

UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Auto-Assessment of Student Learning

Permalink

<https://escholarship.org/uc/item/9vt192fw>

Author

Darvishzadeh, Amirali

Publication Date

2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Auto-Assessment of Student Learning

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Amirali Darvishzadeh

June 2020

Dissertation Committee:

Dr. Thomas F. Stahovich, Chairperson
Dr. Eamonn Keogh
Dr. Christian Shelton
Dr. Stefano Lonardi

Copyright by
Amirali Darvishzadeh
2020

The Dissertation of Amirali Darvishzadeh is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to acknowledge everyone who has helped me throughout my doctoral studies. I would like to thank my research advisor, Dr. Tom Stahovich, for his patience and feedback as I completed my dissertation. I would like to thank the rest of my dissertation committee members: Dr. Shelton, Dr. Keogh, and Dr. Lonardi, for their invaluable advice and insightful comments. A big thanks to my lab-mates in the Smart Tools Lab for their support and friendship throughout the journey: Negin Entezari, Kevin Rawson, Justin Gyllen, Matt Ung, Teresa Wright, Josh Frear, Adam Koehler, Vashishtha Bhatt, and Ryan Mercer. I'd like to thank my family: Mom, Dad, Neda, Masoud, and Amirhossein, for all their support and encouragement. I'd like to thank my aunt, Farah Gholi Khamseh, and her husband, Mansour Charkhechi, for helping me get through school.

The material in Chapter 2 is based on research presented as a poster at ICFHR 2018 in August 2018. The material in Chapter 3 is based on research presented as a paper at ICDAR 2019 in September 2019.

This dissertation is dedicated to my loving wife Negin, for inspiring me and being there for me throughout graduate school. I love you Babe!

ABSTRACT OF THE DISSERTATION

Auto-Assessment of Student Learning

by

Amirali Darvishzadeh

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2020
Dr. Thomas F. Stahovich, Chairperson

Solving open-ended problems with paper and pencil is an important part of education and constitutes a large share of course workload, especially in science and engineering programs. However, grading this kind of work can be prohibitively expensive in large classes. Because of the complexity of handwritten free-form work, there are no existing computational methods capable of interpreting it. In this dissertation, we aim to work toward complete interpretation and semantic analysis of handwritten free-form solutions to homework and exam problems of the sort assigned in undergraduate engineering and science courses. In our study, students wrote their homework and exam solutions on dot patterned paper with Livescribe smartpens. Our work comprises three innovations. First, we developed methods to locate final answers. Our methods locate answers by identifying the boxes drawn around them. Experiments demonstrated that our methods are both accurate and efficient at recognizing answer boxes. Second, we developed a novel CNN-BLSTM-CRF network for semantic labeling of students' handwritten assignments. Semantic labeling is the task of classifying pen strokes according to the type of content they represent. Our

method distinguishes between cross-out strokes, equation strokes, and free body diagram strokes. The input to our network is a set of pen strokes, which are sorted in chronological order, and the output is a sequence of labels for the strokes. Labeling strokes in this manner is an important step in enabling semantic analysis of the writing. Our labeling approach outperforms existing methods. Finally, we developed a novel GRU-CRF network to locate complete mathematical equations. The network exploits the temporal context of consecutive strokes and simultaneously finds and groups equation strokes. Once our method has located the equations, they can be interpreted using existing techniques. Together, this work provides a significant step toward the automated grading of handwritten free-form course work.

Contents

| | |
|--|-------------|
| List of Figures | x |
| List of Tables | xiii |
| 1 Introduction | 1 |
| 1.1 Auto-grading Systems | 3 |
| 1.2 Online and Offline Document | 6 |
| 1.3 Approach | 9 |
| 1.3.1 Finding Final Answers | 9 |
| 1.3.2 Semantic Labeling of Students' Handwritten Assignments | 10 |
| 1.3.3 Finding and Grouping Mathematical Equations | 12 |
| 1.4 Outline | 14 |
| 2 Answer Box Recognition | 15 |
| 2.1 Introduction | 15 |
| 2.2 Answer Localization: System Overview | 17 |
| 2.2.1 Stroke Segmentation | 19 |
| 2.2.2 Stroke Grouping | 21 |
| 2.2.3 Finding Left and Right Edges | 30 |
| 2.2.4 Finding Splitters | 31 |
| 2.2.5 Unit/Value/Problem ID Stroke Grouping | 32 |
| 2.2.6 Answer Box Recognition | 33 |
| 2.3 Dataset | 37 |
| 2.4 Results | 38 |
| 2.4.1 Performance of Pairwise Classifier | 38 |
| 2.4.2 Evaluation of Candidate Group Generation | 39 |
| 2.4.3 Feature Selection and Importance | 40 |
| 2.4.4 Evaluation of Recognition Method | 41 |
| 2.4.5 Ink Interpretation | 43 |

| | | |
|----------|--|------------|
| 3 | Semantic Labeling of Handwritten Assignments | 45 |
| 3.1 | Introduction | 45 |
| 3.2 | Related Work | 48 |
| 3.3 | CNN-BLSTM-CRF Model for Stroke Classification | 50 |
| 3.3.1 | Input sequences | 51 |
| 3.3.2 | CNN (feature extractor) | 54 |
| 3.3.3 | BLSTM (encoder) | 54 |
| 3.3.4 | CRF (decoder) | 56 |
| 3.4 | System workflow | 58 |
| 3.4.1 | Network Training | 59 |
| 3.4.2 | Optimization method | 60 |
| 3.5 | Results | 60 |
| 3.5.1 | Results for Various System Configurations | 61 |
| 3.5.2 | Comparison with Related Methods | 62 |
| 4 | Finding and Segmenting Mathematical Equations in Students' Online Handwritten Assignments | 64 |
| 4.1 | Introduction | 64 |
| 4.2 | Related Work | 67 |
| 4.3 | Approach | 69 |
| 4.4 | Generating Initial Groups | 70 |
| 4.4.1 | GRU-CRF for labeling strokes | 70 |
| 4.4.2 | Input Sequences | 71 |
| 4.4.3 | GRU (encoder) | 78 |
| 4.5 | CRF (decoder) | 80 |
| 4.6 | Combining Initial Groups | 82 |
| 4.7 | System Workflow | 85 |
| 4.8 | Results | 86 |
| 4.8.1 | Results for Equation/Non-equation Classification | 87 |
| 4.8.2 | Results for Equation Segmentation | 88 |
| 4.8.3 | Comparison with Existing Methods | 90 |
| 5 | Conclusions | 93 |
| A | Examples of stroke classification results by the CNN-BLSTM-CRF model. | 100 |
| B | Examples of equation segments recognized by the GRU-CRF-SC model. | 106 |
| | Bibliography | 112 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | An example of a solution to a free response question to a statics problem. | 2 |
| 1.2 | An example of a handwritten document. Each stroke is shown with a unique color. | 7 |
| 1.3 | A typical solution to a statics problem. | 7 |
| 1.4 | The final answer to the problem is written in an answer box. The final answer and the answer box are colored in purple. | 9 |
| 1.5 | A typical solution to a statics problem. Free body diagram strokes are blue, cross-out strokes are red, and text is purple. | 11 |
| 1.6 | Equation strokes are green, and non-equation strokes are black. | 12 |
| 1.7 | Each equation group is enclosed with a red or purple box. | 13 |
| 2.1 | This answer box, which is typical of those in our dataset, contains the value “24.84,” unit “ <i>Slugs</i> ,” and problem ID “ <i>a</i> ,” separated by two vertical lines. | 16 |
| 2.2 | Challenges in answer box recognition: <i>a</i>) An example of an answer box having an irregular quadrilateral shape. Edges of the answer box at the top-left and bottom-right do not intersect at right angles. Also, the opposite sides are not equal in length. <i>b</i>) the left edge of the answer box is disconnected from the bottom and top edges, and the character “1” next to the left edge might be misrecognized as the left edge. <i>c</i>) The top and bottom edges of the answer box are overwritten with multiple strokes. <i>d</i>) Not all rectangular shapes are answer boxes. | 18 |
| 2.3 | Overview of the processing steps for locating answers and classifying strokes into unit, value, and problem ID groups. | 19 |
| 2.4 | An example of stroke segmentation. | 20 |
| 2.5 | Examples of candidate groups generated in the stroke grouping process. Each group is shown with a unique color. | 21 |
| 2.6 | The bottom and top edges of the answer box are drawn with multiple horizontal segments. Here, the strokes inside the answer box are excluded for clarity. | 22 |
| 2.7 | Features used by the pairwise classifier. | 25 |
| 2.8 | Stroke grouping in phase one. | 26 |
| 2.9 | Illustration of “Join” and “NoJoin” instances. | 27 |

| | | |
|------|--|-----|
| 2.10 | Phase two of generating answer box candidates. Here the top and bottom edge of each answer boxes are drawn with a single stroke shown with a unique color. | 29 |
| 2.11 | The blue box represents the search area for candidate strokes for a left edge. Those strokes that fall inside the blue box are colored green and are considered as candidate strokes for the left edge of the answer box. The dashed red line is an imaginary ideal left edge. | 31 |
| 2.12 | y_v is the height of a vertical edge, and y_{BB} is the height of the bounding box of a group. | 33 |
| 2.13 | (<i>left</i>), distances from the bounding box to the answer box strokes are illustrated. Those distances above the Thr_H threshold are colored red. (<i>right</i>), distances have been reduced after transforming the bounding box to a quadrilateral that better fits the boundary of the answer box. | 34 |
| 3.1 | A typical answer to a free response question in a statics course contains a mixture of free body diagrams, text (including equations), and cross-outs. Free body diagram strokes are blue, cross-out strokes are red, and text is green. | 46 |
| 3.2 | The overall architecture of the CNN-BLSTM-CRF model. | 50 |
| 4.1 | Examples of mathematical equations. | 65 |
| 4.2 | Examples of equation segments in our dataset. | 66 |
| 4.3 | The architecture of the GRU-CRF model. | 70 |
| 4.4 | A visual illustration of curvature formed at point P_i | 73 |
| 4.5 | Two equation segments. The numbers next to the strokes indicate the order in which they were drawn. | 82 |
| 4.6 | Many of the strokes in equation segment 1 intersect with equation segment 2. | 83 |
| 4.7 | Most of the equation strokes are sketched consecutively, and they are shown with green color. The red stroke is added to the equation afterward. | 84 |
| 4.8 | An example of the predicted equation segment. Strokes that are correctly included in the group are shown in green, strokes erroneously excluded from the group are shown in red, and strokes erroneously included in the group are shown in blue. | 88 |
| 4.9 | Performance of our equation grouping method without post-processing “GRU-CRF” and with post-processing by our pairwise classifier that combines small groups with neighboring large ones “GRU-CRF-SC”. | 91 |
| 4.10 | Performance of four different grouping methods for equation grouping. | 92 |
| A.1 | Equation and FBD strokes are shown in green and purple, respectively. | 101 |
| A.2 | Equation and FBD strokes are shown in green and purple, respectively. | 102 |
| A.3 | Equation and FBD strokes are shown in green and purple, respectively. | 103 |
| A.4 | Equation and FBD strokes are shown in green and purple, respectively. | 104 |
| A.5 | Equation and FBD strokes are shown in green and purple, respectively. | 105 |
| B.1 | A red box is drawn around each perfectly recognized equation segment. | 107 |

| | | | |
|-----|---|-----|-----|
| B.2 | A red box is drawn around each perfectly recognized equation segment. | . . | 108 |
| B.3 | A red box is drawn around each perfectly recognized equation segment. | . . | 109 |
| B.4 | A red box is drawn around each perfectly recognized equation segment. | . . | 110 |
| B.5 | A red box is drawn around each perfectly recognized equation segment. | . . | 111 |

List of Tables

| | | |
|-----|---|----|
| 2.1 | Pairwise features for grouping strokes in phase one. | 24 |
| 2.2 | Performance of classifiers in phase one of grouping horizontal segments. . . | 38 |
| 2.3 | Performance of grouping methods. | 40 |
| 2.4 | The best features for recognizing answer boxes. Features are ranked according to the absolute values of the weights computed by the SVM classifier. . | 42 |
| 2.5 | Performance of methods for recognition of answer boxes. | 43 |
| 3.1 | Domain dependant stroke features [62]. | 52 |
| 3.2 | Model configuration | 59 |
| 3.3 | Frequencies of stroke types in each dataset. | 59 |
| 3.4 | Performance of our model and three baseline networks. | 61 |
| 3.5 | Performance of models for the stroke classification task. | 63 |
| 4.1 | Single stroke features from [50]. | 72 |
| 4.2 | Pairwise stroke features from [50]. | 77 |
| 4.3 | Pairwise stroke features for grouping small and large equation groups. . . . | 85 |
| 4.4 | Frequencies of stroke types in each dataset. | 87 |
| 4.5 | Confusion matrix for equation/non-equation classification. | 87 |
| 4.6 | Comparison of the average running time for grouping equation strokes on a page. | 91 |

Chapter 1


Introduction

In formal education, assessment is the process of collecting and quantifying student learning. It plays an important role in the teaching and learning processes. Educational assessment serves several purposes: enabling teachers to watch students' progress, providing students with feedback on their performance, motivating students, providing insights on the effectiveness of teaching approaches, etc. There are two forms of educational assessment: summative [10, 65, 64] and formative [8, 13, 55]. Summative assessment is conducted at the end of the learning period and aims to provide a metric to quantify student learning of course objectives. Formative assessment evaluates student learning during the course of study and provides feedback to both the instructor and student. There are many established methods for both summative and formative assessments, with different methods for capturing aspects of student learning.

Despite the benefits of assessment, grading students' assignments may be prohibitively time-consuming, especially in large classes. Due to time constraints, many in-

2/2/17 Quiz 3

Determine moment about point O.



$F = 350 \text{ lb}$

$$\vec{F} = (F \sin 30^\circ) \hat{i} + (F \cos 30^\circ) \hat{j} + (0) \hat{k}$$

$$\vec{r} = 1.5 \hat{i} + 0 \hat{j} + 0.75 \hat{k} \text{ in.}$$

unit
lb·in $\vec{\Sigma M} = \vec{r} \times \vec{F}$

$$= \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ -F \sin 30^\circ & F \cos 30^\circ & 0 \\ 1.5 & 0 & 0.75 \end{vmatrix}$$

$$= (0) \hat{i} + [(-F \cos 30^\circ)(1.5)] \hat{j} + [(-F \sin 30^\circ)(0.75)] \hat{k}$$

$$= -(454.66 \hat{j} - 131.25 \hat{k}) \text{ lb}\cdot\text{in}$$

Magnitude: $\sqrt{454.66^2 + (131.25)^2}$
 $= 473.225 \text{ lb}\cdot\text{in}$

$$\vec{\Sigma M} = (454.66 \hat{j} - 131.25 \hat{k}) \text{ lb}\cdot\text{in}$$

Figure 1.1: An example of a solution to a free response question to a statics problem.

structors limit their assessment to things that can be easily graded by humans or computer systems such as fill-in-the-blank questions, surveys, and multiple-choice questions. However, in many academic subjects, especially in science, technology, engineering, and math (STEM) disciplines, pen and paper solutions to free-response questions are at the heart of education and cannot be effectively replaced by multiple-choice exams.

A typical answer to a free response question from a statics course is shown in Figure 1.1. Statics is the branch of mechanics that is concerned with the equilibrium of mechanical systems subject to forces. The solution to this kind of problem may contain (1) free body diagrams which are a graphical representation of the forces acting on a mechanical system; (2) handwritten equations that include arithmetic symbols, alphanumeric characters, etc.;

(3) diagrams representing geometry; (4) text that may include explanatory text, student name, and identification, organizational information, etc.

Grading such problems is a time-consuming task. Given the free-form nature of the solution, it may take significant effort to interpret the solutions. It would not be unreasonable for a grader to spend 5 minutes on such a problem. In a class of 150 students, this would be 12.5 hours for one problem. Thus, grading an exam or homework assignment with multiple problems could take an exorbitant amount of time. Therefore, there is a need for automated methods of grading.

1.1 Auto-grading Systems

To date, many researchers have studied various methods for automatically grading students' assignments. There are two primary ways to do an automated assessment: (1) change the assessment to make it easier to grade and (2) develop sophisticated tools that attempt to grade traditional free-form work.

The most extreme examples of the first approach are fill-in-the-blanks questions, surveys, and multiple-choice questions. These types of assessments allow no free-form work and instead constrain the answers to make them easy to grade.

More ambitious approaches try to maintain some characteristics of free-form work while constraining the process of constructing the solution so that the individual solution elements can be easily identified and then evaluated. For example, Lee and Stahovich [39] developed *Newton's Pen*, a tutoring system for statics problem. Students use smartpens and dot-patterned papers preprinted with user interface elements to answer statics problems.

Students are required to separate different parts of the answer (e.g., free body diagrams, equations, etc.) and sketch them in predefined locations on the page. The user must follow a specific process for solving a problem so that the system can identify and interpret the individual solution elements. As a result, the system is able to provide real-time feedback about the correctness of each solution element.

Similarly, Field et al. [25] developed a computer-assisted tutoring system called *Mechanix* that aims to improve students' ability to sketch free body diagrams and perform vector analysis. The system relies on the instructor to provide the correct solution to a problem by sketching it with the software. Once a student has sketched a solution, *Mechanix* recognizes the elements and compares them to the elements of the instructor's solution. *Mechanix* then notifies the student if there is any difference, and hence any errors, in the solution.

Silva et al. [18] developed a tutoring system called *Kirchhoff's Pen*, which is designed to teach students how to apply Kirchhoff's voltage law and current law. To utilize the system, a student sketches the schematic of a circuit, annotates its various parts, and writes the corresponding equations. The system is able to interpret the equations and provides feedback about any errors.

While these systems do interpret handwritten work, they constrain the solution process to facilitate interpretation. These systems cannot handle free-form work. There have been some systems that do consider free-form handwritten work, but they do not attempt to interpret the work. Instead, they extract features from the writing and use these features to estimate the correctness of the solutions.

One study conducted by Stahovich and Van Arsdale [66] investigated the relationship between the spatial and temporal organization of a student’s handwritten solution to the statics problem and the correctness of the work. They use 10 features to characterize the organization and hence the student’s fluency in problem-solving. These features quantify properties of the writing such as the extent to which writing is done from the top to the bottom of the page without revisions, the number of breaks (episodes with no writing), the number of digressions (episodes of interrupting work on the present problem or work on another), the amount of writing for equations, etc. They found that these features were correlated with correctness. More specifically, in a study that examined solutions to 13 exam problems from 122 undergraduate students, they found that these features explained 40% of the variance in the scores of the correctness of the solutions.

Similarly, Lin et al. [42] introduced a novel pipeline to identify and segment important types of content in students’ handwritten coursework, and then they performed a lexical analysis to measure the correctness of the solutions. First, their method uses 10 special-purpose features to identify and remove cross-outs (pen strokes used to cross out erroneous or otherwise undesired writing). Second, they perform preliminary recognition to recognize prevalent elements in solutions such as alpha-numeric characters, arrows, algebraic operators, etc. Third, they split the remainder of the written work (i.e., everything except the cross-outs) into free body diagram and equation strokes. They then group the equation strokes into “equation groups,” sets of pen strokes that represent portions of an equation. Then they use Kara’s image-based recognizer [35] to identify letters, digits, math symbols, and Greek letters in the equation groups. Finally, they compute the lexical properties of

the equation groups, such as the number of various classes of characters (e.g., digits, letters, mathematical symbols), ratios of these numbers, and so on. They conducted experiments on a collection of exam solutions produced by 147 students registered in an undergraduate statistics course taught at the University of California, Riverside. They found that these lexical properties correlated positively and significantly with the grades provided by human graders. On a dataset comprising free-form solutions to six midterm exam problems, their model predicted positive correlations between lexical features and the grades provided by teaching assistants, and the correlation coefficients ranged from 0.27 to 0.37.

Due to the complexity of free-form handwritten solutions, there are no existing methods that are capable of interpreting them. The goal of my research is to work toward complete interpretation and semantic analysis of handwritten free-form solutions. My research contributions involve (1) the development of methods for locating the final answers to free-response questions, (2) improving the accuracy of semantic classification of strokes in students' handwritten assignments, and (3) creating a method for finding complete mathematical equations in students' handwritten assignments.

1.2 Online and Offline Document

There are two forms of handwritten documents: offline and online. Offline documents are traditionally created on paper, and we can use scanners to digitize them. The digitized form of offline documents is bitmap images where each pixel denotes the amount of intensity (gray level) to be displayed. Online documents are created by using special pens, digitizing pads, computer tablets, and other devices that are capable of recording

$$W = mg$$

Figure 1.2: An example of a handwritten document. Each stroke is shown with a unique color.

the writing activity. In the online documents, each page is recorded as a series of consecutive strokes where each stroke is a series of points collected from a pen-down to a pen-up event. Each stroke contains the two-dimensional coordinates of the points along with their timestamps. In this thesis, we are interested in providing methods for interpreting online handwritten documents. Figure 1.2 shows an example of online writing.

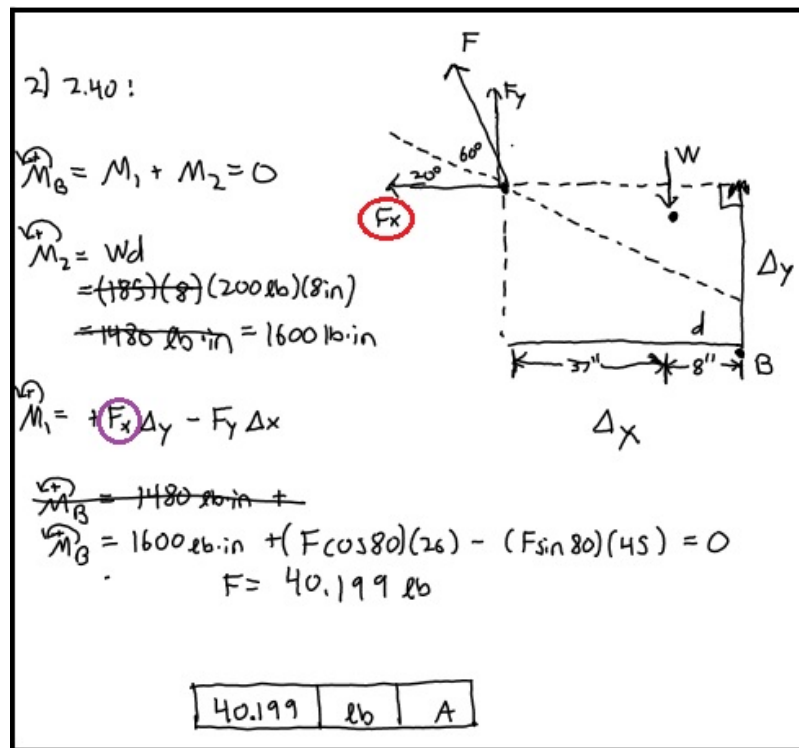


Figure 1.3: A typical solution to a statics problem.

A typical solution to a statics problem is shown in Figure 1.3. Our task is to evaluate the correctness of such solutions automatically. To estimate correctness, we need to build a system that not only recognizes the handwritten shapes and alphanumeric characters in the solution but also is able to understand the meaning of these objects in relationship with each other. For example, the F_x that is contained within a red circle is used to label the arrow next to it, and the F_x that is contained within a purple circle is part of a mathematical equation. The same symbols have different meanings in the two contexts. Thus interpreting these symbols requires both understanding the characters themselves and understanding how they relate to the other objects in the solution.

