Automatic Grading of CITO Mathematics Tests: Multiple Choice Classification

Arend-Jan Tissing

Student number: 7642245

Date: July 1, 2022

UU supervisors:
First supervisor:
Prof. dr. Arno Siebes
Second examiner:
Dr. Thijs van Ommen

CITO supervisors:
First supervisor:
Tjeerd Hans Terpstra
Second supervisor:
Laura Kolbe

MSc Applied Data Science
Utrecht University

Abstract

This thesis attempts to discover whether it is possible to do automatic grading of CITO mathematics tests using Optical Character Recognition (OCR) methods, among others. It is part of a cooperation between three students, where this thesis focuses on the extraction and labeling of questions, as well as classifying the multiple choice questions. It turns out that with the methods discussed in this thesis, it is not possible to extract questions and classify multiple choice questions with high accuracy. Also, there are some robustness concerns discussed in this thesis. In this research, the classification of multiple choice questions turns out to be most successful with a convolving measure of dissimilarity as defined in this paper, in combination with a decision tree classifier, to an accuracy of 99.49%. However, due to an accuracy of the preprocessing of 91.14%, the accuracy of the total process is insufficient to be applicable, since it is 90.68%. In conclusion, it is not advised for CITO to implement the methods discussed in this research for the automatic grading of multiple choice questions.

Contents

1 Introduction

1.1 Motivation and context

In the field of data science, automatically scanning documents is one of the techniques that can automate manual inspection processes. One of these processes that could be improved in such a way, is the scanning of tests. CITO, a Dutch organisation that develops and distributes tests and exams, has requested to see whether it is possible to automate the process of grading mathematics tests. These tests are made for primary school students and are used as practice tests, to monitor the education level of students in general. However, if automating such tests turns out to be succesfull, the decision could be made to extend this to other mathematics tests for students.

These tests consist of two types of questions: open and multiple choice. Open questions consist of one answer line, on which a number should be written by the student. Multiple choice questions consist of answers A to D, of which the chosen letter should be encircled. Lastly, there is an area for drafts on the right of the page, which should be disregarded.

The problem at hand has a few obvious challenge. These challenges are recognising the location of the answer on the scanned page, finding which question numbers the answers belong to, where it is possible that multiple questions are on one page, and lastly recognising the handwriting or encircled letter to see what answer is given. With this last point, a challenge may be that handwritings are not always so clear, especially when belonging to children.

1.2 Literature overview

On the topic of scanning documents, a lot of literature has been written, even though it remains to be a challenging subject. This literature can of course also be applied to mathematics tests.

For extracting separate questions, the computer vision library called OpenCV Itseez [2022] can be used, together with Tesseract Hegghammer [2021]. Then, YOLOv5 can be useful to extract the answer line, as it is known to be used for extracting regions of text from images, as described by Prajwal et al. [2019]. Lastly, there are the challenges of recognising handwritten numbers for the open questions and recognising the encircled letters for the multiple choice questions.

For the open questions, a well-known solution that is known to work relatively well for these types of problems, is the use of Convolutional Neural Networks (CNN). This is described by Prajwal et al. [2019], among others, to achieve an accuracy of over 99% on testing data. However, there are more approaches that are interesting to consider and to compare. For example, much simpler algorithms such as K-nearest neighbours [Babu et al., 2014], Random

Forests [Bernard et al., 2007] and Support Vector Machines [Tuba et al., 2016] are other valid approaches that could be considered. When looking further in these handwritten digit recognition algorithms, it is to be expected that the CNN is superior to other methods, as it has a much lower error rate in literature. All other algorithms seem to have troubles working in more than 95% of the cases, while the CNN has the potential to reach 99.87% accuracy on clean data, as described by Prajwal et al. [2019]. However, it interesting to investigate this further in the context of this research, where the digits are produced by children.

For recognising multiple choice questions, there are usually simpler methods required, as the letter in the multiple choice box should be recognised. One well-known method that is able to recognise such letters is the existing OCR model Tesseract [Hegghammer, 2021]. However, there of course many more options, which is why this research will also explore other possibilities to recognise multiple choice answers.

1.3 Research question and task division

Based on the question presented by CITO, the research question that should be answered is stated as follows.

How should CITO approach automatically scanning mathematics tests and is this approach reliable enough to implement in practice?

This question is researched in a team of three students, each with their own focus. However, there is overlap between the topics and the preprocessing is done only once, to generate one method of extracting the answers. Once this is done, the team members individually tackle the problem of either recognising the numbers or classifying the multiple choice questions. For this, the methods are divided as follows, to ensure that multiple approaches are explored. This should generate at least one method that is a good benchmark for CITO to see whether this product is viable.

This thesis covers the following parts.

- Extracting the student information, separate questions and question numbers.
- Classifying multiple choice questions.

When extracting the questions, of course it is still necessary to recognise where on the page the actual answer is written. Recognising these locations of the question lines and encircled letters from the individual questions can be found in Schreurs [2022], a thesis written in the same research team. Classification of the handwritten open question answers are also not part of this thesis. For the methods and corresponding results for these questions, one would have to consult the theses Klopper [2022] and Schreurs [2022]. For this reason, this

part of the research question is not answered in this thesis and therefore the only focuses of this research are on the initial extraction of questions and the classification of multiple choice questions. Any steps necessary for the processes in this research that are performed by the other team members are briefly explained in this research as well. However, for a thorough explanation, one would have to consult the other theses [Schreurs, 2022, Klopper, 2022].

2 Data

2.1 Obtaining the data

The data consists of 73 scans of tests, where each test contains 15 pages and a total of 30 questions. 10 of these questions are multiple choice, the remaining 20 are open questions. This data was scanned by CITO and delivered into pdf format. The scans are in black and white. During the process of obtaining this data, it turned out that the recognising of numbers required a higher resolutions, since the pictures of numbers were rather noisy.

After obtaining the files in higher resolution, these are converted to jpg image files and structured into separate directories to access them one by one. The quality of these images is 600 dots per inch (dpi), which is high enough for the purpose of this research, and the scanning quality is consistent.

2.2 Data preparation

2.2.1 Student details

On the first page of all tests, the student information is given. To be able to automatically link a test to a student, a unique identifier should be extracted from the page. For this, the student number and school number are chosen, since there are no two students in the same school having the same student number. The process of extracting these numbers is done using Tesseract. This method works in most cases where the student information is available, but in case the method is unable to recognise these numbers or in case this information is not on the page, then the method will output 'unknown', after which a manual inspection can be done by the teacher. It would also be possible to link the observed student numbers to an existing list of student numbers, to see whether every student number actually represents a student or whether an error was made.

2.2.2 Selecting the separate questions

In handling the questions, the first step that is taken, is to separate the questions from each other and place them on a separate page. This is convenient, because this means that there is only one answer that should be found per page and that it is immediately clear to which question the answer belongs.

To split the questions, the following approach was taken. Since any question has a question number on the left, this allows for easy recognition of the starting locations of any new questions. The approach described below relies heavily on OpenCV Itseez [2022], an open source computer vision library that allows for extracting information about the shapes and structures present in the pixels, which can be useful for tackling this problem. OpenCV images are stored as NumPy arrays, which allows for fast computations. For these reasons, this

method is used for the extraction of questions. The following steps are taken to split using the question numbers.

1. Extract the left bar. Careful inspection showed that taking the left 17% of the page contains all question numbers and no extra noise. This part is extracted, as only the vertical coordinates are required to split the questions, together with the question numbers themselves.

2. Set a binary threshold of 100 out of 255 to create an image of only black and white, where all gray is, depending on its intensity, is converted to either black or white. This means that all values below 100 are set to 0 and all values above 100 are set to 255. Careful inspection has shown this to be an appropriate threshold that does not make the numbers disappear, while removing any noise.

