SCISSOR: Simple Circuit Component Recogniser Using Convolutional Neural Networks

Prepared by:
Mpilonhle Ngcoya [NGCMPI002]
Bonga Njamela [NJMLUN002]

Prepared for:
EEE4114F
Department of Electrical Engineering
University of Cape Town

May 16, 2024

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

[illegible]	May 16, 2024
Mpilonhle Ngcoya	Date
[illegible]	May 16, 2024
Bonga Njamela	Date

Abstract

This report explores the development and evaluation of methods for recognizing hand-drawn electrical component symbols, leveraging both manual feature extraction techniques and automated methods such as the Histogram of Oriented Gradients (HOG). Recognizing hand-drawn diagrams is crucial in various fields including healthcare, education, and engineering due to their widespread use in visual communication. The study addresses the inherent challenges posed by variations in symbol scaling, text inclusion, and diagram complexity, aiming to enhance the accuracy and reliability of recognition systems.

The methodology involved two primary approaches: manually extracting features such as aspect ratio, fill percentage, and continuity, and employing the HOG method to derive gradient-based features. The dataset comprised binarized images of various electrical component symbols, which were processed and classified using k-nearest neighbors (kNN) and SVM-based algorithms. The manual method relied on traditional image processing techniques, while the HOG method utilized advanced edge and gradient detection to form robust feature vectors. These approaches were implemented and their performance was compared through classification accuracy and confusion matrix analysis.

Results indicated that the HOG method significantly outperformed the manual feature extraction approach, providing better differentiation between classes and higher overall accuracy. The confusion matrix for the HOG-based classifier demonstrated fewer misclassifications, showcasing the method's ability to handle the variability in hand-drawn symbols effectively. This study underscores the importance of using sophisticated image processing techniques in conjunction with machine learning algorithms to improve the performance of hand-drawn diagram recognition systems.

Contents

List of Figures **vi**

1 Introduction **1**

1.1 Background 1
1.2 Objectives 2
1.3 System Requirements 2
1.4 Scope & Limitations 2
1.4.1 Scope 2
1.4.2 Limitations 2
1.5 Report Outline 3

2 Literature Review **4**

2.1 Introduction 4
2.2 Hand-Drawn Diagram Recognition Approaches 4
2.3 Hand-Drawn Diagram Pre-Processing and Feature Extraction Methods 6
2.4 Selected Features Symbol Recognition Methods 7
2.5 Classification Techniques for Electronic Component Symbols Recognition for Hand-Drawn Sketches 8
2.6 Conclusion 10

3 Proposed Methodology **12**

3.1 Dataset 12
3.2 Pre-Processing and Feature Extraction 13
3.2.1 Histogram of Oriented Gradients (HOG) 13
3.2.2 Manual Line Tracking 15
3.3 Training Methodology 16
3.3.1 Support Vector Machine Classifier 16
3.4 Detection of Circuit Components 16
3.4.1 Histogram of Oriented Gradients (HOG) 16
3.4.2 Manual Line Tracking 16
3.5 Terminal Recognition 17
3.5.1 Histogram of Oriented Gradients (HOG) 17
3.5.2 Manual Line Tracking 17
3.6 Node Recognition 17
3.6.1 Histogram of Oriented Gradients (HOG) 17
3.6.2 Manual Line Tracking 17
3.7 Circuit Schematic Generation 18

3.7.1 Histogram of Oriented Gradients (HOG) 18
3.7.2 Manual Line Tracking 18

4 Results and Discussion **19**
4.1 Training 19
4.1.1 Data Pre-processing and HOG Feature Extraction 19
4.1.2 SVM Classifier Training 20
4.2 Testing SVM Hand-drawing Classifier on Multiple Classes 20
4.2.1 Manual Line Tracking 21

5 Conclusions **23**

6 Recommendations **24**
6.1 Enhance Image Pre-processing Techniques 24
6.2 Advance Feature Extraction Methods 24
6.3 Data Diversity 24
6.4 Model Selection and Optimization 24

Bibliography **25**

List of Figures

2.1 Examples of hand-drawings in health-care, education, business and human-computer interactions from left to right [1]. 5
2.2 The wave shaped sketch on the left is from the Parkinson's disease negative class, while the one on the left belongs to the Parkinson's positive dataset.[2]. 7
2.3 Showing spiral shape data from the Parkinson's negative class (left) and the Parkinson's positive class (right).[2]. 7
2.4 Training and validation accuracy of max-pooling and average pooling achieved using CNN-based feature extraction methods. [3]. 9
2.5 Rachalas proposed framework for component, node and terminal recognition in hand-drawn electronic circuit diagrams.[4]. 10
2.6 Confusion matrix for classification algorithm which takes in screenshots of hand-drawn circuit diagrams.[4]. 10

3.1 Circuit component symbols of interest. 12
3.2 Examples of hand-drawn components in the SolvaData dataset. 13
3.3 Histogram of Oriented Gradients algorithm in the `sklearn` package uses a 64×128 window to extract features. The grid is further divided into overlapping blocks. 14
3.4 Convolution with these gradient selective kernels yields an output that can be used to compute the gradient magnitude. 15
3.5 Ground truth images for training the SVM classifier. 16

4.1 Resulting images from applying HOG feature extraction to find maximum gradients and orientations of the `ammeter` and `voltmeter` templates. 19
4.2 Labels of the classes identified by the SVM classifier. 20
4.3 Classification of hand-drawn electronic components using a standard SVM with a sigmoid kernel. 21
4.4 Classification of Manual Line Tracking of RLC components 22
4.5 Confusion Matrix of Manual Line Tracking of RLC components 22

Chapter 1

Introduction

A baby learns to crawl, walk and then run. We are in the crawling stage when it comes to applying machine learning.

—Dave Waters

The aim of this project is to create a Machine Learning-based classification model that is capable of recognizing and predicting hand-drawn figures. The model is designed to facilitate the design process in CAD environments by providing a more natural and efficient means of input. The project involves developing a small, custom image dataset for classification. Throughout history, hand-drawn diagrams have been used for conveying essential tools and designs. They enable individuals to illustrate objects, however, with moving to computers, such designs have become digitized, necessitating efficient methods at identifying the objects. The task of recognizing some objects is quite difficult due to factors such as text incorporation, symbol scaling, and the requirement to interpret high-level diagrammatic scopes. Despite these challenges, advancements in machine learning (ML) and image processing techniques have opened new avenues for automating HDDR.

1.1 Background

Hand-drawn diagrams have been a crucial tool in various fields, including architecture, engineering, healthcare, education, and business. However, the use of digital tools like KiCAD, AutoCAD, and LTSpice has revolutionized the design process, allowing engineers to create and simulate designs with varying shapes and properties. Despite these advantages, manual drawing can be time-consuming and complex. Humans naturally rely on hand-drawn sketches for illustrating objects and spatial relationships. Consequently, there is a growing interest in developing systems that can recognize and interpret hand-drawn diagrams to streamline the design process. Existing HDDR techniques face challenges such as incorporating text into symbols, handling symbol scaling, and interpreting the overall scope of diagrams. Recent advances in machine learning offer promising solutions for these challenges. Traditional image processing methods, such as noise reduction and edge detection, may not be sufficient for handling the variability and complexity of hand-drawn sketches. Deep learning models like CNNs provide robust tools for feature extraction and classification, enabling more accurate recognition of hand-drawn diagrams.

1.2 Objectives

1. `Data Collection`: Develop a small, custom dataset of hand-drawn circuit component symbols, ensuring the inclusion of diverse and representative samples for training, validation, and testing.
2. `Model Development`: Implement and compare the performance of various ML algorithms, including both traditional and deep learning models, to determine the most effective approach for the dataset.
3. `Pre-Processing and Feature Extraction`: Apply appropriate pre-processing techniques and feature extraction methods, such as Histogram of Oriented Gradients (HOG) and line tracking, to enhance the quality and informativeness of the dataset.
4. `Model Evaluation`: Evaluate the developed models using standard metrics to ensure their reliability and applicability in real-world scenarios.
5. `System Integration`: Integrate the ML model into a 2D CAD tool, providing a proof of concept for how the system can assist in the engineering design process.

1.3 System Requirements

1. `Hardware`: A computer with sufficient processing power and memory to handle ML model training and image processing tasks. A graphics processing unit (GPU) is recommended for training deep learning models.
2. `Software`: Python programming language. Libraries for ML and image processing, such as TensorFlow, Keras, scikit-learn, and OpenCV. CAD software for testing and integration, such as KiCAD or AutoCAD. Tools for data annotation and preprocessing, such as LabelImg or similar.

1.4 Scope & Limitations

1.4.1 Scope

1. `Dataset Size`: The dataset will consist of a limited number of classes (3-5) and images (50-100 per class) to ensure manageability and focus on the core objectives.
2. `Model Comparison`: The project will compare the performance of different ML algorithms on a small dataset, providing insights into the effectiveness of various approaches.
3. `Feature Extraction`: The study will explore traditional feature extraction methods alongside deep learning techniques to identify the most suitable approach for the dataset.