In this thesis, we take several important steps toward enabling the automated understanding of free-form handwritten solutions. First, we develop methods for locating final answers to the free-response questions. Once the final answers are found, we can use existing text recognition techniques to validate the final answers. In the second step, we develop a method that achieves higher accuracy in the semantic labeling of handwritten solutions when compared to existing tools. The method can be utilized by Lin's [42] lexical analyzer for predicting the correctness of the solution, or by future auto-grading systems that require semantic labeling of students' handwritten assignments. To date, many methods [69, 74, 73] have been developed for interpreting mathematical expressions. These methods process one equation at a time and require an entire equation to be provided as an isolated object. However, in the kinds of handwritten solutions we consider, the equations are not isolated, and locating them is challenging. In the third step, we develop a method for finding and grouping complete mathematical expressions, and the output of this method can be passed

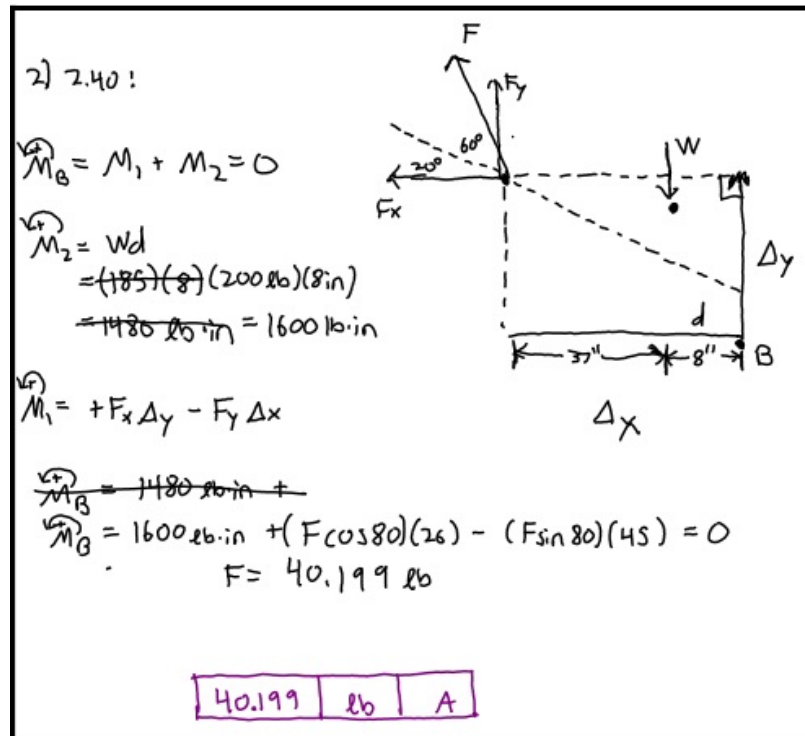


Figure 1.4: The final answer to the problem is written in an answer box. The final answer and the answer box are colored in purple.

to the available equation interpreters [69, 74, 73].

1.3 Approach

Here we provide a brief overview of the methods developed in this thesis. Complete details are provided in the chapters that follow.

1.3.1 Finding Final Answers

Our first step toward enabling the automated interpretation of handwritten free-form solutions to statics questions was to develop techniques for locating final answers. In

the solutions, we consider students drew boxes around their final answers and separated different parts of the answers via vertical lines. An example of an answer box is shown in Figure 1.4. In this example, an answer box is colored in purple along with its corresponding answer. In this case, the answer to part “A” of the problem is “40.199 lb.”. Thus, the answer box contains the value “40.199,” the units “lb,” and the problem ID “A.” These three parts of the answer are separated from each other by vertical lines, which we call splitters in this thesis.

We developed a novel answer box recognizer that can locate these kinds of answer boxes. The recognizer also identifies the splitters and the three parts of the answer. These three parts can then be sent to a text recognizer for interpretation.

1.3.2 Semantic Labeling of Students’ Handwritten Assignments

In the second step, we developed methods to identify the type of contents in students’ handwritten solutions. The pen strokes in a typical handwritten solution to a statics problem can be categorized into the following three classes:

- **Free body diagram:** Pen strokes in this class comprise graphics that show the relationship between the forces and the objects of interest. Diagrams often include arrows, geometric shapes, and text.
- **Text:** Pen strokes in this class comprise alphanumeric characters. Most text in a statics solution comprises equations, although the text is also used for organizational information, explanatory notations, and the like.

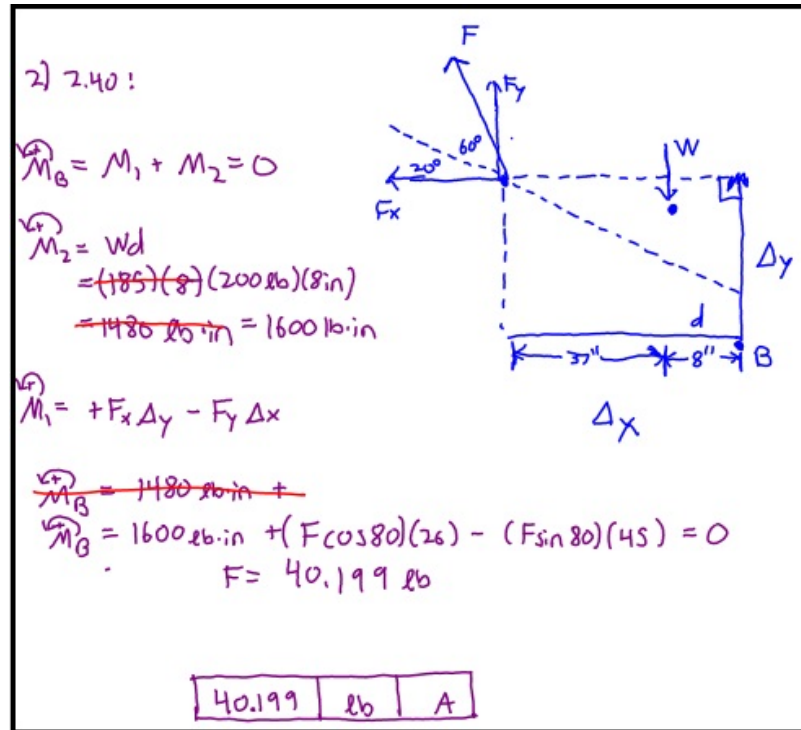


Figure 1.5: A typical solution to a statics problem. Free body diagram strokes are blue, cross-out strokes are red, and text is purple.

- **Cross-out:** Pen strokes in this class are used to indicate that other strokes are intended to be excluded from the solution. Because students wrote their solutions with pens, undesired writing could not be erased, but instead was crossed out.

Figure 1.5 shows examples of pen strokes of the various classes.

Our approach to labeling employs a deep neural network model. The input to the network includes both the raw trajectory of the pen strokes (i.e., the coordinates of the points) and a set of features computed from each pen stroke. The output of the network is a classification of each stroke. Our method is fast and outperforms state-of-the-art methods in semantic labeling of handwritten assignments.

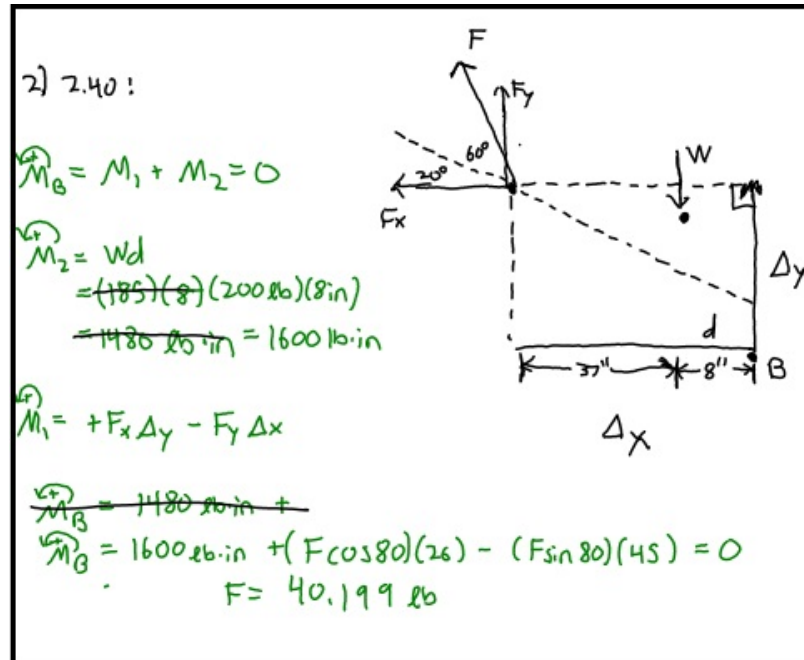


Figure 1.6: Equation strokes are green, and non-equation strokes are black.

1.3.3 Finding and Grouping Mathematical Equations

In the third step, we developed a method to find and group mathematical equations in students' handwritten assignments. To this end, for each stroke, we computed a set of single and pairwise stroke features. Single stroke features are based on the properties of individual strokes and consider the stroke in isolation. Single stroke features are comprised of size, drawing dynamics, location, shape, etc. of the stroke. Pairwise stroke features are based on the properties of consecutive strokes. These features consist of distances, temporal, overlap, perceptual relationships, and ratios of properties of adjacent strokes. The single and pairwise stroke features provide a rich set of observations that can be used for finding equations. To find equations, we utilize a GRU-CRF network to process strokes of a page

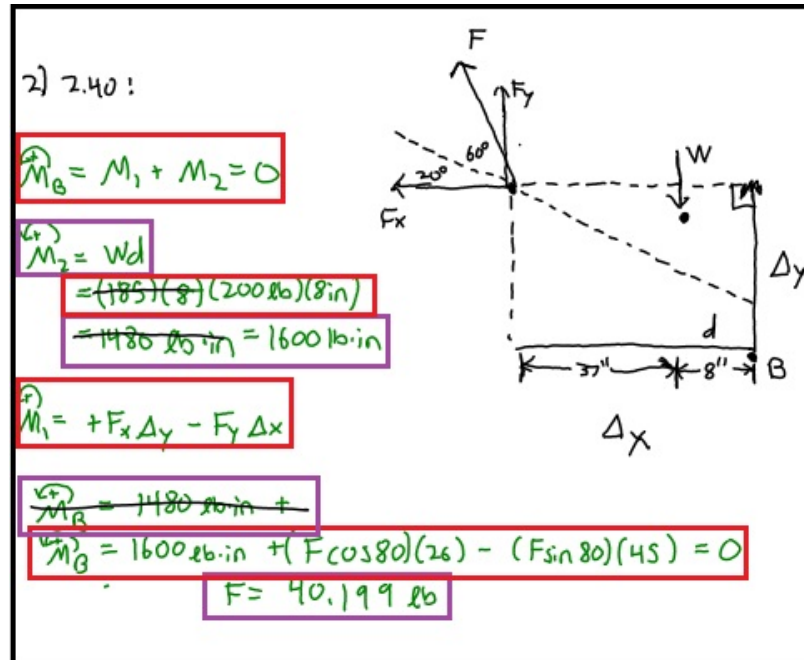


Figure 1.7: Each equation group is enclosed with a red or purple box.

in sequential order. The network receives single and pairwise features of the strokes on the page and produces the equation/non-equation labels in its output for each of the strokes. Figure 1.6 shows the result of the model on finding the equations strokes.

To find individual equations, we extended the model to produce an extra-label for each of the strokes. The extra-label can be one of the following types:

- **Join:** This class denotes that the stroke (equation stroke) should be grouped with the next stroke.
- **NoJoin:** This class denotes that the stroke (equation stroke) should not be grouped with the next stroke.
- **Unknown:** This class denotes that the stroke is a non-equation type and thus should

not be considered for grouping.

Once the labels are generated, we used a chaining technique to cluster equation strokes. The chaining method is a post-processing technique that clusters stroke pairs that have a stroke in common. Finally, we examined the spatial features of the groups generated in the initial clustering to identify clusters that must be combined to generate the final equation clusters. Figure 1.7 shows an example of finding and grouping the equation strokes.

1.4 Outline

This dissertation is organized as follows: Chapter 2 describes our method for locating final answers to free-response questions. Chapter 3 presents our model for the semantic labeling of students' handwritten assignments. Chapter 4 explains our approach for finding and grouping equations in students' handwriting assignments. Chapter 5 provides conclusions.

Chapter 2

Answer Box Recognition

2.1 Introduction

With the rise of highly-enrolled classes over the past decades, new challenges have emerged in the field of education. One of the main challenges is the evaluation of students' work, which is a tedious and time-consuming task. Many tools and interfaces have been designed to help students complete their assignments and assist instructors in evaluating student work [44, 59, 15]; multiple choice exams which can be graded automatically are a major component of such classes.

Despite the usefulness of multiple-choice assessments, only a limited type of knowledge can be examined using them. There are many exams and homework assignments, especially in science and engineering courses, that are best assessed by free-response questions. The most natural and efficient way for students to answer such questions is to write by hand on a piece of paper. However, grading such handwritten answers has always been a tedious and time-consuming task for teachers, especially in large classes. Auto-grading portions



Figure 2.1: This answer box, which is typical of those in our dataset, contains the value “24.84,” unit “*Slugs*,” and problem ID “*a*,” separated by two vertical lines.

of the handwritten answers can reduce grading time for teachers, and provide benefits to students as they can get quicker feedback on their work. Locating final answers to questions is an important step toward auto-grading handwritten assessments. Free response answers may involve equations, free body diagrams, or text, which makes the process of locating final answers to questions a challenging problem.

In our study, students used Livescribe smartpens to do their homework and answer the quiz and exam questions in an undergraduate statics course. We asked students to draw a box around the answer and separate different parts of the answer with vertical lines, which we call *splitters* in this work. Figure 2.1 shows an example of an answer written in an answer box. Splitters are straight lines consisting of one or more strokes that are drawn from the top edge to the bottom edge of an answer box. A final answer to a statics problem consists of three parts: the value, unit (such as newton, meter, etc.) and problem ID. Therefore, each answer box is accompanied by two splitters to help isolate the strokes of each part of the answer. Once our method locates the answer box and identifies groups of strokes for the value, unit, and problem ID, a variety of handwritten recognition methods [36, 40] can be used to recognize and interpret the answers.

Although recognition techniques for identifying isolated shapes exist, our problem

is more difficult because we have to first locate the answer box before we can recognize it. Once we locate it, we could use general-purpose shape recognition techniques to verify its identity, but we developed a new technique that is robust to the wide range of variation in students' real-world drawings.

There are many difficulties in locating and recognizing answer boxes: **1) Irregular quadrilateral shape:** Although most answer boxes are sketched like rectangular shapes, in our dataset, we have observed that some handwritten answer boxes have irregular quadrilateral shapes (Figure 2.2a). **2) Disjoint edges:** Answer boxes are often drawn with disconnected edges (Figure 2.2b). Thus, recognition techniques cannot assume that boxes form closed polygons. This leads to complications as sometimes it is difficult to distinguish the intended end of a box from the contents, particularly when the boxes contain characters such as “L” and “1”. **3) Classifying answer box/answer strokes:** It is challenging to determine a distinction between answer box strokes and answer strokes when they are drawn close to each other. Additional difficulties occur when strokes intersect or are overwritten (Figure 2.2c). **4) Presence of other boxes:** A typical handwritten statics solution may contain boxes and other quadrilateral shapes, which may be identified as answer box candidates (Figure 2.2d); therefore, eliminating boxes that are not true answer boxes is a crucial task.

2.2 Answer Localization: System Overview

An overview of the system is shown in Figure 2.3. Our method searches for answer boxes around the strokes that have large horizontal segments. Each such stroke is a good

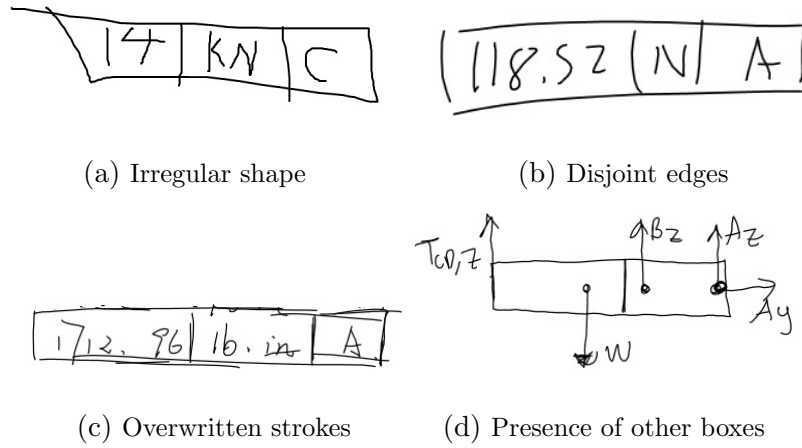


Figure 2.2: Challenges in answer box recognition: *a)* An example of an answer box having an irregular quadrilateral shape. Edges of the answer box at the top-left and bottom-right do not intersect at right angles. Also, the opposite sides are not equal in length. *b)* the left edge of the answer box is disconnected from the bottom and top edges, and the character “1” next to the left edge might be misrecognized as the left edge. *c)* The top and bottom edges of the answer box are overwritten with multiple strokes. *d)* Not all rectangular shapes are answer boxes.

candidate for the bottom or the top edge of an answer box. Therefore, we first segment pen strokes into line segments and identify any strokes with large horizontal segments. Then, we perform a grouping process in which strokes with large horizontal segments are grouped with other nearby strokes to form a long horizontal line. For each group, we identify possible left and right edges and splitters. Groups constructed in this way include both real answer boxes and non-answer boxes. Lastly, we perform the recognition step that identifies true answer boxes and rejects the rest.

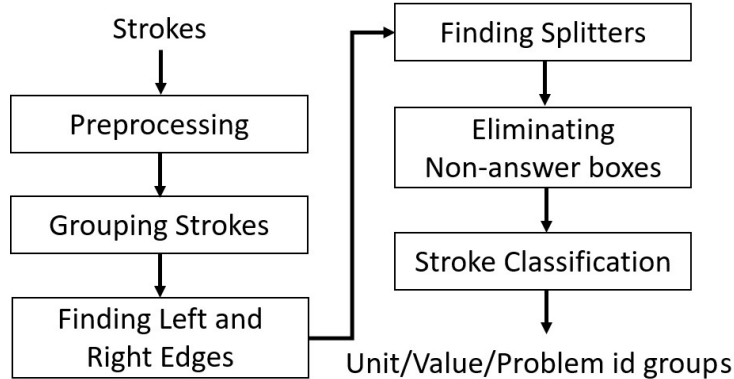


Figure 2.3: Overview of the processing steps for locating answers and classifying strokes into unit, value, and problem ID groups.

2.2.1 Stroke Segmentation

The first step in our stroke segmentation stage is to segment pen strokes into horizontal and vertical segments. We segment strokes using the corner finding technique from [27]. Once corners are found, we break strokes into multiple segments. A segment is a piece of a stroke that starts and ends with corners found by [27]. To identify horizontal and vertical segments, we compute two properties for each segment: straightness and angle to the x-axis. The straightness of a segment is computed by:

$$Straightness = \frac{Path(p_s, p_e)}{\sqrt{(x_s - x_e)^2 + (y_s - y_e)^2}} \quad (2.1)$$

where p_s and p_e are start points and endpoints of the segment, respectively. (x_s, y_s) and (x_e, y_e) are coordinates of p_s and p_e . $Path(p_s, p_e)$ is the summation of euclidean distance between each pair of neighboring points from p_s to p_e . We consider a segment to be *horizontal* if it is straight (has straightness greater than 0.9), and the absolute angle between the x-axis and a line connecting the endpoints of the segment is less than 20° . Similarly,

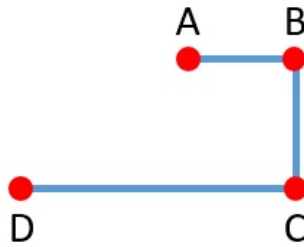


Figure 2.4: An example of stroke segmentation.

we consider a segment to be *vertical* if it is straight (has straightness greater than 0.8), and the absolute angle between the segment and the y-axis is less than 20° . The straightness and angle thresholds are chosen empirically.

An example of stroke segmentation is shown in Figure 2.4. First, a set of corners is found by the speed segmentation method, as shown with red points. Second, we examine each of the segments (i.e., AB, BC, and CD) to see which segments fulfill the requirements for a horizontal or vertical segment. In this thesis, we use “segment” to refer to a piece of stroke that is made by breaking a stroke into smaller pieces, whereas, “stroke” is used for the sequence of points that are collected from a pen down to a pen up and consists of one or more segments.

Once the initial set of horizontal and vertical segments in strokes is determined, we search for long horizontal lines. Each long horizontal line is a candidate to be part of the bottom or top edge of an answer box. For this purpose, we check horizontal segments in nearby strokes to see if they form a long horizontal line. This search process is described in Section 2.2.2.

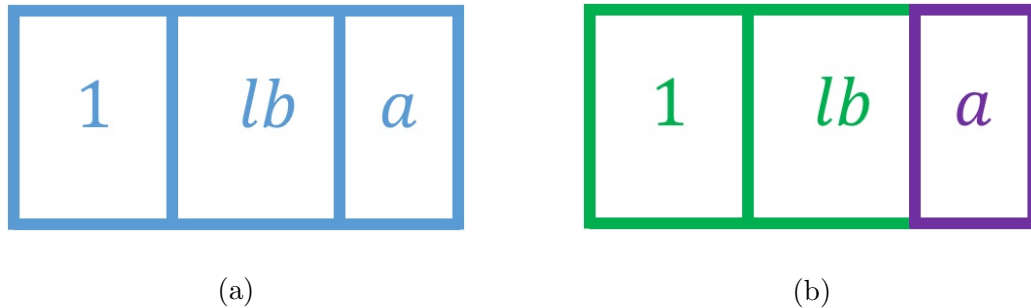


Figure 2.5: Examples of candidate groups generated in the stroke grouping process. Each group is shown with a unique color.

2.2.2 Stroke Grouping

Grouping strokes is a two-phase process. In the first phase, we group sets of strokes that form long horizontal lines. Each such group is a candidate for containing strokes of the top and/or bottom edges of an answer box. In the second phase, for each of the existing groups, we find the best matches from nearby strokes to create new groups. Each of the groups in this step is a candidate to contain strokes of an answer box.

We say a group is an ideal answer box group if it contains only the strokes of an answer box (including the box itself and answer strokes). However, by the end of the grouping process, some of the groups may contain only a partial set of strokes for an answer box. Besides, some of the groups generated in the grouping process may contain non-answer box strokes. The goal of the grouping process is to generate groups that are close to the ideal answer box groups (i.e., contain most of the strokes that belong to an answer box).

In Figure 2.5, three examples of candidate groups are shown. Each of the groups is shown with a unique color. Figure 2.5a shows an ideal group that contains all strokes of

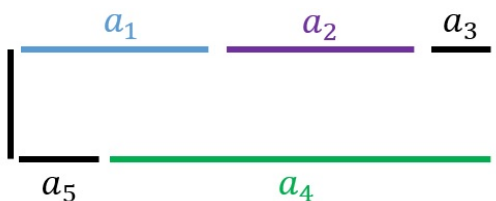


Figure 2.6: The bottom and top edges of the answer box are drawn with multiple horizontal segments. Here, the strokes inside the answer box are excluded for clarity.

an answer box. Figure 2.5b displays two groups where each has only partial strokes of an answer box.

Phase one

Let $H = \{h_1, \dots, h_m\}$ denote a set of strokes that each has a large horizontal segment (horizontal segment's width is greater than $2.5cm$). We begin by placing each h_i in its own isolated group denoted by g_i . Each group potentially contains the top and/or bottom edges of an answer box. The top/bottom edge of an answer box may be drawn with multiple strokes. In such cases, the top/bottom edge of the answer box consists of multiple horizontal segments that are placed in multiple groups.

For example, in Figure 2.6, the top edge of the answer box is drawn with strokes a_1 , a_2 , and a_3 . Similarly, the bottom edge of the answer box is drawn with strokes a_5 and a_4 . Initially, each of the horizontal strokes that is greater than $2.5cm$ in length is placed in an isolated group, which is shown with a unique color. a_5 should be added to the group of a_4 so that there is one single group that contains all the strokes of the bottom edge of the answer box. Similarly, to have one group that contains the top edge of the answer

box, we need to merge the groups of a_1 and a_2 and add a_3 to the resulting group. The horizontal segments in $\{a_1, a_2, a_3\}$, and $\{a_4, a_5\}$ are the horizontal lines that we are looking for in phase one of the stroke grouping. The horizontal segments in each of the groups are intended to form one long horizontal line.

