Review

AI Image Content Extraction Survey

Nick Bray , Michael Hempel * , Matthew Boeding , Hamid Sharif

Posted Date: 9 December 2025

doi: 10.20944/preprints202512.0556.v1

Keywords: visual image understanding; technical illustrations; artificial intelligence; deep learning techniques; optical character recognition; structural context analysis

Review

AI Image Content Extraction Survey

Nick Bray [1], Michael Hempel [1,*], Matthew Boeding [1] and Hamid Sharif [1]

Department of Electrical and Computer Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588, USA
* Correspondence: mhempel@unl.edu

Abstract

With Artificial Intelligence (AI) rapidly increasing in popularity and presence in everyday life, new applications utilizing AI are being explored throughout virtually all domains, from banking and healthcare to cybersecurity to generative AI for images, voice, and video content creation. With that trend comes an inherent need for increased AI capabilities. One cornerstone of AI applications is the ability of Generative AI to consume documents and utilize their content to answer questions, generate new content, correlate it with other data sources, and more. No longer constrained to text alone, we now leverage multimodal AI models to help us understand visual elements within documents, such as images, tables, figures, and charts. Within this realm, capabilities have expanded exponentially from traditional Optical Character Recognition (OCR) approaches towards increasingly utilizing complex AI models for visual content analysis and understanding. Modern approaches, especially those leveraging AI, are now focusing on interpreting more complex diagrams such as flowcharts, block diagrams, electrical schematics, and timing diagrams. These diagrams combine text, symbols, and structured layout, making them challenging to parse and comprehend using conventional techniques. This paper presents a historical analysis and comprehensive survey of scientific literature exploring this domain of visual understanding of complex technical illustrations and diagrams. We explore the use of deep learning models, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer-based architectures. These models, along with OCR, enable the extraction of both textual and structural information from visually complex sources. Despite these advancements, numerous challenges remain, however. These range from hallucinations, where OCR systems produce outputs not grounded in the source image, leading to misinterpretations, to a lack of contextual understanding of diagrammatic elements, such as arrows, grouping, and spatial hierarchy. This survey focuses on four key diagram types: flowcharts, block diagrams, electrical schematics, and timing diagrams. It evaluates the effectiveness, limitations, and practical solutions — both traditional and AI-driven — that aim to enable the extraction of accurate and meaningful information from complex diagrams in a way that is trustworthy and suitable for real-world, high-accuracy AI applications. This survey reveals that virtually all approaches struggle with accurately extracting technical diagram information, and it illustrates a path forward. Pursuing research to improve their accuracy further is crucial for various applications, including complex document question answering and Retrieval Augmented Generation, document-driven AI agents, accessibility applications, and automation.

Keywords: visual image understanding; technical illustrations; artificial intelligence; deep learning techniques; optical character recognition; structural context analysis

1. Introduction

We are experiencing the rapid proliferation of AI in our everyday lives. We can now find AI applications in virtually every domain, from virtual assistants and AI researchers to cybersecurity tools, intelligent industrial automation systems, and much more. Driven by Generative AI, these tools are increasingly multi-modal. They are no longer constrained to text and can be used to generate images, videos, sounds, and voices. They can also be used to analyze multi-modal sources, such as

 doi:10.20944/preprints202512.0556.v1

comprehending text, summarizing image content, and intelligently editing videos. In particular, the ability to extract meaningful information from visual data has long been a central focus of computer vision research, dating back to traditional algorithmic Optical Character Recognition (OCR) methods. Initially, OCR was developed to recognize typed or handwritten text in scanned documents, converting them from pixel-based images into machine-readable formats, such as plain text, source code, and formatted text. Robotics applications have pushed the boundaries of what computer vision algorithms can achieve, but they have always been limited by the capabilities of the underlying algorithms.

In recent years, the rise of artificial intelligence and machine learning has led to significant advancements across many fields, including image content analysis. While traditional OCR was limited to structured, text-heavy inputs, such as books, forms, or printed documents, modern deep learning techniques have been introduced that can support more complex visual data.

One particular emerging application domain for AI image analysis is the automation of extracting complex descriptions of both text and structure from technical diagrams and other technical illustrations. These types of illustrations are commonly found in engineering, software development, and electronics, and contain a mixture of textual, symbolic, and graphical elements. Increasingly, AI models are being used to extract not only the textual content but also vital structural and relational information from these diagrams, enabling greater automation, analysis, and digital transformation. This capability is critical for applications such as Retrieval Augmented Generation (RAG) [1], where Large Language Models (LLMs) are used together with information extracted from documents to answer specific questions related to these documents. For any RAG pipeline, it is critically important to capture all content of the provided documents, not just their textual aspects. It also greatly aids in question-answering processes, enabling the generation of full and accurate answers to user questions by extracting relevant text and visual information from technical documents. This also supports accessibility applications, such as screen readers, in accurately describing visual content, as well as many sophisticated and AI-driven automation applications. Thus, the ability to extract visual details from technical illustrations within these documents, alongside the textual content, is crucial for accurately answering user questions, informed decision-making, and more.

This image content extraction survey focuses on four specific, widely used types of technical illustrations: flowcharts, block diagrams, electrical schematics, and timing diagrams. These are important elements in technical documentation, user guides, network protocol specifications, software and algorithm documentations, and many other types of documents crucial to virtually all scientific fields and end-user applications. Our aim for this survey is to provide an overview of the current state-of-the-art in image content analysis for technical illustrations, including their current capabilities, shortcomings, and suggested areas of further research.

Flowcharts and block diagrams excel at representing complex systems in a clear and easy-to-understand format. They can be used to describe processes, systems, data flow, computer programs and algorithms, document structure, workflows, and decision-making. Also, closely related to flowcharts are State Machine Diagrams and more structured forms such as swimlane flowcharts. Both diagram types are widely used in fields such as engineering, electronics, and telecommunications, where systems need to be broken down into smaller components for analysis, design, and troubleshooting. Figure 1 shows a typical example of a flowchart, and a simple block diagram is shown in Figure 2.

 doi:10.20944/preprints202512.0556.v1

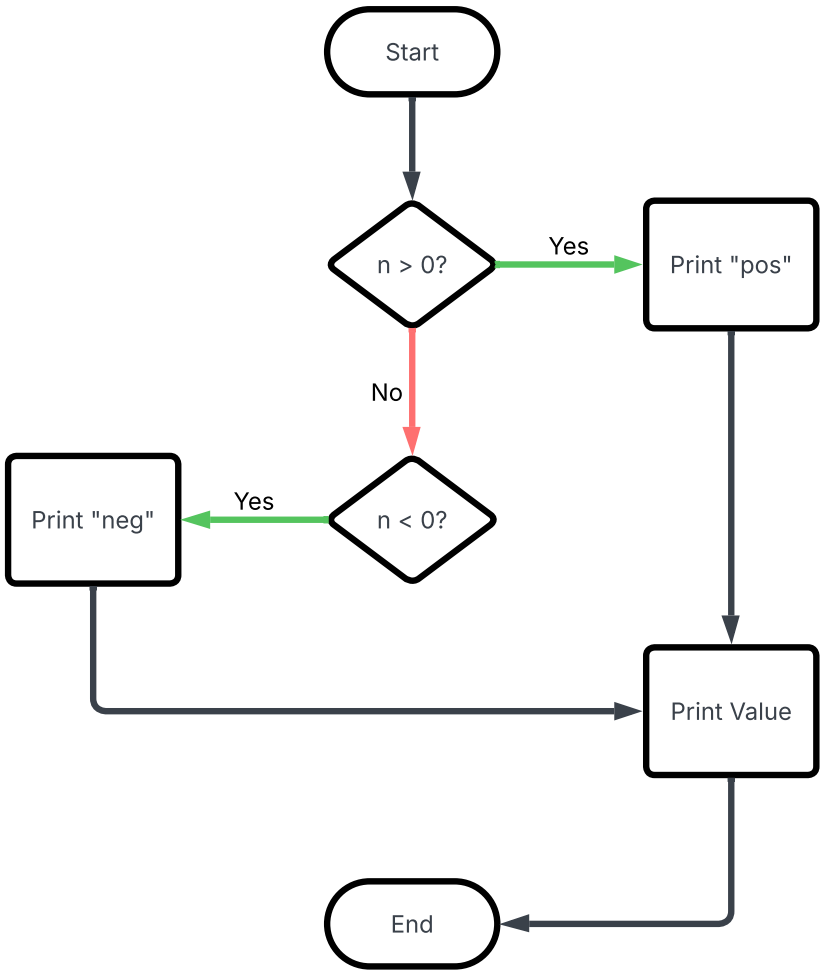

Figure 1. Comparison Flowchart Example.

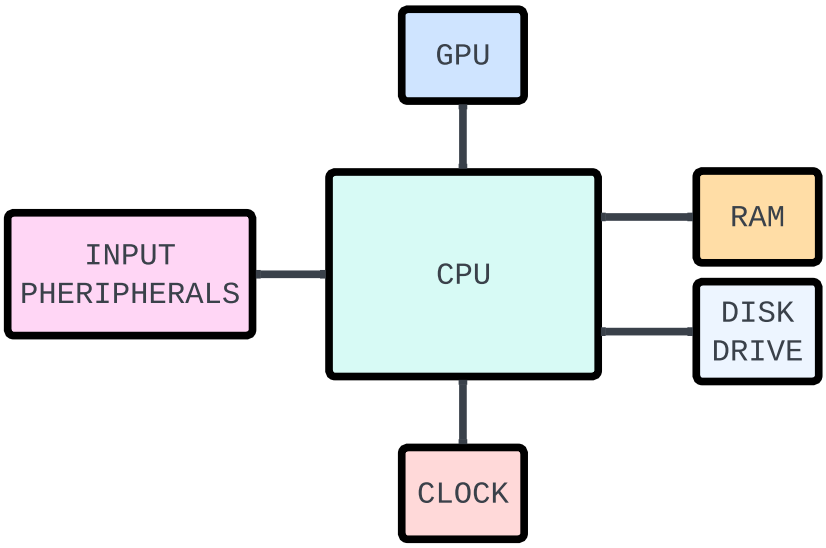

Figure 2. Computer Block Diagram Example.

When it comes to electrical systems and electronics, scientists and engineers often rely on electrical circuit diagrams, which provide a detailed and standardized visual representation of circuits and electronic components, as well as the connections between them, to achieve a specific function. Circuit diagrams serve as blueprints for electrical systems, converting abstract design concepts into a visual representation that is easy to understand. A circuit diagram that describes a microcontroller (MCU) pin-out is shown in Figure 3.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 December 2025 doi:10.20944/preprints202512.0556.v1

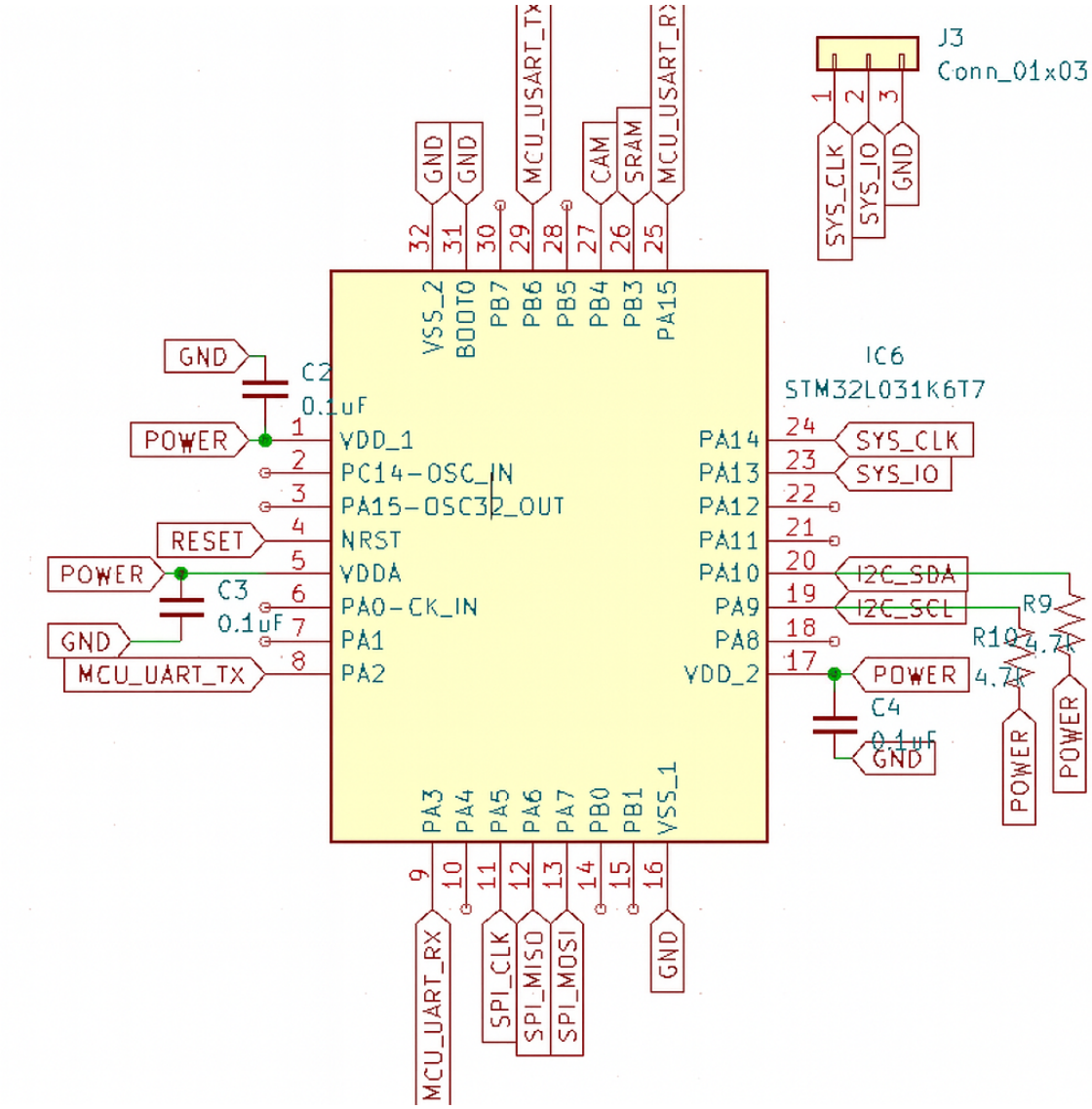

Figure 3. MCU Circuit Diagram Example.

Finally, this survey also specifically reviews approaches related to the analysis of timing diagrams. They are a type of visual representation that describes the behavior of signals or events over time, for example, in digital circuits, systems, or communication networks. Timing diagrams provide a clear and organized way of observing how various signals evolve and interact over specific intervals. An example timing diagram is shown in Figure 4, illustrating the relationship between the clock, reset, address, data, write enable, and ready signals.