1.4.2 Limitations

1. `Dataset Constraints`: The small size of the dataset may limit the generalizability of the model to more extensive and diverse datasets.

2. `Computational Resources`: The availability of computational resources, particularly GPUs, may impact the training time and performance of deep learning models.
3. `Model Complexity`: Complex models may not perform optimally on small datasets, leading to potential overfitting issues.
4. `Integration Challenges`: Integrating the ML model into existing CAD tools may require significant adjustments and validation to ensure seamless operation.

1.5 Report Outline

The report is structured as follows:

Chapter 1 provides an overview of the project, discussing background and high level concept of the project.

Chapter 2 delves into the extensive body of literature and research concerning hand-drawn diagram recognition (HDDR) and its integration into engineering design processes. This comprehensive review offers insights into a wide array of methodologies, techniques, and algorithms utilized for the recognition and interpretation of hand-drawn sketches. From traditional approaches to advanced machine learning and deep learning techniques, this chapter provides a thorough examination of the strategies employed to enhance the efficiency and accuracy of recognizing hand-drawn diagrams in various domains of engineering design.

Chapter 3 presents a detailed methodology for developing a machine learning-based classification model for predicting hand-drawn circuit component symbols. It covers challenges in data preparation, pre-processing, feature extraction techniques like Histogram of Oriented Gradients (HOG) and manual line tracking, training methodology using a Support Vector Machine (SVM) classifier, and comparison of HOG-based methods with manual line tracking approaches.

Chapter 4 discusses the results of the developed machine learning-based classification model, including data pre-processing, HOG feature extraction, SVM classifier training, and identified labels.

Chapter 6 offers key recommendations to enhance the SCISSOR algorithm's effectiveness in recognizing hand-drawn electronic components. These include improving image pre-processing techniques, advancing feature extraction methods, diversifying the dataset, and conducting systematic hyperparameter tuning. These recommendations aim to refine the algorithm and improve its performance across various scenarios.

Chapter 2

Literature Review

2.1 Introduction

Applications of Machine Learning in engineering use the understanding of processes to construct models that fulfill requirements. Engineers use drawing and sketching software tools to expedite the design process. Computed Aided Design tools such as KiCAD, AutoCAD and LTSpice have gained popularity in Electrical engineering disciplines as they allow for creating design simulations of objects with different shapes and physical properties. Typically, CAD tools allow for shapes to be drawn by dragging the mouse from point to point across the view-port of a screen. This can prove to be a time-consuming and complex task for many users since human beings use hand-drawn diagrams as the primary means of illustrating objects and spatial information. Key features of the process of creating hand-drawn diagrams can be collected to classify illustrations and predict the shape that is being drawn.

The aim of this project is to design a Machine Learning-based classification model that can take hand-drawn figures to predict the shape being drawn. This system was intended to be used to augment 2D CAD tools in order to make the engineering design process more intuitive. The following examines different approaches and designs that have been used to predict hand-drawn characters and shapes, with focus on applications to digital design tools that use touchscreens to capture the data of the hand-drawn object.

2.2 Hand-Drawn Diagram Recognition Approaches

Human beings have depended on hand-drawn diagrams to portray complex concepts and designs for most of history that predates computers. Hand-drawings form a central part of visual communication in various industries including architectural design and engineering. This significance of diagrams is emphasised by an increase in the necessity for optimal strategies for identifying, interpreting and predicting the subject of hand-drawings. In their overview on hand-drawn diagram recognition (HDDR), Agrawal et al. [1] describe the applications of HDDR in health-care, education, business and organisations, as well as in human-computer interactions.

Agrawal et al. [1] argue that the computers find HDDR challenging because of factors such as the inclusion of text into symbols, symbol scaling and extension in all directions, and often requirements for interpreting the high level scope of the diagram.

After recognizing that the majority of HDDR techniques were directed at a specific type of diagram,

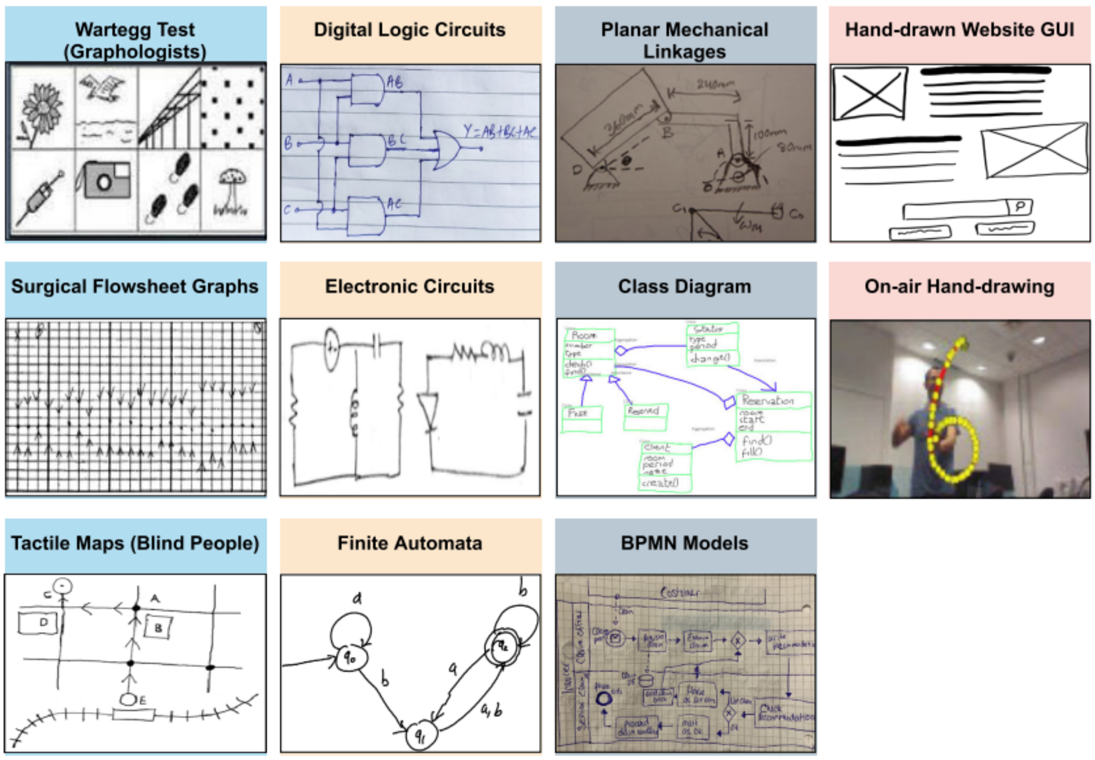


Figure 2.1: Examples of hand-drawings in health-care, education, business and human-computer interactions from left to right [1].

Dorothea Blostein [5], proposed six diagram-recognition processes for general purposes, namely:

1. Early processing which involves noise reduction and straightening of images known as deskewing
2. Segmentation to isolate symbols
3. Symbol recognition
4. Interpretation of spatial information
5. Identifying logical correspondance between symbols
6. Extracting meaning from the symbols

Blostein highlights that all the processes involved in HDDR require prior knowledge of notational conventions so as to preserve the integrity of the data [5]. As an example, Blostein suggests that the segmentation stage requires information about the border edges of symbols to interpret how they overlap [5]. Another example used to describe Blostein's six processes involves removing noise from a symbol. Blostein posits that extra caution needs to be taken so as not to confuse decimal points in mathematical applications and dots in music for noise [5].

In more recent times, the six-stage technique described by Blostein would be classified as an image processing technique. Agrawal posits that the shortcoming of image processing techniques when comes to HDDR is that typically, hand-drawn diagrams exhibit variations in stroke thickness and quality, which is not efficient for classifying more complex images. Alternatively, graph theory-based approaches represent hand-drawings as graphs such that recognition involves a comparison of the sketch to pre-selected templates [1].

Instead of using templates, Machine Learning algorithms such as k-nearest neighbors (kNN) and support vector machines (SVMs) offer more robust predictions using pre-defined models with parameters and hyperparameters that affect the accuracy of the model. Furthermore, Deep Learning algorithms use Convolutional, Recurrent, and Graphical Neural Networks (CNNs, RNNs and GNNs), to achieve more accurate results [1]. By analysing and sorting hand-drawn images, data can be organised more effectively into training, validation and testing sets that produce suitable predictions.

2.3 Hand-Drawn Diagram Pre-Processing and Feature Extraction Methods

Before applying pre-processing pipelines to hand-drawn diagrams, a selection of a suitable dataset is key to training the Machine Learning model. In their study of sketch-recognition using a pre-trained model, Noor et al., used the TU-Berlin sketch dataset having 250 classes [6]. The training set was comprised of 10 000 selected images. Likewise 5000 images were selected for validation and 5000 images were selected for the test set. The selected images were originally available in 1111 $\times$ 1111 dimensions which were read and converted into 224 $\times$ 224 dimensions [6]. This process of selecting the data sets for training, validation and testing is crucial for designing a model that accurately predicts the shape and scope of a hand-drawn diagram.

