,2,3,4	A detailed formulation was needed to be made in Chapter 1. Further improvements using two machine learning techniques was also introduced and implemented.
2. Application of scientific and engineering knowledge: Apply knowledge of mathematics, natural sciences, engineering fundamentals and an engineering speciality to solve complex engineering problems.	1, 3 & 4	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 needed to implement and optimize the software for increased speed.
3. Engineering Design: Perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.	1, 3 & 4	In the design process of implementing Image Processing and Neural Networks a procedural process is demonstrated. Implementing a Probabilistic Graphical Model demonstrates a creative approach to solve 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.	1.4, 1.5 & 5	Engineering methods used in this project include agile development, software version control, through Git, and project scheduling to efficiently manage time.

---

Table B.2: Disrcption of Exit level outcomes and how this project adhere to them.

<b>Outcome</b>	<b>Reference</b>	<b>Description</b>
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 &amp; 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.

# Appendix C

## Overview 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 predict the student entries given the distributions stated. It should be noted that the software exploits additional methods in calculating these values in a efficient manner.

### C.1 System as a whole

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 that he/she wants to portray, signifying the first 5 nodes. Those 5 nodes gives rise to the image, representing the last node. At its core the system is tasked with inferring two types of conditional probabilities namely  $P(StudentNumber/Image)$  and  $P(Answer/Image)$ .

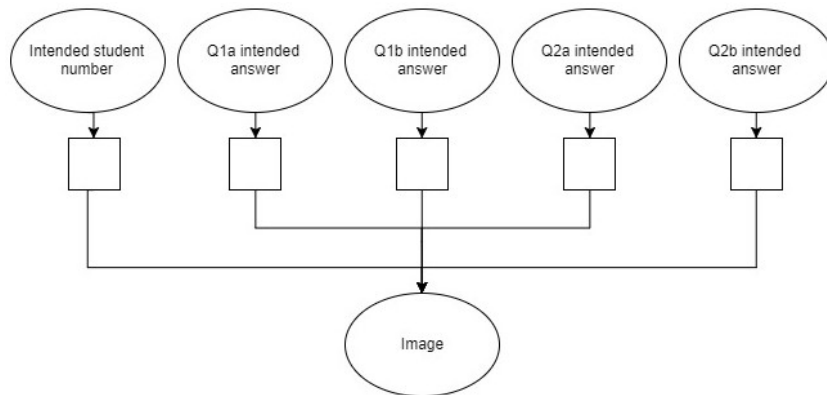


Figure C.1: System overview.

The random variables StudentNumber and Answer(1 to 4) thus represent all the possible values that the student number and answers possibly can have. The image is also a random variable representing the total number of possible states the image can take. Each image has a width and length of 1240 by 1754 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  $1240 \times 1754 \times 256$ . To practically represent this more detailed derivations and assumptions is needed. These derivations are described in the next sections.

## C.2 Student answer

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

$$P(A/I) = P(S_i/I)P(D_1/I)...P(D_7/I). \quad (C.1)$$

Where  $A$  represents an answer for a specific question.  $I$  and  $S_i$  are random variables representing the image and sign of the answer.

To find the most probable answer only  $P(S_n/I)$  and  $P(D_{1-7}/I)$  needs to be calculated. Using image processing techniques described in Section 3,  $P(S_i/I)$  can simply be determined heuristically by determining the probability of the bubble being coloured in,

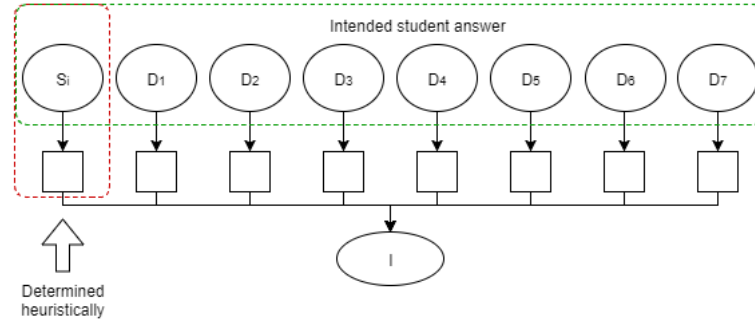


Figure C.2: Graphical setup of determining student answer.

underneath the sign. Thus the only values yet to calculate is  $P(D_{1-7}/I)$ , as derived in the next section.