We iterate through the groups to find nearby strokes that need to be added to them. For each group, we create a set of candidate strokes that potentially can be added to the group. Each candidate should contain at least one horizontal segment that has a minimum point-to-point distance less than $1cm$ to one of the horizontal segments in the group. We create pairs of horizontal segments where the minimum point-to-point distances between segments are less than $1cm$. In these pairs, one horizontal segment is picked from the group's horizontal segments, and one horizontal segment is chosen from the candidate stroke. We use a pairwise classifier (we discuss the pairwise classifier later in this section) to decide whether two horizontal segments in a pair should be joined together. If there is at least one pair where the pairwise classifier decides to join them, the candidate stroke is added to the group. If the candidate stroke is part of another group, then both groups are merged into one single group. We keep iterating through the groups and examine nearby strokes for each of the groups until no more changes can be made to the groups.

An adjacency grid is used to determine the neighborhood of a stroke efficiently. To construct the grid, the $8.5'' \times 11''$ page is divided into a 108×140 grid. Each grid cell has a square shape with a length of $200mm$. Each cell stores pointers to the strokes whose bounding boxes intersect the grid cell. The neighborhood of a stroke is defined by the cells neighboring the cells containing the stroke. For example, if the stroke is in cell $C_{i,j}$ and we

Table 2.1: Pairwise features for grouping strokes in phase one.

| Category | Feature name | Description |
|------------|-----------------|---|
| Distance | d_{min} | Minimum inter-segment point-to-point distance |
| Distance | de_{min} | Minimum inter-segment endpoint-to-endpoint distance |
| Perceptual | $f_{intersect}$ | A boolean feature which is set to 1 if the two segments intersect or zero otherwise |
| Temporal | f_t | Number of strokes sketched between the two segments in the drawing sequence |
| Overlap | f_{IoU} | The intersection of the projections of points onto the x-axis over the union of the projections |

are looking for neighboring strokes that are at most $1cm$ distant, the neighbors are found by pointers located at cells $C_{k,l}$ where k and l are in the ranges $[i - 5, i + 5]$ and $[j - 5, j + 5]$.

We use a pairwise classifier to determine if two horizontal segments are part of a long horizontal line. The classifier considers 5 properties of horizontal segments, and are shown in Table 2.1. If the classifier decides two horizontal segments are *joinable*, we place their strokes into one single group. The classifier is trained by providing examples of segments that are a continuation of one another, i.e., “Joins,” and segments that are not continuations of the another, i.e., “NoJoins.”

The pairwise features in Table 2.1 comprise spatial, temporal, and perceptual measures. d_{min} is the minimum point-to-point distance between the segments. de_{min} is the minimum endpoint-to-endpoint distance between the segments. The $f_{intersect}$ feature is a boolean value indicating whether two segments intersect, and f_t indicates the time gap between the strokes that contain s_i and s_j . The f_{IoU} feature is the intersection of the projections of the segment points onto the x -axis over the union of the projections. Figure 2.7 illustrates the d_{min} , de_{min} , and f_{IoU} features. In this figure, x_i denotes the intersection

of the projection of the points onto the x-axis, and x_u represents the union of the projection of the points onto the x-axis, therefore, $f_{IoU} = \frac{x_i}{x_u}$.

An example of this process is shown in Figure 2.8. Two answer boxes are sketched very close to each other (splitters are excluded for simplicity). The top and bottom answer boxes are sketched with four and two strokes, respectively. Each stroke is shown with a unique color. Assume h_o , h_g , h_p , and h_y are horizontal segments greater than $2.5cm$ in length, whereas h_b and h_r are horizontal segments greater than $1.5cm$ and less than $2.5cm$ in length. Additionally, assume that the minimum point-to-point distance between h_p and each of horizontal segments h_b , h_r , and h_g is less than $1cm$, but all other long horizontal segments are farther than $1cm$ from another horizontal segment.

We start by placing each of the strokes of h_o , h_g , h_p , and h_y in their own groups. Let g_o , g_g , g_p , and g_y denote the groups that contain the strokes of h_o , h_g , h_p , and h_y , respectively. In the first iteration, we use the pairwise classifier to check if any of the following pairs should be joined: (h_g, h_p) and (h_g, h_r) . Each of the (h_g, h_p) and (h_g, h_r) pairs has at least one large horizontal segment, and the minimum point-to-point distance between the horizontal segments in each pair is less than $1cm$, and there is no other pair

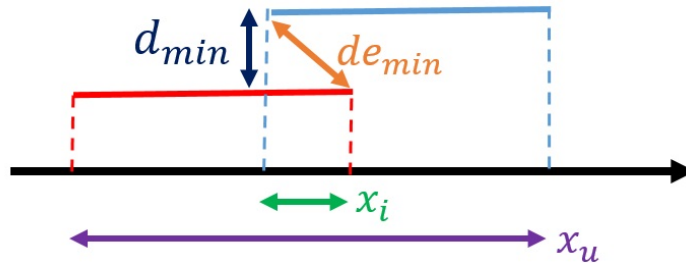


Figure 2.7: Features used by the pairwise classifier.

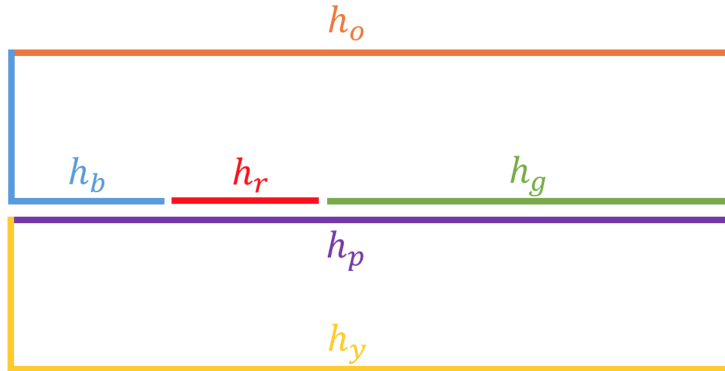


Figure 2.8: Stroke grouping in phase one.

having these two properties. Assume the classifier decides to join (h_g, h_r) and not (h_g, h_p) . Therefore, we add h_r to group g_g . In the second iteration, we check for the following pairs: (h_r, h_b) and (h_r, h_p) . Let us assume the classifier decides to join only (h_r, h_b) . Thus, after the first phase of grouping, there are four groups: $\{h_o\}$, $\{h_b, h_r, h_g\}$, $\{h_p\}$, and $\{h_y\}$.

Generating the Training Set for the Pairwise Classifier

In this section, we describe our approach for generating a training set for training the pairwise classifier used for grouping segments for answer boxes. We started with a set of known answer boxes and then used the corner finder to segment the pen strokes of each box. Our system then used a set of rules to assign the segments to the various parts of the boxes. For example, if a segment was near the top edge of the bounding box of the answer box, the segment was labeled as being part of the top of the box. Once each segment was labeled in this way, nearby horizontal segments were labeled as “Join”, and others were labeled as “NoJoin”.

To understand our method, consider Figure 2.9 that shows an example of a known

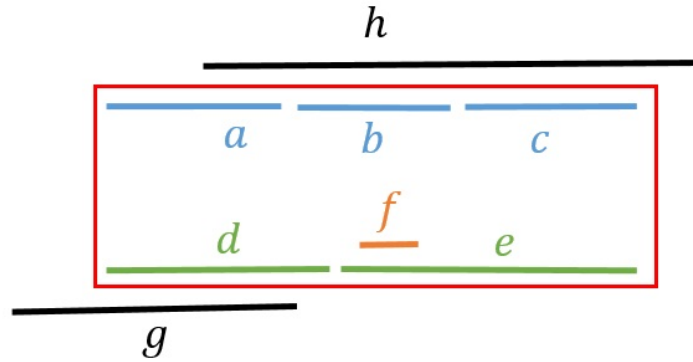


Figure 2.9: Illustration of “Join” and “NoJoin” instances.

answer box. To simplify the Figure, we display only the horizontal segments that are longer than $1.5cm$. Assume that the red box is the bounding box of an answer box. The blue and green strokes comprise horizontal segments that belong to the top and bottom of the answer box, respectively. The orange stroke is a horizontal segment inside the answer box that does not lie on the top or bottom edge of the answer box. The black strokes are horizontal segments that are outside of the box but close to the top and bottom of the answer box.

We say a horizontal segment is placed on the top edge of the answer box if there is at least one point on the segment that has no other segment between it and the top edge of the bounding box. The a , b , and c are horizontal segments that form the top edge of the answer box. Similarly, d and e form the bottom of the answer box.

We create instances of the “Join” class by pairing up the horizontal segments that are placed on the top (i.e., a , b , and c) or bottom (i.e., d and e) edges of the answer box and have a minimum point-to-point distance less than $1cm$. In this example, (a, b) , (b, c) , and (d, e) are instances of the “Join” class. To create instances of the “NoJoin” class, we pair up

horizontal segments that should not be joined together and have a minimum point-to-point distance less than $1cm$. (h, a) , (h, b) , (h, c) , (f, d) , (f, e) , (g, d) , and (g, e) are examples of the “NoJoin” instances.

We selected AdaBoostM1 with J48 decision trees as our classifier and used WEKA to train our model. We started with a seed of 1, no re-sampling, and a weight threshold of 100 and trained the model for 10 iterations. The minimum leaf and confidence values were set to 2 and 0.25, respectively.

Phase two

In this phase, the goal is to generate answer box candidates. First, for each group g_i , we compute the smallest bounding box BB_{g_i} enclosing the group’s strokes. Then, for each pair of (g_i, g_j) , we create a new group containing all of their strokes if and only if it fulfills the following three conditions:

- $(y_i - y_j) < P_y$ where y_i and y_j are the y-coordinates of the centers of the bounding boxes. Parameter $P_y = 5cm$ is designed to make sure groups are vertically close to each other.
- $f_{IoU} > 0.7$. This feature is the same as the one used in phase one.
- The bounding box of the new group does not intersect with the strokes of other groups.

Figure 2.10 shows an example of processing in phase two. In this figure, the top and bottom edges of the answer boxes are sketched with six separate strokes. In phase one, each of the strokes is placed into its own group. Let g_{pk} , g_{pu} , g_c , g_r , g_g , and g_b be the groups

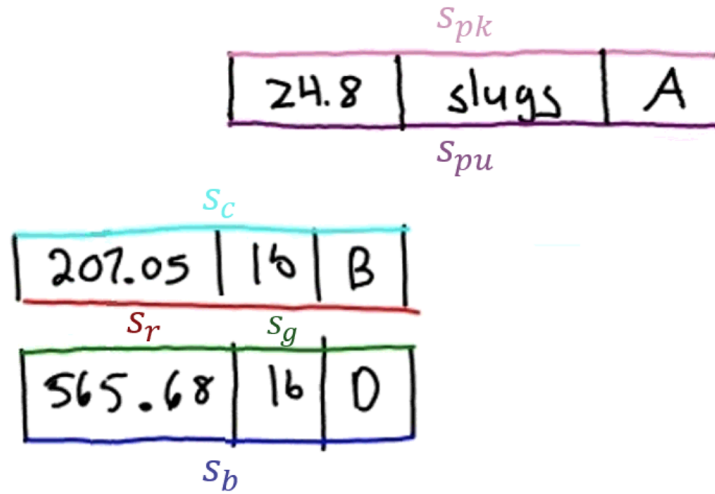


Figure 2.10: Phase two of generating answer box candidates. Here the top and bottom edge of each answer boxes are drawn with a single stroke shown with a unique color.

that contain strokes S_{pk} , S_{pu} , S_c , S_r , S_g , and S_b , respectively. Assume that the vertical distance between g_{pk} and g_b is greater than P_y , but all other pairs of groups are vertically closer than P_y . Also, assume that the value of f_{IoU} for any pair of groups that contains two different groups from $\{g_c, g_r, g_g, g_b\}$ is greater than 70%. Likewise, assume that the value of f_{IoU} for the pair (g_{pk}, g_{pu}) is also greater than the 70%. Conversely, assume that for any pair that contains one group from $\{g_{pk}, g_{pu}\}$ and one from $\{g_c, g_r, g_g, g_b\}$, the value of f_{IoU} that is less than the 70%.

For these conditions, a total of four pairs of groups is generated. g_{pk} and g_{pu} are combined with each other. However, they are not grouped with any of the other groups because the resulting value of f_{IoU} would not meet the threshold of 70%. The other three merged groups that are formed are: (g_c, g_r) , (g_r, g_g) , and (g_g, g_b) . The pair (g_c, g_g) is rejected because its bounding box intersects the bounding box of g_r . Likewise, the pair (g_r, g_b) is rejected because its bounding box intersects the bounding box of g_g . Ultimately

in a subsequent phase of processing, the merged group (g_r, g_g) is rejected as an invalid answer box, and only the other three merged groups are identified as true answer boxes.

2.2.3 Finding Left and Right Edges

Several challenges make the process of finding left and right edges difficult: 1) Left and right edges may not be connected to the bottom and/or top edges of an answer box. 2) We have observed that some students tend to draw these edges with two or more connected or disconnected strokes. 3) Some characters such as ‘1’ or ‘L’ that are sketched very close to the left or right extremes of answer boxes might be misclassified as part of the vertical edges.

We begin with finding candidate strokes for the left and right edges of each group. Any stroke or vertical segment whose centroid falls within a certain square-shaped window at the left and right of the group’s bounding box would be considered as a candidate. The window’s side length is set to be the same as the height of the group’s bounding box. To find the strokes for the left and right edge of a group g_i , we move the center of the window to $(x_{min}, \frac{y_{min}+y_{max}}{2})$ and $(x_{max}, \frac{y_{min}+y_{max}}{2})$, respectively. Here, x_{min} , x_{max} , y_{min} , and y_{max} are the minimum and maximum of x and y in g_i .

Once a candidate set is determined, we search for a subset that best represents the left and right edges of the answer box. Ideally, for each edge, we can find a line that starts at the endpoint of the top edge and ends at the same side endpoint of the bottom edge. We utilize a Hausdorff [30] measure to compute the distance of candidates to an ideal edge. First, we resample each candidate’s points to get 100 equidistantly spaced points. Also, we

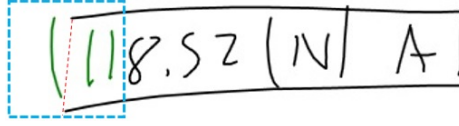


Figure 2.11: The blue box represents the search area for candidate strokes for a left edge. Those strokes that fall inside the blue box are colored green and are considered as candidate strokes for the left edge of the answer box. The dashed red line is an imaginary ideal left edge.

resample ideal edges to obtain 100 points for each edge. Second, for each data point in the resampled candidate set, we compute the minimum distance to the ideal edge. Similarly, for each point on the ideal edge, we compute the minimum distance to the resampled candidate points. We add up all the distances and denote it by $distH(c, IE)$, where c is a candidate subset, and IE is an ideal edge.

Let $C_L = \{c_{L_1} \dots c_{L_m}\}$ and $C_R = \{c_{R_1} \dots c_{R_n}\}$ indicate lists of candidates for left and right edges of a group, respectively. To find the best subsets, we need to minimize the following expression:

$$\arg \min_{c_L \subset C_L, c_R \subset C_R} distH(c_L, IE_L) + distH(c_R, IE_R) \quad (2.2)$$

Where c_L and c_R are the candidate subset for the left and right edges, respectively. IE_L is an ideal edge for left, and IE_R is an ideal edge for the right edge of a group.

2.2.4 Finding Splitters

Identifying splitters enables distinguishing the strokes of the unit, value, and problem ID. Splitters are straight lines that are drawn from the top edge of an answer box to

the bottom edge. Each splitter might be drawn with one or more strokes, and a stroke containing a splitter might be used in drawing the bottom or top edges of an answer box.

For each group g_i (i.e., each candidate answer box), we find all strokes whose centroids fall inside the bounding box of the group. Then, we create a candidate set of splitters that contains all the vertical segments of these strokes and excludes those that are identified as left or right edges. We create splitter groups by placing each vertical segment in a single group. Then, if two vertical segments intersect at their endpoints, their groups are merged.

We compute $\frac{h}{d_{xc}}$ for each of the splitter groups in the candidate set. Here, h is the height of the vertical group, and d_{xc} is the distance between the horizontal lines in the answer box group at the x -coordinate of the centroid of the splitter group. The value of the feature is in the range of 0.0 – 1.0.

Once the feature is computed for each group, we pick the two groups that have the largest value as the left and right splitters.

2.2.5 Unit/Value/Problem ID Stroke Grouping

The splitters are used to identify the strokes that comprise the unit, value, and problem ID of the answer. The splitters divide a candidate answer into three cells, one for the value, one for the unit, and one for the problem ID. Each stroke that lies inside the answer box (except for the splitters and ends) is associated with a cell based on the average x -coordinate of the stroke. Strokes that fall between the left edge and the left splitter are the value. Strokes that fall between the left and right splitters are the unit. Strokes between the right splitter and right end are the ID.

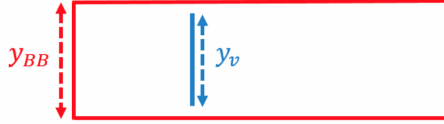


Figure 2.12: y_v is the height of a vertical edge, and y_{BB} is the height of the bounding box of a group.

2.2.6 Answer Box Recognition

A typical statics solution is usually filled with free-body diagrams that are sketched to illustrate the relationship between objects and the forces of interest. It is highly probable that some of the stroke groups created above are non-answer boxes but rather are part of a diagram. In response, we use a series of tests for distinguishing the true answer boxes from the candidate groups.

Basic Test

For each group, the first test is to check if there is at least one stroke for each of the value, unit, and problem ID groups. If not, then the group would be removed.

Vertical Edges Test

An answer box in our dataset should contain at least four vertical edges: two for the left and right edges and two for the splitters. For each of the vertical edges, we compute $\frac{y_v}{y_{BB}}$ (y_v and y_{BB} are shown in Figure 2.12), where y_v and y_{BB} are the height of the vertical edge and the bounding box of the answer box strokes, respectively. If the ratio of the fraction is less than 50% for any of the vertical edges, the group is removed.

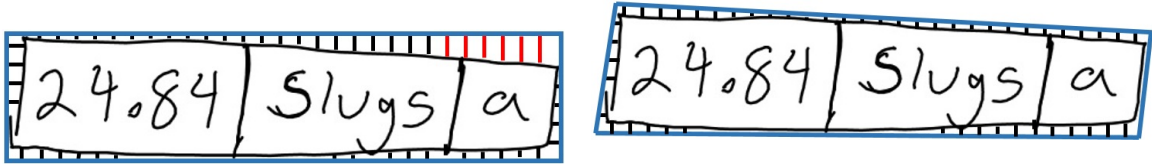


Figure 2.13: (*left*), distances from the bounding box to the answer box strokes are illustrated. Those distances above the Thr_H threshold are colored red. (*right*), distances have been reduced after transforming the bounding box to a quadrilateral that better fits the boundary of the answer box.

Hausdorff Measure

We utilize Hausdorff distance [30] to quantify the quality of sketched answer boxes. For each point on the answer box, we compute the minimum distance to the points on the bounding box. Similarly, for each point on the bounding box, we calculate the minimum distance to the points of the sketched answer box. If there are too many points having distances above a certain threshold Thr_H , the group is unlikely to be an answer box.

Figure 2.13 shows some of the distances from bounding box points to the answer box points. These distances get larger as the angle between the top and bottom edges and horizontal framework becomes larger. In response, first, we compute a quadrilateral that better fits the boundary of the answer box. Then, we compute the Hausdorff distances.

Transforming the Bounding Box

In this step, we transform the bounding box of the answer box into a quadrilateral shape, which is closer to the boundaries of the sketched answer box. To this end, we move

the origin of the coordinate system to the center of the answer box. Then, we create four sets; each contains the points in one of the quadrants of the plane. The points in each set are used to estimate one corner of the transformed quadrilateral.

To compute the corner point in each quadrant, we begin with computing the angle associated with each point. We compute the angle for point (x_t, y_t) as:

$$angle = \arctan\left(\frac{y_t - y_{t-1}}{x_t - x_{t-1}}\right) \quad (2.3)$$

where (x_t, y_t) is the coordinate of the point drawn at timestamp t . The angle at the first point of the stroke is set equal to the angle of the second point.

Once the angles are computed, we divide the points in each set into two new sets: P_V and P_H . Those points that have angles in the range $[-15^\circ, 15^\circ]$ or $[165^\circ, 195^\circ]$ are placed in P_H . Similarly, P_V contains points having angles in the range $[75^\circ, 105^\circ]$ or $[255^\circ, 295^\circ]$. Finally, the x and y coordinates of a corner are computed as:

$$x = \frac{\sum_{i=1}^m x_{vi}}{m}, y = \frac{\sum_{i=1}^n y_{hi}}{n} \quad (2.4)$$

where x_{vi} is the x -coordinate of i th point in P_V , and m is the size of P_V . Similarly, y_{hi} is the y -coordinate of i th point in P_H and n is the size of it.

Once all four corners are computed, we draw lines between each pair of the adjacent corners to form the quadrilateral. Figure 2.13 (right) shows an example of a quadrilateral produced for the answer box. Then, we consider 200 equidistantly spaced points on the edges of the quadrilateral, and for each point, we calculate the minimum distance to the points on the answer box and calculate the percentage of points having distances greater than the Thr_H (in Figure 2.13, red color shows the distances that are greater than the Thr_H

threshold) and denote it by $Perc_{BB2P}$. Similarly, for each point on the answer box, we compute the minimum distance to the quadrilateral points. Then, we count the percentage of points having minimum distances greater than Thr_H and denote this by $Perc_{P2BB}$. In our implementation, Thr_H is set to be $\frac{Y_{BB}}{4}$, where Y_{BB} is the height of the bounding box of the group.

The Hausdorff measure [30] is a powerful method to recognize boxes. By designing two thresholds for $Perc_{BB2P}$ and $Perc_{P2BB}$ (i.e., $Perc_{BB2P} < 15\%$ and $Perc_{P2BB} < 15\%$) we can classify groups into box/non-box groups. However, we found that the solutions often contain boxes that are not answer boxes, such as those contained in free body diagrams. Hence, we developed a classifier to recognize answer boxes from the candidates. The classifier uses several features, including:

- **Group properties:** Width and height of the group.
- **Inner strokes:** Number of inner strokes, i.e., strokes that lie completely inside the group's bounding box.
- **Crossing strokes:** Ratio of the crossing strokes to the inner strokes. Crossing strokes are those that have at least one point inside the bounding box of the group, and at least one point outside of the group's bounding box.
- **Diagonal length:** Ratio of the diagonal lengths of transformed bounding box. The ratio is computed as the length of the longest diagonal divided by the length of the shortest diagonal.
- **Height of strokes:** Ratios of the heights of the bounding boxes of strokes in the

unit, value, and part ID to the height of the group’s bounding box.

We train a linear SVM [51] model, which uses these features to classify groups into answer box and non-answer box groups. We use a grid search to learn the value for the penalty parameter C . In a five-fold cross-validation loop on training data, we compute the accuracy of the SVM classifier for penalty parameter values that were powers of 2 ranging from 2^{-10} to 2^{+10} . The classifier achieved the highest accuracy by using $C = 2^6$. The training data includes examples of both classes.

2.3 Dataset

We used LiveScribe digital pens and dot-patterned paper to collect the data. Participants were 102 undergraduate students enrolled in a statics course offered at the mechanical engineering department at UCR in the winter of 2017. Students were given digital pens and dot-patterned notebooks to complete their homework assignments, quizzes, and exams. A camera at the tip of the pen uses the dots to digitize the writing as timestamped coordinates. This data contains a mixture of solutions to homework and exam assignments. These free-response answers involve free body diagrams, calculation steps, formulas, and text. Each final answer to a statics question includes a tuple of the unit, value, and problem ID. A unit is a physical quantity (such as a meter or newton). Problem ID is a letter from “a” to “e”.