3. Erode the image, to obtain blocks for every individual question. This is shown in for example Figure 1a. As explained by the official OpenCV documentation [Itseez, 2022], eroding an image means taking the minimum value across an area that is specified using a shifting kernel matrix, which means that when eroding the image, black areas are expanded. For this case, a 9x9 square kernel consisting of ones is used. This process is repeated for a total of 8 iterations. This is useful to create one larger box around the number. The 9x9 size of the kernel matrix and the 8 iterations are not exactly required, as the goal is simply to create a box around the number, but careful inspection has shown that numbers much smaller or larger lead to either images where the box does not cover the entire image, or to a box that far exceeds the limits of the individual question number.

4. Determine the coordinates around the eroded block to be able to extract a cropped image, as shown in for example Figure 1b. This is done with OpenCV's `findContours` function, which is performed on the inverse of the eroded image, as the function detects white blocks. There are two settings selected in this function. First of all the mode is set to `RETR_EXTERNAL`. This is an indication that no smaller contours within contours should be returned. Although this will not have much of an influence, due to the nature of question numbers, it is clear that any subcontours would be noise. Second, for the method, `CHAIN_APPROX_NONE` is selected. This simply means that the entire contour is stored, instead of only the boundary points. However, choosing a different setting will not have a large influence on the result, as the only thing of interest is the bounding rectangle, as seen in the following step. Next to hierarchy information, which will not be used here, the result of this function is a list containing all contours around the blocks obtained from the erosion. The following steps are performed for each contour on a page, by iteration using a for loop. It should be noted that all information of this function and its settings are taken from Itseez [2022].

5. Using each contour, the `boundingRect` [Itseez, 2022] function is used, to find the smallest horizontal rectangle around the contour, which indicates the coordinates of a rectangle around the question number and is used for the extraction of this number.

6. There are some issues with the numbers, mostly visible in Figure 1h. To solve this, small horizontal white stripes through the number should be fixed. This is done using OpenCV's `morphologyEx` function. According to the OpenCV documentation Itseez [2022], this can be interpreted as doing an erosion, followed by a dilution (which is the opposite of the erosion). In other words, first black areas are expanded, after which they are shrunk. When doing this expansion and shrinking vertically, this means that the horizontal lines are filled up, after which everything is decreased to its normal size. Since the horizontal gaps have closed in the expansion, these horizontal white lines cease to exist. They will then not reappear in the dilution phase, as they are no longer on the outsides of the shapes. The result is that small horizontal gaps are closed. To achieve this vertical expansion and shrinking, the kernel matrix consists of only a vertical line, as the objective is to expand and shrink vertically, to only solve these horizontal lines. This is done to avoid closing vertical white lines that should not be closed. Also, the kernel is only of size 3x3, as larger sizes could cause closing larger vertical gaps that are actually meant to be in the number, while this size turns out to be large enough to fix the errors in the scans. The order of performing the erosion and the dilution, is done by setting the operation to be `MORPH_OPEN`. Using different settings would only be used when closing for example black stripes in the numbers. The result of this correction of the image can be seen in for example Figure 1i.

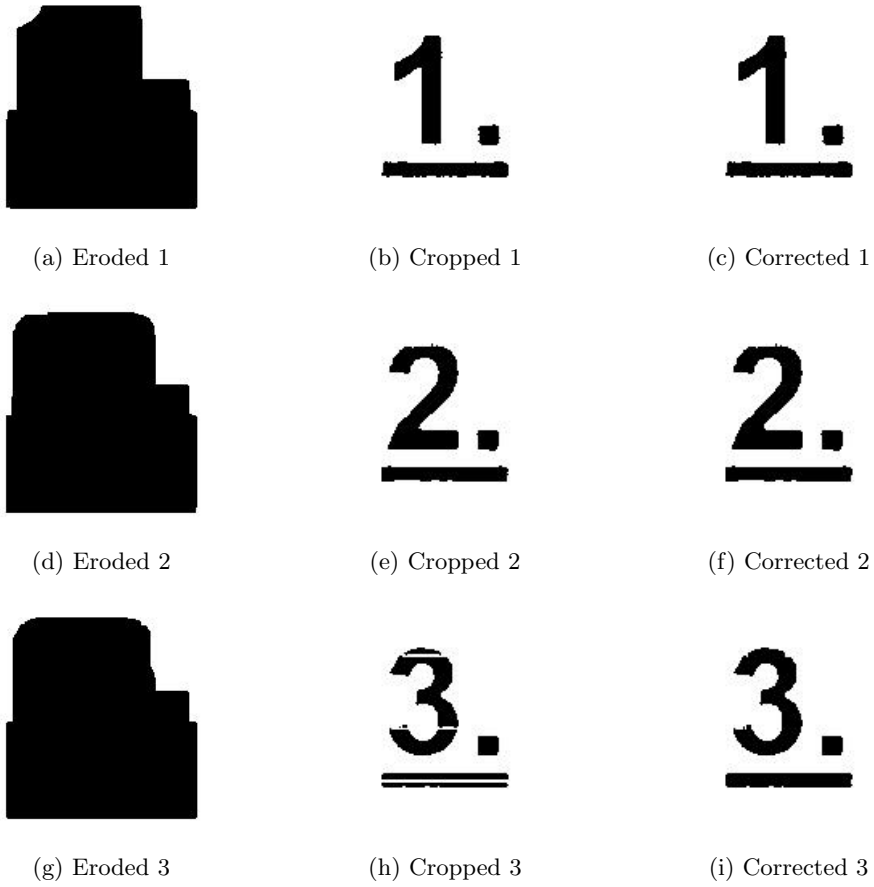
7. The image of the question number should be clear at this point, therefore the number is recognised using Tesseract, an program for Optical Character Recognition (OCR). The following configuration is used to recognise single line answers only containing 0 to 9:

   ```
   config = '-c tessedit_char_whitelist=0123456789 --psm 7'.
   ```

   Then, the confidence level is set to 20, where lower means a higher certainty and the confidence is on a scale from 0 to 100. In the Python script, as can be seen in Appendix C, this is indicated as a threshold of 0.8. This is done by testing out multiple thresholds to minimise the number of misclassifications. The result is that less question numbers are recognised, but also that less incorrect classifications are done. This obviously still causes issues, but this is solved in Section 2.2.3.

8. Lastly, using the coordinates of the observed question numbers, the images are split up 130 pixels above the question numbers, where this specific number was chosen by carefully inspecting the cutoffs at multiple tests.

This ensures that no answer lines are cut off if a student has a large handwriting. In case no number is found, no cutoff is done, since the area is then considered to be noise. These separate questions are all saved in a separate image, so that they can be treated separately and the found question number is stored.



(a) Eroded 1 (b) Cropped 1 (c) Corrected 1

(d) Eroded 2 (e) Cropped 2 (f) Corrected 2

(g) Eroded 3 (h) Cropped 3 (i) Corrected 3

Figure 1: Question numbers from left to right: eroded image, cropped image, corrected image

2.2.3 Improving the question selection

Even though the questions are mostly recognised correctly, there are some issues with this procedure. Most tests contain at least one number where a question is recognised incorrectly. This is an issue that also impacts the other questions. Consider the case where two questions are on an image. If the second question

is not recognised to be a question for example, then the image is not cut off. The result of this is that question 2 is not in the list of questions, but also that question 1 suddenly has two questions on the page, as no cut-off happened. This gives difficulties further along the processing stream.

Since this problem occurs too frequent to be acceptable as an error margin, the following solution is used. A condition for this solution is that all tests in one batch of tests have the same layout, so that no two different types of tests are processed at once. This is not ideal, since it makes the solution less flexible. However, it is not much of an issue, as CITO can choose to do so when processing tests. Also, multiple tests are required to make the process work, so scanning an individual test would not work with this approach.