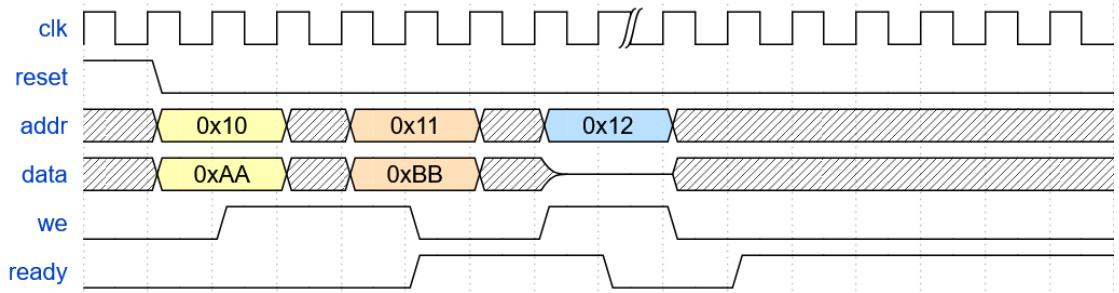

Figure 4. Timing Diagram Example.

These diagram types all pose unique challenges. While OCR can typically be applied to extract the textual content, a comprehensive representation of these diagram types encompasses far more than

 doi:10.20944/preprints202512.0556.v1

simply text. It needs to include shapes, color, and the structural relation between all these elements, in order to fully capture their content and meaning.

OCR itself has evolved over the years, progressing from a purely algorithmic approach to incorporating various Deep Learning techniques.

Despite these improvements, significant limitations remain. One major challenge arising from the use of AI models is hallucination, where AI models generate text or structures that are not grounded in actual content. Another challenge is the lack of contextual understanding. Many image content analysis systems still struggle to accurately interpret connections such as directional arrows, grouped symbols, or hierarchical structures, all of which are crucial in understanding the intent and logic behind technical diagrams. These can lead to flawed outputs, misinterpretations, or even cause system failures when used in automated pipelines.

This paper presents a comprehensive survey and historical analysis of image content extraction techniques, including OCR, applied to the four major diagram types mentioned above. It includes both traditional and AI-driven approaches, highlighting their methodologies, strengths, and specific limitations. By analyzing a wide range of models and algorithms, we assess their effectiveness in extracting both textual and structural information. We also evaluate their readiness for deployment in high-accuracy, real-world environments.

The remainder of this paper will be organized as follows. Section 2 will go over the steps on how this survey was conducted. In Section 3, we present a brief review of the evolution of image content analysis approaches and related evaluation techniques. Section 4 represents our survey, analyzing and discussing the various research papers we included for this survey, exploring their inspiration, techniques, and results. Finally, in Section 5 we present our conclusions and suggest future research directions.

2. Survey Methodology

The literature review method used in this study was designed based on a self-directed process to suit the specific aims of the research. We first conducted numerous literature searches across a wide range of academic publishing platforms. Our search focused on available resources on the following platforms:

- IEEE Xplore,
- ACM Digital Library,
- SpringerLink,
- Elsevier,
- MDPI, and
- Google Scholar.

From these searches, we then collected and compiled all relevant peer-reviewed papers within the scope of the topic. The literature search began with the use of targeted keywords, including "flowchart," "optical character recognition" (OCR), and "extraction," to retrieve publications relevant to the research scope. During this initial phase, no restrictions were applied to the publication date, thereby maximizing the inclusivity of the search results. This approach yielded a total of 210 research papers for preliminary consideration. The search strategy and retrieval process are visually summarized in Figure 5.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 December 2025 doi:10.20944/preprints202512.0556.v1

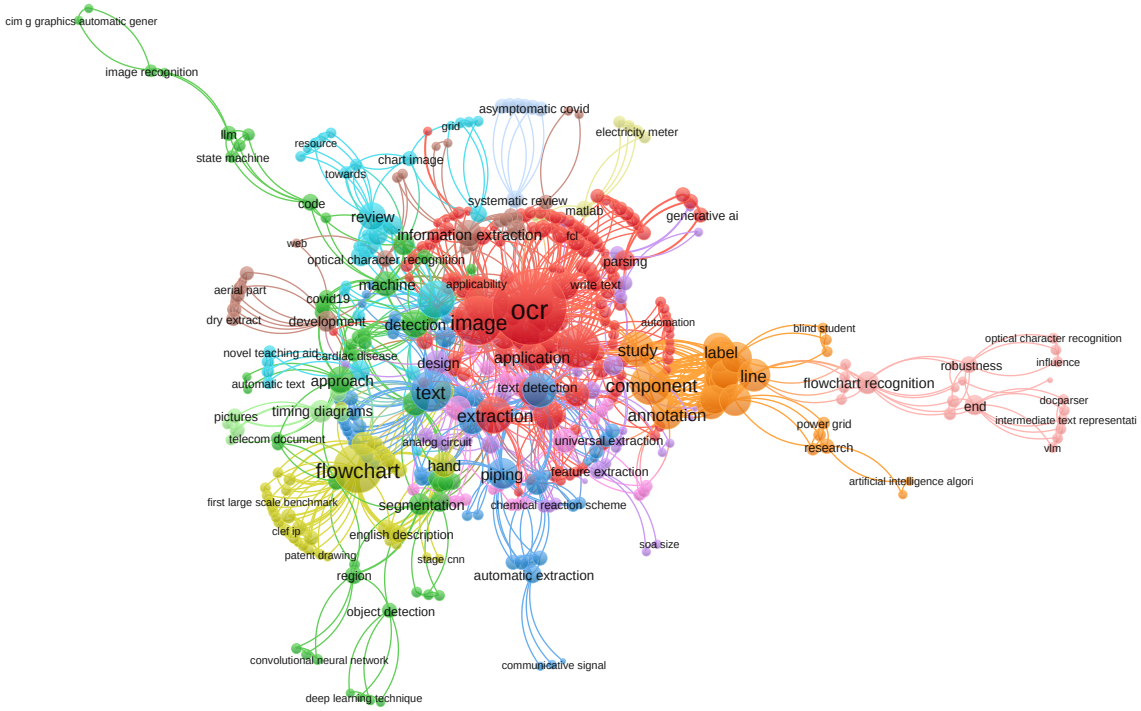

Figure 5. VOSViewer Clusters For Image Content Analysis keywords.

In our pre-filtering step, we then used carefully selected keywords and search techniques to manually identify and filter publications based on relevance, quality, and publication year. Once selected, the papers were reviewed in depth to extract key insights, common themes, and understand the methodological approaches used across the literature. We also, at that time, established relations between these papers. These findings helped establish a strong foundation for the study, ensuring that our work was informed by a broad and up-to-date understanding of current research in the field.

Following the broad initial search and pre-filtering, a more rigorous and focused screening process was implemented to enhance the relevance and modernity of the literature base. Specifically, additional papers were carefully selected for inclusion that were published between 2020 and 2025, ensuring the inclusion of the most recent advancements and findings highly pertinent to the research topic. In addition to this date criterion, further inclusion parameters were applied to align the selected studies with the specific scope and objectives of this paper, with a particular emphasis on research in information retrieval and text recognition from diagrammatic sources.

This systematic refinement culminated in a curated set of 56 research papers, which serve as the foundation for the detailed analysis and discussion in this survey. Among those, 37 papers drive the technical discussion presented below, with other papers providing supporting insights and closely related techniques discussed in relation to the core papers.

During our literature search, we identified a survey published in 2024 [2] that provides an in-depth analysis of advancements in text and shape extraction specifically from Piping and Instrumentation Diagrams (P&ID). Thus, for our survey, we excluded this diagram type and instead refer the reader to the referenced publication, as that paper comprehensively covers the significant progress in the domain of P&ID schematic analysis.

3. Background

3.1. Early Image Content Extraction Approaches

Over the years, numerous methods have been developed for extracting textual and visual information from documents and images. Earlier approaches focused heavily on OCR, which aims to detect printed or handwritten text and convert it into machine-readable formats. As image processing

 doi:10.20944/preprints202512.0556.v1

improved, shape and pattern recognition methods emerged, allowing systems to detect and recognize shapes, diagrams, and geometric figures. In recent years, the emergence of deep learning methods for computer vision tasks, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), has dramatically improved the accuracy and versatility of text and image content extraction techniques. These advancements have enabled the development of more intelligent systems, capable of handling complex layouts, handwritten variants, and providing contextual understanding.

Traditional OCR techniques are the earliest methods for text extraction from images. They typically follow a multi-step process comprised of [3]:

1. Pre-Processing involves noise reduction, binarization, and skew correction to enhance image quality.
2. Segmentation separates text regions from non-text elements, and individual characters or words are isolated.
3. Feature Extraction identifies key characteristics of each symbol, such as edges, corners, and stroke widths.
4. Classification and Recognition compares the features to already trained character models to output machine-readable text.
5. Post-Processing is often utilized by modern OCR systems to correct recognition errors using language models.

While OCR is primarily limited to text recognition, research into shape recognition techniques focuses on the detection and extraction of symbols, icons, diagrams, and geometric figures. The steps associated with shape recognition typically involve [4]:

1. Edge Detection or Contour Extraction is utilized to identify shape boundaries.
2. Shape Description computes area, perimeter, or moments.
3. Feature Matching uses template matching or feature descriptor techniques.
4. Classification maps the detected shapes to predefined categories.

These methods are the cornerstone of content extraction from complex images, which comprise technical diagrams, engineering drawings, flowcharts, and other intricate diagrammatic representations.

The advent of deep learning has enabled significant advances in image content extraction, particularly for complex technical imagery. For example, CNNs [5] are comprised of three primary types of layers, which are typically stacked to extract and combine features from input data before producing a resulting value. By optimally structuring CNNs using these different layer types, the resulting CNNs excel at a variety of computer vision tasks. The three distinct layer types are:

- **Convolutional** layers apply a series of learnable filters or kernels that perform convolutional operations over the input image. These filters detect edges, textures, and color gradients.
- **Pooling** layers perform a down-sampling operation on the feature maps, reducing their spatial dimensions while retaining the significant information.
- **Fully Connected (FC)** layers operate after the feature maps are flattened into a one-dimensional feature vector. Flattening converts the multidimensional feature maps into a single continuous vector. These layers integrate the extracted features and compute the final prediction.

The convolving capabilities of CNNs make them ideal foundational approaches for processing spatial data such as images. For sequential data, such as text, however, RNNs [6] often outperform CNNs by modeling contextual dependencies through the process of remembering and incorporating prior input values. Hybrid versions of these frameworks also exist, such as Convolutional Recurrent Neural Networks (CRNNs), which combine CNN feature extraction with RNN sequence modeling, enabling end-to-end image content analysis. [7]

More recently, Transformer-based models emerged as a dominant architecture, enabling modern Large Language Models (LLMs) and other advances in Deep Neural Networks. LLMs, in particular, have rapidly become a cornerstone of everyday life. Transformers are capable of capturing long-range

 doi:10.20944/preprints202512.0556.v1

dependencies and contextual relationships across an entire document. Transformer architectures also include non-textual adaptations such as Vision Transformers (ViT) [8] and TrOCR [9]. Transformers can often outperform RNNs in recognizing complex layouts. [10] Vision Transformers were introduced to apply the Transformer paradigm to image analysis. They provide the foundational component for building Vision Language Models (VLM), which are multimodal LLMs capable of generating output based on textual and visual content. The initial step in a VLM typically involves vision encoding using ViT (to represent image data using feature vectors), followed by language encoding (to represent the text such as the user query as feature vectors), followed by a projection and integration step (to align both of these embeddings within a common encoding space combined with a Transformer model), and finally the output step that generates the text-based result.

3.2. Existing Evaluation Approaches for Content Extraction Methods

With the development of these architectures, it became essential to establish and implement rigorous evaluation and scoring mechanisms to quantitatively and qualitatively assess the performance of each such framework, particularly under challenging conditions such as noisy inputs, complex layouts, as well as dense or overlapping content, in order to determine their robustness, reliability, accuracy, and suitability for real-world applications. There are multiple methods that assess these frameworks, with one prominent example being Recall-Oriented Understudy for Gisting Evaluation (ROUGE-1) [11]. ROUGE-1 is a metric for evaluating text summarization that measures single-word (uni-gram) overlap between a generated summary and one or more reference summaries. It uses Precision, Recall, and F1-Score to quantify performance. The scores range from 0 to 1, where 0 represents no matching of words and 1 represents perfect matching of words.

Another scoring technique used for recognition evaluation is the Bilingual Evaluation Understudy (BLEU) scoring [12]. BLEU is used for evaluating the quality of the extracted text. The BLEU scoring scale ranges from 0 to 1, where 0 represents the lowest quality and 1 the highest [13]. Some variants of BLEU scoring include CodeBLEU [14], which specifically targets the evaluation of program code extraction, and BLEURT [15], which is based on the Bidirectional Encoder Representation from Transformers (BERT) to mimic a more human-like evaluation approach for extractions. It uses the same scoring scale as the original BLEU. BLEU-4 [16] is primarily used for evaluating extractions, utilizing a comparison to human translations. BLEU-4 is named thus because it utilizes 4 words to calculate precision, using a scoring scale that once again ranges from 0 to 1. Another scoring approach is the "mean Average Precision (mAP)" [17], which evaluates both the localization and classification performance. While precision is employed to measure the accuracy of the model's positive predictions, recall measures the completeness of the model's predictions, and the F1-score represents the mean of precision and recall [18].

4. Diagram Analysis

4.1. Analysis Overview

Since this paper focuses on recent developments for diagram analysis, each paper we are surveying in this section builds upon an extensive foundation of substantial prior work. We fully acknowledge and appreciate this deep historical grounding of the cited works and encourage the reader to delve into these works for a deeper understanding of these techniques. The scientific works we are focusing on in our survey represent a cross-section of the state-of-the-art in image content analysis for the four specific technical illustration types we selected. The authors of these respective papers build upon the extensive prior work to explore new approaches and improvements that push the boundaries for analyzing diagrammatic images. These techniques are critical in numerous application areas, in particular as information extraction tools for AI workflows, multimodal document understanding, document digitization, and literary curation efforts.

The provided analysis examines each article for three key review criteria:

1. inspiration, which covers the foundational ideas that influenced the study;

 doi:10.20944/preprints202512.0556.v1

2. techniques, where we explore specific methods and tools used in the research, along with their results; as well as
3. future directions, which outline the areas the authors wish to enhance or explore further.

Our analysis is divided further into the structure of the image content being extracted, as different methods have been evaluated against flowcharts, block diagrams, electrical circuit schematics, and timing diagrams.