In 2019, Altun and Nooruldeen [7] proposed a stroke-based recognition model which was suitable for structures that can be considered *simple* [7] due to their use of variations of basic shapes such as rectangles, ellipses and triangles. Their recognition model, named SKETRACK, focused on text separation, symbol segmentation, feature extraction, classification and structural analysis [7]. To minimize feature dimensionality, the spectral clustering algorithm was used to cluster strokes based on p-distance and Euclidean distance [7]. The stroke data was stored in ink files. This data was normalised to rectify edges and noises [7]. Text data is also normalised at this stage of pre-processing to improve the performance of the text segmentation process to account for data that is in a non-canonical language [7]. After processing, the symbol recognition stage was performed to classify the subsets of strokes clusters and assign a symbol cluster to each subset [7].

Normalisation involves scaling the pixels to a standard range such as [0,1] or [-1,-1] [1]. Rescaling the pixels makes it possible to differentiate between different components of the image based on the illumination levels [1]. Normalisation is one of many pre-processing methods that can be implemented to organise data. In their study of early identification of Parkinson's Disease, Akter and Ferdib-Al-Islam successfully use image-resizing to prepare the data [2]. Images were resized into 200 $\times$ 200 shape and converted into grayscale format in order to make the comparison process more efficient [2]. The dataset of hand-drawn images that were used in their study contained hand-drawn spiral and wave shaped sketches as illustrated in figure **??**. Hand-drawn images from the Parkinson's disease positive class were expected to exhibit a shaky stroke.

The resized 200 $\times$ 200 grayscale images were used as inputs into a Histogram of Oriented Gradients (HOG) algorithm as it is considered to perform adequately in extracting features for applications in

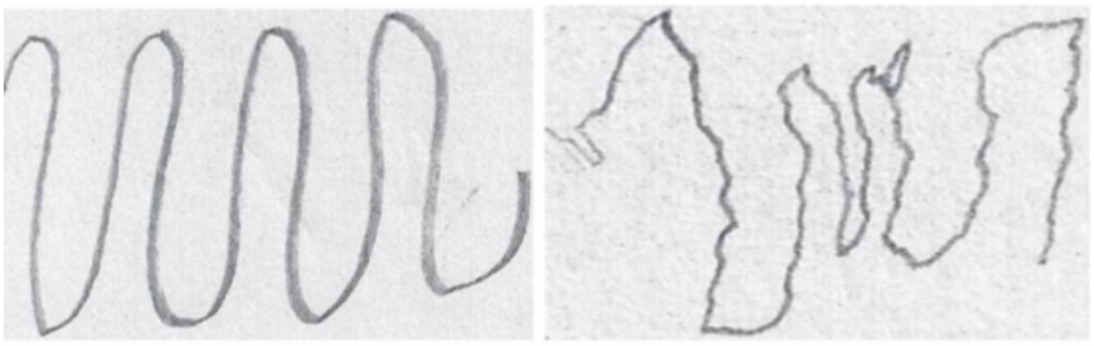


Figure 2.2: The wave shaped sketch on the left is from the Parkinson's disease negative class, while the one on the left belongs to the Parkinson's positive dataset.[2].

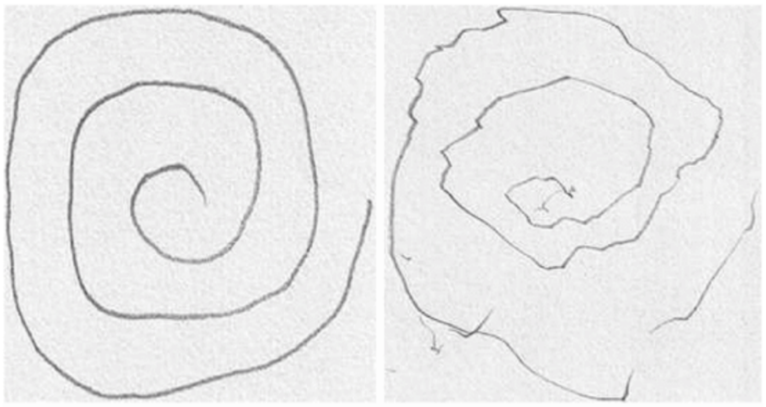


Figure 2.3: Showing spiral shape data from the Parkinson's negative class (left) and the Parkinson's positive class (right).[2].

biomedical engineering [2]. Akter and Ferdib-Al-Islam used HOG descriptors to portray basic shape information of objects that are present in an image [2].

Agrawal et al. argue that performing grayscale conversion on image can streamline pre-processing without decreasing image quality [1]. Another method which helps prepare data is image binarisation which sets a threshold to divide pixels into two groups of different contrast. These methods are suitable for images that depend on the pixel intensity. Other methods such as noise reduction focus on removing high frequency components known as salt-and-pepper noise [1]. Pavithra et al. successfully use Gaussian blurring to reduce noise image in their design of a hand-drawn electronic component recognition model which uses the Oriented FAST and Rotated BRIEF (ORB) feature detector [8]. The same prepared data is used to compare the performance of the ORB algorithm compared to the Scale Invariant Feature Transfrom (SIFT) algorithm which was proposed by Lowe for matching local features in an image [8, 9].

2.4 Selected Features Symbol Recognition Methods

In their paper, Noor et al. explains that the rise in the demand for sketch recognition patterns is because of an increase in the usage of touch screen devices such as tablets and smartphones [6]. Since users can draw on the screen, symbol recognition is used to classify sketches solely on the extracted features of the image [6]. Classification of sketches is considered to be a Supervised Learning algorithm since the labelled sets are pre-defined [6]. Features are individual attributes that can be measured or recorded about a hand-drawing event. Some features have more of a significant impact than others. This means that careful attention needs to be directed to selecting an ideal set of features to use for the symbol recognition algorithm.

Feature extraction techniques of interest to symbol recognition in engineering drawing applications include stroke analysis, contour analysis, and shape descriptors [1] that can be used with algorithms such as ORB. Stroke analysis was successfully implemented by Szwoch and Much to identify properties of sketches by thickness, direction and the intensity of the stroke [10]. Fang et al. successfully used shape descriptors on images with strokes groupings that have regular appearance [11]. Using this technique, along with CNN-based object detection, Fang et al, proposed DrawnNet to for offline hand-drawn diagram recognition [11]. These examples indicate that a sufficient dataset preparations are required before performing Machine Learning algorithms for recognising symbols.

Alternatively, deep learning-based feature extraction methods can be employed to learn features instead of defining them directly. CNNs are particularly effective in this regard. CNN models significantly simplify the learning process by integrating weights into substantially smaller kernel sizes, which reduces the number of weight parameters that need to be trained [3]. This is in contrast to conventional fully connected neural networks, where even small image patches can result in a massive amount of weight parameters to be trained, leading to more complex systems that require numerous training samples to prevent the overfitting problem [3].

In CNNs, convolutional neuron layers are the main building blocks. These layers utilize kernel matrices to extract features from input channels[12]. The learning procedure aims to determine kernel matrices that extract reliable and robust features to classify sketch images. The connection of neurons in a neural network is optimized by the back-propagation process, where connection weights train the kernel matrices [3].

Additionally, pooling layers play a vital role in reducing the dimensionality of features [3]. Pooling algorithms, such as max-pooling, average-pooling, and min-pooling, are used to merge adjacent elements in the output channels of convolution. In the proposed network, max-pooling with a kernel size of 2×2 is utilized, which takes the maximum value from the four adjacent elements of the input channel to create a single component in the output channel. The model achieves a validation accuracy of 95.05% using max-pooling, which is 4% higher than using average-pooling [3]. The performance of using max-pooling and average-pooling is also illustrated in figure **??**.

The activation function employed in Ali's proposed CNN model is the rectified linear unit (ReLU) [3]. ReLU is a non-saturating non-linear function that improves the classification performance and learning speed of CNN compared to saturating non-linear functions such as Sigmoid and TanH [3].