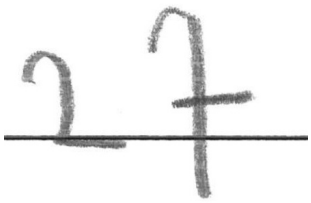
Then, the following process is performed. All tests are processed in one batch to determine the coordinates of the cutoffs between questions, before actually cutting off the questions on an individual basis. All these coordinates are attached to their respective question number. Then, iterate over all tests to see whether the question number is a valid set. This is defined as follows: if there are 30 question numbers found, then all numbers from 1 to 30 should be present. After removing the sets that are not valid, a set of tests is remaining, all containing a valid set of question numbers. To avoid a scenario where the last number is missed (in the case of 30 questions, if question 30 is missed, then the remaining questions 1 to 29 still form a valid question set), a majority vote is taken on which valid question set is most present. In this program, when iterating over all tests, if the program finds at least 5 tests that have the same valid question list and if this is indeed the majority question list, then this is chosen to be the correct one. An issue that one could think of is the theoretical scenario where all last numbers are difficult to read, but this is assumed to be rather unlikely.

In the last step, using these chosen question lists with their attached vertical separation coordinates, the coordinates are averaged per question. In this way, a good separation should be found that should work in almost all cases, especially given the consistent quality of the scans. To give some indication of how well this separation performs, a manual inspection of the first ten tests was done, giving 300 questions, of which 100 are multiple choice. In all cases, the answer is on the page and can be easily identified when classifying by hand, so all the necessary information should be on there. In two of the ten tests, the cutoff sometimes goes slightly through the answer in one or a few open questions, as they have a rather large handwriting. However, when taking another cutoff location, there are larger issues such as having the multiple choice answer completely at the wrong page, so it does seem that this cutoff is the ideal balance.

2.2.4 Locating the answer line

Using YOLOv5, the answer is located. This model was trained as described by Schreurs [2022], which is why it is further elaborated in that thesis. The model extracts the images of the handwritten answers, which can be uses for further processing. The resulting images are shown for a multiple choice question and an open question in Figure 2, of which the multiple choice question is most relevant for the purpose of this research.

(a) Multiple choice question



(b) Open question

Figure 2: Example of an extracted multiple choice question and open question answer.

It is important to discuss how accurate these results are. In Schreurs [2022], a different metric of accuracy is used, as that is specifically designed to look at the quality of the cutouts. This metric is not useful for the purpose of this research. For this thesis, as it focuses on multiple choice questions, it is important to know how many of the cutouts actually contain the letter that the student meant to give as an answer, from A to D. It should be noted that when determining this, in the settings of YOLOv5, the option is selected to only extract the output with the highest assigned probability. Further notes on that choice can be found in Section 5.3.

Most errors being made are due to mistakes of the student, for example when they circle the number after the letter, instead of the letter from A to D themselves. Other cases are when a student crossed out an answer, as it is then harder to recognise for the model. Also, there are cases when there are multiple letters from A to D on the cropped image, which is also considered to be an incorrect classification. In classifying mistakes, four categories are used. These are found in Table 1, together with their counts. It should be noted that this check for mistakes was only done on all the multiple choice questions that the model was not trained on, which is the testing data.

Note here that the categories 'Correctly identified' and 'Missing answer correctly identified' are the two categories that indicate the model worked as ex-

Category	Count
Correctly identified	596
Incorrectly identified	36
Missing answer correctly identified	42
Incorrectly no answer identified	26

Table 1: Counts indicating whether answers were correctly found on the question images.

pected, the other categories indicate that the model has made a mistake. For that reason, the accuracy can be calculated as follows.

$$accuracy = \frac{596 + 42}{596 + 42 + 36 + 26} = 0.9114 \tag{1}$$

In Equation 1, it can be seen that the accuracy of YOLOv5 for the multiple choice questions is 91.14%. In a later stage, this accuracy can be multiplied with the accuracy of multiple choice classification method, to obtain an accuracy for the full process.

2.3 Ethical and privacy considerations for the data

Specifically looking at ethical and privacy considerations for the data, it is important to realise the consequences of faulty data. If for example by a machine error the scans are useless, the methods applied should ideally be able to realise this. The outcome of a test, and specifically final tests such as the ones that CITO takes from students, can be very influential for a student's future. For that reason, CITO should be very hesitant to implement a system that comes without safeguards for faulty data. It should be noted that this specific test is used to determine the overall level of students in the Netherlands, which means that the results of this test would not influence a student's future. However, this thesis does attempt to show a proof of concept that could be applied on similar mathematics tests in general, so any ethical issues should still be kept into consideration.

In this research, an optimal scanning quality is assumed, as provided by CITO in the current state. For that reason, these safeguards as mentioned before are not implemented and therefore caution is needed when considering to implement the methods described in this research.

Lastly, on the contents of this report, no full example tests are shown. This is both for privacy issues, as the tests were made by real students, as well as to prevent the tests from spreading, since these are the property of CITO. These tests include the first names of the students, as well as a school number and a student number. This information is useful for determining which student a

test belongs to, but this also means that caution is required when working with these tests, as they could potentially be traced back to an individual student.

3 Methods

3.1 Translation of the research question to a data science problem

After the data preparation, the resulting dataset consists of the answers in a format that is displayed in Figure 2. The data science problem at hand is to classify the handwritten numbers and multiple choice answers. In this research specifically, the classification of multiple choice questions is the problem of interest.

It is crucial that not too many mistakes are made. There is no specified accuracy or other metric of success on beforehand that should be reached. However, in case of 30 questions, an accuracy of 99% would mean that still almost one in every three students receive an incorrect grade. For that reason, to make a viable product, it is evident that very high performance is essential.

As previously described when describing the research question, within the team of students, a split was made to explore different parts of the research question, which can also be linked to the data science problem. After preprocessing, which is described in Section 2, Klopper [2022] and Schreurs [2022] explore methods of classifying handwritten numbers. This paper focuses on the other side of the problem, which is the classification of multiple choice questions.

3.2 Selection of methods

There are many different approaches possible for this problem. During the exploration to find the best solution for this problem, multiple of them were attempted. These are discussed in this subsection.

Before solving this issue, let's once again see what the multiple choice answers look like. Having an intuition of the pictures to classify can help with finding a good method. The examples can be found in Figure 3. This selection was chosen to show the variety in how answers are given. Note that, as described in Section 2.2.4, there are also incorrect cutouts to begin with, but that this is due to the preprocessing steps. For that reason, when classifying, only the cutouts that actually display a single letter from A to D are taken into consideration for both training and when deciding how well a method performs. This means that there should be a single letter from A to D on the cutout and that the presence of any numbers is allowed.

3.2.1 Labeling

Before describing methods to classify multiple choice answers, the dataset needs to be labeled. In order to do this, a script was written that displays the images one by one, where one of the four options A to D can be selected, after which

(a) Example A

(b) Example B

(c) Example C

(d) Example D

Figure 3: Examples of correctly cropped multiple choice questions, showing the variety in tidiness.

the images are placed in an assigned directory. A fifth option was also available, which was used when no clear letters from A to D are present or when there are multiple on one image. This means that the output from the YOLOv5 preprocessing step was incorrect. Almost all instances that were put in the fifth category were cases where students misunderstood the assignment. They for example circle the number instead of the letter or they would circle both. The amount of instances per category is given in Table 2. Here it becomes clear that there are no huge imbalances, even though there are some differences between the categories. The only category that has a very different count is the category with unknown images. However, this is not an issue, as these are not used during training, as they do not contain the necessary information and any errors coming from faulty crops should be considered in the discussion of the YOLOv5 model, which is described by Schreurs [2022].