We asked students to draw boxes around their final answers, as shown in Figure 2.1. We manually labeled a total of 4,473 answer boxes that were written on 2,022 pages. These 2,022 pages were selected from 4,219 pages of students’ homework, which were sorted

by their earliest timestamps. We excluded any pages that contained no answer boxes, or the answer boxes were not drawn as expected. In other words, if an answer box was missing any elements of the answer box (top/bottom/left/right edge or splitters) as described in 2.1, we removed the page from the collected pages. We label each answer box by grouping the strokes of the box itself (strokes of the top, bottom, left, and right edges and splitters) and answer strokes (strokes of the unit, value, and problem ID). We split the pages containing the answer boxes into the training and testing sets so that the training and testing set contains 1,060 and 962 pages, respectively. In total, there are 2,237 and 2,236 answer boxes in the training and testing set, respectively.

2.4 Results

2.4.1 Performance of Pairwise Classifier

Table 2.2: Performance of classifiers in phase one of grouping horizontal segments.

| Classifier | Accuracy | Precision | Recall |
|------------------------|----------|-----------|--------|
| AdaBoost-J48 | 99.59% | 93.89% | 94.99% |
| Random Forest | 99.51% | 90.28% | 93.39% |
| Bayesian Network | 99.16% | 92.95% | 85.88% |
| Multi-layer Perceptron | 99.14% | 90.75% | 83.37% |
| ADT Tree | 98.92% | 86.16% | 84.75% |

In section 2.2.2, we presented our method that creates instances for Join and NoJoin classes. Here, we create Join and NoJoin instances in the training set, and then we evaluate the performance of several binary classifiers in separating Join and NoJoin instances. Table 2.2 shows the performance of some classifiers in grouping horizontal segments. We have used a 10-fold cross-validation technique to measure the accuracy of each classifier. The AdaBoost-J48 has achieved the highest accuracy of 99.59 in the classification task.

2.4.2 Evaluation of Candidate Group Generation

We benchmark our grouping strategy with two general segmentation methods presented in [49] and [19]. To evaluate the performance of each method in grouping answer box strokes, we train and test methods on training and testing set, respectively.

During the grouping process, each method categorizes strokes on a page into multiple groups. Each of these groups is a candidate answer box. Unlike methods in [49] and [19], which group all of the strokes on a page, our grouping method only produces groups near strokes that have long horizontal segments and leaves many of the strokes ungrouped. Therefore, our grouping method is faster than the general segmentation techniques.

By the end of the grouping process, strokes of an answer box may be placed in one single group, over-segmented in more than one group, or may not be grouped. We say strokes of an answer box are successfully grouped only if all of its strokes are placed in one single group with no additional strokes. We found that in most cases where a few strokes are missing/added to an answer box group, the downstream stroke interpretation process is not harmed. Therefore, we use a tolerant metric for evaluating the grouping process: we

define an answer box’s strokes to be successfully grouped if all of its strokes (including box and answer strokes) are placed in one single group with at most 3 stroke misplacements (i.e., missing/additional strokes).

Table 2.3: Performance of grouping methods.

| Method | Total groups | TP | FP | FN | Time (sec) |
|------------------------|--------------|-------|---------|-----|------------|
| Ours | 12,358 | 2,151 | 10,207 | 85 | 3.6 |
| Delaye and Lee [19] | 358,191 | 1,967 | 356,224 | 269 | 95.1 |
| Perteneder et al. [49] | 297,118 | 1,834 | 295,284 | 402 | 93.2 |

As shown in Table 2.3, our grouping strategy produces 12,358 candidate groups on pages in the test set. Whereas, [19] and [49] produce 358,191 and 297,118 groups, respectively. As a result, our grouping strategy is faster than general segmentation methods. The column of “Time” shows the average computation time for grouping strokes on an ink page.

2.4.3 Feature Selection and Importance

We use recursive feature elimination to find an optimal subset of features to use by the SVM classifier. We recursively prune the least important feature until the classifier’s accuracy drops when the next feature is removed. At each iteration of the feature selection process, we evaluate the performance of the classifier by using five-fold cross-validation. The importance of each feature is computed by taking the absolute value of the weight obtained

by the SVM classifier during the training. The list of features selected by this method is as follows (Table 2.4):

- $Perc_{p2BB}$
- $Perc_{BB2p}$
- Width of the group
- Height of the group
- Inner strokes
- Crossing strokes
- Diagonal length
- Height of strokes

2.4.4 Evaluation of Recognition Method

We use intersection over union (IoU) to measure the accuracy of the recognition methods. The IoU metric is computed as:

$$IoU(p_i) = \frac{Overlap(p_i, G_i)}{Union(p_i, G_i)} \quad (2.5)$$

where p_i is the predicted bounding box, and G_i is the ground truth box. *Overlap* and *Union* are functions that compute the overlap and union of the predicted and ground truth boxes. We define an answer box i is recognized only if there is a candidate group p_i where $IoU(p_i) > 90\%$.

Table 2.4: The best features for recognizing answer boxes. Features are ranked according to the absolute values of the weights computed by the SVM classifier.

| Feature | Weight |
|-------------------------------|--------|
| <i>Perc_{p2BB}</i> | 0.9712 |
| <i>Perc_{BB2p}</i> | 0.9207 |
| Width | 0.8989 |
| Diagonal length | 0.8741 |
| <i>Height_{unit}</i> | 0.7634 |
| Crossing strokes | 0.7348 |
| Height | 0.6724 |
| <i>Height_{value}</i> | 0.6695 |
| <i>Height_{pid}</i> | 0.6427 |
| Inner strokes | 0.6115 |

We benchmark our recognition method with N\$-recognizer [5], Quick draw [15], and Paleo sketch [48] methods. N\$ recognizer [5] is a multi-stroke object recognizer and uses geometry and trigonometry to recognize isolated objects. The Quick draw [15] and Paleo sketch [48] methods use geometric constraints to recognize objects.

Table 2.5: Performance of methods for recognition of answer boxes.

| Method | Accuracy | Precision | Recall | Time (sec) |
|--------------------|----------|-----------|--------|------------|
| Ours (linear SVM) | 95.3% | 83.57% | 91.13% | 4 |
| N\$-recognizer [5] | 88.6% | 67.83% | 68.88% | 28 |
| Quick draw [15] | 78.1% | 42.53% | 63.86% | 1 |
| Paleo sketch [48] | 81.5% | 50.80% | 64.99% | 1 |

We train the classifiers by using answer boxes in the training data and non-answer boxes generated on the training pages. Each method classifies each group as an answer box or a non-answer box. The performance of the recognizers is benchmarked in Table 2.5.

2.4.5 Ink Interpretation

Once we have located answer boxes, we utilize Microsoft ink recognizer [2] to recognize the text in the answer boxes and compare this to the labeled data. The handwriting recognizer was 91% correct when given correct answer box content. We found that the majority of the errors were due to the challenges in recognizing units. Some of the units

consist of special characters or can be written in multiple ways, which make the recognition process difficult.

Chapter 3

Semantic Labeling of Handwritten Assignments

3.1 Introduction

Semantic labeling of online documents is a crucial prerequisite in document analysis and understanding tasks such as retrieval, recognition, and beautification. In online handwritten documents, semantic labeling is the task of classifying pen strokes into meaningful classes. A pen stroke is a series of timestamped coordinates beginning when the pen touches the page and ending when the pen leaves the page. Researchers have developed various domain-dependent classification algorithms for specific types of data and specific tasks [62]. Additionally, general methods have also been developed for the classification of unconstrained documents [34, 19, 63]. However, these methods do not produce satisfactory accuracy for some problems. Therefore, the classification of pen strokes in unconstrained

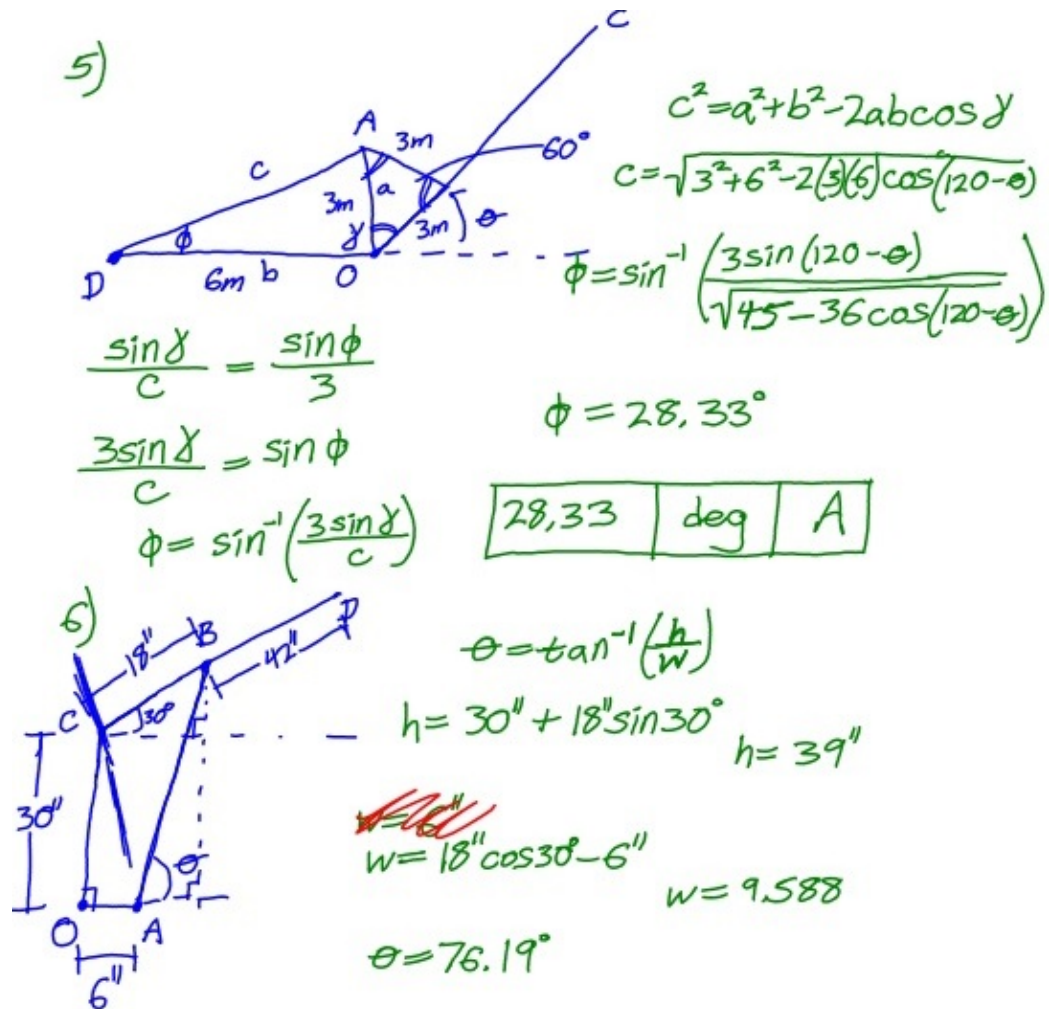


Figure 3.1: A typical answer to a free response question in a statics course contains a mixture of free body diagrams, text (including equations), and cross-outs. Free body diagram strokes are blue, cross-out strokes are red, and text is green.

documents remains a challenging problem that requires more attention from researchers.

In this thesis, we are interested in the task of classifying strokes from students' handwritten homework assignments. More specifically, our dataset comprises handwritten free response solutions to statics problems. Statics is the branch of mechanics that examines

the equilibrium of bodies subjected to forces. The solutions were provided by students enrolled in an undergraduate statics course at UC Riverside. The students used Livescribe smartpens throughout the quarter to take lecture notes and answer questions on exams and homework. The smartpens are used with special dot-patterned paper. A camera at the tip of the pen uses the dots to digitize the writing. Each handwritten page is recorded as a series of strokes that are represented as a sequence of timestamped coordinates. Our semantic labeling task is to classify strokes into the following three classes, which are illustrated in Figure 3.1:

- **Free body diagram (FBD):** Diagrams used to represent the forces acting on a mechanical system. Free body diagrams include drawings, text labels, and arrows.
- **Cross-out:** A cross-out is a set of strokes used to cross out writing. For example, an “X” is a common cross-out mark.
- **Text:** This class comprises all writing that is not free body diagrams or cross-outs. This category primarily comprises equations but also includes explanatory notes, organizational information (e.g., problem number, student name, etc.), and lines and arrows showing relationships between other writing.

In this work, we introduce a novel deep neural network model for classifying strokes into these three classes. This model employs two Convolutional Neural Networks (CNNs), one that extracts features from pen trajectories and another that extracts features from domain-dependent stroke features. The output of each CNN is fed into a Bidirectional Long Short Term Memory (BLSTM) network to encode information characterizing the sequence

of strokes. Finally, the outputs of the two BLSTM networks are concatenated and passed to a Conditional Random Field (CRF) layer, which assigns the final classifications.

3.2 Related Work

Stroke classification is the task of classifying strokes into meaningful classes. Early studies mainly used local information for each stroke and performed isolated stroke classification, which leads to poor performance in many domains. However, recent studies utilize different sources of information to improve classification accuracy. Spatial and temporal information are the two primary sources of information that many methods take advantage of. In this section, we go over the methods and sources of information that were used by researchers for the task of stroke classification.

A method by Jain et al. [34] was one of the first attempts to group strokes of online handwritten documents into homogeneous groups. They computed the length and curvature for each of the strokes, and then they used a linear classifier to classify strokes into text and non-text classes. The method does not exploit contextual information, and as a result, fails to achieve acceptable accuracy in many domains.

Bishop et al. [11] presented a system for distinguishing between text and graphics strokes. They utilize a multilayer perceptron (MLP) classifier to compute a confidence value indicating if a stroke was text or graphics. Then, they used a Hidden Markov Model to incorporate temporal information between neighboring strokes to acquire better accuracy. However, their approach is limited in that it does not consider the spatial distances between strokes.

Another approach by Delaye et al. [20] investigates the use of a conditional random field (CRF) in modeling the spatial and temporal relationships between strokes. They start by classifying strokes using their intrinsic properties and a support vector machine (SVM) classifier. Then they use a CRF to assign labels to the strokes so that each label is most consistent with the strokes' spatial and temporal neighbors.

Van Phan et al. [67] use two recurrent neural networks (RNNs) to label a sequence of strokes. One RNN processes features characterizing the intrinsic properties of the strokes and then produces a class label for each stroke. Another processes pairwise stroke features characterizing consecutive pairs of strokes and then produces a pair of class labels for each stroke. Then, they use an additional classifier to integrate the outputs of the two networks to produce the final labels for the strokes. In our approach, each BLSTM computes an intermediate representation for each stroke, which a CRF layer then uses to produce labels for the strokes jointly. Thus, their RNNs directly compute labels, while our BLSTMs compute new features which are then used to compute labels.

Ye et al. [72] presented a graph attention networks for the stroke classification. They transform the stroke classification problem into the task of classifying nodes in a graph. In the graph, each node represents a stroke in the document, and edges are used to model the temporal and spatial relationships between them. Their method considers an edge between consecutive strokes and between any pair of strokes that are spatially closer than a certain threshold. Their approach requires a large number of hyperparameters to be tuned before it can be used on any domain.

Indermuhle et al. [32] deployed a bidirectional long-short term memory (BLSTM)

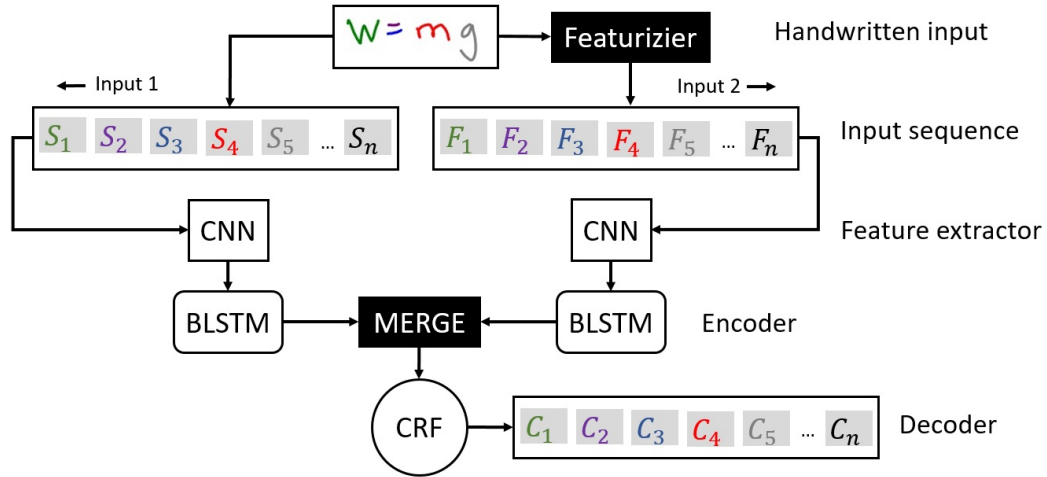


Figure 3.2: The overall architecture of the CNN-BLSTM-CRF model.

neural network to recognize various types of content such as text, diagrams, and tables in online handwritten documents. Their approach extracts a 7-dimensional feature vector for each of the points in the strokes, whereas, we extract features from the strokes. In addition, their network learns to map a sequence of $m \times n$ labels, to n labels where m and n are the number of points and strokes, respectively. However, in our approach, our sequence classifier learns to predict n labels for the n strokes given in the input.

3.3 CNN-BLSTM-CRF Model for Stroke Classification

In this section, we present our hybrid neural network model for classifying pen strokes from online handwritten documents into three classes: free body diagram, cross-out, and text. Figure 3.2 shows an overview of the system. In this network, two sets of input sequences are generated from the pen strokes. The first contains the coordinates of the strokes, and the second consists of features computed from the strokes. Each of the

input sequences is fed to a separate CNN-BLSTM network that extracts new features from the input sequence and encodes the information characterizing the stroke sequence. The outputs of the BLSTM layers are concatenated and become the input to the CRF layer, which decodes this information and generates a probable label for each stroke. A brief description of each layer is provided below.

3.3.1 Input sequences

The input to the network is a page of handwritten pen strokes. Each stroke comprises a sequence of timestamped coordinates. The i^{th} stroke is represented as

$$S_i = \{[x_1^{(i)}, y_1^{(i)}], \dots, [x_m^{(i)}, y_m^{(i)}]\}, \quad (3.1)$$

where $[x_j^{(i)}, y_j^{(i)}]$ are the coordinates of the j^{th} point. Strokes can have an arbitrary number of trajectory points, but in our approach, we represent them as having a fixed number of points (m). The parameter m is set to 250 in our experiments to accommodate most of the strokes in our dataset. If a stroke has fewer than m points, zeros are added to the end. If a stroke has more than m points, it is broken down into smaller strokes; each has equal numbers of points. The first input sequence is constructed by concatenating the trajectory points of all of the strokes on the page:

$$I_1 = \{[x_1^{(1)}, y_1^{(1)}], \dots, [x_m^{(1)}, y_m^{(1)}], \dots, [x_1^{(n)}, y_1^{(n)}], \dots, [x_m^{(n)}, y_m^{(n)}]\} \quad (3.2)$$

The value of n is set to the largest number of strokes observed on a page in our dataset. If the page has fewer than n strokes, strokes containing zero coordinates are added to the end of the sequence.

Table 3.1: Domain dependant stroke features [62].

| | | |
|---------------|-----------|----------------------------------|
| Cross-out | f_{BW} | Bounding box width |
| | f_{BH} | Bounding box height |
| | f_D | Ink density |
| | f_{SR} | Straightness ratio |
| | f_X | Part of a cross? |
| | f_P | Part of a set of parallel lines? |
| | f_{AU} | Area fraction under the stroke |
| | f_{AO} | Area fraction over the stroke |
| | f_{UH} | Average underlying stroke height |
| | f_{TU} | Time to first underlying stroke |
| Miscellaneous | f_{NT} | Normalized time |
| | f_{LS} | Is long stroke? |
| | f_{NL} | Number of nearby long strokes |
| | f_{IN} | Number of intersecting strokes |
| | f_{ID} | Density of intersecting strokes |
| | f_{D2N} | Direction to the next stroke |

We create the second input sequence by computing a 16-dimensional feature vector for each stroke and concatenating them together:

$$I_2 = \{[f_1^{(1)}, \dots, f_{16}^{(1)}], \dots, [f_1^{(n)}, \dots, f_{16}^{(n)}]\} \quad (3.3)$$

Here $[f_1^{(i)}, \dots, f_{16}^{(i)}]$ is the 16-dimensional feature vector corresponding to the i^{th} stroke. As before, if the page has fewer than n strokes, zero features are added to the end of the sequence.

Table 3.1 provides an overview of the 16 features, which are taken from [62]. (See [62] for details.) The first 10 features were designed for identifying cross-out strokes. f_{BW}

and f_{BH} are the height and width of the minimum bounding box containing the stroke, respectively. f_D characterizes the density of the pen stroke as cross-out strokes may be drawn as a dense “blob”. Some cross-out strokes are straight lines; f_{SR} characterizes the straightness of a stroke. Sometimes cross-outs comprise sets of parallel strokes or strokes that form an “X”. f_p and f_X are boolean-valued features indicating whether a stroke is part of a set of parallel lines or a cross, respectively, with nearby strokes. f_{AU} is the ratio of the area of the bounding box of the stroke to the area of the bounding box containing all of the strokes that intersect the stroke and were drawn earlier. Similarly, f_{AO} is the ratio of the area of the bounding box of the stroke to the area of the bounding box containing all of the strokes that intersect the stroke and were drawn later. Finally, f_{TU} is the time difference between the stroke and the earliest stroke that have overlapping bounding boxes.

The next six features are designed to distinguish free body diagram strokes from text strokes. Stahovich and Lin [62] observed that the earliest-drawn strokes are more likely to be part of free body diagrams rather than equations. f_{NT} is a normalized time where the earliest-drawn stroke has a normalized time value of 0, and the last drawn has a value of 1. f_{LS} has a value of 1 if a stroke is three times taller or wider than the average stroke height. f_{NL} denotes the number of nearby long strokes. Here, two strokes are nearby only if the minimum point-to-point distance between them is less than twice the average stroke height. Strokes written in an equation are typically separated from one another, whereas, many pen strokes in a free body diagram intersect each other. This property is captured by f_{IN} , which shows the number of strokes intersecting the stroke. f_{ID} represents the density

of the intersecting strokes and is computed as:

$$f_{ID} = \frac{L^2}{A_{BB}} \quad (3.4)$$

where L is the sum of the arc lengths of the intersecting strokes, and A_{BB} is the area of the bounding box of the stroke. Equations are sometimes drawn from left to right. Thus, f_{D2N} is the direction from one stroke to the subsequent one.

As shown in Figure 3.2, the input sequences I_1 and I_2 are used as the input to the left and right CNN-BLSTM networks, respectively.

3.3.2 CNN (feature extractor)

CNN layers have shown outstanding performance in deriving useful features from images, sequence data, and the like. Each CNN layer performs a linear and a non-linear operation to transform the input data. In this study, CNN layers have one-dimensional kernels that are convolved with the input sequence over a single dimension. Details of the parameter settings are shown in Table 3.2. A CNN network comprises neurons that are arranged in multiple one-dimensional arrays. Each neuron is connected to the neighboring small region of neurons of the previous layer via feed-forward connections. We use a Rectified Linear Unit (ReLU) activation function to apply non-linearity to the neuron values.

3.3.3 BLSTM (encoder)

Recurrent neural networks (RNNs) have shown great promise in processing sequential data. Unlike feed-forward networks that pass information in one direction, RNNs are capable of capturing time dynamics through cyclic connections. LSTM networks are a

popular and successful type of RNN that were introduced by Hochreiter and Schmidhuber [28]. Each LSTM unit is equipped with multiple gates that enable it to control the flow of information. Input, output, and forget gates are at the core of an LSTM cell. More specifically, the input gate (i) monitors the incoming data, the forget gate (f) decides what to be discarded from past memory dynamics, and the output gate (o) determines what piece of information to flow out of the cell.