2.5 Classification Techniques for Electronic Component Symbols Recognition for Hand-Drawn Sketches

The recognition of hand-drawn electronic component symbols in diagrams poses a unique challenge in hand-drawn diagram recognition (HDDR). Various classification techniques have been employed to

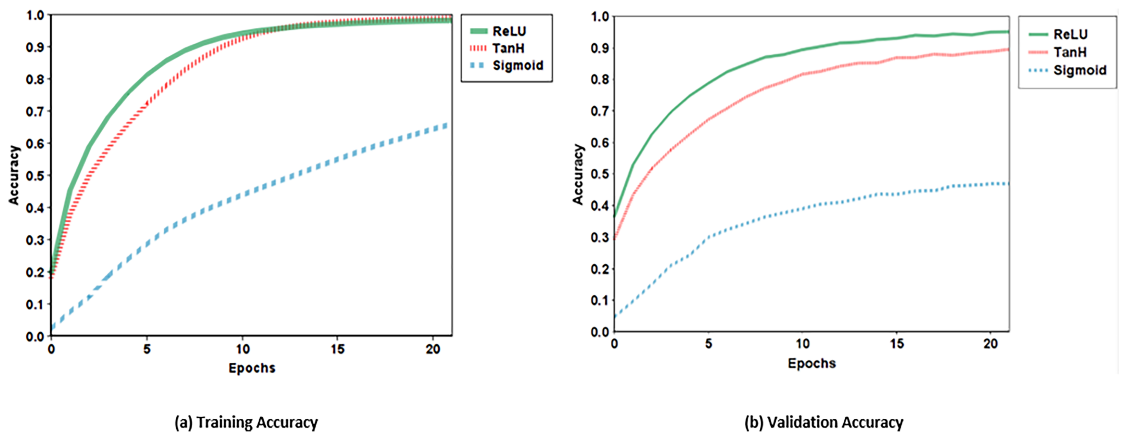


Figure 2.4: Training and validation accuracy of max-pooling and average pooling achieved using CNN-based feature extraction methods. [3].

address this challenge, depending on the complexity of the sketches and the available data. One of the classification methods commonly used in HDDR is Support Vector Machines (SVMs). SVMs are adept at classifying diagrams into binary or multiple categories by determining decision boundaries that separate different classes [13]. They have been widely used in HDDR tasks to classify hand-drawn symbols.

Ensemble methods, such as random forests, are capable of handling complex and multidimensional feature spaces. They combine the predictions of multiple base classifiers to improve classification performance [14]. kNN algorithms classify diagrams based on the feature spaces of their nearest neighbors, making them suitable for symbol recognition tasks. In their study of object recognition in hand-drawn images using machine learning ensembling techniques, Gupta el al. were able to achieve 82.34% accuracy using kNN with 5 nearest neighbours [14]. In the QuickDraw dataset, the kNN algorithm outperformed the SVM and Random Forest Classifier, producing a model precision of 83.4% and a f-score of 82.3% [14]. Other methods that have proven to produce high accuracy and precision include Naive Bayes classifiers which are more efficient for text-based classification problems in images [15].

CNNs excel at recognizing diagrams from images by learning hierarchical features directly from input photos. Rachala and Panicker proposed a framework for circuit component detection, node and terminal recognition and schematic generation [4] as illustrated in Fig. 2.5.

Rachala and Panicker trained this CNN-based object detection model to identify and detect more than one specific object. To extract information about the terminals, the system used two binary images which were generated from the output of the CNN-based object detection model [4]. The second binary image was developed using adaptive thresholding of the input image [4]. This method achieved an accuracy of 92% in node recognition and an accuracy of 86% in component recognition, on a database comprising of 107 nodes and 449 components. The confusion matrix illustrating the accuracy of the proposed framework is shown Fig. 2.6 [4].

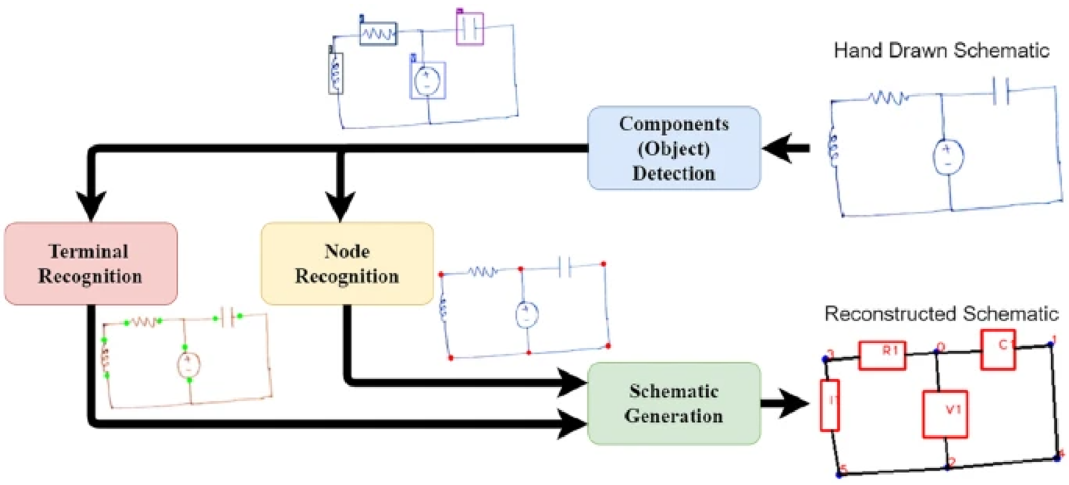


Figure 2.5: Rachalas proposed framework for component, node and terminal recognition in hand-drawn electronic circuit diagrams.[4].

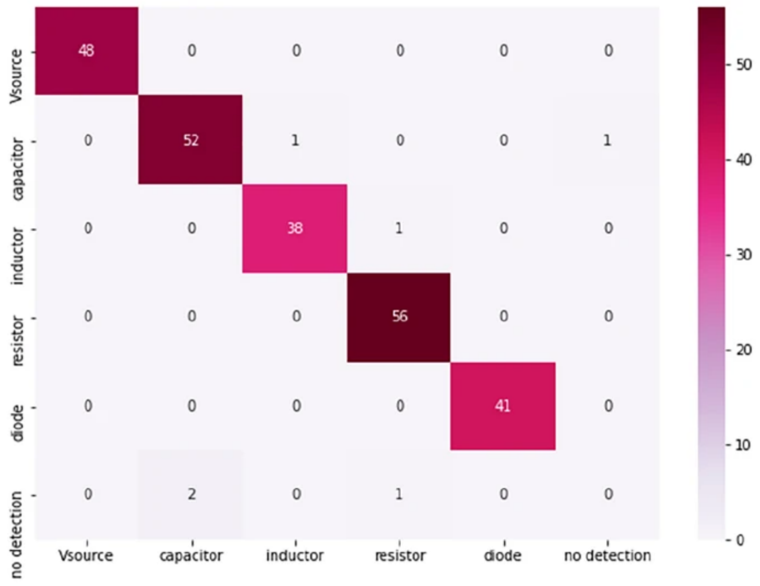


Figure 2.6: Confusion matrix for classification algorithm which takes in screenshots of hand-drawn circuit diagrams.[4].

2.6 Conclusion

In exploring the literature surrounding hand-drawn diagram recognition (HDDR) and its application in engineering design, it becomes evident that a variety of techniques and methodologies have been developed to address this complex task. From the early stages of understanding hand-drawn diagrams to the implementation of advanced classification and recognition algorithms, researchers have made significant strides in improving the accuracy and efficiency of systems designed to interpret hand-drawn sketches.

Historically, hand-drawn diagrams have served as a primary means of visual communication, spanning various industries such as architecture and engineering. As technology has evolved, so too have the methods for recognizing and interpreting these diagrams. Traditional approaches, such as those outlined by Blostein, focused on discrete stages of recognition, from noise reduction to symbol interpretation. While effective in their own right, these methods often struggled with the variability and complexity inherent in hand-drawn sketches.

The emergence of machine learning and deep learning techniques has revolutionized the field of hand-drawn diagram recognition. By leveraging algorithms such as k-nearest neighbors (kNN), support vector machines (SVMs), and convolutional neural networks (CNNs), researchers have been able to achieve remarkable accuracy in identifying hand-drawn symbols and shapes. These methods not only offer superior performance but also exhibit greater flexibility in handling the nuances and intricacies of hand-drawn diagrams.

Pre-processing and feature extraction play crucial roles in preparing hand-drawn data for classification. Techniques such as image resizing, normalization, and noise reduction help streamline the data and enhance the performance of machine learning models. Additionally, advanced feature extraction methods, including stroke analysis, contour analysis, and shape descriptors, provide valuable insights into the structure and characteristics of hand-drawn sketches.

Classification techniques vary depending on the complexity of the symbols and the nature of the data. Support vector machines, ensemble methods like random forests, and k-nearest neighbors algorithms have all been successfully employed in symbol recognition tasks. Furthermore, deep learning approaches, particularly convolutional neural networks, offer unparalleled performance in recognizing hand-drawn diagrams from images.

Looking ahead, the integration of machine learning and deep learning techniques into engineering design tools holds tremendous potential for improving the efficiency and intuitiveness of the design process. By augmenting 2D CAD tools with machine learning-based classification models, engineers can streamline the creation of complex designs and prototypes. As research in this field continues to advance, we can expect even greater innovations in hand-drawn diagram recognition, paving the way for a new era of design automation and optimization.

Chapter 3

Proposed Methodology

3.1 Dataset

A hand-drawn sketch was defined as composing of circuit component symbols. Due to the nature of their shapes and frequent usage, the primary components that were considered are shown in Fig. 3.1. Components such capacitors and cells (batteries), as well as resistors and inductors, were expected to have similar characteristics describing their shape, implying that carefull consideration had to be taken in preparing the data.