Label	A	B	C	D	unknown
Count	125	187	195	118	29

Table 2: Number of instances per label in labeled dataset.

3.2.2 Tesseract method

Since this problem boils down to classifying a printed letter, a good solution could be to use existing models and existing software in the attempt to classify the letters. One example of such software is Tesseract. Smith [2007] mentions that Tesseract is a useful open source OCR tool that can be used to recognise text in pictures, specifically non-handwritten text. This tool has also been used in the preprocessing.

The most important challenge here is that there is in most cases a circle around the letter, or sometimes there are stripes through the letter. For that reason, the crops are not instantly classified, but some preprocessing is performed. The objective of this preprocessing is to filter any pencil writing, in order to only keep any printed text (preferably only letters from A to D). The steps to do this are described below.

1. Open the image as a NumPy array using OpenCV and convert it to grayscale.
2. Set a threshold of 60 out of 255 for the image. Since a simple binary threshold is used, lower numbers are set to 0, higher numbers are set to 255. Since the circles around the letters are written using a pencil while the letters are not, this can remove much of the pencil already. Some examples of this result are shown in Figure 4.
3. Use the `morphologyEx` function to remove further noise. This function was previously explained in Section 2.2.2. This is again done to remove as much of the pencil writing as possible, but also to improve the quality of the letters themselves as well. A square 5x5 kernel is used this time, as there is more noise compared to the situation where this function was previously used. The function is used twice, first with the `MORPH_OPEN` operator, second with the `MORPH_CLOSE` operator, to remove both black areas that should not be present, as well as white gaps in the numbers that should not be in the images, respectively. Examples of this are found in Figure 5.

After the preprocessing is done, it becomes clear that not all letters were extracted perfectly, but that the majority is. An imperfection can be seen from the gap that is present in the D of Figure 5. This are caused by removing too much of the black areas, but this is necessary to prevent a scenario where too much of the pencil writing left in the images after thresholding. However, in the majority of cases, clear letters are extracted. Using these processed images, Tesseract can be used to do predictions on the labels. This is done by setting the configuration in such a way that only the characters A, B, C and D are permitted in Tesseract. After predictions are done, if a multi-letter answer is given, the first letter is extracted, as the answer is always a single letter. Also,

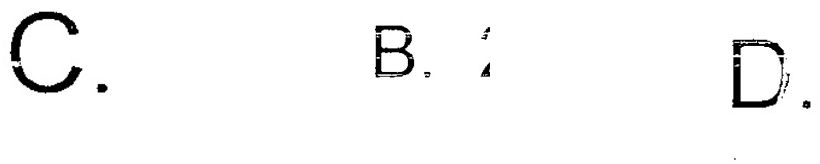


Figure 4: Examples of multiple choice questions after thresholding.

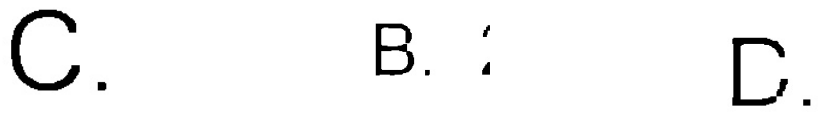


Figure 5: Examples of multiple choice questions after the morphology operator, the last example shows a case where the extraction has a clear imperfection.

if no answer is given, it is classified as 'other'. This is useful to know for the teacher, since they will know that a manual check is required.

One property of this method that is generally an advantage, is that Tesseract is rather robust, as it does not assume any resolution, size of the letters on the page or any font that is used. However, in the current scenario where CITO can simply choose to use the same resolution, font and letter size, it is not a major limitation if a method does not have these properties. Also, conditions on the resolution were already set in the preprocessing in Section 2, which means that this condition not cause any new problems.

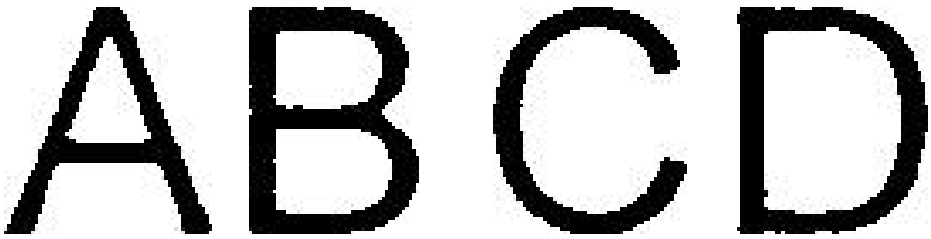
Since this method does not do any training on the cropped multiple choice answers, all data can be used to determine how well the method performs, meaning that no splitting of training and testing data is required. Testing the model is discussed in Section 4.1.

3.2.3 Convolving measure of dissimilarity method

In order to tackle the problem using a different approach, the following method was developed. At the basis of this method, the condition is set that every A, B,

C and D has the same font, size and made in the same resolution. As discussed before, this should be a fair assumption.

From the multiple choice crops, the letters are cleaned as previously described in Section 3.2.2 and one clear example for each of the four letters is chosen in such a way that there is no noise in the letters and that the letters are clear. The objective is to see for each ideal letter, as seen in Figure 6, whether there is a location on the multiple choice cutout that matches the letter well. This is done using a dissimilarity measure.



(a) Ideal A (b) Ideal B (c) Ideal C (d) Ideal D

Figure 6: Ideal letters used to see whether the letters are matched well.

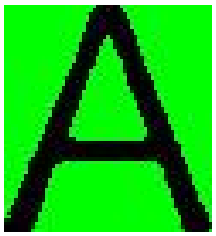
Definition of dissimilarity measure
At every location of the cropped image, the ideal images can be placed. At this location, the idea is to slice the image and find a measure of dissimilarity to determine how similar the images are. One method to determine the similarity would be to simply use the Euclidean distance and do this for all the slices. However, this would be computationally inefficient and since many comparisons are required, one for every location, this would be infeasible.

For that reason, the convolution operation is used. This idea is inspired by the filter in a Convolutional Neural Network. The convolution operation is explained to be the following, according to Chollet [2021]. One can think of the convolution operation as sliding a smaller matrix over a larger matrix. For every location, the values of the smaller matrix are multiplied element-wise with the values of the corresponding slice of the larger matrix and the sum is taken over all these values. Contrary to the Convolutional Neural Network, no filter is calculated, but the ideal letters are slided across the cropped image, as if these images are the filters. The settings are set such that the ideal letters only take positions where the ideal letter is fully on the cropped image, meaning that some border locations are disregarded. The convolutional operator is done using the `convolve2d` function from the Scipy package, as this allows for two-dimensional matrices as an input.

Before doing this operation, both the cropped image and the ideal letter are

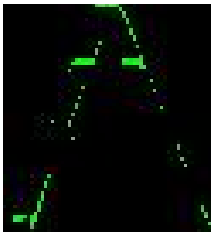
preprocessed by setting a threshold and normalising the pictures to an interval from 0 to 1. The result is then that at every location in the images, which are in fact matrices, the value is either 0 or 1. Then, at every location (which is achieved using the convolutional operator) and for each individual ideal letter, the following calculations are done.

- Multiply the ideal letter with the slice of the cropped image element-wise and take the sum of all values. For the ideal letter, all values outside the letter are equal to 1 (white areas) and all values inside the letter are equal to 0 (black areas). A perfectly matching position would have the same properties. For that reason, this multiplication is in the case of a perfect match equal to the sum of all values in the ideal letter image. The worse the match, the lower the actual multiplication is. A good and bad match are illustrated in Figure 7a and Figure 7b, respectively.
- Multiply the inverse of the ideal letter with the slice of the cropped image element-wise and take the sum of all values. This inverse has all values within the letter equal to 1 and all values outside equal to 0. If multiplying this with a perfect match, this element-wise multiplication results in 0, as all values are opposite. The worse the match, the higher this multiplication is. A good and bad match are illustrated in Figure 7c and Figure 7d, respectively.