## C.3 The intended digit

In determining an intended digit there is 11 information nodes to use as evidence. These nodes represents 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 that has to be calculated for each column is describe by  $P(D_i/I)$ . This value represents the probability that each of the 10 digits is written down in column  $i$ . We know that the digit evidence is calculated heuristically using image processing. 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 digit is now given by  $P(D_i/BE_i, CE_i)$ .  $BE_i$  and  $BI_i$  is further described by

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

where  $P(BE_{i:j})$  represents the digit  $i$  at bubble  $j$  and

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

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

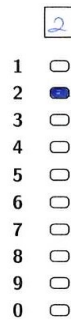


Figure C.3: Column with the evidence that gets considered for the calculation of an intended digit.



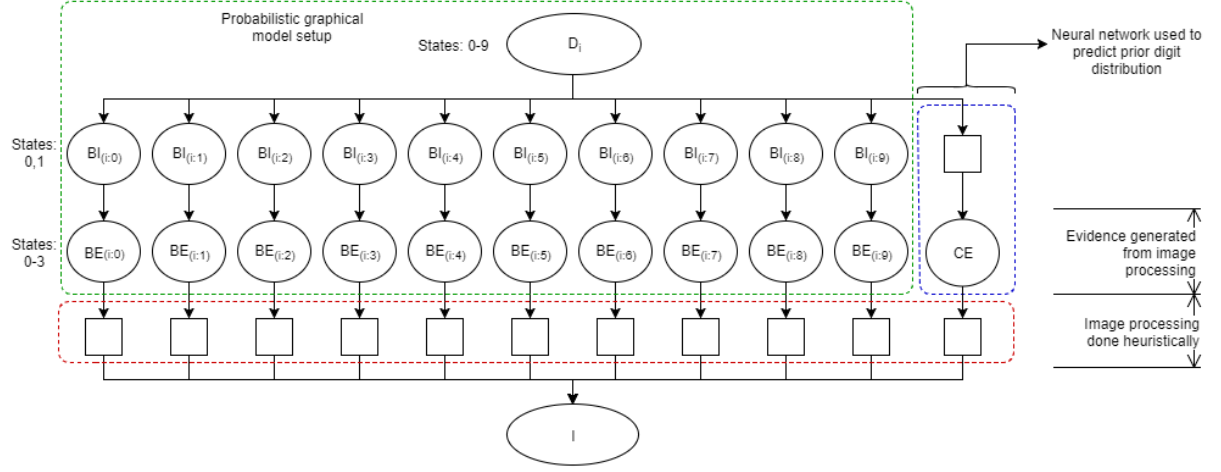


Figure C.4: Graphical setup of determining intended digit.

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

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

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

In Equation C.5, the indented 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  $D_i$  and  $BI_i$  is known. Further it is observed that  $BE_i$  is only reliant on  $BI_i$  and  $CE_i$  on  $D_i$ . Thus by applying a additional product rule results in,

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

Bayes' rule also provides us with,

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

Now by factoring out all the constant terms out of the summation one is left with,

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

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 form training. The final term that is needed is  $P(D_i/CE_i)$ . This term is efficiently represented through the use a the neural network. Thus the pgm system can successfully infer the digit and thus answer probabilities with these 3 distributions specified. Next a derivation on the student number probability is discussed.