4.2. Flowchart Analysis

4.2.1. Review of Identified Papers

In 2020, the authors of [19] proposed using region-based segmentation to convert flowcharts into graphs, introducing a custom approach for precisely extracting 2D elements from flowcharts. They discussed prior attempts, such as [20], which explored flowchart extraction from digital images using a neural network, and [21], which developed Flow2Code, a mobile app that converted photographs of hand-drawn flowcharts into code.

While Flow2Code's original text extraction was performed using offline sketch recognition and computer vision, to advance flowchart text extraction for increased precision and structure recovery, the authors of [19] proposed and evaluated their own technique. The authors followed a traditional preprocessing step using the Otsu method [22] for image thresholding. Feature extraction was achieved through region-based segmentation to separate text and non-text regions, also referred to as regions of interest (ROI) [23]. Morphological labeling was then used to extract text from corresponding regions, and shape identification was performed by analyzing contour vertices to determine flowchart shapes. The flowcharts were ultimately converted into graph-like structures composed of nodes, labels, shapes, and edges. A final plot was generated to visualize the graph. The Open Source Computer Vision Library (OpenCV) [24] and Python were used for implementation.

Their experimental evaluation involved a small dataset of only 25 different flowcharts, which were categorized and converted into graph representations. Each element of the flowcharts was parsed and examined. The approach successfully handled more complex parameters such as scaling, dashed or directed lines, and colored shapes, and achieved 90% accuracy during testing. The content extraction process effectively transformed flowcharts into structured graphs. This result demonstrates the effectiveness of the method in handling diverse graphical features. The authors noted that future work would address greater flowchart complexity and explore alternative extraction methods.

The authors of [25] created their own method of flowchart extraction called Graph Recognition Convolutional Neural Network (GRCNN) in 2020. It utilizes FlashFill [26], an algorithm famously used in Microsoft Excel, which takes a set of given input-output text pairs and determines the user-desired text transformation from those samples. This transformation can then be applied to other input text samples. In GRCNN, a CNN is utilized for image content extraction, with an edge network used to transform the feature vector from the CNN into an edge description of the given flowchart. Using this approach, the nodes of the flowchart are identified and also described.

A custom dataset containing 2490 flowchart images was used for the evaluation of GRCNN, with flowcharts ranging from 3 to 6 nodes and 0 to 2 decisions. These flowcharts were created using Graphviz [27]. An Intel i7 CPU and an NVIDIA GTX 1070 GPU (Graphical Processing Unit) were used for this evaluation. Sequence accuracy was used as the metric in this evaluation of GRCNN's performance. The edge and sequence extraction scores were 94.1% and 90.6%, respectively. However, for node extraction, a relatively low accuracy score of 67.9% was achieved. Consequently, the overall graph extraction accuracy (comprised of edge, sequence, and node extraction) was only 66.4%. This indicates that significantly more research was needed to consistently and accurately extract the structure of a flowchart.

Another approach to flowchart extraction was published in 2020 in [28]. It aims to improve upon other flowchart extraction approaches, such as [29,30], by facilitating the extraction of hand-written flowcharts instead of programmatically generated flowcharts. Images are first acquired using a

 doi:10.20944/preprints202512.0556.v1

camera or scanner. It then utilizes the Otsu method to transform each input image via threshold-based binarization, followed by morphological operations, dilation and application of the Ramer-Douglas-Peucker method [31] for object and line detection, as well as beam search [32] for the hand-written text detection. For evaluating their approach, a small dataset of 20 flowcharts was used. The average word and character accuracy are 66.1% and 86.52%, respectively. While the resulting scores are relatively low, they nonetheless illustrated the potential of this technique and warranted further exploration. The authors themselves stated that their future work aimed to improve the accuracy of flowcharts containing loops and to create a more diverse and complex flowchart.

Another study conducted and published in 2020 utilized deep learning and OCR to convert flowcharts into graphs [33]. The authors referenced earlier works, such as [34], which employed dynamic programming for stroke organization within a Random Forest framework, and [35], which converts flowcharts into source code. They also discussed [20,36]. The first step in their presented approach involved image preprocessing to obtain the flowchart in a planar perspective. This is followed by applying OpenCV contour detection. For shape recognition, the authors first applied Tiny YOLOv3 [37] in conjunction with Darknet [38]—a C and CUDA-based deep neural network (DNN) framework commonly used for implementing YOLO. YOLOv3-SPP (YOLOv3-Spatial Pyramid Pooling) [39] was used for lines and arrow recognition, and Google Cloud's Vision API was used for OCR-based text extraction. Graph Models (GM) were applied to represent the flowcharts. The final step converted this GM data into Flowchart Graphs (FGs). All experiments were implemented using Python on Ubuntu 18.04.

For the performance evaluation of this work, a new dataset comprising 161 images was created, consisting of 38 digital and 123 hand-drawn flowcharts. Using mAP, their presented method achieved an average precision of 99.82% for shape recognition and 88.14% for arrow and line detection. Although these results are promising, the authors emphasize that further improvements could be achieved by using larger datasets and more advanced frameworks. This approach nevertheless demonstrates significant potential for accurately extracting flowcharts. The authors further suggest that it would be worthwhile to develop a mobile application capable of capturing flowcharts and translating them directly into code.

In 2022, [40] proposed FlowChart to Code (FC2Code), a system that extracts text from flowcharts, converts it into pseudocode, and generates executable source code from it. Prior work, such as [36], introduced RAPTOR, a visual programming tool that helps students translate flowcharts into pseudocode. The FC2Code system used a custom dataset of 320 flowcharts with corresponding source code, of which 50 were used for testing. Structure recognition was applied to convert flowcharts into pseudocode, while Bidirectional LSTM [41] and Graph Attention Network (GAT) models [42] extracted textual information. To maintain information integrity, Reversed Flowchart, GAT, and the TranX decoder [43] were utilized to convert pseudocode into executable code. Results were compared against LSTM and Transformer [44] baselines.

Performance was evaluated using the BLEU score, where the proposed method achieved 55.68% compared to 38.25% for LSTM and 44.05% for the Transformer baseline approach. As shown, this method significantly outperforms shows both LSTM and Transformer networks. Although effective, the authors note that further work is required to enhance precision and address challenges in integrating flowchart structure into pseudocode.

In 2023, the authors of [45] introduced a public flowchart dataset named FloCo and a method, FloCo-T5, that converts flowcharts into Python code. This method was evaluated using a dataset containing 11,884 flowcharts, including both digital and handwritten ones. Their presented approach is based on the work in [46], another flowchart dataset but not publicly available, in [21], which converted hand-written flowcharts into executable code, as well as the work in [47] – the seminal work that introduced Bidirectional Encoder Representations from Transformers (BERT) for NLP language representation. Closely related is the work in [48], which digitizes handwritten flowcharts using Faster R-CNN and OCR methods, but only supports hand-drawn images without text. A Hough

 doi:10.20944/preprints202512.0556.v1

transform (HT) [49] was employed for shape recognition, EasyOCR was used for text recognition, while Code-T5 [50] was employed for source code generation. Handwritten text detection was handled using CRAFT [51], and TrOCR was used for text recognition.

To support their evaluation efforts, data augmentation was performed to increase the size of the available dataset, and a BERT-like Code-T5 strategy was employed to enhance the structural and semantic understanding of code and flowcharts. OCR fine-tuning with positional encoding further improved text alignment. A single NVIDIA A6000 with 48GB was used for testing. The proposed method achieved a BLEU score of 21.4 and a CodeBLEU score of 34.6. Although a large dataset could be used for this performance evaluation, the authors note that extracting flowcharts into Python source code requires further study. Future efforts thus aim to reduce computational demands and design models better suited for flowchart-to-code translations.

In 2024, Darda and Jain introduced a system that extracts flowcharts and generates Python code [52]. It builds upon the work in [40,53]. This work was influenced by the preceding efforts in [54], which introduced and presented S-DistilBERT, along with a version of Flow2Code published in 2019 [55] used for processing pictures of handwritten flowcharts and converting them into executable code. Darda and Jain utilized EasyOCR and OpenCV for text extraction, and employed Llama 2 LLM [56] for Python code generation via an API-connected interface. Python code generation was successful for 75% of the flowcharts, and the method produced accurate source code, even when 62% of the flowcharts had added noise. However, the authors also note that improvements are needed for more accurate text extraction and to provide multilingual capabilities.

As shown by these works, the advancements in LLMs have enabled new techniques for flowchart information extraction, which prove to be quite effective across a range of different imagery. However, we can also observe that significantly more work is needed to improve accuracy and precision. The following works aim to address this drawback in the preceding techniques.

In 2025, our team conducted limited experiments to evaluate the performance of a multimodal VLM when tasked with describing a flowchart, aiming to assess the viability of directly utilizing VLMs for such tasks without preprocessing. In these experiments, we utilized Google Gemma-3-27B, and prompted it to describe flowcharts, such as the one shown in Figure 6, with a portion of the resulting description shown in Figure 7. We are highlighting a particularly obvious error produced by the VLM, falsely denoting a loop back to the Start node, rather than a branch to the End node that is actually shown in the flowchart. Such erroneous information would result in significant problems for downstream tasks that depend on the extracted data, and demonstrates that, despite recent advances in LLM capabilities, significant challenges persist.

Another study in 2023, [54] proposed a custom S-BERT model named S-DistilBERT, inspired by [57], which extracted flowcharts from online sources, and [58], which is a lightweight variant of BERT, [28,59] is also referenced. Their experiments utilized 50 handwritten flowcharts, each with manually created pseudocode and descriptions, for validation purposes. The DistilBERT model was used for preprocessing the data, and a third-party text OCR app was used for text recognition. This resulted in an overall accuracy of 75.59%. This work once again demonstrates that, compared to other proposed methods, Transformer-based approaches perform very well for these types of applications. The authors note that recursion in flowcharts may be looked at in the future.

Several studies published between 2024 and 2025, such as [60–62], all reference FlowLearn [63] to enhance dataset annotation and architectural performance. For example, the authors of [60] introduced TEXTFLOW and utilized the datasets FlowVQA [64] (2,272 flowcharts) and FlowchartQA [46] (1 million images) for their performance evaluation. The presented TEXTFLOW combines MERMAID for annotations, Graphviz for node and edge definition, and PlantUML [65] with Visual Language Models (VLMs) [66] such as Qwen2-VL [67] and GPT-4o for logic understanding. The system utilized both VLMs and LLMs to construct flowchart structures and was trained on four NVIDIA A100 GPUs. FlowLearn's reported results demonstrated strong text extraction performance with an F1-score

 doi:10.20944/preprints202512.0556.v1

of 88.23%, with their suggested future work focused on reducing computational complexity. The approaches presented in [60–62] demonstrate similar results, overall.

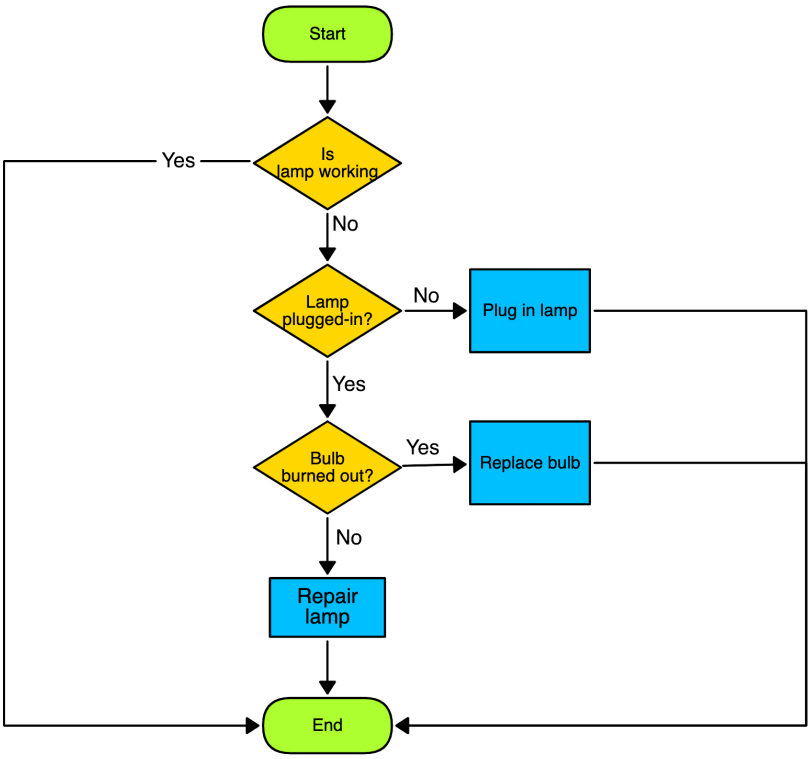

Figure 6. Flowchart diagnosing a Lamp Fault,

{
"detailed_description": "The image presents a flowchart diagram illustrating a troubleshooting process for a non-working lamp. The chart is laid out linearly, starting with an oval 'Start' node at the top center and ending with a rounded rectangle labeled 'End' at the bottom. The flowchart uses standard flowchart symbols: ovals for start/end, diamonds for decision points, and rectangles for processes.
The first decision point asks 'Is lamp working?' If the answer is 'Yes', a loop back to the start node is indicated by an arrow.
If the answer is 'No', the chart proceeds to a second decision point: 'Lamp plugged-in?'. A 'Yes' answer leads to another question, 'Bulb burned out?', and a 'No' answer directs the user to 'Plug in lamp'. If the bulb is burned out, the process instructs the user to 'Replace bulb'. If the bulb isn't burned out, the chart directs the user to 'Repair lamp'. After either replacing the bulb or repairing the lamp, an arrow leads back to the start node.

The flowchart is simple and uses clear text labels within each shape. The arrows indicate the flow of logic based on the answers to the questions. The shapes are colored: 'Start' and 'End' are green, decision points are white with black text, and process steps (Plug in lamp, Replace bulb, Repair Lamp) are light blue. The background is white. The overall style is clean and functional, typical of a technical diagram.",
"entity_info": {
"entity_name": "Lamp Troubleshooting Flowchart",
"entity_type": "image",
"summary": "This image depicts a flowchart outlining the steps to troubleshoot a malfunctioning lamp. The process begins by checking if the lamp is working, then progresses through questions about power

Figure 7. VLM Text Output of Lamp Fault Diagnosis Flowchart.

Inspired by [53], the authors of [68] in 2024 proposed GenFlowchart, a flowchart extraction method influenced by earlier transformation algorithms such as the PAD approach [69], along with the technique presented in [70] that iterates through flowcharts with dialogs. Image preprocessing was

 doi:10.20944/preprints202512.0556.v1

conducted using grayscale conversion, followed by Pytesseract OCR [71] configured with the Page Segmentation Mode (PSM) [72] selected to be mode 4, and the OCR Engine Mode (OEM) configured for mode 3. Additionally, the Segment Anything Model (SAM) [73], and PyMuPDF [74] were integrated for text and shape recognition.