Formally, we use following equations to update an LSTM unit at time step t :

$$\begin{aligned}
i_t &= \sigma(W_{h_i}h_{t-1} + U_{x_i}x_t + b_i) \\
f_t &= \sigma(W_{h_f}h_{t-1} + U_{x_f}x_t + b_f) \\
o_t &= \sigma(W_{h_o}h_{t-1} + U_{x_o}x_t + b_o) \\
\tilde{c}_t &= \tanh(W_{h_c}h_{t-1} + U_{x_c}x_t + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned} \tag{3.5}$$

where the input vector at time step t is x_t , and the hidden state at t is h_t . $U_{x_i}, U_{x_f}, U_{x_o}$, and U_{x_c} are weight matrices for the gates, and $W_{x_i}, W_{x_f}, W_{x_o}$, and W_{x_c} are weight matrices for the hidden state. b_i, b_f, b_o , and b_c are biases for the hidden state. \tilde{c}_t is an intermediate value that is used to update the cell state c_t . A pointwise multiplication operation is denoted by \odot . σ and \tanh are the sigmoid and hyperbolic tangent functions, respectively.

LSTM units are capable of capturing long-term, past dependencies in their hidden states. Our network computes two sets of contextual information from neighboring strokes. The LSTM network on the right channel of our network (as shown in Figure 3.2) produces contextual information from domain-dependent features extracted from the pen strokes.

The LSTM network on the left channel of our network produces contextual information directly from the pen stroke coordinates.

The temporal information about strokes contains valuable information for the classification task. For each stroke, the contextual information provided by other strokes, which are sketched near in time, has a positive effect in most cases. In contrast, the context information computed for two strokes that have a large temporal gap can have a negative effect. Therefore, we only consider the temporal context between nearby strokes.

For many sequence labeling problems, accessing future contextual information enables the model to encode more complex contextual knowledge from the sequential data. As proposed in [22], we use bi-directional LSTM (BLSTM) units in our model to provide a means of considering future contextual information. BLSTM layers are composed of two separate LSTM networks where one of them processes the input in the forward direction (i.e., from start to end), and the other operates in the backward direction (i.e., from end to start). The hidden states of the two anti-parallel LSTMs are concatenated together to produce the final output.

3.3.4 CRF (decoder)

Many sequence labeling problems require incorporating relations between neighbors in order to predict a label for any individual in a given input sequence. The Conditional Random Fields (CRF) is a class of undirected graphical models that are used to compute the conditional probability of the output nodes given the values in the input nodes. In this way, they are able to jointly predict the best label for each element in the input sequence.

Let $h = \{h_1, \dots, h_n\}$ denote an input sequence to the CRF layer, which is obtained

by concatenating the outputs of the two BLSTMs. More specifically, each h_i comprises the i^{th} hidden state of both the forward and backward LSTMs from both the left and right channels of the model. $y = \{y_1, \dots, y_n\}$ is the output sequence of the CRF where each y_i is the predicted label for stroke S_i .

We use a CRF from [38] to model our sequence labeling task:

$$P(y|h; \theta) = \frac{1}{Z(h)} \exp\{\phi(h, y; \theta)\} \quad (3.6)$$

where $\theta = \{W, b\}$ represents the set of parameters and $Z(h)$ is the normalization factor. $\phi(h, y; \theta)$ is the potential function which is formulated as:

$$\phi(h, y; \theta) = \sum_{i=1}^n \phi_U(h_i, y_i; \theta) + \sum_{i=1}^{n-1} \phi_P(h_i, h_{i+1}, y_i, y_{i+1}; \theta) \quad (3.7)$$

where $\phi_U(h_i, y_i; \theta)$ is the unary potential function that measures the compatibility of the i^{th} stroke and the label y_i . $\phi_P(h_i, h_{i+1}, y_i, y_{i+1}; \theta)$ is the pairwise potential function that captures the dependency between adjacent strokes. These potential functions are formulated as

$$\phi_U(h_i, y_i; \theta) = h_i^T \theta_{y_i}^u \quad (3.8)$$

$$\phi_P(h_i, h_{i+1}, y_i, y_{i+1}; \theta) = h_i^T \theta_{y_i, y_{i+1}}^{p,1} + h_{i+1}^T \theta_{y_i, y_{i+1}}^{p,2}$$

Here, C is the number of classes and $\theta_{y_i}^u$, $\theta_{y_i, y_{i+1}}^{p,1}$, and $\theta_{y_i, y_{i+1}}^{p,2}$ are the parameters of the CRF.

For training purposes, we search for the optimal solution to maximize the conditional likelihood in our training data. We use the logarithm of the likelihood as follows:

$$L(\theta) = \sum_{i=1}^N \log p(y|h; \theta) \quad (3.9)$$

Here, N is the number of instances (pages) in our training data. We employ a maximum likelihood estimation technique to estimate parameters that maximize the log-likelihood.

Finally, decoding is the task of finding the optimal label sequence y^* that achieves the highest conditional probability:

$$y^* = \operatorname{argmax}_{y \in \mathcal{Y}(h)} p(y|h; \theta) \quad (3.10)$$

We find the most probable label sequence by using the Viterbi algorithm [53]. The Viterbi algorithm is a dynamic programming approach for efficiently finding the most likely sequence of labels. Here the maximization occurs over all of the possible labels for each stroke.

3.4 System workflow

We preprocess the pen strokes before feeding them to the neural network. For the input sequence I_1 , we use min-max normalization to linearly transform the x and y coordinate values to map into the range $[0, 1]$. Likewise, we use z-normalization to normalize the feature values in the input sequence I_2 .

As shown in Figure 3.2, I_1 and I_2 are fed to the right and left CNN-BLSTM networks, respectively. We observed that BLSTM networks easily overfit the training data. As a remedy, we apply dropout to the recurrent input signal on the BLSTM networks. During network training, the dropout rate is set to 0.2. The configuration of the CNN-BLSTM networks is shown in Table 3.2. Both of the CNN-BLSTM networks utilize the same hyperparameters. The output vectors of the BLSTM networks are combined together to form the input to the CRF layer. Finally, the CRF layer jointly predicts the best label

Table 3.2: Model configuration

| Layer no. | Type | Specifications |
|-----------|-------------|--|
| 1 | Convolution | Filters = 16, Kernel size = 9, strides = 1 |
| 2 | Convolution | Filters = 32, Kernel size = 9, strides = 1 |
| 3 | BLSTM | Output dimension = 64 |
| 4 | BLSTM | Output dimension = 64 |
| 5 | BLSTM | Output dimension = 64 |

Table 3.3: Frequencies of stroke types in each dataset.

| Dataset | Stroke Type | | |
|---------------|-------------|-----------|-------------|
| | FBD | Cross-out | Misc. Notes |
| Training Data | 31.2% | 1.2% | 67.6% |
| Testing Data | 20.5% | 1.4% | 78.1% |

sequence for the output nodes.

3.4.1 Network Training

We use Keras [16] to implement our neural network model. The entire model contains 216,996 trainable parameters. The training and testing experiments were run on a machine equipped with a 2.66 GHz Xeon(R) CPU. For the configuration described above, the model training requires approximately 49 hours.

3.4.2 Optimization method

We use RMSProp [17] to optimize the parameters of the network. At each step of network training, we feed batches of size 10 to the network. The learning rate is initially set to 0.001, and at each step is updated according to [17]. Our model achieves the highest accuracy on the validation set (we discuss the validation set in Section 3.5) at 278 epochs.

3.5 Results

We evaluate the performance of our model on a database of homework assignments, quizzes, and exams collected from 132 undergraduate students enrolled in a mechanical engineering course on statics. The students produced writing using Livescribe digital pens. These pens are used with special dot-patterned paper. A camera integrated into the tip of the pen uses the dots to digitize the writing as timestamped coordinates.

6,562 pages of handwritten coursework were collected from 12 exam problems, 30 homework problems, and 7 quiz problems. From this, we manually labeled 1,060 pages comprising solutions to 5 exam problems (293 pages) and 8 homework problems (776 pages). The exam pages contained 122,058 pen strokes, and the homework pages contained 298,527 pen strokes. The frequencies of the various stroke types in the labeled exam and homework pages are shown in Table 3.3. We used the homework data for training and the exam data for testing. We held out 155 pages from the training set for validation.

Table 3.4: Performance of our model and three baseline networks.

| Model | Accuracy | Precision | Recall | F1 Score |
|--------------------------------------|---------------|---------------|---------------|---------------|
| <i>CNN – BLSTM₁ – CRF</i> | 88.30% | 88.42% | 88.30% | 88.36% |
| <i>CNN – BLSTM₂ – CRF</i> | 91.34% | 91.43% | 91.22% | 91.32% |
| <i>CNN – BLSTM</i> | 90.76% | 89.51% | 90.46% | 89.98% |
| <i>CNN – BLSTM – CRF</i> | 94.70% | 96.14% | 94.54% | 95.33% |

3.5.1 Results for Various System Configurations

To evaluate the power of each part of our model, we computed the accuracy with various parts of the model removed. More specifically, we considered 3 sub-models: (1) The CNN-BLSTM₁-CRF model uses only the I_1 input sequence (i.e., the pen stroke coordinates) for input and excludes the CNN-BLSTM network on the right of the network (i.e., the domain-dependent features). (2) Conversely, the CNN-BLSTM₂-CRF model uses only the I_2 input sequence and ignores the CNN-BLSTM network on the left side of the network. (3) The CNN-BLSTM model removes the CRF layer and replaces it with a dense layer. All of these networks utilize the same hyper-parameters as displayed in Table 3.2. As shown in Table 3.4, our complete model achieved higher accuracy than any of the sub-models, indicating the importance of all of the elements of our model.

3.5.2 Comparison with Related Methods

We benchmark our model with three existing methods: (1) The method of Stavovich and Lin [62]; (2) the GSC26_BCC26_19Q method [67]; and (3) the CRF_NN [71] method. The first of these methods was specifically designed for free-form statics solutions, and the other two are the top-performing systems for classifying strokes into text/non-text classes on the IAMonDo database [33].

Table 3.5 compares the performance of our method to that of the three benchmark methods on the testing dataset. Our method performed better than the GSC26_BCC26_19Q and CRF_NN methods on all performance measures. Our method also achieved the highest overall accuracy of 94.7%. For the most part, our method performed better than the method from [62] for FBD and text strokes: Our method achieved higher recall rates for both types of strokes and achieved a higher precision rate for FBD strokes. The precision of our method for text strokes was only slightly less (about one percentage point) than that of the method from [62]. Due to the small number of cross-out instances in the training set, most of the systems have low performance in recognizing cross-out strokes. However, [62] uses a special purpose, hand-coded cross-out detection technique that achieves high accuracy for this class.

Table 3.5: Performance of models for the stroke classification task.

| Model | Acc. | Precision | | | Recall | | |
|-----------------|--------------|---------------|---------------|---------------|---------------|---------------|---------------|
| | | FBD | Cross-out | Misc. | FBD | Cross-out | Misc. |
| Our Model | 94.7% | 88.57% | 15% | 96.14% | 87.88% | 0.68% | 97.00% |
| Stahovich [62] | 92.23% | 77.32% | 53.91% | 97.26% | 87.74% | 74.26% | 93.59% |
| CRF_NN [71] | 89.55% | 71.84% | 9.49% | 93.95% | 78.24% | 0.97% | 93.66% |
| GSC26_LSTM [67] | 91.33% | 79.97% | 27.65% | 93.61% | 74.81% | 0.74% | 96.61% |

Chapter 4

Finding and Segmenting

Mathematical Equations in

Students' Online Handwritten

Assignments

4.1 Introduction

The previous chapter presented techniques for classifying individual pen strokes into various classes. Here we consider how we group pen strokes into meaningful objects. More specifically, we describe a method for identifying the pen strokes that comprise individual equations.

Mathematical equations are the most abundant type of content in written solutions

(a) $x^2 + y^2 = r^2$

(b) $\vec{M} = \vec{r} \times \vec{F}$

(c) $I = \int r^2 dm$

Figure 4.1: Examples of mathematical equations.

to free-response problems in a variety of academic subjects such as engineering, physics, and science. Mathematical expressions provide visual explanations of the thinking process in computing the final answers. Our goal is to locate equations so they can be recognized with existing recognition techniques such as [69, 74, 73].

In the simplest case, an equation is two expressions equated by an “=” sign, where an expression is a collection of mathematical symbols that are combined according to a set of formal rules. Figure 4.1 shows examples of equations: (a) is an algebraic expression, (b) is a vector algebra equation, and (c) is an integral calculus equation.

There are many ways of organizing the equations on a page. For example, each of the equations in Figure 4.1 is written on a single line on the page. Likewise, each of these contains exactly two expressions that are set equal to each other. However, it is

$$\begin{array}{l}
 (a) \quad F_y = 100 \cos 60 = 50N \\
 (b) \quad F_x = 100 \sin 60 \\
 (c) \quad = 80N
 \end{array}$$

Figure 4.2: Examples of equation segments in our dataset.

common to combine multiple equations on a single line as, shown in Figure 4.2 (a). Here, the equation $F_y = 100 \cos(60)$ is combined with the equation $100 \cos(60) = 50N$ to produce a single combined equation $F_y = 100 \cos(60) = 50N$. Thus, this equation comprises three expressions. Likewise, it is common to split equations across multiple lines as in 4.2 (b) and (c). This equation contains two expressions on one line (b) and a third expression on a second line (c). For our purposes, we seek to group together all of these expressions for an equation that are on the same line. We call each such group an *equation segment*. Thus, each of the equations in 4.1 comprises its own equation segment. Likewise, each of the three parts in 4.2 comprises its own equation segment.

Grouping strokes of mathematical equations is a challenging task for the following reasons:

- **Complicated geometric structure:** unlike the traditional problem of recognizing handwritten text, where most of the strokes are written close to the baseline, strokes of mathematical equations may spread over more than one baseline to form numerators,

denominators, subscripts, superscripts, etc. Handwritten text is typically organized into lines and written from left to right and top to bottom, but the writing order for mathematical equations can be written in varied and complicated arrangements.

- **Complicated contextual information:** due to the irregular placement of equation strokes on the baselines, there are many cases where strokes from different equations are close to one another. As a result, using only spatial information is insufficient. Other forms of contextual information are needed to identify equation segments accurately.

4.2 Related Work

Clustering strokes into meaningful objects has been the subject of many research studies. Identifying complete objects in unconstrained documents is a difficult task due to variations in handwriting and the lack of apriori knowledge about the structure of the writing. Context-integrated methods use several sources of contextual information (spatial, temporal, and local) to split strokes into multiple groups.

Many existing approaches consider only local pairwise features. Clustering decisions are based primarily on the geometric relationships between pairs of strokes. This kind of local information may not be definitive. For example, inter-cluster strokes may overlap with one another, which could cause the clusters to be merged erroneously. Our approach considers a much broader temporal context by considering information from the entire sequence of strokes.

Stahovich and Lin [62] presented a method to split equation strokes into individ-

ual groups. Their method produces groups such that the strokes of a group belong to a single equation and are written on the same baseline. With this method, if an equation is written on multiple lines, such as an equation that has an expression with numerator and denominator, then the equation is likely to be placed into multiple groups. In our method, we aim to find complete mathematical equations rather than finding pieces of the equations. Moreover, they use local pairwise stroke features to group strokes, whereas, in our approach, we use global information.

Peterson [50] developed a general-purpose technique for clustering pen strokes. His method leverages local pairwise stroke features to decide whether or not to place a pair of strokes into the same group. The pairwise features capture both temporal and spatial aspects of the stroke pairs. Once pairwise features are computed for every pair of strokes on the page, these features are passed to a binary classifier (AdaBoostM1 with J48 decision trees) that computes a “Join” or “NoJoin” label for each pair of strokes. Initial groups of size 2 are created by grouping pairs of strokes labeled “Join”. Some of the initial groups may have a stroke in common; therefore, a “chainer” is deployed to join groups that have at least one stroke in common, producing final clusters.

In Peterson’s method, the decision to join a pair of clusters relies on local contextual information. We observed that in many cases, using only local information can lead to errors in grouping. For example, in our dataset, there are cases where the strokes of one equation are very near to, or even intersect, the strokes from another equation. Local information is often insufficient for distinguishing the equations in such cases. As a remedy, we use global contextual information. More specifically, we use context-integrated sequential

information to produce the initial groups. In our approach, each stroke has access to information coming from both previous and subsequent strokes in the chronological sequence of strokes. Using such information enables more accurate grouping.

4.3 Approach

In this section, we present our method for finding and grouping equation strokes in students' online handwritten assignments. Our approach uses both intrinsic properties of the strokes and contextual information to do this. The intrinsic properties describe the shapes of the strokes, and the contextual information comes in part from pairwise features describing geometric and temporal relationships between strokes. A classifier uses these features to assign two different labels to each stroke. The first label is the type of stroke, indicating whether it is part of an equation or not – the stroke is “equation” or “other”. The second label indicates whether or not the stroke is part of the same group as the subsequent stroke. This label has one of two values, either “Join” indicating that the stroke belongs in the same group with the next stroke in the chronological order, or “NoJoin” indicating it does not.

We use the two labels assigned to each stroke to group the strokes into equation segments via a chaining process. This process groups together consecutive strokes that are labeled as “equation” and “Join”. We then use a second classifier that examines the initial groups to determine if they need to be combined.

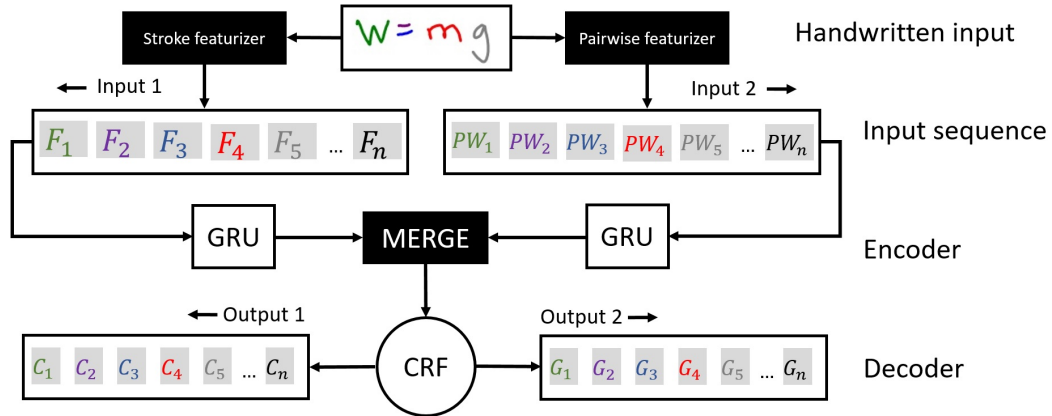


Figure 4.3: The architecture of the GRU-CRF model.

4.4 Generating Initial Groups

4.4.1 GRU-CRF for labeling strokes

In this section, we present our model that assigns the stroke type label and the Join/NoJoin label to each stroke. The model contains two gated recurrent unit (GRU) layers that both feed into a conditional random field (CRF) layer. The input to the first GRU is a set of features computed from the stroke itself. The input to the other GRU is a set of pairwise features. Both sets of features are taken from Peterson [50]. Peterson used these features with an Ada-Boosted J48 decision tree to group pen strokes into objects. Here, we use them with a GRU network that computes contextual information, resulting in a more powerful and accurate grouping approach. We combine the outputs of the two GRUs with a CRF to provide additional contextual information. Figure 4.3 shows a schematic of the system.

4.4.2 Input Sequences

The input to the network is a set of features computed from the pen strokes written on a single page. Each stroke consists of an arbitrary number of points collected from a pen-down to pen-up. Formally, the i^{th} stroke is recorded as:

$$S_i = \{(x_1^{(i)}, y_1^{(i)}, t_1^{(i)}), \dots, (x_m^{(i)}, y_m^{(i)}, t_m^{(i)})\}, \quad (4.1)$$

where $[x_j^{(i)}, y_j^{(i)}]$ are the coordinates of the j^{th} point and $t_j^{(i)}$ indicates the time it was drawn. m is the number of points in the stroke. We extract a 21-dimensional feature vector for each stroke and combine them together to produce the first input sequence:

$$I_1 = \{[f_1^{(1)}, \dots, f_{21}^{(1)}], \dots, [f_1^{(n)}, \dots, f_{21}^{(n)}]\} \quad (4.2)$$

where $[f_1^{(1)}, \dots, f_{21}^{(1)}]$ is the 21-dimensional feature vector computed for the i^{th} stroke. Each page can contain an arbitrary number of strokes, but in our method, we assume that they all have n strokes. The parameter n is set to 1650, which is the maximum number of strokes on a page in our training data. If a page has fewer than n strokes, zeros are added to the end of I_1 .

Table 4.1 shows an overview of the 21 stroke features which are taken from [50]. The first four features describe the size of the stroke. f_{BW} and f_{BH} denote the height and width of the minimum bounding box containing the stroke, respectively. Likewise, f_{BA} is the area of the bounding box. f_{AL} is the arc length of the stroke and is computed by summing up the euclidean distance between consecutive points in the stroke. These features

Table 4.1: Single stroke features from [50].

| Category | Name | Description |
|--------------------|--------------|--|
| Size | f_{BW} | Bounding box width |
| | f_{BH} | Bounding box height |
| | f_{BA} | Bounding box area |
| | f_{AL} | Arc length |
| Location | f_{D2LR} | Distance to left / right |
| | f_{D2TB} | Distance to top / bottom |
| | f_{AVGX} | Average x of the points in the stroke |
| | f_{AVGY} | Average y of the points in the stroke |
| Shape | f_{CSS} | Sum of the curvatures along the stroke |
| | f_{CSA} | Sum of the absolute value of the curvatures along the stroke |
| | f_{CSQ} | Sum of the square of the curvatures along the stroke |
| | f_{CSSQRT} | Sum of the square root of the curvatures along the stroke |
| | f_D | Ink density |
| | f_{SR} | Straightness ratio |
| Drawing kinematics | f_{SAVG} | Average speed in drawing the stroke |
| | f_{SMAX} | Maximum speed while drawing the stroke |
| | f_{SMIN} | Minimum speed while drawing the stroke |
| | f_{SDIFF} | Difference of the maximum and minimum pen speed |
| | f_{STIME} | Total time to draw the stroke |
| Temporal relation | f_{TN} | Temporal distance to the next stroke |
| | f_{TP} | Temporal distance to the previous stroke |

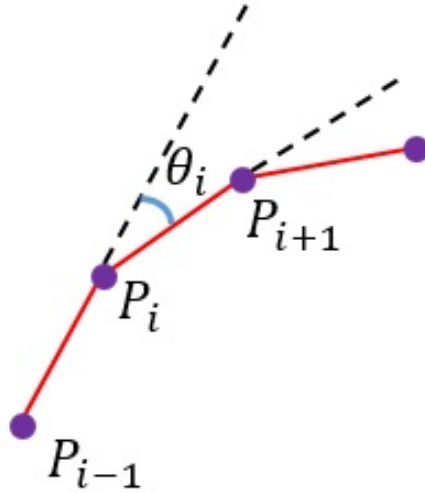


Figure 4.4: A visual illustration of curvature formed at point P_i .

are normalized using z-score normalization technique. Z-normalization subtracts the mean of a feature from original feature values and divides the difference by the standard deviation value.

One important insight is that certain types of content are more likely to be found at some places on the page rather than in other places. For example, the text for a student's name is likely to be found at the very top of the page. Likewise, free body diagrams tend to be written higher on the page than equations. For this reason, we use four features that characterize the location of a stroke on the drawing canvas. f_{D2LR} is the minimum distance from the stroke to the left or right edge of the canvas, and f_{D2TB} is the minimum distance from the stroke to the top and bottom edge of the canvas. f_{AVGX} and f_{AVGY} are the averages of the x and y-coordinates of the points, respectively. These features are normalized using the min-max normalization method. Min-max normalization linearly transforms the feature values into the range $[0, 1]$.