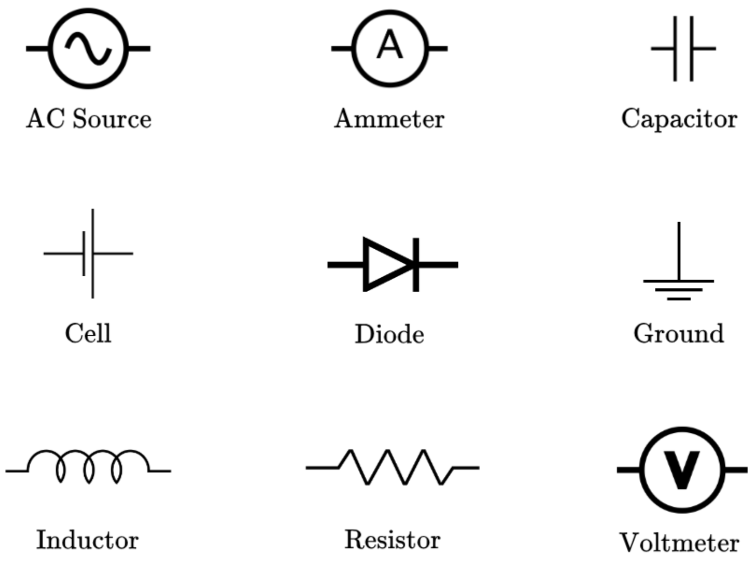
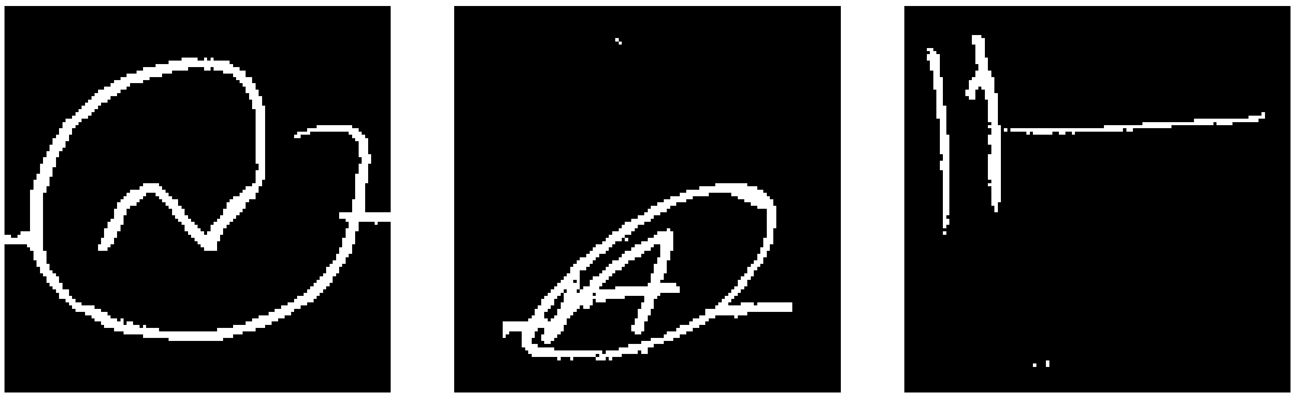


Figure 3.1: Circuit component symbols of interest.

The system used the SolvaDataset consisting of 2952 binary images of electrical circuit component symbols. The dataset consisted of 15 directories corresponding to a component symbol, however, the only symbols considered from this dataset were as illustrated in Fig. 3.1 above. Each directory in the selected data subset contained 200 120×120 bitmap image files (`.bmp`). Each file was assumed to include a single component sketched with a white stroke on a black background. An example of a hand-drawn sketch of an AC source, an ammeter and capacitor from the SolvaDaset set are shown in Fig. 3.2.

Based on the examples above, the following key observation can be made about the hand-drawn data:

- A symbol is may or may not be centered in the 120×120 window. In other words, the relative



(a) Hand-drawn AC source. (b) Hand-drawn ammeter. (c) Hand-drawn apacitor.

Figure 3.2: Examples of hand-drawn components in the SolvaData dataset.

spatial location of the edges of a symbol is not constant.

- The size and scale of symbols is not consistent among sketches in the dataset.
- The contours of circular components are not always connected or they may be connected to other parts of the symbol, as shown in 3.2a.
- Component sketches may or may not include canonical text symbols
- The length of lines that are expected to be equal are not always sketched as having the same distance. This is the case for the capacitor in 3.2c which can be misconstrued as the symbol for a cell.
- Circuit symbols have lines extending out horizontally to illustrate the wire input and output of the component.
- Sketch data may include salt-and-pepper noise which appears as white does that are not part of the edges of the circuit component hand-drawing.

Using binarized images was expected to streamline the pre-processing stage by eliminating the need to convert the sketch to grayscale. Images also have the same size, i.e. 120×120, allowing for easier application of algorithms that would have required an implementation of resizing techniques. The following section details the extraction of features from the sketches in the dataset.

3.2 Pre-Processing and Feature Extraction

3.2.1 Histogram of Oriented Gradients (HOG)

The Histogram of Oriented Gradients (HOG) feature descriptor is applied extensively in computer vision and image processing applications for extracting information about objects in an image . HOG extracts information about edge magnitudes and edge orientation by counting events of gradient orientation in a windowed region of an image [16]. The method proposes the use of HOG for obtaining a feature vector for each training sample image. The `sklearn` library in Python performs the HOG algorithm by automatically resizing the image to a `(64, 128)` shape [16]. This is so that a detection window of 64×128 pixels can be used before dividing the image into smaller parts. The divisions have a dimension of 8×16 cells with 50% overlap so $7 \times 15 = 105$ blocks in total [16]. The library takes the 64 output gradient vectors of each 8×8 pixel cell and transfers them to a 9-bin histogram [16]. The HOG feature extraction logic is illustrated in Fig. 3.3 where red and blue blocks, each with 2×2 cells and 8×8 pixels.

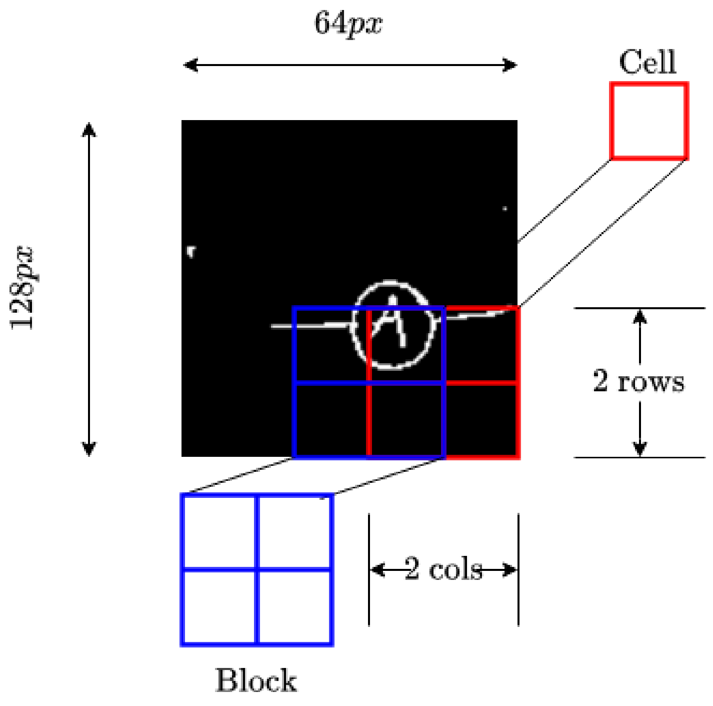


Figure 3.3: Histogram of Oriented Gradients algorithm in the `sklearn` package uses a 64×128 window to extract features. The grid is further divided into overlapping blocks.

After resizing the images, the gradient in the x and y directions are determined using convolution by a gradient kernel as illustrated in Fig. ??. The result from the convolution of each block with both kernels is used to calculate the magnitude of the gradients. The magnitude of the gradient $\|\nabla f\|$ can be determined using

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{3.1}$$

The orientation of the gradient was associated with an angle, θ, given by the ratio of the rate of changes in both directions, i.e.

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \tag{3.2}$$

The angle was taken as zero along the horizontal axis and increases between 0° and 180°.

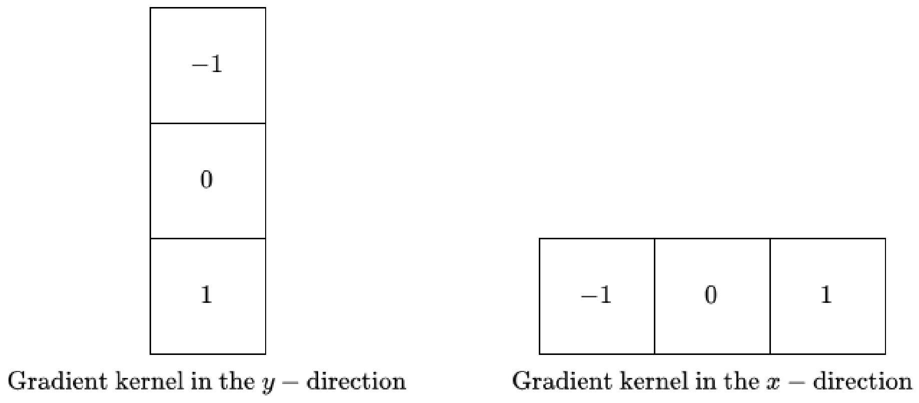

−1
0
1
Gradient kernel in the $y-$ direction
−1
0
1
Gradient kernel in the $x-$ direction

Figure 3.4: Convolution with these gradient selective kernels yields an output that can be used to compute the gradient magnitude.