The study evaluated 550 flowcharts based on text extraction and the description of the flowchart, with scoring methods such as BERT-P, BERT-R, BERT-F1, and Cosine Similarity. Their presented approach achieved an overall accuracy of 86.49% using BERT-F1 for instruction-based prompting. Based on the previous methods discussed, it is evident that using generative AI can enhance the overall effectiveness of flowchart extraction.

In [75], the authors present FR-DETR (Flowchart Recognition Detection Transformer)–an end-to-end multitask network. Prior work had used text–graphic separation and OCR-based extraction [76], while [77] and [78] employed DETR for object recognition. FR-DETR, on the other hand, integrates CNNs for feature extraction and Transformers for encoding and decoding, with the obtained results further improved using a Feedforward Neural Network (FNN) [79]. Finally, a coarse-to-fine strategy is employed for detecting arrows and lines.

Using CLEF-IP [80] and a custom dataset of 1,000 flowcharts that were randomly divided into 800 training images and 200 testing images, FR-DETR achieved 98.7% precision and 98.1% recall on CLEF-IP, and 94% precision and 93.1% recall on the custom dataset. This once again illustrates the benefits of using transformer-based approaches for extraction when compared to other, more traditional, methods. FR-DETR's computational requirements, however, preclude its use in mobile on-device applications. That area remains for future research efforts.

Another image content extraction method for hand-written flowcharts was introduced in 2022 by [59], building on the work shown in [81,82], among others, with the goal of improving upon their overall performance and accuracy. In their approach, each flowchart is represented in graph form using Instance GNN (InstGNN) [83]. Feature extraction is performed to simplify the data, and a GAT is used for learning and updating the nodes and edges of each flowchart. Training was conducted using the Deep Graph Library (DGL). Edge classification and node learning were conducted at the postprocessing stage, and symbols were verified using ResNet-18 [84]. A custom dataset called CASIA-OHFC, containing 2957 hand-drawn, annotated flowcharts, was used for testing. The Stroke Classification Accuracy (SCA) was used for evaluation, and the proposed method achieved a score of 76.44%. To further improve their approach, the authors suggest exploring various neural networks for feature extraction and the real-time identification of strokes and symbols.

A 2024 study introduced LS-DETR [85], targeting handwritten flowchart recognition. It builds upon [75] and earlier diagram recognition work published in [86] along with FLoCo-T5 [45]. [25, 40,48,54] is also referenced. LS-DETR integrates CNN encoders and DETR decoders with DPText-DETR [87] for angled and polygonal text, and Enhanced Factorized Self-Attention (EFSA) for precise shape prediction. FLoCo-T5 was used to convert recognized flowcharts into source code. Using four NVIDIA A100 GPUs, the authors evaluated their approach using mAP on multiple datasets (ImageNet, ICDAR19 ArT V3-Flowchart, GFTotal-Text, and a custom 85-flowchart set), demonstrating improved accuracy over prior methods: With the use of multiple datasets and their proposed transformer-based algorithm, the Average Precision (AP) at 50% Intersection over Union (IoU) is 99.18%. The authors demonstrate that their approach can significantly improve both accuracy and precision compared to previous efforts. However, improvements are needed for enhancing handwritten flowchart-to-code translations.

Flowchart extraction has recently received increasing attention, with 2025 seeing the highest number of methods developed, as illustrated in Figure 8. This view was generated using Litmaps, showing the connectedness of the papers discussed in this section, as well as miscellaneous related works organized in the blue ring.

 doi:10.20944/preprints202512.0556.v1

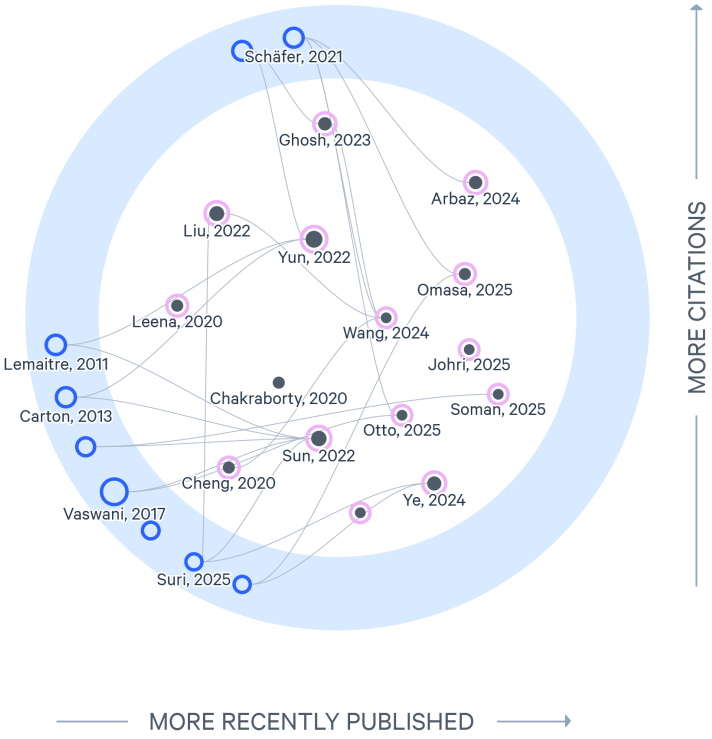

Figure 8. Citation Graph Visualization For Flowcharts.

In 2025, Omasa *et al.* [61] focused on improving the arrow precision and noise reduction provided by GenFlowchart [68]. Their work expands upon GenFlowchart, along with [63], and also discusses their work in the context of FloCo-T5 [45] as one of the foundations and inspirations for their own technique. [25] is also referenced in their paper. Using OCR for text extraction and DAMO-YOLO [88] for object detection, they combined OCR and detection data to identify arrow start–end pairs, constructing graph structures and performing reasoning analysis with GPT-4o's multimodal capabilities. On a set of 30 flowcharts, the method achieved a 88.9% OCR accuracy, as assessed and compared to results obtained via human evaluation. However, additional work is needed to enhance graph-based prompting, integrate confidence-aware reasoning, and refine evaluation methods for robust and generalizable real-world deployment.

Another VLM-based approach was published in 2025 by Soman *et al.* [62]. Their flowchart extraction method builds upon 2024's FlowVQA [64] and FlowLearn [63], and realizes graph structures from images using a fine-tuned version of a Qwen2-VL VLM. To evaluate the performance of their approach, the authors utilized the FlowLearn dataset, with graph performance measured via Graph Edit Distance (GED) [89]. Additionally, they implemented a Retrieval-Augmented Generation (RAG) system and evaluated it using TeleRoBERTa [90] and bge-large [91]. Using the top-k retrieval metric performance evaluation, which spanned 1,586 images, the system achieved an accuracy of 71.91% using TeleRoBERTa. The authors discussed their plans to extend their approach by implementing JSON structures through RAG, along with expanding to other forms of technical illustrations, such as Unified Modeling Language (UML) diagrams.

The author of [92], published in 2025, leveraged LLMs for flowchart extraction using the C4 model dataset. Their work builds upon [52,93,94]. Here, they first pre-process images to remove textual and visual noise. They also implement the LLM-based Tesseract OCR framework for diagram enhancement. RAG is used to enable Tesseract OCR to utilize external documents, code, and system details to create diagrams that are accurate, up-to-date, and match the real system. Datasets were created using Mermaid, PlantUML, Structurizr [95], Graphviz, and Diagrams as Code. Their evaluation was conducted with a focus on knowledge extraction, standardization, toolchain support, flexibility, and ease of use. The code generation post-extraction demonstrated very high compilation success rates, reaching 100% for Mermaid flowcharts, for those flowcharts where content extraction succeeded

 doi:10.20944/preprints202512.0556.v1

prior to Mermaid code generation. Future work will focus on developing an Agentic AI to dynamically determine diagram types, enhancing usability and deployment through improved UI and integration, refining document structuring with flexible rules and additional LLM processing, strengthening diagram compilation with multi-DSL validation and feedback loops, and expanding the dataset to improve generalization and output quality.

Another 2025 study employed third-party APIs combined with LLMs [96]. It utilizes automated computing [97,98] involving API calls to online image processing services, followed by leveraging GPT-4 Vision for content extraction, and finally using AI-based code generation in multiple programming languages to turn flowcharts into code. Using GPT-4, the system achieved an Artificial Analysis Quality Index score of 77, and a 92% efficiency in human-evaluated code conversion. This proposed method demonstrates that with more capable LLMs, the task of accurately extracting information from flowcharts can be improved. It also demonstrates that this task can be built upon online services to offload the processing requirements from the actual devices, thereby widening the possible target platforms and application domains. Similar to many other surveyed papers, however, the authors note that additional work is needed to improve accuracy and incorporate support for handwritten diagrams.

Several papers published in 2025 demonstrated that multi-modal LLMs can compete with, and in some cases outperform, purpose-built image content analysis frameworks. One such 2025 study [99] incorporated LLMs for recognition and code generation, building upon earlier work on diagram digitization [100]. In their approach, preprocessing is conducted first to scramble the text in the diagrams due to copyright concerns. Manual cropping enables enhanced precision before OCR-based abbreviation recognition and GPT-4o LLM-driven code generation. Using the OPC UA and PROFINET datasets, with 15 and 80 diagrams, respectively, the model achieved 63% and 45% edge detection accuracy. The key observation from this study is that even when using an LLM for flowchart extraction, significant preprocessing work is required beforehand to improve the obtained content extraction results.

Finally, the work presented in [101] introduced the Dynamic Flow-to-Code Generator (DFCG), which extracts flowcharts and generates source code. This was influenced by the sketch-based flowchart detection research presented in [102] and code generation from flowcharts [103], as well as the works in [40,45]. Their presented method uses binarization for noise reduction, an approach called Faster Region-based CNN (Faster R-CNN) and Visual Geometry Group (VGG-16) [104] for shape detection, Region Proposal Network (RPN) to identify any objects in the feature map, Keras OCR with C-RNN for text extraction, Hough Transform for arrow and line detection, and finally a Graph Neural Network (GNN) [105] for graph construction and conversion into several programming languages. For evaluation, the authors used a dataset containing 775 digital and 100 handwritten flowcharts, achieving a 95.8% mAP, a 93.4% precision, a 94.3% recall, and a 93.8% for F1. By employing a multi-layered algorithm and a sufficiently large dataset, they demonstrated the high performance achievable by their approach, albeit at the expense of significant computational resources, without utilizing transformers. Future DFCG improvements include expanding language support, integrating LLMs for better code quality, enhancing handwriting and graph recognition, optimizing models for mobile use, and adding multimodal inputs like speech and sketch-based flowchart creation.

In this section, we surveyed a wide variety of image content analysis approaches for flowchart images. Table 1 summarizes these studies.

 doi:10.20944/preprints202512.0556.v1

Table 1. Table of Flowchart Schematic Extraction Scoring.

Paper	Publication Year	Method	Dataset Size	Performance
[19]	2020	ROI	25	(Accuracy) 90%
[25]	2020	GRCNN	2490	(Accuracy) 94.1%
[33]	2020	Darknet, YOLOv3	161	(mAP) 99.82%
[28]	2020	Otsu, beam search Dilation, RDP	20	(Accuracy) 86.52%
[40]	2022	LSTM	50	(BLEU) 55.68%
[59]	2022	InstGNN, DGL	2957	(Accuracy) 76.44%
[75]	2022	FR-DETR	1000	(F1-Score) 98.7%
[45]	2023	easyOCR, Code-T5	11,884	(BLEU) 21.4%
[54]	2023	S-Distil-BERT	50	(Accuracy) 75.59%
[52]	2024	EasyOCR, OpenCV, Llama 2	Unknown	(Success Rate) 75%
[85]	2024	LS-DETR	1685	(mAP) 99.18%
[68]	2024	LSTM	550	(F1-Score) 86.49%
[60]	2024	GPT-4o, Qwen2-VL	Unknown	(F1-Score) 88.23%
[92]	2025	GPT-4o, RAG	Unknown	(Accuracy) 100%
[96]	2025	GPT-4o	Unknown	(Accuracy) 92%
[101]	2025	GNN, R-CNN, Keras OCR	875	(mAP) 95.8%
[61]	2025	OCR, DAMO-YOLO, GPT-4o	30	(Accuracy) 89%
[99]	2025	GPT-4o	95	(Accuracy) 63%
[62]	2025	Qwen2-VL, RAG	1586	(Accuracy) 71.91%

4.2.2. Flowchart Review Summary

Of the 19 methods reviewed, we observe that 8 reported performance metrics that demonstrate a strong ability to successfully conduct this task, while 11 of the surveyed papers reported results that we would deem insufficient for many real-world applications. Despite substantial progress in this domain, we observe that none of the techniques have demonstrated consistently high performance across all tested flowchart images. Thus, we conclude that further improvement is needed in areas such as dataset quality, arrow and line recognition, and comprehensive flowchart extraction.

4.3. Block Diagram Analysis

4.3.1. Review of Identified Papers

Our survey identified only very few research works specifically related to extracting content from Block Diagrams. We believe that this stems from the close relationship between flowcharts and block

 doi:10.20944/preprints202512.0556.v1

diagrams, as most works related to image content analysis can utilize the work already performed for flowcharts.

In 2022, BloSum [106] was published, which utilizes OCR and deep learning models for extracting block diagram content. It builds upon Chart-Text [107], which generates descriptions from chart-based images. It excels at chart type classification, achieving 99.72% classification accuracy. However, it is not well-adapted to producing suitable descriptions for different chart types. Instead, T5 [108] is utilized in BloSum for generating text sentences.

A custom dataset of complex Computerized Block Diagram (CBD) was created from search results for block diagrams. 96 images were selected and subsequently used for testing of BloSum. The dataset annotation was performed using the LabelImg tool. The authors of BloSum also tested their approach on additional datasets called FC_A, which contains 171 hand-drawn images, and FC_B, comprising 196 hand-drawn diagrams. The BloSum model also used Faster R-CNN with Feature Pyramid Network (FPN) [109], along with Inception-ResNet-V2 [110] for shape prediction. EasyOCR was used for text prediction, and a Hough Line Transformer (HLT) was used for arrow prediction. A new architecture called Triplet Generator (TG) connected the shapes and arrows in the form of triplets. Image captioning was performed by using the Show, Attend, and Tell (SAT) model. [111]. Finally, T5 and Bidirectional and Auto-Regressive Transformer (BART) [112] language models were both modified and utilized within BloSum for text extraction.