The next six features characterize the shape of the stroke. The first four of these describe the curvature of the stroke. As shown in Figure 4.4, the curvature at point P_i (denoted by θ_i) is defined as the angle between the lines formed by connecting point P_{i-1} to point P_i and connecting point P_i to P_{i+1} . θ_i is computed as:

$$\theta_i = \arctan \frac{\delta x_i \delta y_{i-1} + \delta x_{i-1} \delta y_i}{\delta x_i \delta x_{i-1} + \delta y_i \delta y_{i-1}} \quad (4.3)$$

where δx_i and δy_i are the horizontal and vertical distances between P_{i+1} and P_i , respectively. The four curvature features are obtained as follows:

$$f_{CSS} = \sum_{i=2}^{N-1} \theta_i \quad (4.4)$$

$$f_{CSA} = \sum_{i=2}^{N-1} |\theta_i| \quad (4.5)$$

$$f_{CSQ} = \sum_{i=2}^{N-1} \theta_i^2 \quad (4.6)$$

$$f_{CSSQRT} = \sum_{i=2}^{N-1} \sqrt{|\theta_i|} \quad (4.7)$$

f_{CSS} (Equation 4.4) is the sum of the signed curvatures along the stroke. Here, turning in the positive direction cancels turns in the negative direction and vice versa. Also, this value is computed modulo 360° so that if the pen stroke turns through a net rotation of 360° , the signed curvature value is computed as zero. f_{CSA} (Equation 4.5) is the sum of the absolute value of the curvature along the stroke. Whereas f_{CSS} represents the net change direction of the stroke path, f_{CSA} represents the total amount of bending

of the path. f_{CSQ} (Equation 4.6) represents the sum of the squared curvature along the stroke, which emphasizes the points that have high curvatures, such as corners. Finally, f_{CSSQRT} (Equation 4.7) is the sum of the square root of the curvature along the stroke, which emphasizes points of low curvature. These curvature features are normalized using z-score normalization.

f_D is the ink density and measures the compactness of the stroke. Ink density is defined as:

$$f_D = \frac{f_{AL}^2}{f_{BA}} \quad (4.8)$$

where the arc length is squared to scale similar to the bounding box area. f_D is normalized by the z-score normalization method. f_{SR} is the straightness ratio, which is defined as the euclidean distance between the endpoints of the stroke divided by the arc length. f_{SR} ranges between 0 to 1, where the value of 0 denotes that the endpoints have similar coordinates, and 1 represents a straight line with no curvature along the path.

Stahovich and Peterson [50] observed that students tend to draw familiar objects (e.g., student name or identification number) faster than they draw complicated shapes and diagrams. For that reason, we use five features that provide kinematic characteristics of the stroke. f_{SAVG} , f_{SMAX} , and f_{SMIN} are the average, maximum, and minimum of the pen speed along the stroke, respectively. f_{SDIFF} is the difference between the value of f_{SMAX} and the value of f_{SMIN} . Finally, f_{STIME} denotes the total time of drawing the stroke. These speed features are normalized using the z-score normalization technique.

The next two features provide the temporal distance to the next and previous

stroke. f_{TN} is the elapsed time between the last point of the stroke and the first point of the next stroke. Similarly, f_{TP} is the elapsed time between the last point of the previous stroke and the first point of the stroke. We use z-score normalization to normalize the f_{TN} and f_{TP} features.

We also extract 10 pairwise features for each pair of consecutive strokes and concatenate them together to form the second input sequence:

$$I_2 = \{[pw_1^{(1)}, \dots, pw_{10}^{(1)}], \dots, [pw_1^{(n-1)}, \dots, pw_{10}^{(n-1)}]\} \quad (4.9)$$

where $[pw_1^{(i)}, \dots, pw_{10}^{(i)}]$ is the 10-dimensional pairwise feature vector computed for the i^{th} and $(i+1)^{th}$ strokes. Table 4.2 provides an overview of the 10 pairwise features which are taken from [50]. The first five features characterize the distances between the strokes. pw_{min} and pw_{max} are the minimum and maximum distances between a point in one stroke and a point on the other stroke. $pw_{centroid}$ measures the distances between the centroids of the strokes' bounding boxes. pw_{minLL} is the minimum of the distances between an endpoint on one stroke and an endpoint on the other stroke. pw_{minXL} is the minimum distance between an endpoint on one stroke and any point on the other stroke. These features are normalized using z-score normalization.

The next two features measure the overlap of the strokes along the x- and y-axes. $pw_{XOverlap}$ denotes the length of the intersection of the projections of the strokes along the x-axis to the union of the projections. Similarly, $pw_{YOverlap}$ is the length of the intersection of the projections of the strokes along the y-axis to the union of the projections.

pw_T is the temporal distance (time) between the last point of the stroke and

Table 4.2: Pairwise stroke features from [50].

| Category | Name | Description |
|----------|-----------------|---|
| Distance | pw_{min} | Minimum point-to-point distance |
| | pw_{max} | Maximum point-to-point distance |
| | $pw_{centroid}$ | Distance between centroids of the strokes |
| | pw_{minLL} | Minimum endpoint-to-endpoint distance |
| | pw_{minXL} | Minimum endpoint-to-anypoint distance |
| Overlap | $pw_{XOverlap}$ | Intersection of strokes' projection onto x-axis |
| | $pw_{YOverlap}$ | Intersection of strokes' projection onto y-axis |
| Temporal | pw_T | Temporal gap between the strokes |
| Ratios | $pw_{RatioLL}$ | Ratio of pw_{minLL} to pw_{min} |
| | $pw_{RatioXL}$ | Ratio of pw_{minXL} to pw_{min} |

the first point of the next stroke. This feature is normalized with z-score normalization. $pw_{RatioLL}$ and $pw_{RatioXL}$ characterize whether strokes are closest to each other at their endpoints or at other points, and they are computed as follows:

$$pw_{RatioLL} = \frac{pw_{min} + k}{pw_{minLL} + k} \quad (4.10)$$

$$pw_{RatioXL} = \frac{pw_{min} + k}{pw_{minXL} + k} \quad (4.11)$$

where k is a constant value and is set to 10,000 metrics. Both $pw_{RatioLL}$ and $pw_{RatioXL}$ have values in the range of 0.0 to 1.0.

4.4.3 GRU (encoder)

A Gated Recurrent Unit (GRU) ¹ network is an improved version of an RNN and is used for processing sequential data. Similar to RNNs, GRUs are equipped with feedback loops that allow them to capture time dynamics. A GRU aims to address vanishing and exploding gradient problems, which are classical problems in traditional neural networks that destabilize the network during back-propagation training. These problems introduce challenges in the training process and make it difficult for the network to capture long-range dependencies.

To solve the vanishing and exploding gradient problems, GRUs are equipped with update and reset gates. These gates control the flow of information throughout the network and can be trained to hold information between time steps with large temporal distances and add/remove the important/irrelevant information.

Each GRU consists of three main components: an update gate, a reset gate, and a memory content. The update gate in a GRU network governs the flow of the information from earlier time steps to the future ones. The update gate determines the fraction of the earlier information to be passed along to the future time steps. The reset gate indicates the amount of past information to be dismissed. Current memory content is a vector of contextual information computed for the current unit.

Formally, we use the following equations to update components of a GRU at time

¹This presentation is adapted from [1].

step t :

$$\begin{aligned}
u_t &= \sigma(W_{xu}x_t + W_{hu}h_{t-1}) \\
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1}) \\
\tilde{h}_t &= \tanh(W_{xh}x_t + W_{rh}(r_t \otimes h_{t-1})) \\
h_t &= (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h}_t
\end{aligned} \tag{4.12}$$

where u_t , r_t , \tilde{h}_t and h_t are the update gate, reset gate, candidate activation vector, and memory content vector, respectively. W_{xu} , W_{hu} , W_{xr} , W_{hr} , W_{xh} , and W_{rh} are weight matrices.

Our network computes two sets of contextual information from temporally neighboring strokes. The GRU channel on the left of our network (as depicted in Figure 4.3) generates contextual information from features computed for each stroke. The GRU network on the right of our network produces contextual information from pairwise stroke features.

In many sequence labeling problems, future contextual information is just as significant as the prior information. To leverage both the past and future contextual information, we process the input sequences in both forward (i.e., from start to end) and backward (i.e., from end to start) directions. The final output of each GRU channel is produced by concatenating the hidden states of the two anti-parallel networks. Lastly, the left and right GRU networks are combined by merging the output of the networks at each time step. Using bi-directional GRU networks, along with contextual information computed for both the single and the pairwise stroke features, encodes complex contextual knowledge that exists in the pen strokes.

4.5 CRF (decoder)

In handwriting datasets, temporally neighboring strokes are highly correlated. In classification and segmentation tasks, incorporation of statistical dependencies between neighboring strokes can lead to better performance. CRF's are a family of undirected graphical models that provide a statistical framework to model the dependencies between correlated entities. Here, the role of the CRF is to predict a sequence of labels for the input elements jointly.

The input to the CRF layer is a sequence of hidden states computed in the GRU networks. Let $h = \{h_1, \dots, h_n\}$ denote an input sequence to the CRF layer. Each h_i is the concatenation of the hidden states computed at the i^{th} time step of the left and right GRU channels. Let $y = \{y_1, \dots, y_n\}$ represent a sequence of labels predicted by the CRF layer.

We model the dependencies between the GRU's hidden states by using the linear chain CRF method:

$$P(y|h; \theta) = \frac{1}{Z(h)} \exp\{\phi(h, y; \theta)\} \quad (4.13)$$

where $\theta = \{W, b\}$ is a set of weight matrices and bias parameters of the model. $Z(h)$ is the partition factor that transforms the distribution into a probability distribution. $\phi(h, y; \theta)$ represents the potential function which consists of unary and pairwise potential functions as:

$$\phi(h, y; \theta) = \sum_{i=1}^n \phi_U(h_i, y_i; \theta) + \sum_{i=1}^{n-1} \phi_P(h_i, h_{i+1}, y_i, y_{i+1}; \theta) \quad (4.14)$$

where $\phi_U(h_i, y_i; \theta)$ denotes the unary potential function that measures the consistency of the i^{th} stroke and the label y_i . $\phi_P(h_i, y_i, y_{i+1}; \theta)$ is the pairwise potential function that measures the consistency of the predicted labels for adjacent strokes. Unary and pairwise

potential functions are formulated as

$$\begin{aligned}\phi_U(h_i, y_i; \theta) &= h_i^T \theta_{y_i}^u \\ \phi_P(h_i, h_{i+1}, y_i, y_{i+1}; \theta) &= h_i^T \theta_{y_i, y_{i+1}}^{p,1} + h_{i+1}^T \theta_{y_i, y_{i+1}}^{p,2}\end{aligned}\tag{4.15}$$

Here, $\theta_{y_i}^u$ is the parameter for the unary potential function and $\theta_{y_i, y_{i+1}}^{p,1}$ and $\theta_{y_i, y_{i+1}}^{p,2}$ are the parameters for the pairwise potential function.

We use the maximum likelihood estimation technique to search for the optimal solution. During the training process, we look for parameters that maximize the following conditional likelihood:

$$L(\theta) = \sum_{i=1}^N \log p(y|h; \theta)\tag{4.16}$$

where N is the number of pages in our training set. We use the gradient descent method to estimate the parameters.

During testing, we use the Viterbi algorithm to find the optimal labels for the given input so as to maximize the following conditional probability:

$$y^* = \operatorname{argmax}_{y \in \mathcal{Y}(h)} p(y|h; \theta)\tag{4.17}$$

where $\mathcal{Y}(h)$ is the set of all possible labels for the given input.

The two labels assigned to each stroke are used to group the strokes into equation segments via a chaining process. If two consecutive strokes are both of the “equation” type, and the first is labeled “Join”, then the two are placed in the same equation segment. Applying this rule exhaustively yields the set of equation segments. The equation segments that are produced in this step are called “initial groups”.

Consider the example in Figure 4.5 in which all of the strokes have been labeled as type “equation” and all except the fifth and 11th have been labeled as “Join”. (The number

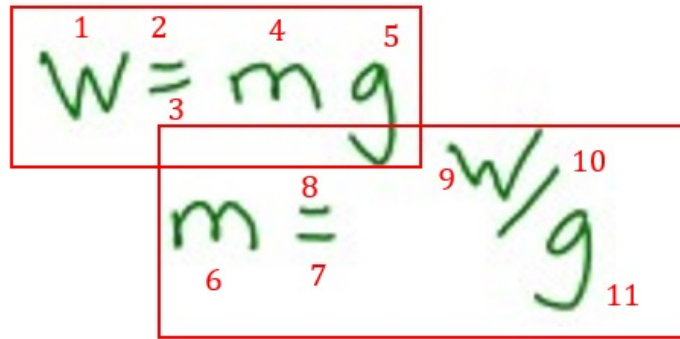


Figure 4.5: Two equation segments. The numbers next to the strokes indicate the order in which they were drawn.

next to each pen strokes indicates that order in which it was drawn.) In this case, the first five strokes are chained together to form one initial group, and the last six are chained to form a second initial group.

4.6 Combining Initial Groups

Our goal is to find all of the strokes of an equation segment while excluding all other strokes. Many of the initial groups produced in the previous step are complete equation segments. However, there are many cases in which equation strokes are not written in contiguous temporal sequences. In such situations, an equation segment may be split over multiple initial groups.

To remedy this, we use a second classifier that examines the initial groups to determine if they need to be combined. More specifically, this classifier is used to determine if large initial groups (i.e., those that have a large number of strokes) should be joined with nearby small groups (i.e., those that have a small number of strokes). This binary classifier

Equation 1 \longrightarrow $F = 100(.96\hat{i} - 1.73\hat{j} - 1\hat{k})$
 Equation 2 \longrightarrow $F = 96\hat{i} - 173\hat{j} - 100\hat{k}$

The image shows two equations. The first is $F = 100(.96\hat{i} - 1.73\hat{j} - 1\hat{k})$ and the second is $F = 96\hat{i} - 173\hat{j} - 100\hat{k}$. A green box is drawn around the second equation, and a red arrow points from the label 'Equation 2' to it. The first equation also has a red arrow pointing to it.

Figure 4.6: Many of the strokes in equation segment 1 intersect with equation segment 2.

uses pairwise spatial features computed from the large group and the neighboring small group to determine if they should be joined together.

Typically, equations are written such that most strokes that belong to the same equation segment are placed close to one another, and writers try to place spatial distances between strokes of different segments. Consequently, spatial distances are fruitful sources of information for segmenting equation strokes. However, due to the variations in placements of symbols such as subscripts, superscripts, fractions, matrices, etc., many inter-cluster strokes intersect with one another, or their bounding boxes largely overlap. As a result, it is not easy to rely on spatial distances to merge equation groups.

In Figure 4.6, two equations are sketched, and each of them is properly grouped by the GRU-CRF network and placed in one of the initial groups. If we decide to consider spatial distances between equations to combine initial groups, it is highly likely that these two groups would be combined. As shown in Figure 4.6, the answer box stroke in equation 2 intersects with many strokes in equation one, and most of the strokes in equation 1 are vertically close to strokes in equation 2. Therefore, to reliably combine initial groups, spatial distances are not definitive. Instead, we perform a semantic analysis of equations to determine if equation groups belong to the same equation segment.

$$66.27N = 750 + \frac{25By}{20}$$

Figure 4.7: Most of the equation strokes are sketched consecutively, and they are shown with green color. The red stroke is added to the equation afterward.

However, cases exist where students write most of the equation strokes consecutively, and after sketching some unrelated strokes, they add minor changes to the initially drawn equation. In these cases, most of the equation strokes are drawn in one run, and minor changes consist of a few strokes. We use spatial features to find and group such cases.

We identify large groups among initial groups, and for each of the large groups, we perform a local search to find candidate small groups to be merged. We compute pairwise features between a large group and each of its candidate small groups and use a binary classifier to decide whether or not the groups should be merged with one another. We empirically define a group as a large group if it contains at least 10 equation strokes, and a group is a small group if it at most contains 5 equation strokes. We consider a set of candidate small groups for a large group wherein each of the small groups there is at least one stroke that has a minimum point-to-point distance with one of the strokes in the large group less than $2cm$.

Table 4.3 shows an overview of the spatial features used for merging large and small groups. pw_{minX} and pw_{minY} are the minimum and maximum distances between the bounding boxes of the groups along the x and y-axis, respectively. The description of the $pw_{XOverlap}$ and $pw_{YOverlap}$ are similar to what we described for the features in Table 4.2.

Table 4.3: Pairwise stroke features for grouping small and large equation groups.

| Category | Name | Description |
|----------|-----------------|---|
| Distance | pw_{minX} | Minimum distance along the x-axis |
| | pw_{minY} | Minimum distance along the y-axis |
| Overlap | $pw_{XOverlap}$ | Intersection of strokes' projection onto the x-axis |
| | $pw_{YOverlap}$ | Intersection of strokes' projection onto the y-axis |

We explored a variety of classification methods for the task of combining small and large groups, and we found that the highest accuracy was achieved by using the neural network classifier.

4.7 System Workflow

We sort the pen strokes in chronological order before feeding them to the network. As described in Section 4.4.2, we compute and normalize single stroke and pairwise stroke features to produce input sequences I_1 and I_2 . As shown in Figure 4.3, I_1 and I_2 are fed to two separate Bidirectional GRU networks, and the output of the networks are concatenated and fed to the CRF layer. The CRF layer jointly predicts two sequences of labels for the output nodes. The first sequence label identifies equation strokes, and the second sequence label determines if the stroke is part of the same equation as the subsequent stroke. Finally, we use a binary classifier to combine some of the initial groups created by our network.

We use the L2-regularization method with the parameter set to 0.001, to overcome overfitting with the GRU networks. Additionally, we apply dropout to the recurrent input

signal, with the dropout rate set to 0.2.

We use the Keras [16] framework to build our neural network model. The entire network contains 94,830 trainable parameters. We use RMSProp [17] as the optimization method, and at each step of training, we feed samples of size 10 to the network. We performed the experiments on a machine equipped with a 2.66 GHz Xeon(R) CPU. The model training required approximately 17 hours.

4.8 Results

To evaluate the performance of our method, we conducted experiments on a database [62] of handwritten homework assignments, quizzes, and exams collected from 132 undergraduate students that were registered in a mechanical engineering course on statics at UC Riverside. Throughout the course of study, the students used Livescribe digital pens and dot-patterned paper to do their written work. The pen tip of the smartpen contains a built-in camera to digitize the writing activity as timestamped coordinates.

After manually labeling 1,060 pages containing solutions to free-response questions from 5 exam problems (293 pages) and 8 homework problems (776 pages). In total, there are 121,740 strokes in the exam solutions and 298,527 in the homework solutions. The frequencies of the various stroke types are shown in Table 4.4. Miscellaneous Notes strokes are strokes that are not classified as free body diagram, cross-out, or equation. We used the homework pages and exam pages for training and testing, respectively. For validation, during the training phase, we withhold 155 pages of the training data.

We manually labeled 6,029 equation segments in the training data and 3,920

Table 4.4: Frequencies of stroke types in each dataset.

| Dataset | Stroke Type | | | |
|---------------|-------------|-------|-----------|-------------|
| | Equation | FBD | Cross-out | Misc. Notes |
| Training Data | 63.8% | 31.2% | 1.2% | 3.8% |
| Testing Data | 76.6% | 20.5% | 1.4% | 1.5% |

equation segments in the testing data. The equation segments in the training data contain an average of 27 strokes, and those in the testing data contain an average of 23.

4.8.1 Results for Equation/Non-equation Classification

In chapter 3, we presented the performance of our approach to classifying strokes into multiple classes. In this chapter, we are concerned about the accuracy of our network in distinguishing between two classes: equation strokes and non-equation strokes. Table 4.5 shows the performance of our GRU-CRF network in classifying strokes into these two

Table 4.5: Confusion matrix for equation/non-equation classification.

| Actual \ Prediction | Equation | Non-equation |
|---------------------|----------|--------------|
| | Equation | 91,970 |
| Non-equation | 5,115 | 23,331 |

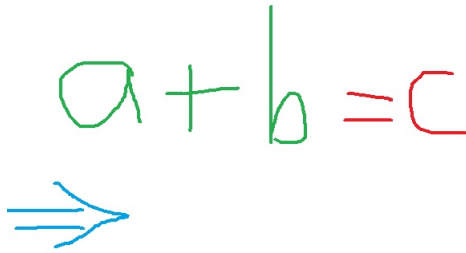


Figure 4.8: An example of the predicted equation segment. Strokes that are correctly included in the group are shown in green, strokes erroneously excluded from the group are shown in red, and strokes erroneously included in the group are shown in blue.

classes. Our network achieves an overall accuracy of 94.7%.

4.8.2 Results for Equation Segmentation

We use two types of metrics [50] to measure the performance of our method for grouping equation strokes. The first type of metric concerns the length of the ink that is correctly grouped. We define *InkFound* as the percentage of the true ink contained in the group generated by our program. Additionally, a program-generated group may contain extra ink not contained in the true group. We define *InkExtra* as the ratio of the length of the extra ink to the length of the true ink. For example, if the true equation segment contains 10 cm of ink, and 8 cm of that ink is contained in the program-generated group, the value of *InkFound* is 80%. Likewise, if the program-generated group contains 1.5 cm of ink that is not contained in the true group, the value of *InkExtra* is 15%. The second type of metric concerns the number of strokes that are correctly grouped. Here we count the number of strokes that are erroneously included in or excluded from the machine-generated group.

Consider the equation segment in Figure 4.8 consisting of strokes forming the characters “a”, “+”, “b”, “=” and “c”, which have arc lengths of 150, 110, 130, 90, and 140 units, respectively. In this example, imagine that the strokes of “a”, “+” and “b” are correctly grouped together, but the strokes of “=” and “c” are excluded. Also, imagine that the strokes of an additional arrow, which has an arc length of 175 units, are erroneously included in the group. In the case we would obtain $InkFound = \frac{150+110+130}{150+110+130+90+140} = 62.9\%$ and $InkExtra = \frac{175}{150+110+130+90+140} = 28.2\%$. Likewise, the number of incorrect strokes would be six, including three erroneously excluded strokes from “=” and “c” and three erroneously included strokes from the arrow.

Figure 4.9 shows the performance of our equation grouping technique on our test set containing 3,920 equation segments. We report the performance for a model comprising only the GRUs and CRF and for a model that also includes post-processing with our pairwise classifier that combines small groups with neighboring large ones. Without post-processing, our methods achieve an average *InkFound* of 86.2%, and 64.4% of the groups have three or fewer erroneous strokes. Recall that the equation segments in the testing set have an average of 23 strokes. With post-processing, our methods achieve an average *InkFound* of 89.6% and 70.0% of the groups have three or fewer erroneous strokes. The post-processing achieves a small increase in *InkFound*, but also produces an increase in *InkExtra*. Thus post-processing does correctly add missing strokes to the groups, but also erroneously adds strokes that do not belong.

4.8.3 Comparison with Existing Methods

We benchmark our technique against three existing methods: (1) The method of Stahovich and Lin [62], (2) the IPC-IRL method [63], and (3) the method of Delaye and Kibok [19]. The first is specifically designed for grouping equation strokes that are written on a single baseline. If an equation is written on multiple baselines, it is likely to be grouped into multiple groups, one for each baseline. For example, a fraction might be grouped into two groups. However, as most of the equation segments in our dataset are written on a single baseline, this method is appropriate for our data. The other two methods are general purpose pen stroke grouping algorithms.