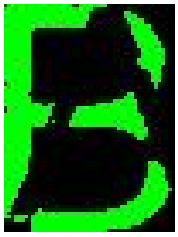


(a) Good match using the ideal letter A and a cropped image of an A.

(b) Bad match using the ideal letter B and a cropped image of an A.



(c) Good match using the inverted ideal letter A and a cropped image of an A.



(d) Bad match using the inverted ideal letter B and a cropped image of an A.

Figure 7: Overlaying the ideal images with locations on the crops. Green areas indicate locations where the element-wise multiplication results in a 1. This comparison shows how A's match with each other and how A's match with B's.

It should be noted that since this multiplication is performed at every location, the power of such a convolutional operator really comes to use. This is much more efficient compared to doing a Euclidean distance for all slices.

To obtain a measure of how good of a match an ideal letter gives with the cropped image, it is necessary to do both calculations above and combine them. This is because a completely white area would give the ideal value in the first multiplication and a completely black area would give the ideal value in the second multiplication. Hence, both are combined by squaring the distance to the ideal value and summing those, again for every location. This gives a measure of dissimilarity to the ideal letter at every location.

The formula explained previously can be described as given in Equation 2. It should be noted that this calculation is only for one separate location and that this is repeated efficiently using the convolutional operator.

$$\begin{aligned} image\_dissimilarity_{id} =& (sum(img_{id} \cdot img_{crop}) - sum(img_{id}))^2 + \\ & (sum(inv(img_{id}) \cdot img_{crop}) - 0)^2 \end{aligned} \tag{2}$$

In this equation, img_{id} is the image of an ideal letter and img_{crop} is the image slice, at the set location, of the cropped image. These are both in matrix format. inv is a function taking the inverse, setting all 0's to 1's and vice versa. The multiplication sign is an element-wise multiplication of matrices. This formula achieves the goals described above and can be seen, for any location in the convolutional operator, as the dissimilarity between the ideal letter and that slice of the image.

When applying the procedure above for a single ideal letter and a single cropped image to classify, the result is that at each location where the ideal letter is placed at the cropped image, a number is given for the dissimilarity. That means that the result is a matrix, where each entry in this matrix indicates the dissimilarity at one of these locations. When doing this for all four ideal letters, A to D, this gives four matrices, all indicating the dissimilarities to the one cropped image at each location. The minimum of each matrix is taken, since this should represent the dissimilarity at the location of the best match for each ideal letter. This makes sense, as one is interested to know the best match, as that is likely a candidate location for the letter. The result of this full measure of dissimilarity calculation, is that a number can be assigned for the dissimilarity between the cropped image and any of the four letters, represented by their respective ideal images.

Lastly, it should be noted why this calculation is in fact a measure of dissimilarity and not a metric. A metric has a very specific definition. As stated by Chen et al. [2009], the conditions that should be satisfied are non-negativity, symmetry, the triangle inequality and identity of indescernibles. These are not mathematically stated in this paper, as they would require a broader introduction into all elements used in this definition. However, as a smaller matrix is convolved over a larger matrix, after which the minimum value is taken, it is clear that there is no symmetry present. It is not possible to do this operation

Class	Precision	Recall	F1-score	Support
A	0.99	0.95	0.97	125
B	1.00	0.97	0.98	187
C	0.60	1.00	0.75	195
D	0.00	0.00	0.00	118
accuracy			0.79	625
macro avg	0.65	0.73	0.68	625
weighted avg	0.69	0.79	0.72	625

Table 3: Classification report of convolving measure of dissimilarity method.

in the reverse order, as one cannot move a larger matrix over a smaller matrix using the convolving operation. Even though other conditions are also not satisfied, showing that there is no symmetry is enough to show that a measure of dissimilarity is required instead of a metric.

Application of method
Now that a measure of dissimilarity is known, this can be applied to the crops. This measure of dissimilarity is applied in a straightforward way at first, simply taking the letter with the minimum dissimilarity. To measure its performance, the labeled crops from Section 3.2.1 are used.

As the results of this method are not good at this point and as this result is necessary for the further improvement of this method, they are already discussed in this section. There are very large differences in the performance of each of the letters. Interestingly, on the data, all A, B and C are classified in a rather good way. However, every occurrence of D is classified as a C. This becomes clear from Table 3 and Table 4.

Correct label	**Predicted label**			
	A	B	C	D
A	119	0	6	0
B	1	181	5	0
C	0	0	195	0
D	0	0	118	0

Table 4: Confusion matrix of Tesseract method.

Here it is clear that the method definitely has some predictive power, but that it runs into troubles with the C and D. This means that in its current state the model is useless, as it is completely unable to recognise any D's. However, it can be interesting to see whether there is any 'hidden' predictive power in these dissimilarity values for all four labels. It can be the case that there are patterns within these dissimilarity values that allow for correct classification of

the labels. This option is explored below.

3.2.4 Tree based methods

For the classification of the labels using the minimum dissimilarities per label, multiple methods were attempted. Since this problem could turn out to be a simple classification problem, the Logistic Regression was used at first to attempt to solve the problem. The performance of the simple Logistic Regression model's performance can be found in Appendix B. Without using interaction terms, this method is unable to discover more complex patterns, which seems to be the case in this situation. Adding interaction terms would mean that the model would increase in size significantly. To avoid using too many useless predictors and to avoid overfitting with so many variables, one solution would be to use penalty terms, as for example done in the LASSO regression. [Ranstam and Cook, 2018] However, instead of exploring this option completely, simpler approaches are attempted.

Since specific combinations of the minimum dissimilarities could be determining the label to classify, tree based methods can be a good choice for this classification problem. Below, two of these algorithms are tried. First of all the decision tree, which is a rather simple model, able to discover simple patters only. However, it could be the case that the patterns are much more complex, which is why the random forest could be a good classifier as well.

It should be noted that the information on tree based methods that the text below is based on, can be found in James et al. [2021].

Decision Tree Classifier
The decision tree classifier is an algorithm that splits the data into classes that are as much alike as possible. So in this case, the decision tree splits the data in groups containing majorities of A, B, C and D with as little impurity as possible. The decisions on when to split are based on a cutoff value, where all instances with a value below the cutoff value are split into one group and all instances with a value above the cutoff are split into another group. A thorough explanation of this algorithm is outside of the scope op this thesis, as only the main idea behind the algorithm is relevant for this process.

Random Forest Classifier
A more complex method is the random forest classifier. This method is a tree based method, as it is built using many decision trees. By sampling from both the data as well as the predictors and by doing many runs, this model becomes a good predictor in a large amount of cases. A more technical explanation can again be found in James et al. [2021], but for the scope of this research, a basic understanding is all that is required.

Both the decision tree and random forest are implemented using Scikit Learn

[Pedregosa et al., 2011]. For the decision tree, a maximum depth of 3 is used to achieve good results. For the random forest, the standard settings are used, since no hyperparameter tuning is required to obtain good results, which is shown in Section 4.

Comparison of tree based methods
In general, a good habit in the field of data science is to use models as simple and as interpretable as possible. When comparing these two tree based models, the decision tree classifier is clearly the simplest model, as the random forest is made using many decision trees. Also, the decision tree classifier is easy to interpret, an advantage that the random forest does not have. However, in this case this last point is not much of a benefit, since the input consists of the minimum dissimilarities with the ideal letters, which is already not too trivial to understand to begin with.

James et al. [2021] states that *"trees generally do not have the same level of predictive accuracy as some other regression and classification approaches"*. Therefore, it will be interesting to see whether the random forest is indeed better at classifying the labels, or whether the underlying pattern is simple enough. Keep in mind that the maximum tree depth is set to 3, which means that the trees used for this research are really simple.