For CBD, BloSum with the T5 model, which was modified to utilize 770M model parameters overall, achieved the highest BLEU score of 42.18% and ROUGE-1 score of 80.78%, along with a low perplexity (PPL) score of 7.54. BloSum with the same T5 model also scored the highest BLEU score and ROUGE-1 scored 51.73% and 88.24% for FC_A, respectively. For FC_B, BloSum with the same T5 model also scored the highest for BLEU at 53.17% and ROUGE-1 at 89.56%. Although this method employs traditional extraction techniques in conjunction with transformers, it nonetheless demonstrates that significant work remains to be conducted to further improve the performance of block diagram extraction. One area the authors identified is the need to further explore alternative methods for extracting shapes and arrows, as well as improving support for more complex block diagrams.

In 2024, another framework for block diagram extraction was introduced called BlockNet [113]. This paper referenced BloSum [106], due to the similarity of block diagram extraction. The authors state that BloSum can suffer from hallucinations due to the use of language models in BloSum. They also point out the smaller data set size of only 463 images total as a contributing factor. BlockNet, by comparison, uses a dataset of 76K images. Other related work mentioned involved symbol detection using Faster R-CNN [114].

To create the dataset used by BlockNet, Mermaid code was utilized for block diagram creation and visualization, along with adding noise to the images to enhance the robustness of the resulting BlockNet approach. GPT-3.5 was used for the annotation of the block diagrams. The dataset has 83,394 English and Korean block diagrams notated as BD-EnKo dataset. The authors divide their technique into 3 key components: a local extractor, a global extractor, and an integrator. In the local extractor stage, YOLOv5 was utilized for object detection, trained on the aforementioned CBD, FC_A, and FC_B datasets. The Pororo OCR package was used due to its support of both English and Korean text. A custom algorithm called BlockSplit divides block diagrams into sub-diagrams and integrates the Triplet Generator to represent shapes, arrows, and their relationships to each other. In the global information extraction stage, the Swin Transformer (ST) [115] was employed as the visual decoder, and BART was used for text decoding. In the integrator stage, to minimize errors, GPT-4V was used to combine the local and global data and to output a summary of the given block diagram.

The BLEU score, ROUGE-1 score, and BLEURT score were 72.31%, 88.9%, and 41.0%, respectively, using the CBD dataset. This demonstrates that using a larger dataset and a more complex framework can achieve higher precision and accuracy. Nonetheless, the reliability of these approaches remains a significant challenge when applied to a wide range of diagram images of varying quality.

 doi:10.20944/preprints202512.0556.v1

4.3.2. Block Diagram Review Summary

Table 2 shows the high-level comparison of the surveyed approaches. For both BloSum and BlockNet, the published results demonstrate the viability of the presented approaches, while also highlighting the potential for further performance improvements. Both methods indicate that they encountered significant errors at the end stage, making them unsuitable for everyday use.

Table 2. Table of Block Diagram Extraction Scoring.

Paper	Publication Year	Method	Dataset Size	Performance
[106]	2022	Faster R-CNN, T5 EasyOCR, HLT	463	(BLEU) 42.8%
[113]	2024	YOLOv5, Pororo OCR BART, ST, GPT-4V	76k	(BLEU) 72.31%

4.4. Electrical Circuit Diagram Analysis

4.4.1. Review of Identified Papers

While flowcharts and block diagrams are seen as closely related, electrical circuit diagrams and schematics are significantly different. Image content extraction from electrical circuit diagrams often targets circuit simulators via netlist generation, and thus demands very high extraction accuracy. Netlists are files that utilize a text representation to describe an electrical circuit, including its components, their parameters, and the connections between them.

In 2020, the work in [116] describes a circuit diagram approach that expands upon the works of [117–119]. Their approach starts by first binarizing the image and performing noise reduction. This is followed by localization, to extract circuit components without any of the diagram's wires after applying the Run Length Smoothing Algorithm (RLSA) [120]. The authors also generated their own dataset of 60 hand-drawn circuit diagrams, as they were unable to locate any publicly accessible datasets applicable to their research. A localization accuracy evaluation was performed, yielding an overall accuracy of 91.28% in extracting circuit components.

The work in [121], called Img2Sim and published in 2022, uses an Artificial Neural Network (ANN) [122] approach to convert circuit schematics into a netlist. This work builds upon a prior approach published in [123] that used deep learning to recognize circuit components, and [124], which involved identifying hand-drawn circuit images using OCR. The paper [125] is also referenced. However, while these methods primarily focus on passive components, Img2Sim also aims to support active components.

A dataset comprising 330 circuit diagrams was used for testing, utilizing YOLOv5 for element recognition, a Hough Transform for node detection, and OCR for text recognition, with the final results generated as a netlist. Their presented method detects circuit components with an accuracy of 98%, and generates correct netlists with an accuracy of 90%. This demonstrates that, while there is room for improvement in circuit diagram extraction from images, significantly more research is needed to accurately represent these diagrams using netlist files post-extraction, in order to support the intended downstream applications.

Another method, presented in 2022 [126], utilizes a DNN to generate netlists from hand-drawn circuit diagrams, and the CNN-based classification approach published in [125] to classify circuit components. Other works, such as [124,127–129] influenced their research and the published extraction approach. YOLOv5 is utilized for component recognition, adaptive thresholding, and bounding box extraction for terminal recognition, as well as k-means clustering for detecting intersections. The Hough Transform is applied for line detection. This approach was implemented using PyTorch and subsequently evaluated using a dataset comprised of 388 images. It achieved an average mAP score with 0.5 IoU of 99.19%. Their results indicate good extraction performance for simple circuit diagrams using passive components, but more research is needed to improve the accuracy of more complex diagrams involving active components and Integrated Circuit (IC) components. Additionally, these

 doi:10.20944/preprints202512.0556.v1

methods require further evaluation using larger datasets spanning a wider range of circuit diagram representations.

In 2022, [130] utilized machine learning methods to convert hand-drawn circuit diagrams into netlists. It builds on the work in [116] for component detection in hand-drawn circuit diagrams, as well as the work in [124]. A custom dataset containing hundreds of images of resistors, diodes, inductors, and op-amps was used for evaluation. These images were converted to grayscale to facilitate noise removal prior to binarization. Their presented scheme incorporates dilation and skeletonization [131], as well as OCR for text extraction, the use of Histogram of Gradients (HOG) [132] and Support Vector Machine (SVM) [133] for classifying the components using OpenCV, along with the identification of the resulting nodes. The method achieved an average precision of 94%. Even though this score is high, since their scheme focused only on simple passive components, their approach does not scale well for more complex circuit diagrams.

The authors of [134], published in 2021, developed a method that leverages augmented reality techniques to convert hand-drawn circuit diagrams into other representations using a smartphone. The author's work expands upon [135,136], which used an SVM approach for circuit recognition. Feature extraction is performed using a custom algorithm that combines region proposal using SURF, Eigen, and FAST. Dilation and Hough Transform are also incorporated as part of the processing workflow. A capsule network architecture (CapsNet) [137] is used instead of a CNN due to limitations with CNNs, as discussed by the authors in their paper, primarily related to the significantly higher training dataset size required for CNNs compared to CapsNets. The final step in their approach is circuit simulation, which determines whether the extraction produces a viable netlist, concluding with the export of the netlist generated by the previous steps. With a custom dataset comprising 800 circuit diagrams used for evaluation, and utilizing the mAP scoring metric with an IoU threshold of 0.3, the resulting score for this approach is 93.64%.

Circuit diagram extraction has attracted increasing attention in recent years, driven by advancements in machine learning that enable more accurate recognition and interpretation of complex schematics, as shown in Figure 9. Using Litmaps, this analysis illustrates the relationships and interdependencies among the reviewed papers.

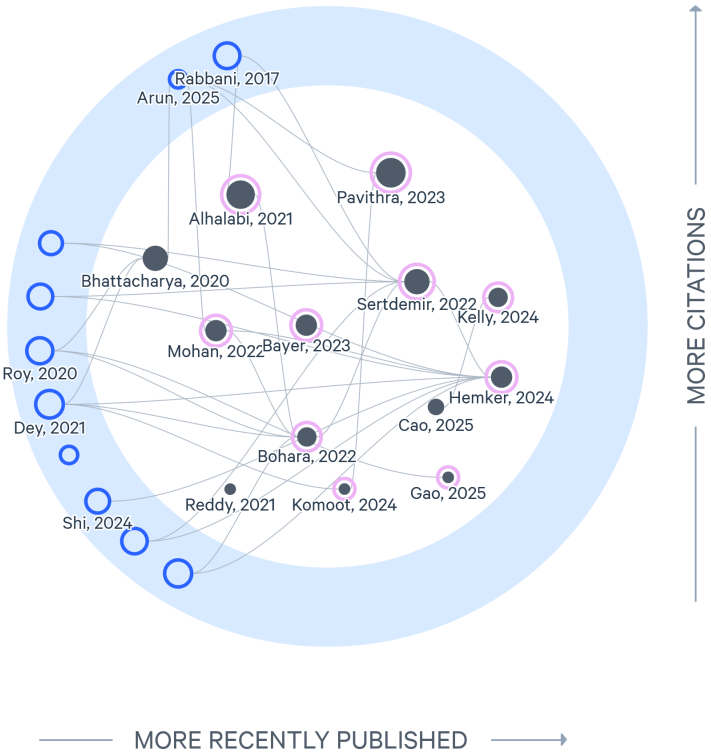

Figure 9. Citation Graph Visualization For Circuit Diagrams.

 doi:10.20944/preprints202512.0556.v1

The authors of [138–140] each developed their own circuit extraction techniques that advance upon [129] by widening its scope, which focused exclusively on text extraction from hand-drawn diagrams. The techniques discussed in [138,141] also reference the research [127], which focuses on recognizing hand-drawn schematics. Published in 2022, the work in [138] presents a model that can analyze hand-drawn or computer-generated schematics specifically of power converter circuits, generate netlists from them, and simulate them in real-time. It is inspired by older works, such as [121,123,126,128,130,134]. For evaluating their presented technique, they developed their own dataset. The images underwent noise filtering to help build a more robust system. A variant of YOLO called You Only Look Once Representation (YOLOR) [142] was used to detect components. Hough Transform was used to detect lines. Edge detection is performed to improve the performance of the HT. Node recognition was conducted using a K-Means clustering technique [143]. The resulting netlist is then evaluated and simulated using PySpice. The technique's assessed mAP0.5 score was 91.6%. This work's biggest shortcoming is in diagrams involving surface-mount components in Printed Circuit Board (PCB) schematics. The authors also compared their performance obtained using PySpice with performance metrics from LTSpice, which confirmed the accuracy of the generated netlists for diagrams where extraction was conducted successfully.

In 2023, Bayer *et al.* published [140], which aimed to facilitate extracting component labels and junctions from circuit diagram extraction using keypoint extraction [144] and instance segmentation [145]. For evaluation, it utilized the Circuit Graph Hand-Draw (CGHD) dataset, containing 2,304 hand-drawn circuit diagrams. Additionally, their approach is inspired by [122,124,126]. The presented approach first separates the background from objects using binarization maps, which are then further refined using the LabelME annotation tool to clean up the coarse polygon masks. The image was then simplified by layering coarse polygons with binarization maps. Contour detection was employed to enhance the precision of the polygons, reducing ambiguity in the annotations, before proceeding to contour detection. Keypoint Generation precisely identifies connection points on electrical symbols after applying the morphological operator to reduce the impact of noise. A technique called Keypoint Port Assignment is then performed to link detected connection points on electrical symbols to their correct terminals, with the final step being the creation of electrical circuit graph representations.

The results obtained from applying this approach with the CGHD dataset demonstrate a mask accuracy of 94%. This demonstrates that traditional computer vision information extraction approaches can yield commendable results. However, they are often outperformed by modern ML-based approaches, as shown in this survey.

Closely related to [138], the authors of [139] in 2023 published a technique that combined OCR, structured pattern extraction with regular expressions, and shape matching using geometric methods, building upon their own prior work published by [124,140]. The aforementioned CGHD dataset was used for their testing. During the text detection stage, Faster R-CNN was used. For handwritten recognition, the approach relies on a combination of CNN and LSTM, as well as Regular expressions (RegEx) for text label identification. The Euclidean distance was the metric employed by the authors for text symbol assignment, and prefix resolution was used for unit of measurement resolution. The work demonstrated promising results. It excels at resistors, capacitors, and voltage labels. However, subscript annotations and negation lines are not yet supported, and additional work is needed to address these and related capabilities. More work is also needed for the normalization of electrical values. Lastly, a consistency evaluation must be conducted to confirm the integrity of the graph's generated output representation.

This 2024 research paper [146] proposed another technique for circuit diagram extraction, influenced by earlier techniques such as [147], which was created to help to recognize faulty analog circuits using neural networks, as well as [126,128,148]. Following the initial image acquisition, they first convert the image into a grayscale representation. YOLOv8 is leveraged to perform text recognition with high accuracy, whereas a CNN and Line Detection are used during image segmentation to classify the structure of the circuit components and their corresponding value labels. The resulting text

 doi:10.20944/preprints202512.0556.v1

representation of the circuit diagram is used in a subsequent analysis stage to evaluate the accuracy of current and voltage levels at each node present in the circuit. 50 diagrams were used for testing, performed on a NVIDIA RTX 4070 GPU, and implemented using Python. This evaluation yielded an 84% success rate, with 42 out of 50 images extracted without errors. However, a more comprehensive evaluation using a significantly larger dataset is needed to properly assess this technique in comparison to others in this domain.

In 2024, [149] presented another technique that aims to convert schematics into netlists, based on discussions from [121,126,128,130]. Line detection was performed by using a Lightweight Convolutional Neural Network (L-CNN) [150]. EasyOCR was employed with a detection model called Character-Region Awareness For Text Detection (CRAFT). This model was fine-tuned from a model checkpoint of both SynthText and ICDAR 2015. A custom dataset containing 124 circuit diagrams was used in their performance testing to evaluate symbol detection. Although the obtained results were promising for component and line recognition, significant improvements are needed, particularly in terms of robustness for varying symbol orientations, and detecting small text labels such as pin numbers used on integrated circuit components.

A similar effort, published in 2024 in [151], aimed to address the process of extracting information from digitized circuit diagram images by using a custom OCR model, improving on [152]. Here, Otsu's method was used for binarization and skeletonization to gain a specific contour size of 1 pixel wide for the circuit. A Progressive Probabilistic Hough Transform (PPHT) [153] was then employed for recognizing lines in the circuit, and Tesseract OCR was used to read the text of each symbol. This approach was then implemented as a software solution and evaluated using a custom dataset of 15 schematics, with 3 being hand-drawn and 12 obtained from various publishers. Their testing showed that 166 out of 194 symbols in the circuit were correctly identified, resulting in an accuracy of 85.6%.