The `hog()` function from the `sklearn` library takes in 6 parameters [16]:

1. `image`: training dataset image.
2. `orientations`: define the number, h of histogram bins. Defaults to a 9-bin histogram.
3. `pixels_per_cell`: defaults to 8×8 pixels in a cell.
4. `cells_per_block`: number of cells per block is 2×2 by default.
5. `visualize`: set to `True` to display output.
6. `multichannel`: set to `True` to tell the function that the last dimension is considered as a color channel instead of a spatial channel.

3.2.2 Manual Line Tracking

There are various features that one can find if you track the line of the actual component. It might be useful to convert all the pixels of the images to coordinates. This way we are then able to look at how the features of a specific component behaves verses another. From here we are able to look for specific features that the lines will follow when reconstructing the points. Looking at how it is drawn. From this you are able to get things such a:

1. `Aspect Ratio`: By looking at how the aspect ratio of one component might differ from another, for instance are some components more square drawn or are they more triangular drawn
2. `Fill %`: We can look at how much does each component fill up its associated drawing space
3. `Continuity`: Are the components continuous over the horizontal and vertical axis, some might require a spacing inside the component
4. `Deviation`: When drawing the component, is there a point whereby the line begins to deviate. Also how is that deviation?

3.3 Training Methodology

3.3.1 Support Vector Machine Classifier

The proposed classification scheme followed a Supervised Learning scheme since it used templates to train a sigmoid Support Vector Machine (SVM) model. Three of the samples of the templates that were used to train the SVM classifier are displayed in Fig. 3.5. The training dataset used the assumptions that were made about each image, namely:

- Each template image size is 120 × 120 px.
- An image consists of a component drawn with a white stroke on a black background.
- Templates images may or may not include text.
- Each component contains nodes and wires represented as horizontal lines.
- Template images do not include salt-and-pepper (high frequency) noise.

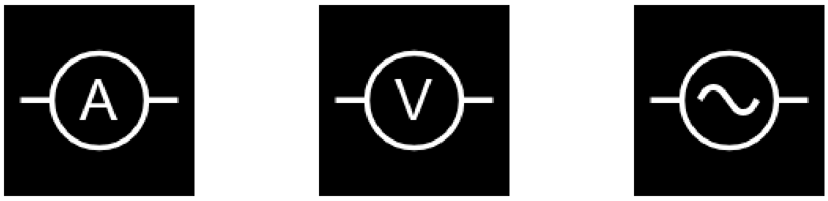


Figure 3.5: Ground truth images for training the SVM classifier.

SVM was originally designed to classify problems that require two-class classification [17]. In this method, a multi-class SVM was used to find a suitable hyperplane that could distinguish the different classes representing each component. Data was transferred from the HOG features extractor to a higher dimensional feature space. This method was chosen due to its suitable time complexity for relatively small datasets, which leads to faster execution times [17].

3.4 Detection of Circuit Components

3.4.1 Histogram of Oriented Gradients (HOG)

The HOG method for detecting circuit components involved using gradient orientation histograms to extract robust features from the binarized images. This method captured the edge structure and distribution of gradients within each symbol, providing a detailed representation that is invariant to minor variations in drawing style. The HOG features were then used to train a k-nearest neighbors (kNN) classifier, which effectively differentiated between various electrical components with higher accuracy compared to the manual method. This approach significantly improved the detection reliability and reduced the likelihood of misclassification.

3.4.2 Manual Line Tracking

In this approach, the detection of circuit components involved manually coding line tracking algorithms to identify and classify hand-drawn electrical symbols. The process began with preprocessing the images to binarize them and remove noise. The line tracking algorithm then converted the pixel data into coordinate points, allowing for the analysis of geometric features such as aspect ratio, fill percentage, and continuity. These features were used to classify different components based on predefined thresholds and heuristics. Although this method provided a basic level of accuracy, it was prone to errors due to variations in hand-drawn symbols and required extensive tuning of parameters.

3.5 Terminal Recognition

3.5.1 Histogram of Oriented Gradients (HOG)

Using HOG for terminal recognition involved analyzing the gradient orientation and magnitude at the edges of the symbols to detect points where lines terminated. The HOG features provided a detailed edge map that highlighted terminal points more clearly than the manual method. By examining the gradient patterns, the classifier could reliably identify terminals even in the presence of noise or slight variations in drawing. This method offered a more robust solution for terminal recognition, ensuring consistent detection across different hand-drawn instances.

3.5.2 Manual Line Tracking

Terminal recognition using the manual method involved identifying the endpoints of lines that extended from the main body of the component symbols. The line tracking algorithm was adjusted to detect these endpoints by analyzing the continuity and direction of the strokes. Terminals were characterized by abrupt changes in stroke direction and isolated points at the periphery of the symbols. This method required careful calibration to distinguish between terminals and other parts of the symbols, and it was sensitive to inconsistencies in hand-drawn lines.

3.6 Node Recognition

3.6.1 Histogram of Oriented Gradients (HOG)

For node recognition using HOG, the gradient orientation and magnitude information were used to identify areas with high gradient density, indicating potential nodes. The HOG features provided a comprehensive view of the edge structure, making it easier to detect intersections and connection points accurately. This method leveraged the detailed gradient information to distinguish true nodes from noise, offering improved reliability and precision compared to the manual approach.

3.6.2 Manual Line Tracking

Node recognition with the manual line tracking method focused on detecting intersections where multiple lines converged, indicating connection points in the circuit. The algorithm identified nodes by tracking the continuity of lines and detecting points where they intersected or changed direction sharply. This method required complex logic to differentiate between actual nodes and accidental overlaps or noise, making it challenging to achieve high accuracy consistently.

3.7 Circuit Schematic Generation

3.7.1 Histogram of Oriented Gradients (HOG)

The HOG-based approach for circuit schematic generation utilized the accurately recognized components, terminals, and nodes to create a precise schematic layout. By leveraging the robust features extracted through HOG, the algorithm could reliably connect terminals and nodes, ensuring a correct representation of the circuit. This method minimized the need for extensive post-processing and provided a more automated and accurate schematic generation process. The detailed gradient information helped maintain the integrity of the connections, resulting in a clearer and more reliable circuit diagram.

3.7.2 Manual Line Tracking

Generating circuit schematics manually involved using the recognized components, terminals, and nodes to reconstruct the overall circuit layout. The algorithm connected the detected terminals and nodes based on their spatial relationships and continuity of lines. This process was highly dependent on the accuracy of the preceding steps, and errors in component or terminal recognition could lead to incorrect schematic generation. The manual method required significant post-processing to ensure the schematic's correctness and completeness.

Chapter 4

Results and Discussion

4.1 Training

4.1.1 Data Pre-processing and HOG Feature Extraction

HOG features were successfully extracted in Python using the `sklearn` package whose `hog()` function produced output images as shown in one of the output images below. Data pre-processing was performed to resize the image to the 64×128 dimensions that are suitable for the application of the HOG feature extraction method. To formulate a set G of ground truth and predicted values, filenames of the template images were set to the form 'ammeter' for an ammeter, 'voltmeter' for voltmeters, and so forth.

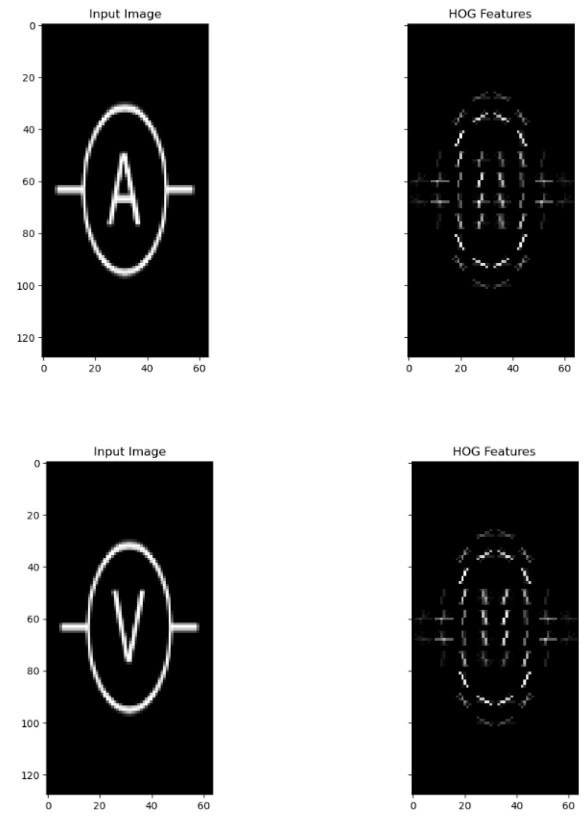


Figure 4.1: Resulting images from applying HOG feature extraction to find maximum gradients and orientations of the `ammeter` and `voltmeter` templates.