## C.4 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 will have  $\pm 900$  states, depending on the number of student numbers. The student number graph can be seen in Figure C.5. Next a derivation for  $P(S/I)$  is needed.  $S$  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  $P(S/DE)$ .

$DE$  and  $DI$  is further described by

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

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

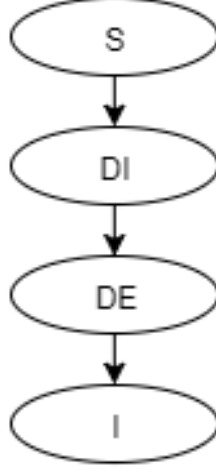


Figure C.5: Graphical setup of determining student number.

$P(S/DE)$  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 seen to be independent from  $S$  if  $DI$  is known 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 is also known,

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

$$(C.15)$$

$P(S)$  can be initialized as a equal distribution, because every student number is equal as likely to be in a given test.  $P(DI/S)$  are values that are trained from data using the

independence property,

$$P(DI/S) = P(D_1/S)...P(D_8/S). \quad (C.16)$$

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, digit 1 will have a high probability of being 1. Thus these two random variables are thus strongly correlated. The only values that still needs to be calculated are thus  $P(DE/ID)$ . Using the indepenency property of

$$P(DE/ID) = P(BE_1/BI_1)P(CE/D_1)...P(BE_8/BI_8)P(CE/D_8), \quad (C.17)$$

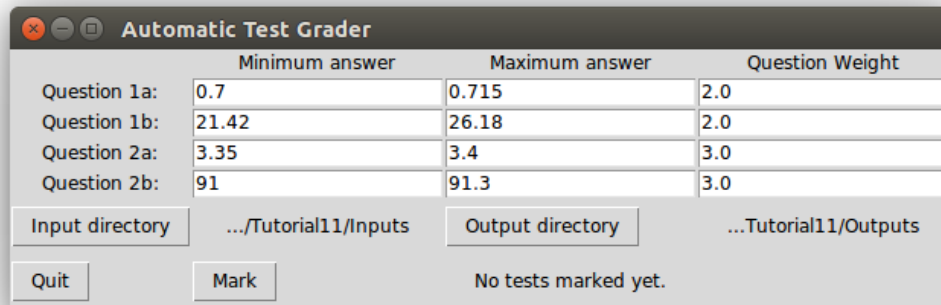
the intededded digit model's conditional distributions can be used. The two PGM models can now be fully defined and used to infer the intended student entries from an image.

# Appendix D

## Systems diagrams

### D.1 Interface

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

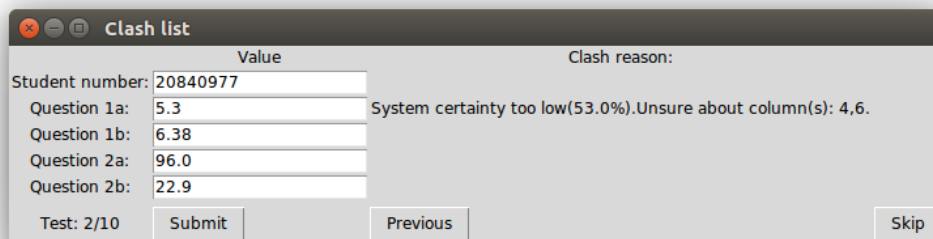


	Minimum answer	Maximum answer	Question Weight
Question 1a:	0.7	0.715	2.0
Question 1b:	21.42	26.18	2.0
Question 2a:	3.35	3.4	3.0
Question 2b:	91	91.3	3.0

Input directory: .../Tutorial11/Inputs      Output directory: ...Tutorial11/Outputs

Quit      Mark      No tests marked yet.

Figure D.1: Main interface of test grader.



	Value	Clash reason:
Student number:	20840977	
Question 1a:	5.3	System certainty too low(53.0%).Unsure about column(s): 4,6.
Question 1b:	6.38	
Question 2a:	96.0	
Question 2b:	22.9	

Test: 2/10      Submit      Previous      Skip

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

## **D.2    Templates**

The original template can be seen in the Figures [D.3](#).