In 2025, the work in [141] evaluated circuit diagram recognition using a graph structure. This was inspired by efforts on the automatic generation of detection rules for the identification of a circuit published in [154,155], which both leverage machine learning to recognize circuit components, as well as the work published in [127]. Otsu's method is used for the preprocessing stage to perform noise reduction. It then utilizes ResNet-101 [156] and VGG for circuit retrieval. Their work used a more optimized structured graph for the circuit diagrams. GED was employed as their graph matching measurement [157], while Intersection of Union (IoU) and mAP were used for object detection metrics. For the evaluation of this technique, a dataset of 227 images was employed together with GAM-YOLO [158]. It resulted in a mAP score of 91.4%. For circuit diagram extraction, the results show significant promise compared to previous methods. Its high accuracy enables a variety of downstream tasks to be performed successfully using the circuit diagram's graph structure representation.

Additionally, a 2025 paper [159] discussed using DL for circuit diagram extraction, referencing [116,121,126,129,130] as inspirations for their work. The Kaggle dataset [160] was used for testing, with additional complex circuit symbols, such as operational amplifiers (OP-AMPs), included. The authors developed their own CNN Multi-Layer Model. The method scored a 96% test accuracy using mAP, which is significantly better than prior approaches. The authors also created a web application that allowed users to upload images and generate netlists. This work demonstrates notable improvements in the automated extraction and interpretation of circuit diagrams.

Finally, building on [151], the authors of [161] conducted an experiment that individually extracted each layer of a circuit diagram, allowing for a more accurate extraction. It aims to improve upon prior efforts such as [151] for better support of digitizing circuit diagrams. It also incorporates efforts from [162], which specifically aims to improve the detection of small objects in circuit diagrams. YOLOv7 was used for element layer detection within a circuit diagram, PaddleOCR [163] and PaddleClas image classification [164] were used for text layer extraction. The author used a connection relationship algorithm to understand the relationship between each diagram. Their framework was implemented using Python 3.9, PyTorch 1.13, and OpenCV to run on a host with Ubuntu 20.04 LTS and a single NVIDIA RTX 3090 GPU. Using a dataset containing 5,000 diagrams and an Element Feature Library

 doi:10.20944/preprints202512.0556.v1

(EFL), this method achieved an F1-score of 99.6% with diagrams featuring various backgrounds. This highly promising result shows the significant advances made in this research domain in recent years.

4.4.2. Electrical Circuit Diagram Review Summary

Table 3 contains the 14 schematic diagram extraction papers that were discussed in this part of our survey. It is our estimation that 10 out of 14 methods are suitable for various application domains, whereas the remaining 4 papers demonstrate earlier efforts that paved the way for later significant advances in the accurate and detailed extraction of electrical circuit diagrams from images. All of these efforts, however, could benefit from further evaluation using larger benchmark datasets to better compare their individual performance across the variety of algorithms and ML techniques proposed and studied over the years for this application domain.

Table 3. Table of Circuit Schematic Extraction Scoring.

Paper	Publication Year	Method	Dataset	Performance
[116]	2020	RLSA, Localization	60	(Accuracy) 91.28%
[134]	2021	CapsNet, HT	800	(mAP) 93.64%
[126]	2022	YOLOv5, HT, K-Means	388	(mAP) 99.19%
[130]	2022	OpenCV, OCR	Unknown	(Accuracy) 94%
[121]	2022	YOLOv5, OCR, HT	330	(Accuracy) 98%
[138]	2022	YOLOR, HT, K-Means	176	(mAP) 91.6%
[139]	2023	LSTM, CNN, RegEx	2304	Unknown
[140]	2023	Masked RCNN	2208	(Accuracy) 94%
[149]	2024	EasyOCR, L-CNN, CRAFT	124	(mAP) 43%
[146]	2024	CNN, OCR	50	(BLEU) 84%
[151]	2024	Tesseract OCR, PPHT	15	(BLEU) 85.6%
[159]	2025	Custom CNN	Unknown	(mAP) 96%
[141]	2025	GAM-YOLO, GED, VGG-16	227	(mAP) 91.4%
[161]	2025	YOLOv7, PaddleClas, PaddleOCR	5000	(F1-Score) 96.6%

4.5. Timing Diagram Analysis

4.5.1. Review of Identified Papers

A category of technical illustrations that stands on its own is timing diagrams. Unlike many other diagram types, the importance of line positioning is significantly elevated, and thus it mandates highly accurate extraction to accurately represent signal change timing.

A study published in 2023 developed a method named TD-Magic [165], using object detection for rising and falling edge detection, along with OCR to detect text labels. It also incorporated the use of image processing techniques to extract synchronization indications. In addition to influences from [166], its approach to text extraction from timing diagrams was inspired by work from the medical domain [167], focusing on Electrocardiogram (ECG) signal extraction. Feature recognition was performed using a Signal-Edge Detector (SED) [168] and YOLOv5 for edge recognition. PaddleOCR

 doi:10.20944/preprints202512.0556.v1

was used for text detection. The Line-and-Arrow-Detection (LAD) method was employed for identifying arrows and lines in the timing diagrams. Semantic Interpretation (SEI) [169] was employed to create annotations for each aspect of the timing diagram using Strict Partial Order (SPO) [170]. For experimentation and evaluation of their approach, the authors also utilized Formal Methods (FM). The dataset for evaluation included 29 timing diagrams from various sources. Out of the 29 timing diagrams, 23 were effectively obtained at the template stage. However, only 15 of the test cases could be processed successfully and accurately. Using the mAP scoring metric, an overall score of 76.7% was achieved. Hence, significantly more work is needed to make this approach more robust and performant. Furthermore, a larger dataset is needed for the evaluation to be more informative and applicable to different downstream application domains.

Another timing diagram extraction technique was introduced in 2025, called TD-Interpreter [171]. While it was inspired by TD-Magic [165], the only aspect they have in common is that TD-Magic and TD-Interpreter both read timing diagrams. Another inspiration for this work came from LLaVA–a Visual Language Model (VLM) [172]. TD-Interpreter utilizes a ViT to process images and extract visual features. The LLaVA VLM processes the extracted information, performing reasoning and generating responses. Low-rank-adaptation (LoRA) [173] was employed for fine-tuning the layers of the proposed algorithm. A custom dataset was created for testing using caption-based data. It is comprised of 4942 timing diagram images. Additionally, for reasoning-based testing, 5292 timing diagram images were used. The evaluations were conducted on a system with 8 NVIDIA A800 GPUs, and achieved a BLEU-4 score of 95.9%, and ROUGE-1 score of 96.7%. Thus, the authors demonstrated that, with the use of transformers and a large dataset, along with access to sufficiently powerful computational hardware, it is feasible to achieve highly precise and accurate timing diagram extraction.

4.5.2. Timing Diagram Review Summary

Table 4 shows the comparison of TD-Magic and TD-Interpreter. The applicability of TD-Magic is limited by its low score achieved over a comparatively small dataset. However, it did succeed on several test cases and thus warrants further research to improve its performance, especially for use cases with limited computational resources such as in edge devices. The TD-Interpreter, on the other hand, employs a significantly more advanced VLM-driven approach that includes reasoning capabilities and was tested on a much larger dataset. Thus, among the two, TD-Interpreter is a more promising approach that, given the required computational resources, will be able to achieve significantly better results for downstream applications depending on accurate timing diagram extraction.

Table 4. Table of Timing Diagram Extraction Scoring.

Paper	Publication Year	Method	Dataset	Overall Score
[165]	2023	Paddle OCR, YOLOv5 SED, LAD	29	(mAP) 76.7%
[171]	2025	LlaVA, ViT, LoRA	10,234	(ROUGE-1) 96.7%

4.6. Comparative Observations and Summary Across Illustration Types

Based on the reviewed research papers across all four technical diagram types, Transformer-based approaches consistently demonstrate strong performance in terms of accuracy, adaptability, and robustness, regardless of diagram type. While these Transformer-based approaches are computationally expensive, more lightweight models are available for application scenarios where efficiency outweighs accuracy, such as in edge computing or mobile applications. Object detection frameworks such as YOLO and LLMs such as the various GPT models also offer reliable performance across diverse scenarios. Traditional extraction methods remain useful for specific tasks, but combining them with modern techniques generally yields the most effective results across a wide range of situations.

 doi:10.20944/preprints202512.0556.v1

It is important to note that different diagram types attract varying levels of research attention, which influences how rapidly performance improvements can be achieved. Flowcharts are the most extensively examined type of technical diagram, which has led to more advanced, fine-tuned, and higher-accuracy extraction techniques, along with publicly accessible benchmark datasets. Circuit diagrams have also received considerable research focus. However, progress in this area remains constrained by the reliance on small, domain-specific, custom-built datasets. In contrast, timing diagrams and block diagrams have received relatively limited attention, leading to slower methodological advancements and lower overall extraction performance.

A notable issue identified in the reviewed research is the scarcity of publicly available datasets that can be reliably used for testing and evaluating extraction methods. Most studies relied on custom datasets tailored to their specific needs, which may introduce bias and limit the generalizability of their results. The availability of a standard public dataset could enable more robust and applicable benchmarking, supporting fairness in comparing various image content analysis approaches.

5. Conclusions

With the rapid proliferation of LLMs and Agentic AI in our everyday lives comes a resurgence in the need for multimodal data access to inform queries and produce accurate responses. As a result, we see growing research activities in image content analysis and extraction for complex technical illustrations, such as flowcharts, block diagrams, schematics, and timing diagrams—illustrations where a significant portion of the information is represented not just by text, but by the contained shapes, their styling, and their relationships. To cope with increasingly complex diagrams and the need to reduce extraction errors, the techniques utilized for extracting information from these technical illustrations must also evolve. The rise of AI-powered techniques resulted in a number of highly promising image content analysis techniques, achieving higher accuracy and richer contextual representations than ever before, across a variety of diagram types.

Our survey found that the majority of recent research activities focused on flowcharts, while circuit schematics represent the second-most studied diagram type. In contrast, block diagrams and timing diagrams have received only limited interest, with only two studies dedicated to each of these categories in recent years.

While the advancements made so far are promising, significant work remains as future research avenues to optimize extraction methods for various diagram types, reduce extraction errors, and improve the cost and complexity of these algorithms. Reducing complexity, in particular, is crucial for accelerating information extraction in an era where an ever-increasing amount of multimodal information is being generated. Similarly, for applications where accuracy is paramount, further research is necessary to enhance information extraction in terms of both completeness and accuracy. This is vital for reducing hallucinations and downstream generative errors in Agentic AI systems. Finally, we also observed that the significant lack of benchmark datasets severely impedes objective comparisons across the various techniques. While several research works utilize larger datasets with thousands of images, others use much smaller datasets, sometimes comprised of only a few dozen images. Thus, there is a strong need to establish and publicly release benchmark datasets that span the full range of technical diagram types, with sufficient representation in terms of complexity, color usage, multilingual aspects, diagram correctness, and other features.

This survey aims to provide both a cross-section of research work related to information extraction from technical diagrams, as well as inspire researchers to pursue the various open research questions in this field, in an effort to advance the accuracy, fidelity, and efficiency of these techniques, because there are numerous applications that would be enabled by these capabilities.

Author Contributions: Conceptualization, N.B., M.H., M.B., and H.S.; Investigation, N.B. and M.H; Writing—original draft preparation, N.B., M.H, M.B, and H.S.; Writing—review and editing, N.B., M.H., M.B., and H.S.; Supervision, M.H, M.B., and H.S.; Project administration, M.H. and H.S. All authors have read and agreed to the published version of the manuscript.

 doi:10.20944/preprints202512.0556.v1

Funding: This research received no external funding.

Data Availability Statement: No new datasets were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Arslan, M.; Ghanem, H.; Munawar, S.; Cruz, C. A Survey on RAG with LLMs. *Procedia computer science* **2024**, *246*, 3781–3790.
2. Jamieson, L.; Moreno-García, C.F.; Elyan, E. A review of deep learning methods for digitisation of complex documents and engineering diagrams. *Artificial Intelligence Review* **2024**, *57*, 136. https://doi.org/10.1007/s10462-024-10779-2.
3. Mittal, R.; Garg, A. Text extraction using OCR: A Systematic Review. In Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 357–362. https://doi.org/10.1109/ICIRCA48905.2020.9183326.
4. Kumar, G.; Bhatia, P.K. A Detailed Review of Feature Extraction in Image Processing Systems. In Proceedings of the 2014 Fourth International Conference on Advanced Computing & Communication Technologies, 2014, pp. 5–12. https://doi.org/10.1109/ACCT.2014.74.
5. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights into imaging* **2018**, *9*, 611–629.
6. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* **2020**, *404*, 132306.
7. Demiss, B.A.; Elsaigh, W.A. Application of novel hybrid deep learning architectures combining convolutional neural networks (CNN) and recurrent neural networks (RNN): construction duration estimates prediction considering preconstruction uncertainties. *Engineering Research Express* **2024**, *6*, 032102.
8. Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. A Survey on Vision Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2023**, *45*, 87–110. https://doi.org/10.1109/TPAMI.2022.3152247.
9. Li, M.; Lv, T.; Chen, J.; Cui, L.; Lu, Y.; Florencio, D.; Zhang, C.; Li, Z.; Wei, F. Trocr: Transformer-based optical character recognition with pre-trained models. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2023, Vol. 37, pp. 13094–13102.
10. Shao, T.; Guo, Y.; Chen, H.; Hao, Z. Transformer-based neural network for answer selection in question answering. *IEEE Access* **2019**, *7*, 26146–26156.
11. Lin, C.Y. Rouge: A package for automatic evaluation of summaries. In Proceedings of the Text summarization branches out, 2004, pp. 74–81.
12. Saadany, H.; Orasan, C. BLEU, METEOR, BERTScore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text. *arXiv preprint arXiv:2109.14250* **2021**.
13. Reiter, E. A structured review of the validity of BLEU. *Computational Linguistics* **2018**, *44*, 393–401.
14. Ren, S.; Guo, D.; Lu, S.; Zhou, L.; Liu, S.; Tang, D.; Sundaresan, N.; Zhou, M.; Blanco, A.; Ma, S. Codebleu: a method for automatic evaluation of code synthesis. *arXiv preprint arXiv:2009.10297* **2020**.
15. Sellam, T.; Das, D.; Parikh, A.P. BLEURT: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696* **2020**.
16. Yang, M.; Zhu, J.; Li, J.; Wang, L.; Qi, H.; Li, S.; Daxin, L. Extending BLEU Evaluation Method with Linguistic Weight. In Proceedings of the 2008 The 9th International Conference for Young Computer Scientists, 2008, pp. 1683–1688. https://doi.org/10.1109/ICYCS.2008.362.
17. Henderson, P.; Ferrari, V. End-to-end training of object class detectors for mean average precision. In Proceedings of the Asian conference on computer vision. Springer, 2016, pp. 198–213.
18. Hicks, S.A.; Strümke, I.; Thambawita, V.; Hammou, M.; Riegler, M.A.; Halvorsen, P.; Parasa, S. On evaluation metrics for medical applications of artificial intelligence. *Scientific reports* **2022**, *12*, 5979.
19. Leena, C.; Ganesh, M. Generating Graph from 2D Flowchart using Region-Based Segmentation. In Proceedings of the 2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS), 2020, pp. 1–5. https://doi.org/10.1109/SCEECS48394.2020.165.
20. Vasudevan, B.G.; Dhanapanichkul, S.; Balakrishnan, R. Flowchart knowledge extraction on image processing. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 4075–4082. https://doi.org/10.1109/IJCNN.2008.4634384.