A risk of using trees, according to James et al. [2021], is that trees can be very sensitive to small changes in data. Therefore, it is important to see whether the results coming from trees are stable enough. This will be assessed with K-fold cross-validation.

Repeated K-fold cross-validation
To determine which model performs best at classifying the labels among the two presented tree based models, training and testing data should be separated and the performance should be tested on the testing data. However, when testing this, there is some instability in the results that should be overcome, as in different runs different models are preferred. This could be caused by a lack of testing data. One way to overcome this, is by taking different combinations of training and testing data, which can be done with K-fold cross-validation. According to James et al. [2021], this means splitting the data in K parts. Every part serves once at the testing data, where the rest of the data is the training data.

One example of this goes as follows. If $K = 5$, that means that the data is split up into 5 parts, each containing 20% of the data. In the first run, the first part is the testing data and the other 80% is the training data. Then, the same happens with all the other parts one by one. This means that it is possible to use the whole dataset for testing using the 5 splits, either by keeping 5 metrics of success, or by concatenating all parts again and checking the performance on the whole dataset. Note that in this case, it is possible to check performance

on the whole dataset, while the model used to predict on a testing instance has never 'seen' that specific instance before.

For this research, 10-fold cross-validation is used. This is done using the `RepeatedKFold` function from Scikit Learn [Pedregosa et al., 2011]. Note that repeated K-fold cross-validation differs from normal K-fold cross-validation by repeating the process a number of times. In the context of this research, this means that next to doing the 10-fold cross-validation, this function also repeats the process, in this case 10 times. In any of the 10 runs, the 10 folds are different random selections. This increases the stability of the performance assessment and allows for making good inferences about the performance of the two models.

Lastly, keep in mind that, as already discussed in Section 3.2.2, no validation with training and testing data is required for the Tesseract method, as no training is done on the data. For that reason, K-fold cross-validation is also not used on this method.

4 Results

4.1 Tesseract method

As a result of applying the method discussed in Section 3.2.2, the classification report can be found in Table 5. In this classification report, a few things should be noted. First of all, the precision is very high, but the recall is lower. This is because in some of the pictures, no letter from A to D was discovered, which means that it was classified as 'other'. For that reason, other has all 0's in the classification report, as there was no cropped image labeled as 'other' in the labeled dataset. This also becomes clear from the confusion matrix, which can be found in Table 6. This table shows more clearly where the mistakes are being made and it also shows the situation with the label 'other'.

Class	Precision	Recall	F1-score	Support
A	1.00	0.97	0.98	125
B	1.00	0.97	0.99	187
C	0.99	0.92	0.95	195
D	0.99	0.92	0.95	118
other	0.00	0.00	0.00	0
accuracy			0.94	625
macro avg	0.80	0.75	0.77	625
weighted avg	0.99	0.94	0.97	625

Table 5: Classification report of Tesseract method.

Correct label	**Predicted label**				
	A	B	C	D	other
A	121	0	0	0	4
B	0	182	0	1	4
C	0	0	179	0	16
D	0	0	2	108	8
other	0	0	0	0	0

Table 6: Confusion matrix of Tesseract method.

When discussing the performance, the category 'other' deserves special attention. The presence of 'other' is two-sided, because of the following. On one hand, having a category that shows that no classification was found can be essential for determining when a teacher should review the test. On the other hand, the category 'other' should ideally only be used when an error took place at selecting the crop. In this relatively clean dataset one letter from A to D should be present in any of the images, so selecting 'other' shows that the model is not able to find a letter in some cases while it is present in the cropped image.

In Table 5, it can be seen that the accuracy is 94%. It is preferred to see an improvement there, as this error margin is too large for implementation. Although referring some questions to the teacher is not problematic for implementation, it is preferred to be able to classify a higher percentage of the multiple choice questions.

4.2 Convolving measure of dissimilarity with tree based methods

4.2.1 Decision Tree Classifier

When performing the decision tree as described in Section 3.2.4, the results are shown in Table 7 and Table 8. Recall that this is done using a 10 times repeated 10-fold cross-validation, which is the reason why the numbers in the confusion matrix and in the support of the classification report are much higher for this method. To get an idea of what such a decision tree looks like in this scenario, an example is added in Appendix A with a short explanation of what this example entails.

Class	Precision	Recall	F1-score	Support
A	1.00	1.00	1.00	1250
B	1.00	1.00	1.00	1870
C	0.99	0.99	0.99	1950
D	0.98	0.99	0.99	1180
accuracy			0.99	6250
macro avg	0.99	0.99	0.99	6250
weighted avg	0.99	0.99	0.99	6250

Table 7: Classification report of convolving measure of dissimilarity method with decision tree.

Correct label	**Predicted label**			
	A	B	C	D
A	1245	0	0	5
B	0	1869	0	1
C	0	0	1934	16
D	0	0	10	1170

Table 8: Confusion matrix of convolving measure of dissimilarity method with decision tree.

In these tables, it becomes clear that this model has a much better performance. The accuracy of this method is equal to 99.49%. Although there are still some mistakes being made, this is much lower than in the model based on Tesseract. Note that in the confusion matrix, the total values are 10 times as high compared to Tesseract, meaning that the errors are relatively much lower

than in the Tesseract-based model.

As a result, it can clearly be said that the method based on the convolving measure of dissimilarity with a decision tree outperforms the approach with Tesseract, when doing a short comparison. The model's performance is further discussed and compared with the other models in Section 5.

4.2.2 Random Forest Classifier

When performing the random forest that was also described in Section 3.2.4, the results are found in Table 9 and Table 10.

Class	Precision	Recall	F1-score	Support
A	1.00	1.00	1.00	1250
B	1.00	1.00	1.00	1870
C	0.99	0.99	0.99	1950
D	0.99	0.99	0.99	1180
accuracy			1.00	6250
macro avg	1.00	0.99	0.99	6250
weighted avg	1.00	1.00	1.00	6250

Table 9: Classification report of convolving measure of dissimilarity method with random forest.

Correct label	**Predicted label**			
	A	B	C	D
A	1245	0	0	5
B	0	1870	0	0
C	0	0	1940	10
D	0	0	14	1166

Table 10: Confusion matrix of convolving measure of dissimilarity method with random forest.

This model also clearly performs very well. The accuracy of this method is 99.54%. When doing a short comparison with the decision tree based model, this model seems to do slightly better. In Section 5, these methods are compared more thoroughly to determine whether one model is preferred over the other.

5 Conclusion and Discussion

Based on the findings in Section 4, the following comparison of the methods can be made.

5.1 Tesseract method

First, the Tesseract model is discussed. As described in 3.2.2, an advantage of this method compared to the other methods, is that the model indicates when no solution has been found. This could be useful for the teacher, so that they know that manual inspection is required. However, when looking at the confusion matrix, it becomes clear that this is not enough of an advantage, for the following reason.

The ideal situation would be that the cases that the other models classify incorrectly, are classified as 'other' in this model. However, this is not the case. In the confusion matrix of the Tesseract based model, 3 misclassifications take place when ignoring the category 'other'. If the ideal situation is taking place, this number should be lower than the misclassifications in the tree based methods using the convolving measure of dissimilarity. To compare this to these other methods, this number has to be multiplied by 10 (making 30 in total), as the sample size of the tree based methods is 10 times as large, caused by the repeated cross-validation process. However, when this comparison is done, the decision tree classifier misclassifies 32 labels and the random forest classifier misclassifies 29 labels. Since there are no large differences, it must be concluded that the Tesseract based method simply classifies letters to be 'other' in cases where the tree based methods does make the correct prediction.