Two additional templates has also been developed and implemented for the department. These templates gives the department the capabilities 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      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="checkbox"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<input type="checkbox"/>	0

### Section A:

	A	B	C	D	E	
(1)	<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"/>	(2)
(3)	<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"/>	(4)
(5)	<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"/>	(6)
(7)	<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"/>	(8)
(9)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(9)
(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(10)
(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(11)
(12)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(12)
(13)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(13)
(14)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(14)
(15)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(15)
(16)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(16)
(17)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(17)
(18)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(18)
(19)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(19)
(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(20)
(21)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(21)
(22)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(22)
(23)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(23)
(24)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(24)
(25)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(25)

	A	B	C	D	E	
(26)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(26)
(27)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(27)
(28)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(28)
(29)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(29)
(30)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(30)
(31)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(31)
(32)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(32)
(33)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(33)
(34)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(34)
(35)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(35)
(36)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(36)
(37)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(37)
(38)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(38)
(39)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(39)
(40)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(40)
(41)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(41)
(42)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(42)
(43)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(43)
(44)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(44)
(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(45)
(46)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(46)
(47)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(47)
(48)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(48)
(49)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(49)
(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(50)

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

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

---

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="checkbox"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<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"/>	<input type="checkbox"/>	0

### Section A:

	A	B	C	D	E	
(1)	<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"/>	(2)
(3)	<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"/>	(4)
(5)	<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"/>	(6)
(7)	<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"/>	(8)
(9)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(9)
(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(10)
(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(11)
(12)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(12)
(13)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(13)
(14)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(14)
(15)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(15)
(16)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(16)
(17)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(17)
(18)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(18)
(19)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(19)
(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(20)
(21)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(21)
(22)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(22)
(23)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(23)
(24)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(24)
(25)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(25)

	A	B	C	D	E	
(26)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(26)
(27)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(27)
(28)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(28)
(29)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(29)
(30)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(30)
(31)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(31)
(32)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(32)
(33)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(33)
(34)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(34)
(35)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(35)
(36)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(36)
(37)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(37)
(38)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(38)
(39)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(39)
(40)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(40)
(41)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(41)
(42)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(42)
(43)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(43)
(44)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(44)
(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(45)
(46)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(46)
(47)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(47)
(48)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(48)
(49)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(49)
(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(50)

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

### D.3 DCNN TensorFlow setup

The DCNN structural setup diagram can be seen in Figure [D.6](#). This structure was directly used as constructed was constructed by the TensorFlow team.