 doi:10.20944/preprints202512.0556.v1

21. Herrera-Camara, J.I.; Hammond, T. Flow2Code: from hand-drawn flowcharts to code execution. In Proceedings of the Proceedings of the Symposium on Sketch-Based Interfaces and Modeling, 2017, pp. 1–13.
22. Bangare, S.L.; Dubal, A.; Bangare, P.S.; Patil, S. Reviewing Otsu's method for image thresholding. *International Journal of Applied Engineering Research* **2015**, *10*, 21777–21783.
23. Gould, S.; Gao, T.; Koller, D. Region-based segmentation and object detection. *Advances in neural information processing systems* **2009**, *22*.
24. Mohamad, M.; Saman, M.Y.M.; Hitam, M.S.; Telipot, M. A review on OpenCV. *Terengganu: Universitas Malaysia Terengganu* **2015**, *3*, 1.
25. Cheng, L.; Yang, Z. GRCNN: Graph Recognition Convolutional Neural Network for synthesizing programs from flow charts. *arXiv preprint arXiv:2011.05980* **2020**.
26. Gulwani, S. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices* **2011**, *46*, 317–330.
27. Gansner, E.R. Drawing graphs with Graphviz. *Technical report, AT&T Bell Laboratories, Murray, Tech. Rep, Tech. Rep.* **2009**.
28. Chakraborty, S.; Paul, S.; Masudul Ahsan, S.M. A Novel Approach to Rapidly Generate Document from Hand Drawn Flowcharts. In Proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 702–705. https://doi.org/10.1109/TENSYMP50017.2020.9231033.
29. Vogel, M.; Warnecke, T.; Bartelt, C.; Rausch, A. Scribbler—drawing models in a creative and collaborative environment: from hand-drawn sketches to domain specific models and vice versa. In Proceedings of the Proceedings of the Fifteenth Australasian User Interface Conference-Volume 150, 2014, pp. 93–94.
30. MENG, W.K. Development of program flowchart drawing tool. Technical report, Universiti Tunku Abdul Rahman, Malaysia, 2016.
31. Ramer-Douglas-Peucker, N.p. Ramer-douglas-peucker algorithm. *Online Referencing* **1972**.
32. Scheidl, H.; Fiel, S.; Sablatnig, R. Word beam search: A connectionist temporal classification decoding algorithm. In Proceedings of the 2018 16th International conference on frontiers in handwriting recognition (ICFHR). IEEE, 2018, pp. 253–258.
33. Chakraborti, A.; Naik, A.; Pansare, A.; Pant, U. Extracting Flowchart Features into a Structured Representation. Technical report, Mukesh Patel School of Technology Management and Engineering, 2020.
34. Chen, Q.; Shi, D.; Feng, G.; Zhao, X.; Luo, B. On-line handwritten flowchart recognition based on logical structure and graph grammar. In Proceedings of the 2015 5th International Conference on Information Science and Technology (ICIST), 2015, pp. 424–429. https://doi.org/10.1109/ICIST.2015.7289009.
35. Supaartagorn, C. Web application for automatic code generator using a structured flowchart. In Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2017, pp. 114–117.
36. Carlisle, M.C.; Wilson, T.A.; Humphries, J.W.; Hadfield, S.M.; et al. Raptor: introducing programming to non-majors with flowcharts. *Journal of Computing Sciences in Colleges* **2004**, *19*, 52–60.
37. Adarsh, P.; Rathi, P.; Kumar, M. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In Proceedings of the 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 687–694. https://doi.org/10.1109/ICACCS48705.2020.9074315.
38. Omar, Z.M.; Ibrahim, J. An overview of Darknet, rise and challenges and its assumptions. *International Journal of Computer Science and Information Technology* **2020**, *8*, 110–116.
39. Pebrianto, W.; Mudjirahardjo, P.; Pramono, S.H.; Setyawan, R.A.; et al. YOLOv3 with spatial pyramid pooling for object detection with unmanned aerial vehicles. *arXiv preprint arXiv:2305.12344* **2023**.
40. Liu, Z.; Hu, X.; Zhou, D.; Li, L.; Zhang, X.; Xiang, Y. Code generation from flowcharts with texts: A benchmark dataset and an approach. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2022, 2022, pp. 6069–6077.
41. Alizadegan, H.; Rashidi Malki, B.; Radmehr, A.; Karimi, H.; Ilani, M.A. Comparative study of long short-term memory (LSTM), bidirectional LSTM, and traditional machine learning approaches for energy consumption prediction. *Energy Exploration & Exploitation* **2025**, *43*, 281–301.
42. Vrahatis, A.G.; Lazaros, K.; Kotsiantis, S. Graph attention networks: a comprehensive review of methods and applications. *Future Internet* **2024**, *16*, 318.
43. Yin, P.; Neubig, G. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. *arXiv preprint arXiv:1810.02720* **2018**.
44. Khan, S.; Naseer, M.; Hayat, M.; Zamir, S.W.; Khan, F.S.; Shah, M. Transformers in vision: A survey. *ACM computing surveys (CSUR)* **2022**, *54*, 1–41.

 doi:10.20944/preprints202512.0556.v1

45. Shukla, S.; Gatti, P.; Kumar, Y.; Yadav, V.; Mishra, A. Towards making flowchart images machine interpretable. In Proceedings of the International Conference on Document Analysis and Recognition. Springer, 2023, pp. 505–521.
46. Tannert, S.; Feighelstein, M.G.; Bogojeska, J.; Shtok, J.; Arbelle, A.; Staar, P.W.; Schumann, A.; Kuhn, J.; Karlinsky, L. FlowchartQA: the first large-scale benchmark for reasoning over flowcharts. In Proceedings of the Proceedings of the 1st Workshop on Linguistic Insights from and for Multimodal Language Processing, 2023, pp. 34–46.
47. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.
48. Montellano, C.D.B.; Garcia, C.; Leija, R.O.C. Recognition of handwritten flowcharts using convolutional neural networks. *International Journal of Computer Applications* **2022**, *184*, 37–41.
49. Mukhopadhyay, P.; Chaudhuri, B.B. A survey of Hough Transform. *Pattern Recognition* **2015**, *48*, 993–1010.
50. Reddy, P.Y.; Sri Satya Aarthi, N.; Pooja, S.; Veerendra, B.; D, S. Enhancing Code Intelligence with CodeT5: A Unified Approach to Code Analysis and Generation. In Proceedings of the 2025 International Conference on Artificial Intelligence and Data Engineering (AIDE), 2025, pp. 135–140. https://doi.org/10.1109/AIDE64228.2025.10987356.
51. Baek, Y.; Lee, B.; Han, D.; Yun, S.; Lee, H. Character region awareness for text detection. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 9365–9374.
52. Darda, A.; Jain, R. Code Generation from Flowchart using Optical Character Recognition & Large Language Model. *Authorea Preprints* **2024**.
53. Supaartagorn, C. Web application for automatic code generator using a structured flowchart. In Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 114–117. https://doi.org/10.1109/ICSESS.2017.8342876.
54. Ghosh, S.; Pratihar, S.; Chatterji, S.; Basu, A. Matching of hand-drawn flowchart, pseudocode, and english description using transfer learning. *Multimedia Tools and Applications* **2023**, *82*, 27027–27055.
55. Ray, S.; Herrera-Cámara, J.I.; Runyon, M.; Hammond, T. Flow2code: Transforming hand-drawn flowcharts into executable code to enhance learning. In *Inspiring Students with Digital Ink: Impact of Pen and Touch on Education*; Springer, 2019; pp. 79–103.
56. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* **2023**.
57. Miyao, H.; Maruyama, R. On-Line Handwritten flowchart Recognition, Beautification and Editing System. In Proceedings of the 2012 International Conference on Frontiers in Handwriting Recognition, 2012, pp. 83–88. https://doi.org/10.1109/ICFHR.2012.250.
58. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* **2019**.
59. Yun, X.L.; Zhang, Y.M.; Yin, F.; Liu, C.L. Instance GNN: A Learning Framework for Joint Symbol Segmentation and Recognition in Online Handwritten Diagrams. *IEEE Transactions on Multimedia* **2022**, *24*, 2580–2594. https://doi.org/10.1109/TMM.2021.3087000.
60. Ye, J.; Dash, A.; Yin, W.; Wang, G. Beyond end-to-end vlms: Leveraging intermediate text representations for superior flowchart understanding. *arXiv preprint arXiv:2412.16420* **2024**.
61. Omasa, T.; Koshihara, R.; Morishige, M. Arrow-Guided VLM: Enhancing Flowchart Understanding via Arrow Direction Encoding. *arXiv preprint arXiv:2505.07864* **2025**.
62. Soman, S.; Ranjani, H.; Roychowdhury, S.; Sastry, V.D.S.N.; Jain, A.; Gangrade, P.; Khan, A. A Graph-based Approach for Multi-Modal Question Answering from Flowcharts in Telecom Documents. *arXiv preprint arXiv:2507.22938* **2025**.
63. Pan, H.; Zhang, Q.; Caragea, C.; Dragut, E.; Latecki, L.J. Flowlearn: Evaluating large vision-language models on flowchart understanding. *arXiv preprint arXiv:2407.05183* **2024**.
64. Singh, S.; Chaurasia, P.; Varun, Y.; Pandya, P.; Gupta, V.; Gupta, V.; Roth, D. FlowVQA: Mapping multimodal logic in visual question answering with flowcharts. *arXiv preprint arXiv:2406.19237* **2024**.
65. Carruthers, S.; Thomas, A.; Kaufman-Willis, L.; Wang, A. Growing an accessible and inclusive systems design course with PlantUML. In Proceedings of the Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, 2023, pp. 249–255.

 doi:10.20944/preprints202512.0556.v1

66. Bordes, F.; Pang, R.Y.; Ajay, A.; Li, A.C.; Bardes, A.; Petryk, S.; Mañas, O.; Lin, Z.; Mahmoud, A.; Jayaraman, B.; et al. An introduction to vision-language modeling. *arXiv preprint arXiv:2405.17247* **2024**.
67. Wang, P.; Bai, S.; Tan, S.; Wang, S.; Fan, Z.; Bai, J.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191* **2024**.
68. Arbaz, A.; Fan, H.; Ding, J.; Qiu, M.; Feng, Y. GenFlowchart: parsing and understanding flowchart using generative AI. In Proceedings of the International Conference on Knowledge Science, Engineering and Management. Springer, 2024, pp. 99–111.
69. WU, X.H.; QU, M.C.; Liu, Z.Q.; Li, J.Z. A code automatic generation algorithm based on structured flowchart. *Appl. Math* **2012**, *6*, 1S–8S.
70. Raghu, D.; Agarwal, S.; Joshi, S.; et al. End-to-end learning of flowchart grounded task-oriented dialogs. *arXiv preprint arXiv:2109.07263* **2021**.
71. Saoji, S.; Eqbal, A.; Vidyapeeth, B. Text recognition and detection from images using pytesseract. *J Interdiscip Cycle Res* **2021**, *13*, 1674–1679.
72. Pavlidis, T.; Zhou, J. Page segmentation and classification. *CVGIP: Graphical models and image processing* **1992**, *54*, 484–496.
73. Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A.C.; Lo, W.Y.; et al. Segment anything. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision, 2023, pp. 4015–4026.
74. Philippovich, V.A.; Philippovich, A.Y. Modern Approaches to Extraction Text Data From Documents: Review, Analysis and Practical Implementation. In Proceedings of the 2025 7th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), 2025, pp. 1–6. https://doi.org/10.1109/REEPE63962.2025.10970923.
75. Sun, L.; Du, H.; Hou, T. FR-DETR: End-to-End Flowchart Recognition With Precision and Robustness. *IEEE Access* **2022**, *10*, 64292–64301. https://doi.org/10.1109/ACCESS.2022.3183068.
76. Schäfer, B.; Stuckenschmidt, H. Arrow R-CNN for Flowchart Recognition. In Proceedings of the 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), 2019, Vol. 1, pp. 7–13. https://doi.org/10.1109/ICDARW.2019.00007.
77. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European conference on computer vision. Springer, 2020, pp. 213–229.
78. Shehzadi, T.; Hashmi, K.A.; Stricker, D.; Afzal, M.Z. Object detection with transformers: A review. *arXiv preprint arXiv:2306.04670* **2023**.
79. Bebis, G.; Georgiopoulos, M. Feed-forward neural networks. *Ieee Potentials* **2002**, *13*, 27–31.
80. Piroi, F.; Hanbury, A. Multilingual patent text retrieval evaluation: CLEF–IP. In *Information Retrieval Evaluation in a Changing World: Lessons Learned from 20 Years of CLEF*; Springer, 2019; pp. 365–387.
81. Miyao, H.; Maruyama, R. On-line handwritten flowchart recognition, beautification and editing system. In Proceedings of the 2012 International Conference on Frontiers in Handwriting Recognition. IEEE, 2012, pp. 83–88.
82. Bresler, M.; Van Phan, T.; Prusa, D.; Nakagawa, M.; Hlavác, V. Recognition system for on-line sketched diagrams. In Proceedings of the 2014 14th International Conference on Frontiers in Handwriting Recognition. IEEE, 2014, pp. 563–568.
83. Zhuang, D.; Xu, H.; Guo, X.; Zheng, Y.; Wang, S.; Zhao, J. Mitigating Spatial Disparity in Urban Prediction Using Residual-Aware Spatiotemporal Graph Neural Networks: A Chicago Case Study. In Proceedings of the Companion Proceedings of the ACM on Web Conference 2025, 2025, pp. 2351–2360.
84. Pandey, G.K.; Srivastava, S. ResNet-18 comparative analysis of various activation functions for image classification. In Proceedings of the 2023 International Conference on Inventive Computation Technologies (ICICT). IEEE, 2023, pp. 595–601.
85. Wang, H.; Gao, S.; Zhang, X. A method for analyzing handwritten program flowchart based on detection transformer and logic rules. *International Journal on Document Analysis and Recognition (IJDAR)* **2024**, pp. 1–18.
86. Bresler, M.; Průša, D.; Hlaváč, V. Recognizing Off-Line Flowcharts by Reconstructing Strokes and Using On-Line Recognition Techniques. In Proceedings of the 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2016, pp. 48–53. https://doi.org/10.1109/ICFHR.2016.0022.