Even though there are some more advantages previously discussed, relating to the fact that Tesseract has a better flexibility and robustness compared to the other models, it can be concluded that the Tesseract based method does not have sufficiently high performance compared to the other methods and that it should not be used for the final solution to classify multiple choice questions.

5.2 Convolving measure of dissimilarity with tree based methods

Since the Tesseract based method seems to have clear disadvantages, the tree based methods using the convolving measure of dissimilarity can be compared with each other.

At first, notice the very high values in the classification reports found in Table 7 and Table 9. Almost all metrics are either 99% or 100%, except for the precision of class D in the decision tree classifier. This is what is required for a good solution. Of course, metrics being even closer to 100% would be better in both cases, but is does appear that both models are very well able to distinguish

the classes from each other.

Note that all mistakes are made in the confusion matrices with labels including a D, whether there is a correct label D that is not predicted or whether there is a predicted label D that is not correct. This makes sense, as one can recall that the original problem with the convolving measure of dissimilarity was with classifying the D's.

If one is to closely compare the precise values in the confusion matrices, the decision tree based method makes slightly more mistakes. On these 6250 attempts for labeling crops, coming from the repeated cross-validation, the decision tree based method makes 32 mistakes, where the random forest based method makes 29 mistakes. In terms of accuracy, the decision tree based method has an accuracy of 99.49%, where the random forest based method has an accuracy of 99.54%. This difference is so small, that this difference could even be due to random chance. For this reason, it is more important to consider other properties of both methods, although both methods can be considered to be good candidates.

It can be said to be somewhat surprising that the decision tree method performs so well. As previously discussed, trees do not tend to have the same predictive accuracy. The fact that this method still has such a high accuracy, likely means that the underlying patterns in the data are very simple, especially considering that a maximum tree depth of 3 is used. This means that a method such as the random forest classifier would be unnecessarily complex, especially since there is not a significant gain in performance. The concern about a lack of robustness can also be disregarded for the decision trees. The cross-validation shows that for many subsets of training data, the predictive power remains high. For this reason, it can be concluded that there is a high generalisability of the decision trees in this specific use case.

After seeing all the advantages of the decision tree based method when comparing it to the random forest based method, it becomes clear that the decision tree based method is the preferred one of all methods to classify multiple choice questions.

5.3 Performance multiple choice classification

It is now clear that the performance of the method using the convolving measure of dissimilarity with the decision tree is preferred, with an accuracy of 99.49%. However, it must be noted that this method's performance cannot be considered to be the total performance of classifying multiple choice questions. If any mistakes are made in the process or extracting the crops, these lower the total accuracy of the multiple choice classification.

In Section 2.2.4, it is discussed that YOLOv5's accuracy is 91.14%. This

means that in order to obtain the accuracy of the total process of extracting the answers and classifying them into options from A to D, the two accuracies must be multiplied, as it is known that for the correctly extracted 91.14% of the answers, 99.49% is correctly classified. This was previously introduced in Section 2.2.4. In Equation 3, the total multiple choice accuracy is calculated.

$$accuracy_{tot} = 0.9114 \cdot 0.9949 = 0.9068 \tag{3}$$

The total accuracy of the multiple choice questions is 90.68%. Unfortunately, this is much lower than the accuracy of classifying correctly extracted multiple choice questions, but this is a realistic number describing many questions can be correctly classified when taking the full process into account.

Lastly, note that in the settings of YOLOv5, when determining the accuracy of YOLOv5 on the multiple choice questions, the feature was selected that allows only one answer to be classified. This was described in Section 2.2.4. This is done to ensure that only one answer is selected, which is the answer with the highest assigned probability, which is one of the outputs of YOLOv5. However, it should be kept in mind that this answer with the highest probability is not necessarily the correct answer. It could for example be the case that this is an answer that was erased. For that reason, when implementing YOLOv5 with the multiple choice questions, a more critical look at finding the correct answer is necessary, instead of simply finding any handwritten text. This could improve the performance of YOLOv5, meaning that this accuracy is possibly an underestimation of the maximum accuracy that can be achieved with YOLOv5. However, as this thesis focuses on classifying multiple choice questions, this is outside of the scope of this research. More detailed discussions can therefore be found in Schreurs [2022].

5.4 Answer to the data science problem

In Section 3.1, the data science problem was formulated. The goal of this research was to see whether it is possible to classify the handwritten numbers and multiple choice answers. The classification of handwritten numbers was touched upon by other team members, which is why this will not be answered in this thesis.

When taking the steps as described in this research, it turns out not to be feasible to grade multiple choice answers automatically in the provided format. As previously discussed, this is due to difficulties with extracting the multiple choice questions correctly. Every test in the provided format contains around 10 multiple choice question. This means that the found accuracy that is close to 90% implies that on average, it is expected that the methods described in this research incorrectly classify 1 question. This could mean that most students would not get the correct grade on the multiple choice part of the test, or

possibly that some students get a grade that is very far off their actual grade. This is of course not an acceptable margin of error.

5.5 Answer to the research question

Recall from Section 1.3 that the research question was formulated as follows.

How should CITO approach automatically scanning mathematics tests and is this approach reliable enough to implement in practice?

Based on the answer of the data science question, taking only the multiple choice questions into consideration, the answer to this research question can be given as follows. Multiple choice questions cannot be reliably classified with a convolving measure of dissimilarity in combination with a decision tree classifier and YOLOv5 as preprocessing. The approach is not reliable enough to implement in practice, due to a high error rate. For that reason, if no improvements are made to the extraction process, CITO should keep checking mathematics tests manually.

To answer this research question for the handwritten open questions, please refer to the theses Klopper [2022] and Schreurs [2022].

5.6 Implications for the proper domain settings

The implications for the domain of automatic test recognition is as follows. With this thesis, it has been shown that with the methods described in this research, it is difficult to classify multiple choice answers correctly. If one is to attempt implementing an automatic grading of tests, the preprocessing and extraction of the given answers are processes to improve, as these are the main limiting factors. However, even when these issues would be solved, it is not possible to simply apply the methods without considering any implications regarding errors and their consequences. For this reason, any organisation that is in doubt, should very carefully describe the possible consequences and how influential incorrectly classified test answers are.

5.7 Ethical considerations

One of the most important ethical considerations, is that it is crucial to understand the implications of misclassifications. Since no true context is known for where these tests would be implemented, as these tests are practice tests, it is not possible to discuss the implications of misclassifications in this report. However, CITO or any party implementing such methods for tests that the students depend on should be aware that misclassifying the answers in only a small amount of cases could lead to large consequences for some students, especially if their level of future education or the decision on whether to graduate is de-

pendent on it.

For that reason, even systems that have an accuracy well above 99% should be carefully considered before implementation. To avoid any unethical practices by negatively influencing students, one could for example think of doing a manual check for all edge cases, such as students who are on the border of getting a different advice for their future education or students who are on the edge of graduation. In that way, the consequences can be managed relatively well. However, even then it is the question whether saving the manual work is worth the risk of making mistakes on the classification of answers.

5.8 Additional discussions

There are a few additional notes to make on the contents and the implications of this research.

First of all, one thing to discuss is the preprocessing, specifically the first part where the individual questions are recognised and where a question number is attached. Even though a lot of the focus of this research is on recognising the answers, the preprocessing is a thorough process and there are disadvantages there that have not been discussed yet. In future attempts to solve this problem, the methods should become more robust and more flexible. The selection of questions is very dependent on the layout of the page, so scanning the document in a different resolution would require adaptions to the preprocessing process. Also, to determine the locations on the page, many scans are used. Of course, ideally a preprocessing method would be able to classify individual tests with high accuracy without the context of other similar tests. Lastly, imagine the scenario that a student or teacher lost a page of a test. In that case, prefixing the locations of the questions could result in failing to recognise questions on all successive pages after the missing page. So in general, improving these concerns would be important to allow for the practical implementation of this method.