For this comparison, we provide our method with all of the strokes on the page, and it must first identify the equation strokes before grouping them. For the other two methods, we provide only the equation strokes on the page. Thus, in this comparison, our method is performing a more challenging task than the other methods. Despite this, as shown in Figure 4.10, our method outperforms the other methods on all performance measures. Table 4.6 shows the average processing time for our method and the three benchmark methods. Here, we report processing time for our method both with and without post-processing by the pairwise classifier. Only Lin’s method is faster than our method. However, our method is performing a more challenging task as it is both identifying and grouping the equation strokes. With Lin’s method, additional time would be needed to identify the equation strokes.

Table 4.6: Comparison of the average running time for grouping equation strokes on a page.

| Method | GRU-CRF | GRU-CRF-SC | Lin [62] | Delaye [19] | IPC-IRL [63] |
|----------------------------|---------|------------|----------|-------------|--------------|
| Avg. processing time (sec) | 1.25 | 2.54 | 0.28 | 39.72 | 18.96 |

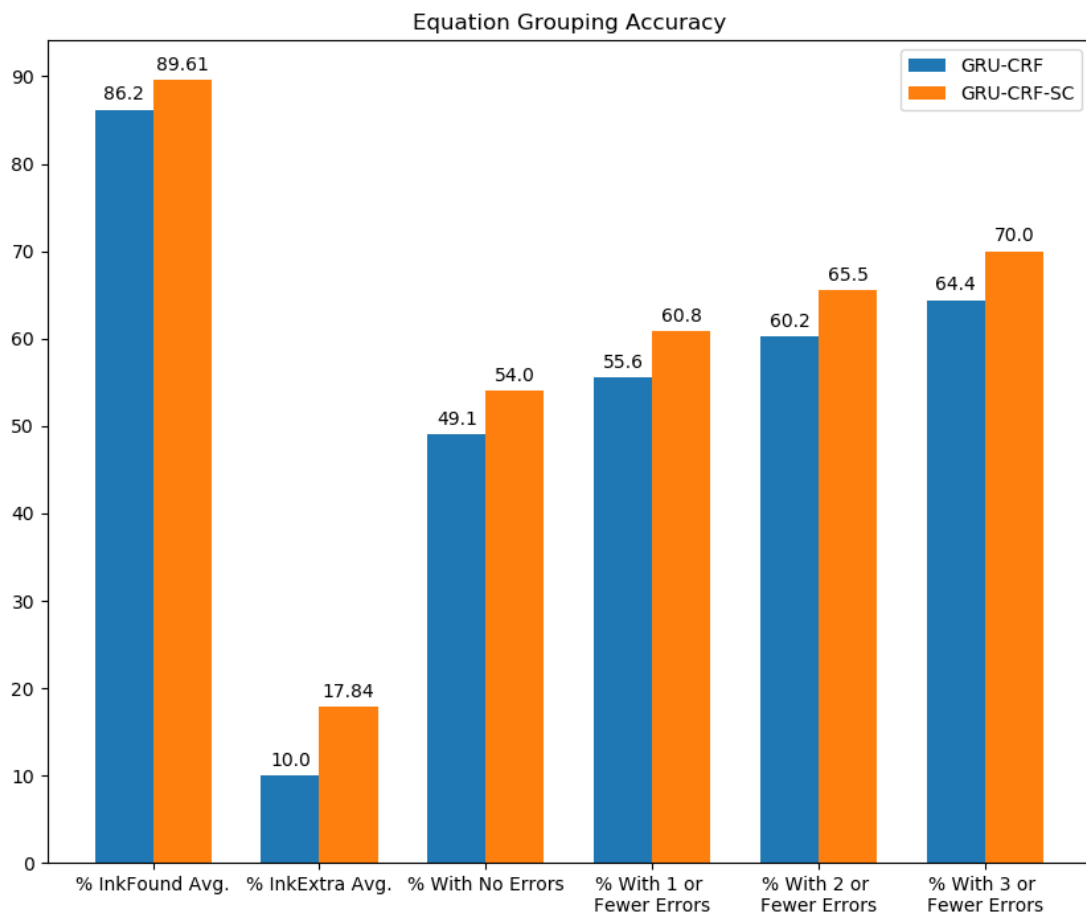


Figure 4.9: Performance of our equation grouping method without post-processing “GRU-CRF” and with post-processing by our pairwise classifier that combines small groups with neighboring large ones “GRU-CRF-SC”.

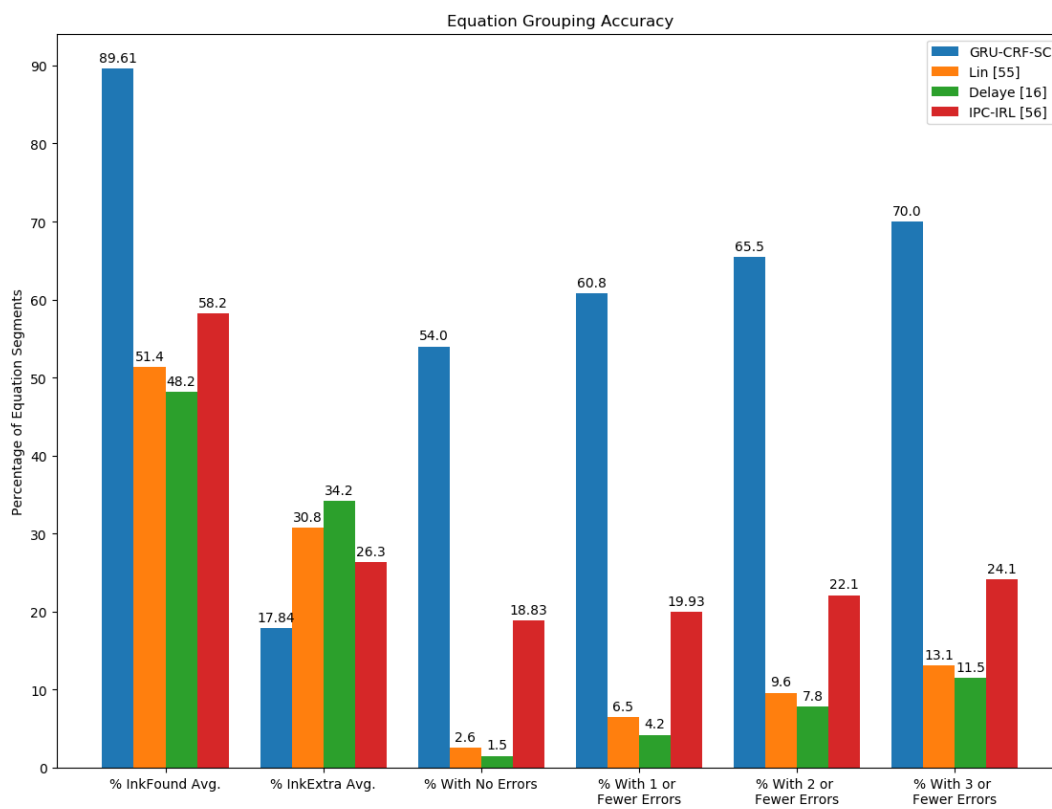


Figure 4.10: Performance of four different grouping methods for equation grouping.

Chapter 5

Conclusions

In this thesis, we have developed methods that support the automatic grading of students' handwritten assignments. In this chapter, we review our contributions, discuss the limitations of our methods, and present ideas for future work.

Our first contribution is the development of techniques for finding final answers in handwritten solutions to free-response questions. Finding the answers is an important step toward automated grading. In our study, students in an undergraduate statics course used Livescribe smartpens to do their homework assignments and answer quiz and exam questions. We asked the students to draw a box around each final answer and to separate different components of the answer with vertical lines. The input to our method is a set of pen strokes on a page. The output is the list of strokes corresponding to each final answer. Our method is efficient because it avoids expensive search. Processing begins with a search whose cost is linear in the number of pen strokes on the page. The results of this are then used to drive local searches with small search spaces.

On average, our method took only four seconds to process a page of writing from our data set. Our method proved to be accurate, achieving an overall accuracy of 95.3% in recognizing answer boxes in our data set. Our methods for locating answer boxes could be combined with text recognition techniques to enable the automatic interpretation of handwritten answers.

Our second contribution is the development of a novel machine learning model for semantic labeling of students' handwritten solutions to free-response questions. Our task is to classify the strokes into three classes: strokes that are part of a free body diagram, strokes that are part of a cross-out, and strokes that belong to text. (Alphanumeric characters that are included in a free body diagram are classified as free body diagram strokes and not text strokes.) Semantic labeling reduces the complexity of document interpretation tasks by separating strokes of different types so that special-purpose techniques can be applied to each type of stroke separately. Additionally, semantic labeling is required by some existing techniques [62, 66] that estimate the correctness of a handwritten solution by examining the properties of the various types of content.

Many methods exist for interpreting handwriting. These methods are effective because they exploit the known structure of the writing. For example, a page of text may be written from left to right and from the top of the page to the bottom. For our problem, there is no fixed spatial organization. A student can write anywhere on the page in any order. While there is no fixed spatial organization to guide our task, we found that we could exploit temporal organization. We found that pen strokes in the free-form solutions frequently have the same semantic type as their temporal neighbors. More specifically,

each stroke in our data set was preceded and succeeded by an average of 46 consecutive temporal neighbors of the same semantic type. Our methods exploit this by processing the strokes on a page as a temporal sequence. This is a key element of the effectiveness of our methods.

There are two sequences of inputs to our novel machine learning model: one consists of the raw data points of the sequence of strokes, the other consists of features that characterize the shapes of the strokes and the spatial and temporal relationships between each stroke in the sequence and its immediate succeeding neighbor. The output of the model is a semantic label for each stroke in the sequence. Each input sequence is processed by its own convolutional neural network (CNN) that extracts new features characterizing the sequence of strokes. The output of each CNN is then processed by its own bidirectional-long-short-term-memory (BLSTM) network. The BLSTM networks produce the global context of the entire sequence of strokes and are used to produce an intermediate feature vector for each stroke. The BLSTM networks sometimes misclassifying strokes at the boundary between two semantic types in the input sequence. We remedy this using a conditional random field (CRF). The outputs of the two BLSTM networks are concatenated and fed to the CRF, which models the interactions between temporally nearby strokes and predicts a label for each stroke, such that the label is consistent with the stroke’s neighbors.

In testing, our method proved to work well. On average, our method achieved 94.7% accuracy for semantic labeling, which is 2.47 percentage points more accurate than a previous state-of-the-art method. Also, our method is faster than existing methods. On average, our method took about 3.2 seconds to process a page of writing, whereas a state-

of-the-art method took about 285 seconds.

Our third contribution is the development of a two-step grouping method to find individual equations in students' handwritten solutions to free-response questions. Mathematical equations are the most abundant type of content in handwritten solutions in many academic subjects, especially in science, engineering, math, and technology. The purpose of our grouping method is to enable existing equation recognition techniques such as [69, 74, 73] to be applied to free form writing.

Many existing methods for finding meaningful objects in handwritten documents employ a two-step process. The pen strokes are first categorized into various classes. Then the strokes of each class are clustered into objects. By contrast, we developed a novel method that simultaneously finds equation strokes and groups them into equations. Based on the same insights behind our semantic labeling method, we use sequential information to do this.

The input to our method is two sequences: one consists of features that describe the shapes of the strokes, the other consists of pairwise features that characterize the spatial and temporal relationships between each stroke in the sequence and its immediate succeeding neighbor. Each input sequence is processed with a separate gated recurrent unit (GRU) network that computes the global context of each stroke in the sequence and produces an intermediate representation for each stroke. The outputs of the GRU networks are combined and fed to a CRF layer to predict labels for the strokes. The model produces two labels for each stroke. The first indicates whether the stroke is an equation stroke or not. The second label indicates whether or not the stroke is part of the same equation group as the

subsequent stroke. Using these two labels, a chaining process assembles the strokes into groups representing equations. Sometimes students revisit equations to add to them. This can result in the equation being grouped into multiple pieces. We remedy this by using a pairwise classifier that examines spatial relationships between pairs of equation groups to determine if they should be combined.

Our method can find 70% of the equations with no more than three missing or extra pen strokes, which is significantly better than existing segmentation methods. On average, our method takes 2.54 seconds to process a page. The output of our method is suitable for use with mathematical expression recognizers that could be used to interpret the equations.

Limitations and Future Works

Our answer box recognizer is designed to locate and find final answers to free-response questions where each final answer comprises three components (i.e., unit, value, and problem identification). The next step is to integrate our methods with text recognition methods to enable interpretation of the answers. One limitation of our method is that it is designed for applications in which each answer has exactly three components. To use our method for other applications, it will be necessary to remove this requirement.

Our semantic labeler accurately finds the strokes of free body diagrams and equations. However, we observed that our method performs poorly in recognizing cross-out strokes. This is due to an imbalance in our dataset: less than 3% of the dataset is cross-outs. To improve the accuracy of our model, we need to use training methods that support imbalanced data sets. Additionally, we would like to extend our method to include ad-

ditional semantic classes such as explanatory notes, organizational information, geometric diagrams (rather than free body diagrams), and so on.

While our equation grouper works well and is superior to existing grouping methods, more work is needed to improve its accuracy. We believe that this could be accomplished by combining our grouping method with semantic analysis. More specifically, combining our method with an equation interpreter could improve accuracy by augmenting the spatial and temporal properties we currently consider with an analysis of the meaning of the strokes in each group. In some cases, knowledge of the actual mathematical meaning of the pen strokes is needed to make grouping decisions.

Conclusion

There are no existing methods for automatically grading handwritten solutions to free-response problems. There are methods [62, 66] that can estimate the correctness of a solution by extracting features from the writing, but these methods do not attempt to interpret the work. Handwriting recognizers cannot be directly applied to this problem as they require individual objects (e.g., equation, word, etc.) to be distinguished from one another. Unlike with handwritten text, solutions to free-response questions have no well-defined organization that can be used to distinguish objects. For such solutions, students can write anywhere on the page at any time.

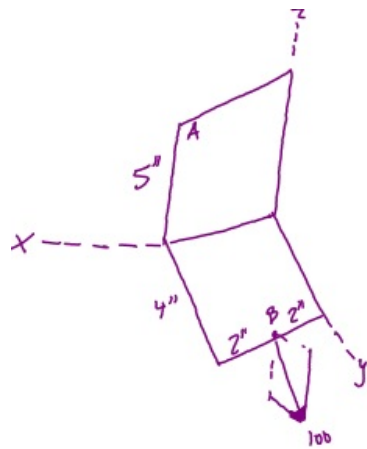
In this dissertation, we have presented methods to locate various types of objects in solutions to free-response problems so that interpretation methods can be applied. First, we developed a method for identifying final answers. Once the answers are found, a text recognizer can be used to interpret them. Second, we developed a method for semantic

labeling of the solutions. Our method treats the writing as a temporal sequence, which proved to be a powerful approach to interpreting the writing. Semantic labeling reduces the complexity of analyzing the writing by decomposing it into smaller problems, one for each type of content. Additionally, our labeler can support existing methods [62, 66] for estimating the correctness of solution from properties of the various types of content. Third, we developed a method for identifying mathematical equations. The output of our method can be used with existing methods [69, 74, 73] to interpret the equations.

We have not yet achieved the goal of complete interpretation and semantic analysis of handwritten solutions to free-response problems. However, we have made significant advances toward that goal by developing methods for locating important objects, including answers and equations, in free-form writing.

Appendix A

**Examples of stroke classification
results by the CNN-BLSTM-CRF
model.**



$$F = 100\text{lb}$$

$$\sum M_A =$$

$$\begin{pmatrix} 5\hat{i} + 4\hat{j} + 2\hat{k} \\ 2\hat{i} + 4\hat{j} + 5\hat{k} \end{pmatrix}$$

$$100(5 \quad ?)$$

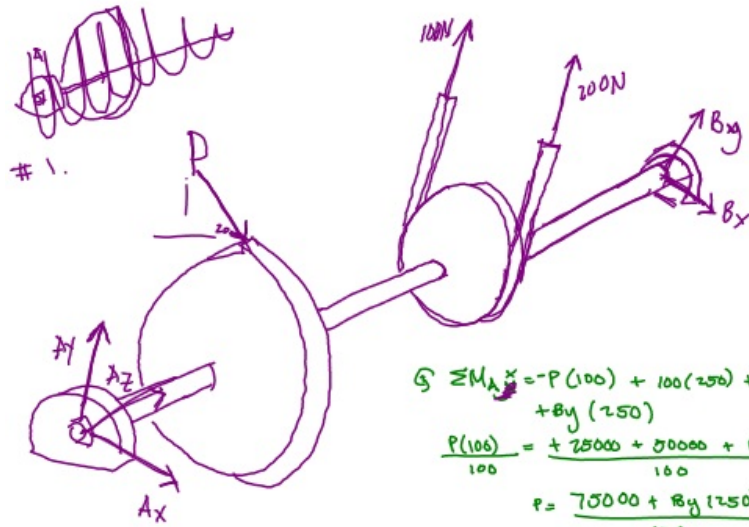
$$\begin{aligned} & (5\hat{i} + 4\hat{j} + 2\hat{k}) \cdot (2\hat{i} + 4\hat{j} + 5\hat{k}) \\ & 100(10\hat{i} - 16\hat{j} - 10\hat{k}) \\ & = 100(10\hat{i} - 16\hat{j} - 10\hat{k}) \\ & 100 \cdot \sqrt{10^2 - 16^2 - 10^2} \\ & \quad \underline{\underline{-1886.8 \text{ N}\cdot\text{m}}} \end{aligned}$$

Figure A.1: Equation and FBD strokes are shown in green and purple, respectively.

SOH
CAH

$$P_y = -P \sin(20^\circ)$$

$$P_x = P \cos(20^\circ)$$



$$\sum M_A = -P(100) + 100(250) + 200(250) + B_y(250)$$

$$\frac{P(100)}{100} = \frac{+25000 + 50000 + B_y(250)}{100}$$

$$P = \frac{75000 + B_y(250)}{100}$$

$$\uparrow \sum F_y = -P_y + A_y + 100N + 200N + B_y = 0$$

$$-P \sin 20 + A_y + 300N + B_y = 0$$
~~$$-750 + 25B_y + A_y + 300 + B_y = 0$$~~

$$P = 750 + 25B_y$$

$$\rightarrow \sum F_x = P_x + A_x + B_x = 0$$

$$P \cos 20 + A_x + B_x = 0$$

$$\updownarrow \sum F_z = 0$$

$$A_z = 0$$

Figure A.2: Equation and FBD strokes are shown in green and purple, respectively.

#1

$$\begin{aligned} \sum M_A = 0 & \Rightarrow -P(120) - 100\left(\frac{160}{2}\right) + 200\left(\frac{160}{2}\right) = 0 \\ P(120) &= -8000 + 16000 \\ \frac{P(120)}{120} &= \frac{8000}{120} \\ P &= 66.67 \text{ N} \end{aligned}$$

$$\begin{aligned} P &= 750 + 25B_y \\ 66.67 \text{ N} &= 750 + \frac{25B_y}{25} \\ -27.33 \text{ N} &= B_y \end{aligned}$$

$$\begin{aligned} \sum F_y = 0 & \Rightarrow -P \sin 20 + A_y + 300 \text{ N} + B_y = 0 \\ -(66.67) \sin 20 + 300 \text{ N} + (-27.33 \text{ N}) &= -A_y \\ 249.87 &= -A_y \\ -249.87 &= A_y \end{aligned}$$

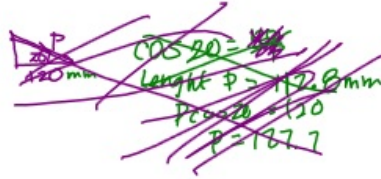
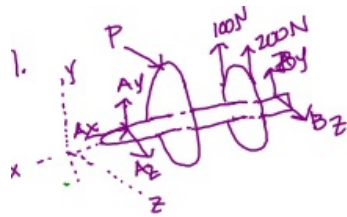
$$\begin{aligned} \sum M_B = 0 & \Rightarrow -P(100) - B_x(350) = 0 \\ B_x &= \frac{-P(100)}{350} \\ &= \frac{66.67(100)}{350} \\ B_x &= 19.05 \text{ N} \end{aligned}$$

$$\begin{aligned} \sum F_x = 0 & \Rightarrow (66.67) \cos 20 + A_x + 19.05 \text{ N} = 0 \\ A_x &= -66.67(\cos 20) - 19.05 \\ A_x &= -81.70 \text{ N} \end{aligned}$$

$$\begin{aligned} \|A\| &= \sqrt{(-81.70)^2 + (-249.87)^2 + 0^2} \\ A &= 262.89 \text{ N} \end{aligned}$$

$$\begin{aligned} \|B\| &= \sqrt{(19.05)^2 + (27.33)^2} \\ B &= 33.31 \text{ N} \end{aligned}$$

Figure A.3: Equation and FBD strokes are shown in green and purple, respectively.



$$\begin{aligned} \Sigma F_x &= 0 \\ -A_x &= 0 \\ A_x &= 0 \\ \Sigma F_y &= 0 \\ A_y + B_y + 100\text{ N} + 200\text{ N} + P \sin 20 &= 0 \\ \Sigma F_z &= 0 \\ A_z + B_z + P \cos 20 &= 0 \end{aligned}$$

$$\Sigma M_{A, y} = 0$$

$$-P \cos 20 (120\text{ mm}) - B_z (350\text{ mm}) = 0$$

$$\Sigma M_{A, x} = 0$$

$$-P \cos 20 (120\text{ mm}) - 100\text{ N}(80) + 200(80) = 0$$

$$-P \cos 20 (120\text{ mm}) = -8000\text{ N mm}$$

$$P = 7.1\text{ N}$$

$$\therefore -P \sin 20 (100\text{ mm}) = B_z (350\text{ mm})$$

$$\Sigma M_{A, x} = 0$$

$$\cos 20 (127.7) (120\text{ mm}) - 100\text{ N}(80) + 200(80) = 0$$

$$P (127.7) (120\text{ mm}) - 100\text{ N}(80) + 200(80) = 0$$

$$P =$$

$$\Sigma M_{A, x} = 0$$

$$P \cos 20 (120\text{ mm}) - 100\text{ N}(80) + 200(80) = 0$$

$$D = -7.1\text{ N}$$

Figure A.4: Equation and FBD strokes are shown in green and purple, respectively.

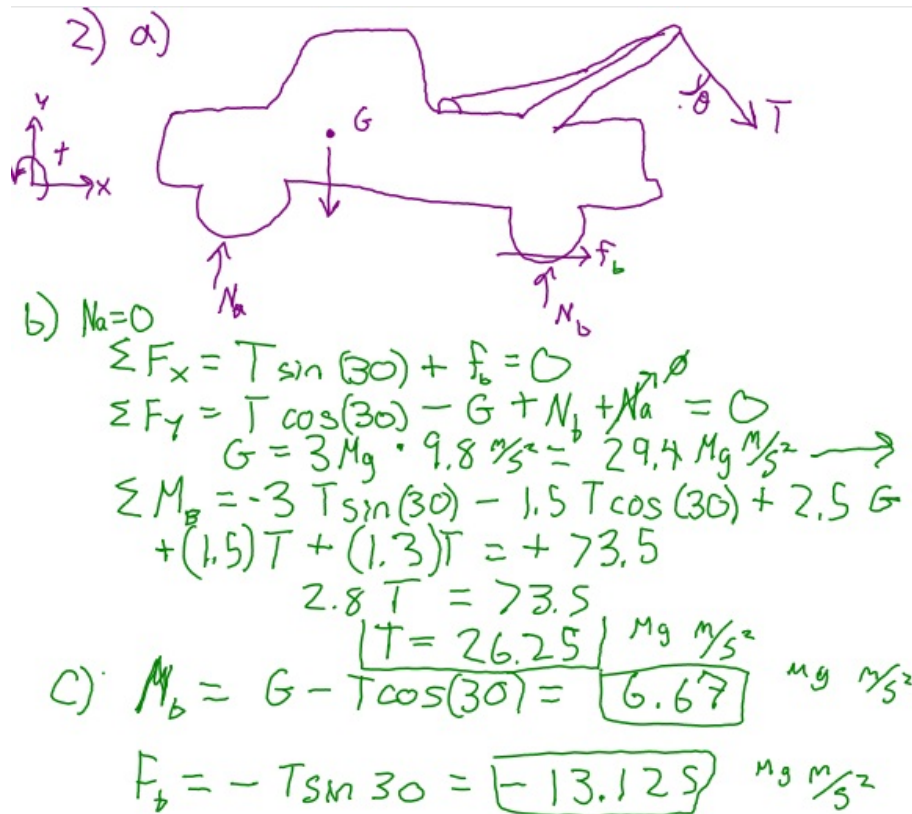


Figure A.5: Equation and FBD strokes are shown in green and purple, respectively.