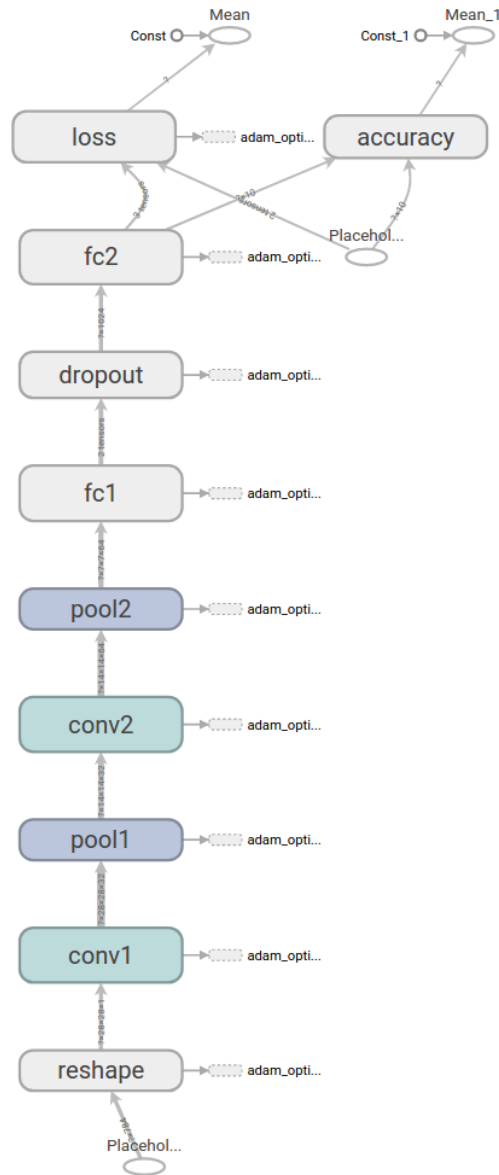


Figure D.6: DCNN structural setup diagram, from [Google \(2017\)](#)

# Appendix E

## Validation and results

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

### E.1 All tutorial results

#### E.1.1 Overview

This automatic test grader was successfully used to grade 11 tutorial test in 2017. The amount of tests 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. Non were found. Thus it is assumed that tests that have mistakes in, but is not reported are unlikely and is not interoperated into the calculations.

In the 11 tutorials an average of 99.3% of test is estimated to be graded correctly, as no corrections were made by students. This result is seen to be brought down by the basic system having a lower accuracy rate. When only taking the complete grading system's result and average correctly grading tests is calculated to be 99.9%.

Table E.1: Description of tutorial results.

<b>Tutorial number</b>	<b>Percentage tests graded correctly</b>	<b>Reason for results</b>
Basic system is now implemented.		
Tutorial 1	98.4% (14 mistakes)	The system had problems with identifying crossed out answers.
Tutorial 2	98.8% (11 mistakes)	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 mistake 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)	The system made a few mistakes with answers with only character information.

## E.2 Deep Convolutional Neural Network results

---

Table E.2: Description of tutorial results.

Tutorial number	Number of tests graded incorrectly	Reason for results
Compete system 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 a crossed out answer 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 the results chapter. 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 is conducted on 3 neural networks trained on different datasets and compared with each other. This testing proses is conducted to find the neural network weights that will classified had written digits the most accurately. The neural networks is tested on a test dataset generated by grading 900 student tests, while extracting the character images. The answers from these tests is used to create labels for each digit image. Thus each 28 by 28 pixel digit image has a accompanied label specifies what that digit is. The database contains 16 000 labelled images and was thus 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), was also used in this proses. This dataset contains 60 000 training set digits and a test set of 10 000 digits. Each neural network was tested on both datasets. The results of these test is shown in the next section. Every network was trained for 8 hours on the same processor, before being tested.

---

## E.2 Deep Convolutional Neural Network results

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

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

### E.2.1 Trained on generated database

In a first attempt at training the 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 in on each dataset's test set is shown in Table [E.3](#).

#### E.2.1.2 Conclusion on accuracy

The results are promising, but an the average on the MNIST dataset is still to low. A standard digit classifier has an 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 neural network thus does not have enough data to accurately model each digit.

### E.2.2 Trained on MNIST database

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

#### E.2.2.1 Accuracy of network

The test accuracy of the neural network in on each dataset's test set is shown in Table [E.4](#).

#### E.2.2.2 Conclusion on accuracy

The results is very poor in classifying the generated data. A main reason was found to be that the MNIST dataset having very preprocessed digits. In the test grader example



---

## E.2 Deep Convolutional Neural Network results

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

Test dataset	Percentage accuracy
MNIST dataset	99.35%
Test generated dataset	82.23%

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

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

digits are sometimes written to the side and the system cannot find a segment to recentre. This causes a few of centred digits that is not present in the MNIST dataset.

### E.2.3 Trained on mixed database

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

#### E.2.3.1 Accuracy of network

The test accuracy of the neural network in on each dataset's test set is shown in Table [E.5](#).

#### 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 usefull information to the test grading system.