Second, as mentioned before, the methods applied in this thesis should not be compared on their own, but should ideally be compared with the status quo. For that reason, there should be some metrics available on how this method performs in comparison with manual inspection. It should not be forgotten that these methods are in practice not intelligent, as they simply follow a set of learned instructions. If a student changes their mind, crosses out an answer, puts an arrow at a new one, changes their mind again and makes a mess for that reason, teachers are still able to interpret what the student meant, where automated systems are usually not. For that reason, it is important to recognise such cases, as teachers will still be needed to do a manual check in these scenarios. Also, teachers are much more flexible in understanding the intentions of a student in case the student failed to understand how to fill in the test, where these methods are not so flexible. All these limitations contributed to the lower accuracy of recognising the right answer, but even if a higher accuracy

is achieved in future research, it is still important to keep such considerations in mind.

One consideration that could benefit future attempts to solve this research question, is to critically look at the instructions given to students. Students were supposed to circle the correct letter in the multiple choice questions. However, many students did not do so. They chose to cross out all wrong answers, circled both the letter and the attached number, or sometimes only circled the number. If it turns out that the instructions were indeed unclear, providing students with better instruction about how to fill in the test and how to correct their own mistakes, could solve some of the issues with the preprocessing. Another way to decrease the influence of this issue, it that teachers do a very brief initial check whether a student actually used the right format, as it should immediately stand out when students did not understand the instructions.

When looking at the method of recognising the multiple choice letters, as was already briefly touched upon in Section 5.1, a disadvantage is that both tree based methods are unable to determine a confidence level for the predicted label. An ideal situation would be that edge cases are separately classified for a teacher to do a manual inspection. This would be something to implement in future research.

Lastly, it should not be forgotten that this thesis is part of a broader research with two other students, whose research is described in Klopper [2022] and Schreurs [2022]. The applicability of this method goes together with their ability to classify handwritten answers. For that reason, all three theses should be considered to determine the total feasibility of the automatic grading of mathematics tests.

Acknowledgements

I would like to express my gratitude to all those who helped me with the construction of this thesis. First of all, my thanks goes out to prof. dr. Arno Siebes, who has provided our team with weekly support and guidance, to know whether we were still on the right track. Also, I would like to thank my CITO supervisors, Tjeerd Hans Terpstra and Laura Kolbe, for their weekly support and for providing us with the problem statement.

Lastly, my special thanks goes out to my team members, Tom Klopper and Aico Schreurs. I have enjoyed working with you and I believe that we managed to learn a lot from each other.

References

U Ravi Babu, Y Venkateswarlu, and Aneel Kumar Chintha. Handwritten digit recognition using k-nearest neighbour classifier. In *2014 World Congress on Computing and Communication Technologies*, pages 60–65. IEEE, 2014.

Simon Bernard, Sébastien Adam, and Laurent Heutte. Using random forests for handwritten digit recognition. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 1043–1047. IEEE, 2007.

Shihyen Chen, Bin Ma, and Kaizhong Zhang. On the similarity metric and the distance metric. *Theoretical Computer Science*, 410(24-25):2365–2376, 2009.

Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

Thomas Hegghammer. Ocr with tesseract, amazon textract, and google document ai: a benchmarking experiment. *Journal of Computational Social Science*, pages 1–22, 2021.

Itseez. Open source computer vision library. `https://docs.opencv.org/4.x/`, 2022.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. Statistical learning. In *An introduction to statistical learning*, pages 15–57. Springer, 2021.

TS Klopper. Automatic grading of handwritten math tests: A convolutional neural network approach (unpublished master's thesis). *Utrecht University*, 2022.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

MJ Prajwal, KB Tejas, V Varshad, MM Murgod, and R Shashidhar. Detection of non-helmet riders and extraction of license plate number using yolo v2 and ocr method. *Int. J. Innov. Technol. Exploring Eng.(IJITEE)*, 9(2), 2019.

J Ranstam and JA Cook. Lasso regression. *Journal of British Surgery*, 105(10): 1348–1348, 2018.

A Schreurs. Automatic school handwriting detection and classification based on yolo and vision transformers models (unpublished master's thesis). *Utrecht University*, 2022.

Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

Eva Tuba, Milan Tuba, and Dana Simian. Handwritten digit recognition by support vector machine optimized by bat algorithm. 2016.

A Appendix: Decision tree figure

In Figure 8, an example of a decision tree made for the convolving measure of dissimilarity model can be found. It should be noted that this is just one example, in fact it is the last example of the 10 times repeated 10-fold cross-validation. For this reason, the example might differ from other folds and runs in this cross-validation method.

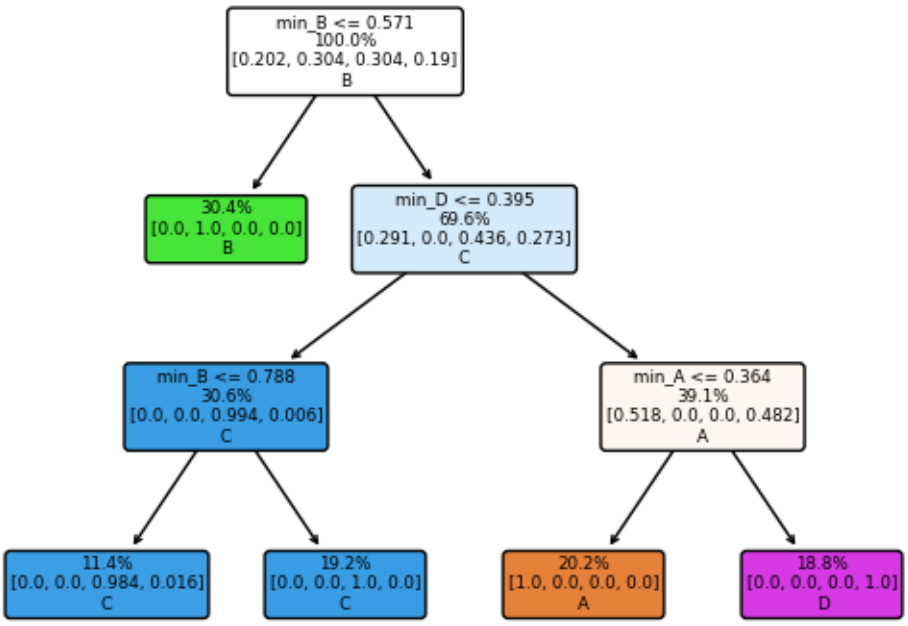


Figure 8: Example of decision tree made for the classification of multiple choice labels.

B Appendix: Logistic regression performance

Table 11 and Table 12 display the classification report and the confusion matrix of a 10 times repeated 10-fold cross-validation run of a simple Logistic Regression on the dissimilarity values to predict the labels. It can be concluded that the performance is too low to accurately predict the labels, as half of the images containing a D are still classified as a C.

Class	Precision	Recall	F1-score	Support
A	0.99	1.00	1.00	1250
B	1.00	1.00	1.00	1870
C	0.77	1.00	0.87	1950
D	1.00	0.49	0.66	1180
accuracy			0.90	6250
macro avg	0.94	0.87	0.88	6250
weighted avg	0.93	0.90	0.89	6250

Table 11: Classification report of convolving measure of dissimilarity method with logistic regression.

Correct label	**Predicted label**			
	A	B	C	D
A	1250	0	0	0
B	0	1870	0	0
C	0	0	1950	0
D	10	0	586	584

Table 12: Confusion matrix of convolving measure of dissimilarity method with logistic regression.

C Appendix: Python scripts

All code used in this thesis can be found in this project's GitHub repository, by clicking this link. In this repository, all code from the three team members in this research project is present.