By applying horizontal and gradient filters on the template images and storing the results in a 9-bin histogram, the hand-drawn component recognition scheme was able to extract features that could be used to classify the data using a SVM classifier. Using the assigned filenames as ground truth values, the SVM classifier was trained as follows.

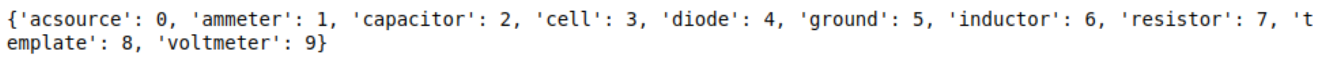
4.1.2 SVM Classifier Training

A sigmoid kernel was implemented to perform the SVM classification. The was based on the features that were extracted from the template images as shown above. The Python code for performing the sigmoid SVM classification on the training set is shown in the listing below.

```python
for img in template_images:
    img = cv2.resize(img, (64, 128))
    hog_feature = hog(img, orientations=9, pixels_per_cell=(8, 8),
                      cells_per_block=(2, 2), transform_sqrt=True,
                      block_norm="L1")
    hog_features.append(hog_feature)

label_mapping = {label: idx for idx, label in enumerate(np.unique(template_labels))}
true_labels_encoded = [label_mapping[label] for label in template_labels]

X_train, X_test, y_train, y_test = train_test_split(hog_features, true_labels_encoded, test_size=0.2, random_state=42)

svm_classifier = SVC(kernel='sigmoid')
svm_classifier.fit(X_train, y_train)
```

Listing 4.1: Code snippet for training the SVM classifier with a sigmoid kernel.

Fig. 4.2 shows the labels of the classes that were successfully extracted using the SVM classifier.



```
{'acsource': 0, 'ammeter': 1, 'capacitor': 2, 'cell': 3, 'diode': 4, 'ground': 5, 'inductor': 6, 'resistor': 7, 't
emplate': 8, 'voltmeter': 9}
```

Figure 4.2: Labels of the classes identified by the SVM classifier.

4.2 Testing SVM Hand-drawing Classifier on Multiple Classes

The SVM classifier with a sigmoid kernel was tested on the a subset of the SolvaDataset as described in section 3.1 above. The results of the original implementation of the classifier that was trained using the template images are summarised in the confusion matrix below.

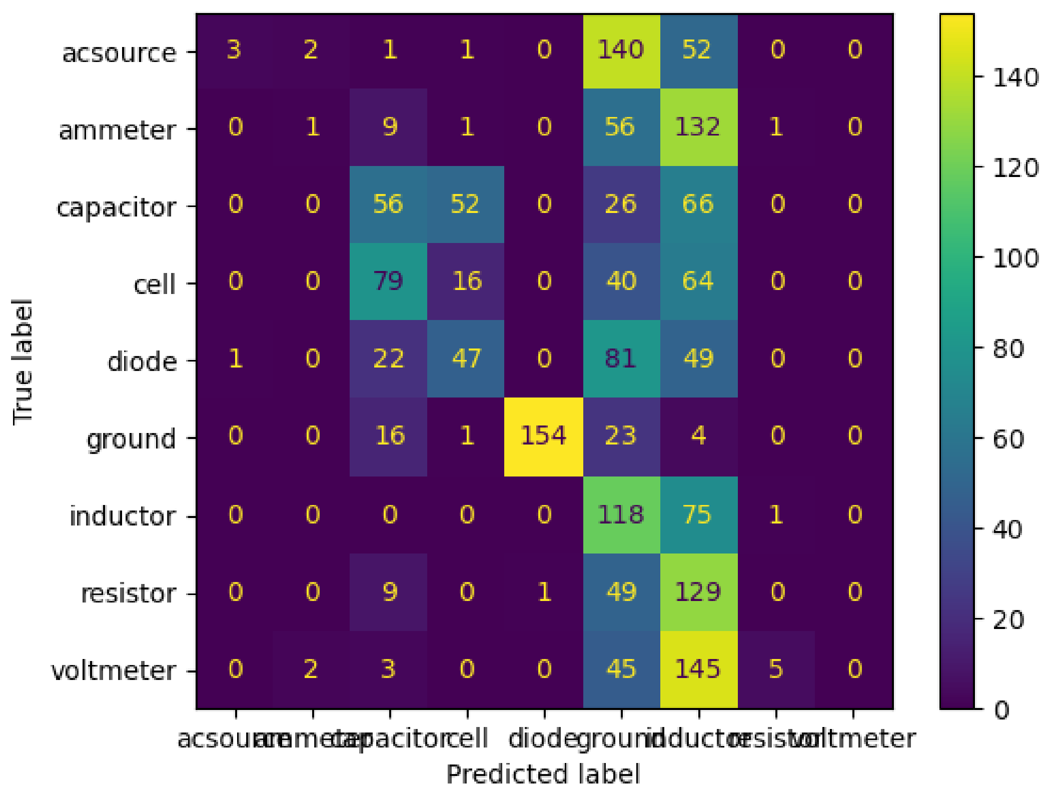


Figure 4.3: Classification of hand-drawn electronic components using a standard SVM with a sigmoid kernel.

The confusion matrix in Fig. 4.3 show that the classifier performed well for capacitors, obtaining the target correct 28% of the time. However, the classifier performed best for inductors, having successfully classified the hand-drawn sketch of the component correctly 37.5% of the time. The classifier also successfully predicted ground, cell, and AC source hand-drawings, albeit, having been wrong most of the time. This classifier was unsuccessful in grouping diodes, resistors and voltmeters. This can be accounted for by the high correspondance between the shapes of diodes and ground, resistors and inductors, as well voltmeters and ammeters.

Overall, the SVM performed poorly in classifying datasets with more classes of components.

4.2.1 Manual Line Tracking

When it came down to tracking the coordinates of the drawn pixels as well as as using them to track how the line was drawn, we needed to first find the features that we wish to work with. Doing this Manually is very tedious as you exhaust out, trying different features to classify the data, attempting different models as well. In the end we were able to create a classification making use of the Aspect Ratio on the x-axis and a combination of fill, continuity and deviation on the y-axis. Due to the nature of manually doing it, it comes with a lot of time restraints, so one of the things we had to do was reduce the amount of classes used. It resulted in a classification that returns the following figure:

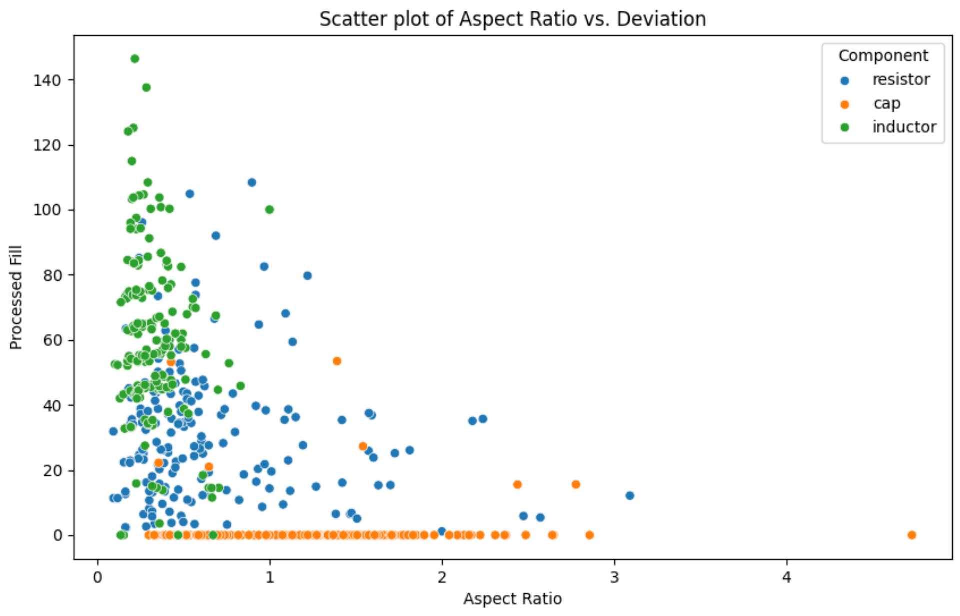


Figure 4.4: Classification of Manual Line Tracking of RLC components

With this we were able to then get the confusion matrix and found that there was a lot of similarities between the inductor and resistor so we adjusted the model till it gave a decent spread. Afterwards then obtaining the Confusion Matrix in Fig. 4.5 below.