Appendix B

**Examples of equation segments
recognized by the GRU-CRF-SC
model.**

Problem 3

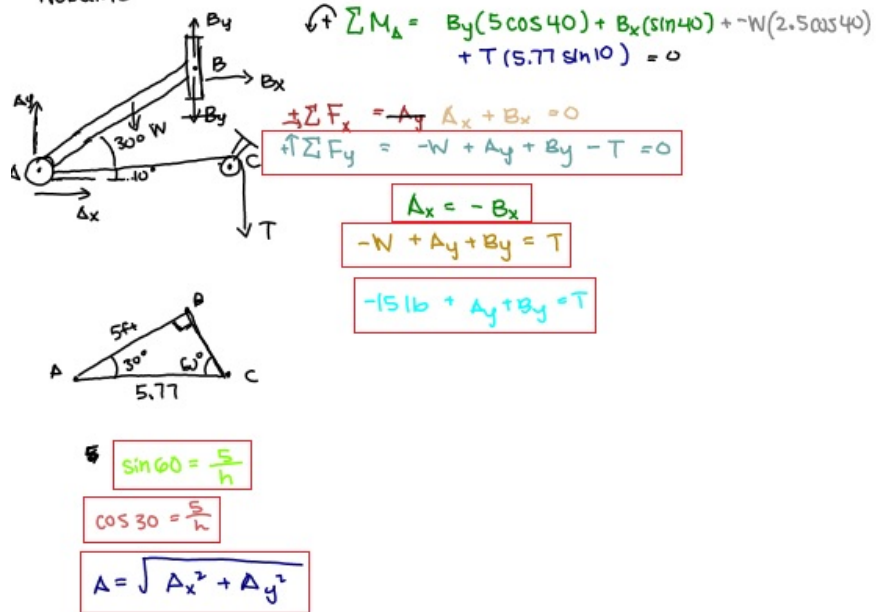
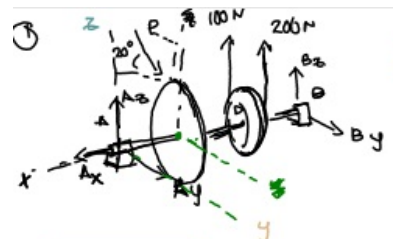


Figure B.1: A red box is drawn around each perfectly recognized equation segment.



Free body diagram showing a system with forces A_x, A_y, A_z at point A, reaction forces B_y, B_z at point B, and applied forces of 100 N and 200 N. A force P is applied at a distance of 0.12 m from the center, at an angle of 20° to the vertical. The diagram includes coordinate axes x, y, z .

Equations and solutions:

$$\begin{aligned} \textcircled{a} M_{A,x} &= 0 \\ -P \cos 20^\circ (.12) + P \sin 20^\circ (.12) \\ -100 (.08) + 200 (.08) &= 0 \\ -.0717 P &= -8 \\ \boxed{P = 111.54 \text{ N}} \end{aligned}$$

$$\textcircled{b} M_{A,y} = 0$$

$$\begin{aligned} -P \cos 20^\circ (.1) - B_y (.35) &= 0 \\ -111.54 \cos 20^\circ (.1) &= .35 B_y \\ -10.48 &= .35 B_y \\ \boxed{B_y = -29.95 \text{ N}} \end{aligned}$$

$$\begin{aligned} \Rightarrow F_x &= 0 & \boxed{A_x = 0} \\ \uparrow F_y &= 0 & A_y + B_y = 0 & \quad A_y = -B_y = 29.95 \text{ N} \\ \uparrow F_z &= 0 & A_z & \quad A_y + B_y + (-P) \sin 20^\circ = 0 \\ & & & \quad \boxed{A_y = 68.099 \text{ N}} \\ & & & \quad A_z + B_z + 100 + 200 - P \sin 20^\circ = 0 \\ & & & \quad \boxed{B_z = -203.386 \text{ N}} \end{aligned}$$

$$\textcircled{c} M_{A,y} = 0$$

$$\begin{aligned} -P \sin 20^\circ (.18) + 100 (.25) + 200 (.25) + B_z (.35) &= 0 \\ \boxed{B_z = -203.386 \text{ N}} \end{aligned}$$

$$\begin{aligned} A_z &= 203.386 \text{ N} + (-100 \text{ N}) - 200 \text{ N} + 111.54 \sin 20^\circ \\ \boxed{A_z = -58.47 \text{ N}} \end{aligned}$$

$$\begin{aligned} A &= \sqrt{A_x^2 + A_y^2 + A_z^2} = \boxed{89.753 \text{ N}} \\ B &= \sqrt{B_y^2 + B_z^2} = \boxed{205.58 \text{ N}} \end{aligned}$$

Figure B.2: A red box is drawn around each perfectly recognized equation segment.

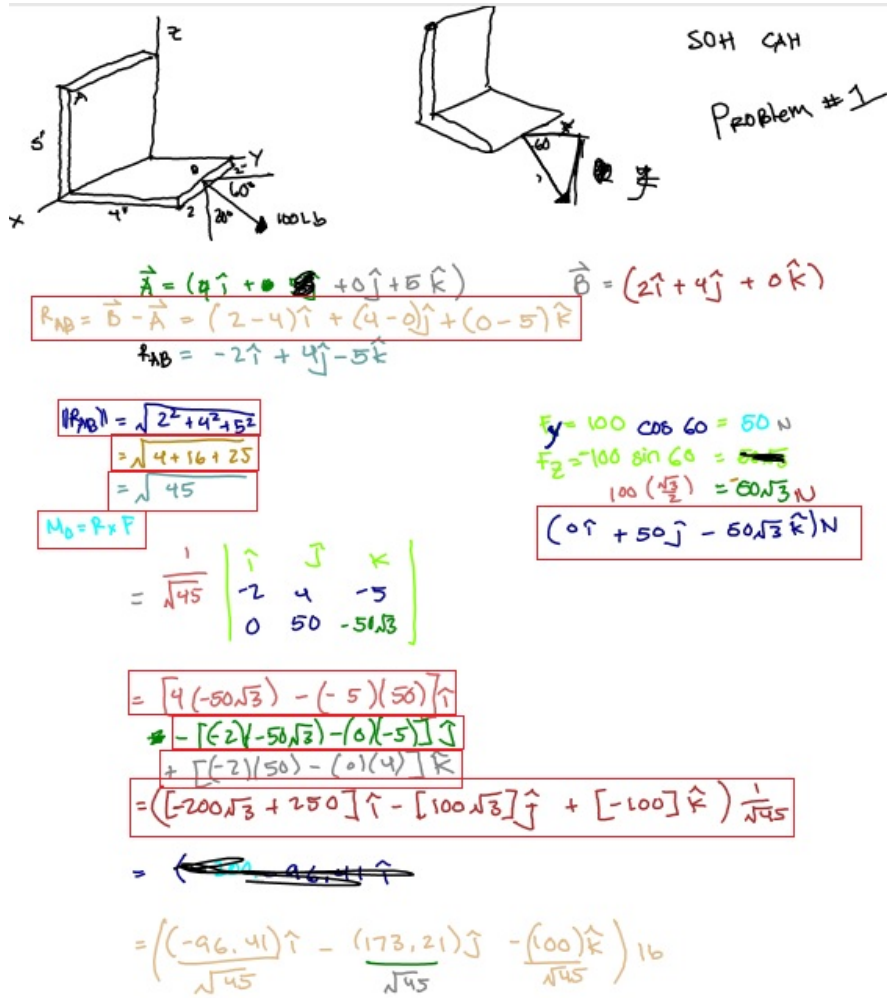


Figure B.3: A red box is drawn around each perfectly recognized equation segment.

$$\sum F_y = 0$$

$$B_y + A_y - 200 + 100 - P \sin 20 = 0$$

$$| A_y = 113 \text{ Nm.}$$

$$\sum F_z = 0$$

$$B_z + A_z + P \cos 20 = 0$$

$$104.86 + 29.96 + A_z = 0$$

$$| A_z = -134.8 \text{ Nm}$$

Figure B.4: A red box is drawn around each perfectly recognized equation segment.

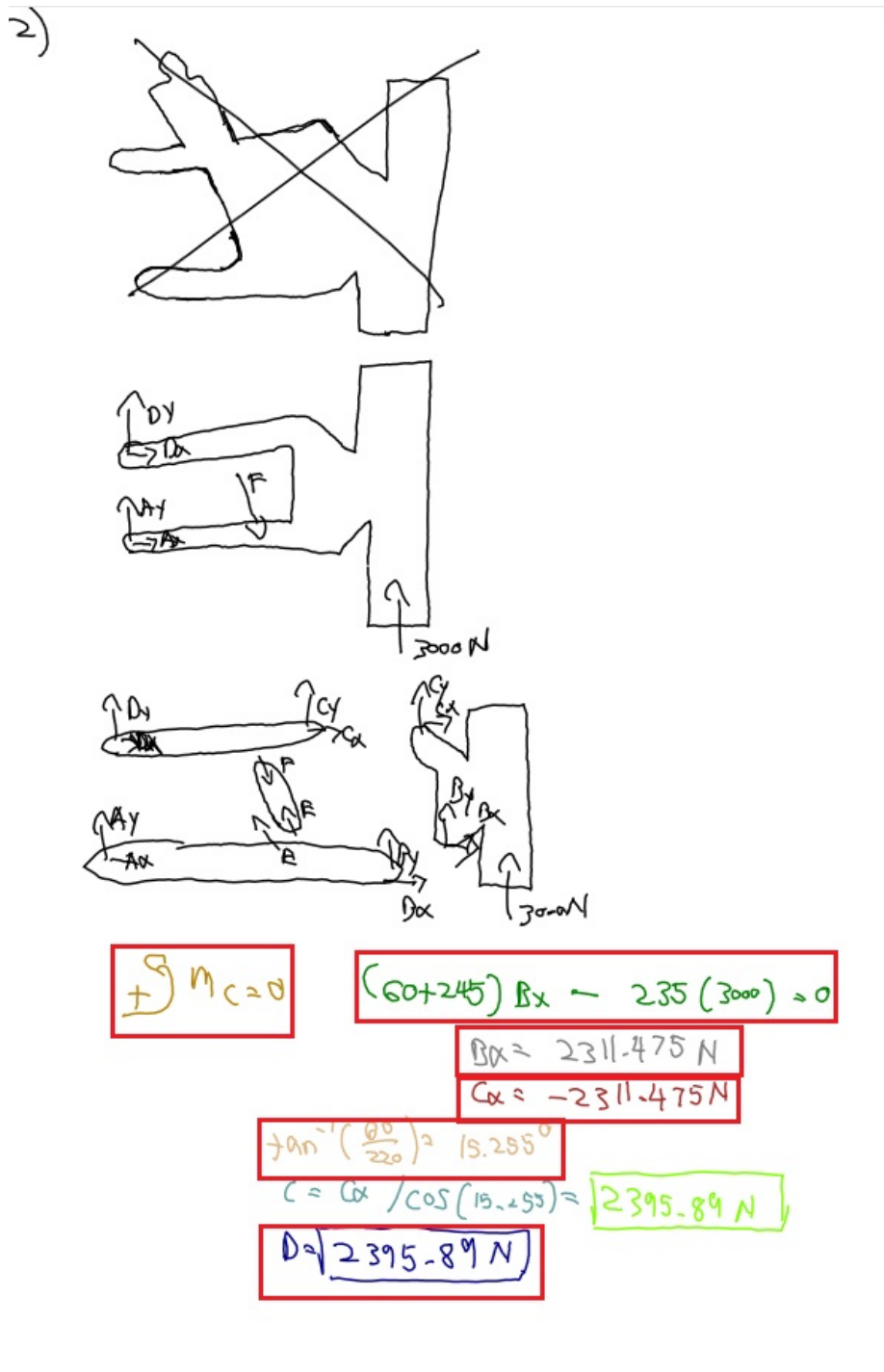


Figure B.5: A red box is drawn around each perfectly recognized equation segment.

Bibliography

- [1] Gated recurrent unit. <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>.
- [2] Microsoft ink recognizer. <https://docs.microsoft.com/en-us/windows/win32/tablet/ink-recognition>.
- [3] Christine Alvarado and Randall Davis. Dynamically constructed bayes nets for multi-domain sketch understanding. In *ACM SIGGRAPH 2006 Courses*, page 32. ACM, 2006.
- [4] Christine Alvarado and Michael Lazzareschi. Properties of real-world digital logic diagrams. In *Pen-Based Learning Technologies, 2007. PLT 2007. First International Workshop on*, pages 1–6. IEEE, 2007.
- [5] Lisa Anthony and Jacob O Wobbrock. \$ n-protractor: A fast and accurate multi-stroke recognizer. In *Proceedings of Graphics Interface 2012*, pages 117–120. Canadian Information Processing Society, 2012.
- [6] Ahmad-Montaser Awal, Guihuan Feng, Harold Mouchere, Christian Viard-Gaudin, et al. First experiments on a new online handwritten flowchart database. *DRR*, 11:1–10, 2011.
- [7] Ahmad-Montaser Awal, Harold Mouchère, and Christian Viard-Gaudin. A global learning approach for an online handwritten mathematical expression recognition system. *Pattern Recognition Letters*, 35:68–77, 2014.
- [8] Randy Elliot Bennett. Formative assessment: A critical review. *Assessment in Education: principles, policy & practice*, 18(1):5–25, 2011.
- [9] Akshay Bhat and Tracy Hammond. Using entropy to distinguish shape versus text in hand-drawn diagrams. In *IJCAI*, volume 9, pages 1395–400, 2009.
- [10] John Biggs. Assessment and classroom learning: a role for summative assessment? *Assessment in Education: Principles, Policy & Practice*, 5(1):103–110, 1998.

- [11] Christopher M Bishop, Markus Svensen, and Geoffrey E Hinton. Distinguishing text from graphics in on-line handwritten ink. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 142–147. IEEE, 2004.
- [12] Rachel Blagojevic, Beryl Plimmer, John Grundy, and Yong Wang. Using data mining for digital ink recognition: Dividing text and shapes in sketched diagrams. *Computers & Graphics*, 35(5):976–991, 2011.
- [13] Carol Boston. The concept of formative assessment. *Practical Assessment, Research, and Evaluation*, 8(1):9, 2002.
- [14] Martin Bresler, Truyen Van Phan, Daniel Prusa, Masaki Nakagawa, and Václav Hlavác. Recognition system for on-line sketched diagrams. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 563–568. IEEE, 2014.
- [15] Salman Cheema, Sumit Gulwani, and Joseph LaViola. Quickdraw: improving drawing experience for geometric diagrams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1037–1064. ACM, 2012.
- [16] François Chollet et al. Keras. <https://keras.io>, 2015.
- [17] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in neural information processing systems*, pages 1504–1512, 2015.
- [18] Ruwanee De Silva, David Tyler Bischel, WeeSan Lee, Eric J Peterson, Robert C Calfee, and Thomas F Stahovich. Kirchhoff’s pen: a pen-based circuit analysis tutor. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 75–82, 2007.
- [19] Adrien Delaye and Kibok Lee. A flexible framework for online document segmentation by pairwise stroke distance learning. *Pattern Recognition*, 48(4):1197–1210, 2015.
- [20] Adrien Delaye and Cheng-Lin Liu. Contextual text/non-text stroke classification in on-line handwritten notes with conditional random fields. *Pattern Recognition*, 47(3):959–968, 2014.
- [21] Ashish Dutt, Maizatul Akmar Ismail, and Tutut Herawan. A systematic review on educational data mining. *IEEE Access*, 5:15991–16005, 2017.
- [22] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.
- [23] Guihuan Feng, Christian Viard-Gaudin, and Zhengxing Sun. On-line hand-drawn electric circuit diagram recognition using 2d dynamic programming. *Pattern Recognition*, 42(12):3215–3223, 2009.
- [24] G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

- [25] Tracy Hammond. *Learning Through the Lens of Sketch*, pages 301–341. Springer International Publishing, Cham, 2017.
- [26] J. Herold and T. F. Stahovich. The 1¢ recognizer: A fast, accurate, and easy-to-implement handwritten gesture recognition technique. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pages 39–46, Goslar Germany, Germany, 2012. Eurographics Association.
- [27] James Herold and Thomas F Stahovich. Speedseg: A technique for segmenting pen strokes using pen speed. *Computers & Graphics*, 35(2):250–264, 2011.
- [28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [29] Heloise Hwawen Hse and A. Richard Newton. Recognition and beautification of multi-stroke symbols in digital ink. *Comput. Graph.*, 29(4):533–546, August 2005.
- [30] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.
- [31] Emanuel Indermühle, Horst Bunke, Faisal Shafait, and Thomas Breuel. Text versus non-text distinction in online handwritten documents. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 3–7. ACM, 2010.
- [32] Emanuel Indermühle, Volkmar Frinken, and Horst Bunke. Mode detection in online handwritten documents using blstm neural networks. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 302–307. IEEE, 2012.
- [33] Emanuel Indermühle, Marcus Liwicki, and Horst Bunke. Iamondo-database: an online handwritten document database with non-uniform contents. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 97–104. ACM, 2010.
- [34] K Jain, Anoop M Namboodiri, and Jayashree Subrahmonia. Structure in on-line documents. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 844–848. IEEE, 2001.
- [35] Levent Burak Kara and Thomas F Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers & Graphics*, 29(4):501–517, 2005.
- [36] Levent Burak Kara and Thomas F. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Comput. Graph.*, 29(4):501–517, August 2005.
- [37] Youngwook Paul Kwon and Sara McMains. An automated grading/feedback system for 3-view engineering drawings using ransac. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale, L@S '15*, pages 157–166, New York, NY, USA, 2015. ACM.

- [38] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.
- [39] WeeSan Lee, Ruwanee de Silva, Eric J Peterson, Robert C Calfee, and Thomas F Stahovich. Newton’s pen: A pen-based tutoring system for statics. *Computers & Graphics*, 32(5):511–524, 2008.
- [40] WeeSan Lee, Levent Burak Kara, and Thomas F Stahovich. An efficient graph-based recognizer for hand-drawn symbols. *Computers & Graphics*, 31(4):554–567, 2007.
- [41] Aurélie Lemaitre, Harold Mouchere, Jean Camillerapp, and Bertrand Coüasnon. Interest of syntactic knowledge for on-line flowchart recognition. In *Graphics recognition. new trends and challenges*, pages 89–98. Springer, 2013.
- [42] Han-Lung Lin. *Estimating Student Competence in Engineering Statics From a Lexical Analysis of Handwritten Equations*. PhD thesis, UC Riverside, 2014.
- [43] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.
- [44] Trevor Nelligan, Seth Polsley, Jaideep Ray, Michael Helms, Julie Linsey, and Tracy Hammond. Mechanix: A sketch-based educational interface. In *Proceedings of the 20th International Conference on Intelligent User Interfaces Companion*, IUI Companion ’15, pages 53–56, New York, NY, USA, 2015. ACM.
- [45] Tom Ouyang and Randall Davis. A visual approach to sketched symbol recognition., 01 2009.
- [46] Tom Y Ouyang and Randall Davis. A visual approach to sketched symbol recognition. In *IJCAI*, volume 9, pages 1463–1468, 2009.
- [47] Rachel Patel, Beryl Plimmer, John Grundy, and Ross Ihaka. Ink features for diagram recognition. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 131–138. ACM, 2007.
- [48] Brandon Paulson and Tracy Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 1–10. ACM, 2008.
- [49] Florian Perteneder, Martin Bresler, Eva-Maria Grossauer, Joanne Leong, and Michael Haller. cluster: Smart clustering of free-hand sketches on large interactive surfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 37–46. ACM, 2015.
- [50] Eric Jeffrey Peterson. *An efficient and general-purpose technique for grouping hand-drawn pen strokes into objects*. PhD thesis, UC Riverside, 2010.
- [51] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

- [52] Yuan Qi, Martin Szummer, and Thomas P Minka. Diagram structure recognition by bayesian conditional random fields. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 191–196. IEEE, 2005.
- [53] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [54] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–521–II–527 vol.2, June 2003.
- [55] D Royce Sadler. Formative assessment: Revisiting the territory. *Assessment in education: principles, policy & practice*, 5(1):77–84, 1998.
- [56] Susan C Schneider. ” paperless grading” of handwritten homework: Electronic process and assessment. 2014.
- [57] Michael Shilman and Paul Viola. Spatial recognition and grouping of text and graphics. In *Proceedings of the First Eurographics conference on Sketch-Based Interfaces and Modeling*, pages 91–95. Eurographics Association, 2004.
- [58] Michael Shilman, Zile Wei, Sashi Raghupathy, Patrice Simard, and David Jones. Discerning structure from freeform handwritten notes. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 60–65. IEEE, 2003.
- [59] Arjun Singh, Sergey Karayev, Kevin Gutowski, and Pieter Abbeel. Gradescope: A fast, flexible, and fair system for scalable assessment of handwritten work. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale, L@S ’17*, pages 81–88, New York, NY, USA, 2017. ACM.
- [60] Yale Song, Randall Davis, Kaichen Ma, and Dana L Penny. Balancing appearance and context in sketch interpretation. *arXiv preprint arXiv:1604.07429*, 2016.
- [61] Thomas F Stahovich. Segmentation of pen strokes using pen speed. In *AAAI Fall Symposium Series*, pages 21–24, 2004.
- [62] Thomas F Stahovich and Hanlung Lin. Enabling data mining of handwritten coursework. *Computers & Graphics*, 57:31–45, 2016.
- [63] Thomas F Stahovich, Eric J Peterson, and Hanlung Lin. An efficient, classification-based approach for grouping pen strokes into objects. *Computers & Graphics*, 42:14–30, 2014.
- [64] Maddalena Taras. The use of tutor feedback and student self-assessment in summative assessment tasks: Towards transparency for students and for tutors. *Assessment & Evaluation in Higher Education*, 26(6):605–614, 2001.

- [65] Maddalena Taras. Summative assessment: The missing link for formative assessment. *Journal of Further and Higher Education*, 33(1):57–69, 2009.
- [66] Timothy S Van Arsdale and Thomas Stahovich. Does neatness count? what the organization of student work says about understanding. In *American Society for Engineering Education*. American Society for Engineering Education, 2012.
- [67] Truyen Van Phan and Masaki Nakagawa. Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents. *Pattern Recognition*, 51:112–124, 2016.
- [68] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.
- [69] Jiaming Wang, Jun Du, Jianshu Zhang, and Zi-Rui Wang. Multi-modal attention network for handwritten mathematical expression recognition.
- [70] A. Wolin, B. Paulson, and T. Hammond. Sort, merge, repeat: An algorithm for effectively finding corners in hand-sketched strokes. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, pages 93–99, New York, NY, USA, 2009. ACM.
- [71] Jun-Yu Ye, Yan-Ming Zhang, and Cheng-Lin Liu. Joint training of conditional random fields and neural networks for stroke classification in online handwritten documents. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 3264–3269. IEEE, 2016.
- [72] Jun-Yu Ye, Yan-Ming Zhang, Qing Yang, and Cheng-Lin Liu. Contextual stroke classification in online handwritten documents with graph attention networks. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 993–998. IEEE, 2019.
- [73] Jianshu Zhang, Jun Du, and Lirong Dai. A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 902–907. IEEE, 2017.
- [74] Jianshu Zhang, Jun Du, and Lirong Dai. Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia*, 21(1):221–233, 2018.