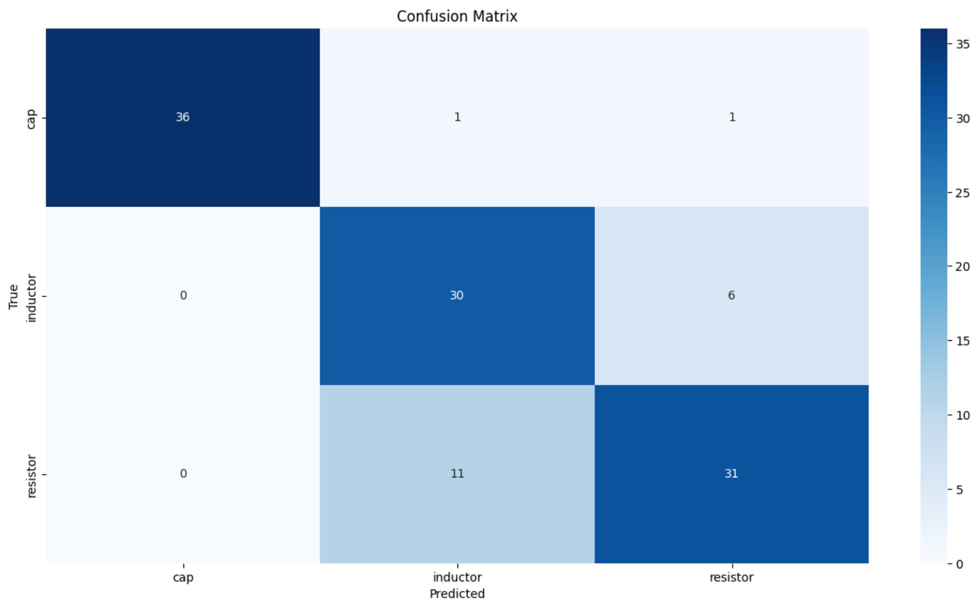


Figure 4.5: Confusion Matrix of Manual Line Tracking of RLC components

Chapter 5

Conclusions

The purpose of this project was to develop and evaluate methods for recognizing hand-drawn electrical component symbols, using both manual feature extraction techniques and automated methods such as the Histogram of Oriented Gradients (HOG).

This report began with an introduction to the significance of hand-drawn diagram recognition (HDDR) in various fields including healthcare, education, and engineering. It outlined the challenges faced by computers in interpreting hand-drawn diagrams due to variations in symbol scaling, inclusion of text, and other factors. The introduction also covered different approaches to HDDR, including image processing and machine learning techniques.

The literature review was followed in Chapter 2, where we discussed various existing methodologies for HDDR, including Blostein's six-stage process for diagram recognition and recent advancements in machine learning algorithms such as k-nearest neighbors (kNN) and support vector machines (SVMs). This chapter also highlighted the advantages of deep learning techniques, particularly Convolutional Neural Networks (CNNs), for improving recognition accuracy.

The bulk of the work for this project followed next, in Chapter 3, where we described the dataset preparation and feature extraction methods. The dataset comprised binarized images of hand-drawn electrical component symbols. Manual feature extraction involved calculating aspects like aspect ratio, fill percentage, and continuity, while the HOG method focused on extracting edge and gradient information to form robust feature vectors.

We implemented and compared two approaches for symbol recognition: one based on manually extracted features and the other using the HOG method. The manual method involved using kNN for classification based on features such as aspect ratio and processed fill. The HOG method, on the other hand, utilized the scikit-image library to extract gradient-based features, which were then used for classification.

We attempted to evaluate the performance of both methods. The HOG method significantly outperformed the manual feature extraction approach. It provided better differentiation between classes, as evidenced by higher classification accuracy and more clearly defined confusion matrix results. The confusion matrix for the HOG-based classifier showed fewer misclassifications and higher overall accuracy compared to the manual method.

Chapter 6

Recommendations

6.1 Enhance Image Pre-processing Techniques

- Adaptive thresholding: the current design did not require to account for noise because the images were black and white, i.e. binarized. Adaptive thresholding can be adopted to handle variations in background noise for datasets that include non-binarized images.
- Edge detection: Canny edge detection is a state of the art algorithm that can significantly improve the method's performance in recognising boundary edges in inputs that are non-binarized.

6.2 Advance Feature Extraction Methods

- Deep learning approaches: Currently, the SCISSOR algorithm uses HOG descriptors to define features, however deep learning-based approaches, such as RNN or CNN-based methods, could be explored for better performance in feature extraction.
- Geometric feature extraction: Improvements on the geometric feature extraction for properties such as image aspect ratio, curvature and symmetry could provide additional discriminative information for classification.

6.3 Data Diversity

- Dataset expansion: The current dataset is limited to 120×120, binarized images with no noise and a fixed orientation. The dataset can be expanded or altered by applying transformations to make the model invariant to rotations, scaling and translations to increase sample diversity and improve the model's ability to generalize unseen data.
- Include text variations: since some electronic components include text in their diagramatic representations, incorparating variations of symbols with and without accompanying text can enhance the model's accuracy to different drawing styles and annotations.

6.4 Model Selection and Optimization

- Hyperparameter tuning: future work can perform systematic hyperparameter tuning using techniques such as grid search or Bayesian optimization to maximize the performance of the model.

Bibliography

[1] V. Agrawal, J. Jagtap, and M. P. Kantipudi, "An overview of hand-drawn diagram recognition methods and applications," *IEEE Access*, vol. 12, pp. 19 739–19 751, 2024.

[2] L. Akter *et al.*, "Early identification of parkinson's disease from hand-drawn images using histogram of oriented gradients and machine learning techniques," in *2020 Emerging Technology in Computing, Communication and Electronics (ETCCE)*. IEEE, 2020, pp. 1–6.

[3] S. Ali, N. Aslam, D. Kim, A. Abbas, S. Tufail, and B. Azhar, "Context awareness based sketch-deepnet architecture for hand-drawn sketches classification and recognition in aiot," *PeerJ Computer Science*, vol. 9, p. e1186, 2023.

[4] R. R. Rachala and M. R. Panicker, "Hand-drawn electrical circuit recognition using object detection and node recognition," *SN Computer Science*, vol. 3, no. 3, p. 244, 2022.

[5] D. Blostein, "General diagram-recognition methodologies," in *Graphics Recognition Methods and Applications: First International Workshop University Park, PA, USA, August 10–11, 1995 Selected Papers 1*. Springer, 1996, pp. 106–122.

[6] M. N. Noor, M. Nazir, S. Rehman, and J. Tariq, "Sketch-recognition using pre-trained model," in *Proceedings of the National Conference on Engineering and Computing Technology, Islamabad, Pakistan*, vol. 8, 2021.

[7] O. Altun and O. Nooruldeen, "Sketrack: stroke-based recognition of online hand-drawn sketches of arrow-connected diagrams and digital logic circuit diagrams," *Scientific Programming*, vol. 2019, pp. 1–17, 2019.

[8] S. Pavithra, N. Shreyashwini, H. Bhavana, G. Nikhitha, and T. Kavitha, "Hand-drawn electronic component recognition using orb," *Procedia Computer Science*, vol. 218, pp. 504–513, 2023.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[10] W. Szwoch and M. Mucha, "Recognition of hand drawn flowcharts," in *Image Processing and Communications Challenges 4*. Springer, 2013, pp. 65–72.

[11] J. Fang, Z. Feng, and B. Cai, "Drawnnet: offline hand-drawn diagram recognition based on keypoint prediction of aggregating geometric characteristics," *Entropy*, vol. 24, no. 3, p. 425, 2022.

[12] M. Dey, S. M. Mia, N. Sarkar, A. Bhattacharya, S. Roy, S. Malakar, and R. Sarkar, "A two-stage cnn-based hand-drawn electrical and electronic circuit component recognition system," *Neural Computing and Applications*, vol. 33, pp. 13 367–13 390, 2021.

[13] M. Moetesum, S. W. Younus, M. A. Warsi, and I. Siddiqi, “Segmentation and recognition of electronic components in hand-drawn circuit diagrams,” *EAI Endorsed Transactions on Scalable Information Systems*, vol. 5, no. 16, pp. e12–e12, 2018.

[14] M. Gupta, P. Mehndiratta, and A. Bhardwaj, “Object recognition in hand drawn images using machine ensembling techniques and smote sampling,” in *Information, Communication and Computing Technology: 4th International Conference, ICICCT 2019, New Delhi, India, May 11, 2019, Revised Selected Papers 4*. Springer, 2019, pp. 228–239.

[15] C. C. Le, P. Prasad, A. Alsadoon, L. Pham, and A. Elchouemi, “Text classification: Naïve bayes classifier with sentiment lexicon,” *IAENG International journal of computer science*, vol. 46, no. 2, pp. 141–148, 2019.

[16] A. Waheed, “How to apply hog feature extraction in python,” Last Accessed: 15/05/24. [Online]. Available: https://thepythoncode.com/article/hog-feature-extraction-in-python

[17] M. Günay and M. Köseoğlu, “Classification of hand-drawn circuit components by considering the analysis of current methods,” in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE, 2020, pp. 1–5.