 doi:10.20944/preprints202512.0556.v1

87. Ye, M.; Zhang, J.; Zhao, S.; Liu, J.; Du, B.; Tao, D. Dptext-detr: Towards better scene text detection with dynamic points in transformer. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2023, Vol. 37, pp. 3241–3249.
88. Xu, X.; Jiang, Y.; Chen, W.; Huang, Y.; Zhang, Y.; Sun, X. Damo-yolo: A report on real-time object detection design. *arXiv preprint arXiv:2211.15444* **2022**.
89. Gao, X.; Xiao, B.; Tao, D.; Li, X. A survey of graph edit distance. *Pattern Analysis and applications* **2010**, *13*, 113–129.
90. Karapantelakis, A.; Thakur, M.; Nikou, A.; Moradi, F.; Olrog, C.; Gaim, F.; Holm, H.; Nimara, D.D.; Huang, V. Using Large Language Models to Understand Telecom Standards. In Proceedings of the 2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), 2024, pp. 440–446. https://doi.org/10.1109/ICMLCN59089.2024.10624786.
91. Xiao, S.; Liu, Z.; Zhang, P.; Muennighoff, N.; Lian, D.; Nie, J.Y. C-pack: Packed resources for general chinese embeddings. In Proceedings of the Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, 2024, pp. 641–649.
92. Guernsey, G. Harnessing Large Language Models for Automated Software Diagram Generation. Master's thesis, University of Cincinnati, 2025.
93. Fill, H.G.; Fettke, P.; Köpke, J. Conceptual modeling and large language models: impressions from first experiments with ChatGPT. *Enterprise Modelling and Information Systems Architectures (EMISAJ)* **2023**, *18*, 1–15.
94. Conrardy, A.; Cabot, J. From image to uml: First results of image based uml diagram generation using llms. *arXiv preprint arXiv:2404.11376* **2024**.
95. Thomas, R.; Webb, B. Architectural views, 2023.
96. Johri, A.; Sharma, S.; Chaudhary, V.; Raj, G.; Khurana, S. Analysis & Modeling of Deep Learning Techniques for Flowchart to Code Generation. In Proceedings of the 2025 International Conference on Networks and Cryptology (NETCRYPT), 2025, pp. 578–583. https://doi.org/10.1109/NETCRYPT65877.2025.11102763.
97. D'Souza, J.; Kabongo, S.; Giglou, H.B.; Zhang, Y. Overview of the CLEF 2024 simpletext task 4: SOTA? tracking the state-of-the-art in scholarly publications. *Working Notes of CLEF* **2024**.
98. Abeywickrama, D.B.; Bicocchi, N.; Zambonelli, F. SOTA: Towards a General Model for Self-Adaptive Systems. In Proceedings of the 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2012, pp. 48–53. https://doi.org/10.1109/WETICE.2012.48.
99. Otto, B.; Aidarkhan, A.; Ristin, M.; Braunisch, N.; Diedrich, C.; van de Venn, H.W.; Wollschlaeger, M. Code and Test Generation for I4.0 State Machines with LLM-based Diagram Recognition. In Proceedings of the 2025 IEEE 21st International Conference on Factory Communication Systems (WFCS), 2025, pp. 1–8. https://doi.org/10.1109/WFCS63373.2025.11077624.
100. Moreno-García, C.F.; Elyan, E.; Jayne, C. New trends on digitisation of complex engineering drawings. *Neural computing and applications* **2019**, *31*, 1695–1712.
101. Pawar, S.S. Automated Code Generation from Flowcharts: A Multimodal Deep Learning Framework for Accurate Translation and Debugging. Master's thesis, Texas Tech University, 2025.
102. Yuan, Z.; Pan, H.; Zhang, L. A novel pen-based flowchart recognition system for programming teaching. In Proceedings of the Workshop on Blended Learning. Springer, 2008, pp. 55–64.
103. Hussein, B.M.; Salah, A. A Framework for Model-Based Code Generation from a Flowchart. *International Journal of Computing Academic Research* **2013**, *2*, 167–181.
104. Tammina, S.; et al. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications (IJSRP)* **2019**, *9*, 143–150.
105. Khemani, B.; Patil, S.; Kotecha, K.; Tanwar, S. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data* **2024**, *11*, 18.
106. Bhushan, S.; Lee, M. Block diagram-to-text: Understanding block diagram images by generating natural language descriptors. In Proceedings of the Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022, 2022, pp. 153–168.
107. Balaji, A.; Ramanathan, T.; Sonathi, V. Chart-text: A fully automated chart image descriptor. *arXiv preprint arXiv:1812.10636* **2018**.
108. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* **2020**, *21*, 1–67.

 doi:10.20944/preprints202512.0556.v1

109. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.
110. Peng, C.; Liu, Y.; Yuan, X.; Chen, Q. Research of image recognition method based on enhanced inception-ResNet-V2. *Multimedia Tools and Applications* **2022**, *81*, 34345–34365.
111. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International conference on machine learning. PMLR, 2015, pp. 2048–2057.
112. Orna, M.A.; Akther, F.; Masud, M.A. OCR Generated Text Summarization using BART. In Proceedings of the 2024 27th International Conference on Computer and Information Technology (ICCIT), 2024, pp. 2128–2133. https://doi.org/10.1109/ICCIT64611.2024.11022545.
113. Bhushan, S.; Jung, E.S.; Lee, M. Unveiling the Power of Integration: Block Diagram Summarization through Local-Global Fusion. In Proceedings of the Findings of the Association for Computational Linguistics ACL 2024, 2024, pp. 13837–13856.
114. Julca-Aguilar, F.D.; Hirata, N.S.T. Symbol Detection in Online Handwritten Graphics Using Faster R-CNN. In Proceedings of the 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), 2018, pp. 151–156. https://doi.org/10.1109/DAS.2018.79.
115. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 10012–10022.
116. Bhattacharya, A.; Roy, S.; Sarkar, N.; Malakar, S.; Sarkar, R. Circuit component detection in offline handdrawn electrical/electronic circuit diagram. In Proceedings of the 2020 IEEE Calcutta Conference (CALCON). IEEE, 2020, pp. 80–84.
117. Okazaki, A.; Kondo, T.; Mori, K.; Tsunekawa, S.; Kawamoto, E. An automatic circuit diagram reader with loop-structure-based symbol recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1988**, *10*, 331–341.
118. Barta, A.; Vajk, I. Document image analysis by probabilistic network and circuit diagram extraction. *Informatica* **2005**, *29*.
119. De Jesus, E.O.; Lotufo, R.D.A. ECIR-an electronic circuit diagram image recognizer. In Proceedings of the Proceedings SIBGRAPI'98. International Symposium on Computer Graphics, Image Processing, and Vision (Cat. No. 98EX237). IEEE, 1998, pp. 254–260.
120. Malakar, S.; Halder, S.; Sarkar, R.; Das, N.; Basu, S.; Nasipuri, M. Text line extraction from handwritten document pages using spiral run length smearing algorithm. In Proceedings of the 2012 international conference on communications, devices and intelligent systems (CODIS). IEEE, 2012, pp. 616–619.
121. Sertdemir, A.E.; Besenk, M.; Dalyan, T.; Gokdel, Y.D.; Afacan, E. From Image to Simulation: An ANN-based Automatic Circuit Netlist Generator (Img2Sim). In Proceedings of the 2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2022, pp. 1–4. https://doi.org/10.1109/SMACD55068.2022.9816254.
122. Zupan, J. Introduction to artificial neural network (ANN) methods: what they are and how to use them. *Acta Chimica Slovenica* **1994**, *41*, 327.
123. Wang, H.; Pan, T.; Ahsan, M.K. Hand-drawn electronic component recognition using deep learning algorithm. *International Journal of Computer Applications in Technology* **2020**, *62*, 13–19.
124. Rabbani, M.; Khoshkangini, R.; Nagendraswamy, H.; Conti, M. Hand drawn optical circuit recognition. *Procedia Computer Science* **2016**, *84*, 41–48.
125. Günay, M.; Köseoğlu, M.; Yıldırım, Ö. Classification of hand-drawn basic circuit components using convolutional neural networks. In Proceedings of the 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). IEEE, 2020, pp. 1–5.
126. Rachala, R.R.; Panicker, M.R. Hand-drawn electrical circuit recognition using object detection and node recognition. *SN Computer Science* **2022**, *3*, 244.
127. Roy, S.; Bhattacharya, A.; Sarkar, N.; Malakar, S.; Sarkar, R. Offline hand-drawn circuit component recognition using texture and shape-based features. *Multimedia Tools and Applications* **2020**, *79*, 31353–31373.
128. Dey, M.; Mia, S.M.; Sarkar, N.; Bhattacharya, A.; Roy, S.; Malakar, S.; Sarkar, R. A two-stage CNN-based hand-drawn electrical and electronic circuit component recognition system. *Neural Computing and Applications* **2021**, *33*, 13367–13390.

 doi:10.20944/preprints202512.0556.v1

129. Lakshman Naika, R.; Dinesh, R.; Prabhanjan, S. Handwritten electric circuit diagram recognition: An approach based on finite state machine. *Int J Mach Learn Comput* **2019**, *9*, 374–380.
130. Mohan, A.; Mohan, A.; Indushree, B.; Malavikaa, M.; Narendra, C. Generation of Netlist from a Hand drawn Circuit through Image Processing and Machine Learning. In Proceedings of the 2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP). IEEE, 2022, pp. 1–4.
131. Zhou, Y.; Toga, A.W. Efficient skeletonization of volumetric objects. *IEEE Transactions on visualization and computer graphics* **1999**, *5*, 196–209.
132. Tomasi, C. Histograms of oriented gradients. *Computer Vision Sampler* **2012**, *1*, 1–6.
133. Jakkula, V. Tutorial on support vector machine (svm). *School of EECS, Washington State University* **2006**, *37*, 3.
134. Alhalabi, M.; Ghazal, M.; Haneefa, F.; Yousaf, J.; El-Baz, A. Smartphone handwritten circuits solver using augmented reality and capsule deep networks for engineering education. *Education Sciences* **2021**, *11*, 661.
135. Liu, Y.; Xiao, Y. Circuit sketch recognition. *Department of Electrical Engineering Stanford University: Stanford, CA, USA* **2013**.
136. Patare, M.D.; Joshi, M.S. Hand-drawn digital logic circuit component recognition using svm. *International Journal of Computer Applications* **2016**, *143*, 24–28.
137. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. *Advances in neural information processing systems* **2017**, *30*.
138. Bohara, B.; Krishnamoorthy, H.S. Computer Vision based Framework for Power Converter Identification and Analysis. In Proceedings of the 2022 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES), 2022, pp. 1–6. https://doi.org/10.1109/PEDES56012.2022.10080752.
139. Bayer, J.; Turabi, S.H.; Dengel, A. Text extraction for handwritten circuit diagram images. In Proceedings of the International Conference on Document Analysis and Recognition. Springer, 2023, pp. 192–198.
140. Bayer, J.; Roy, A.K.; Dengel, A. Instance segmentation based graph extraction for handwritten circuit diagram images. *arXiv preprint arXiv:2301.03155* **2023**.
141. Gao, M.; Qiu, R.; Chang, Z.H.; Zhang, K.; Wei, H.; Chen, H.C. Circuit Diagram Retrieval Based on Hierarchical Circuit Graph Representation. *arXiv preprint arXiv:2503.11658* **2025**.
142. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206* **2021**.
143. Likas, A.; Vlassis, N.; Verbeek, J.J. The global k-means clustering algorithm. *Pattern recognition* **2003**, *36*, 451–461.
144. Liu, C.; Xu, J.; Wang, F. A review of keypoints' detection and feature description in image registration. *Scientific programming* **2021**, *2021*, 8509164.
145. Hafiz, A.M.; Bhat, G.M. A survey on instance segmentation: state of the art. *International journal of multimedia information retrieval* **2020**, *9*, 171–189.
146. Komoot, N.; Mruetusatorn, S. An Analysis of Electronic Circuits Diagram Using Computer Vision Techniques. In Proceedings of the 2024 9th International Conference on Business and Industrial Research (ICBIR), 2024, pp. 1581–1586. https://doi.org/10.1109/ICBIR61386.2024.10875940.
147. Dieste-Velasco, M. Application of a Pattern-Recognition Neural Network for Detecting Analog Electronic Circuit Faults. Mathematics 2021, 9, 3247. *Modeling and Simulation in Engineering* **2021**, p. 81.
148. Pavithra, S.; Shreyashwini, N.; Bhavana, H.; Nikhitha, G.; Kavitha, T. Hand-drawn electronic component recognition using orb. *Procedia Computer Science* **2023**, *218*, 504–513.
149. Hemker, D.; Maalouly, J.; Mathis, H.; Klos, R.; Ravanan, E. From Schematics to Netlists–Electrical Circuit Analysis Using Deep-Learning Methods. *Advances in Radio Science* **2024**, *22*, 61–75.
150. Li, X.; Li, J.; Hu, X.; Yang, J. Line-CNN: End-to-End Traffic Line Detection With Line Proposal Unit. *IEEE Transactions on Intelligent Transportation Systems* **2020**, *21*, 248–258. https://doi.org/10.1109/TITS.2019.2890870.
151. Kelly, C.R.; Cole, J.M. Digitizing images of electrical-circuit schematics. *APL Machine Learning* **2024**, *2*.
152. Mishra, D.; Vinayak, C. Image based Circuit Simulation. In Proceedings of the 2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE), 2013, pp. 1–4. https://doi.org/10.1109/CARE.2013.6733773.
153. Matasyx, J.; Kittlery, C.G.J. Progressive probabilistic hough transform. In Proceedings of the Proceedings of the British Machine Vision Conference, Southampton, UK, 1998, pp. 14–17.
154. Neuner, M.; Abel, I.; Graeb, H. Library-free Structure Recognition for Analog Circuits. In Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021, pp. 1366–1371. https://doi.org/10.23919/DATE51398.2021.9474102.

 doi:10.20944/preprints202512.0556.v1

155. Vasudevan, S.K.; Venkatachalam, K.; Sundaram, R. A novel mobile application for circuit component identification and recognition through machine learning and image processing techniques. *International Journal of Intelligent Systems Technologies and Applications* **2017**, *16*, 342–358.
156. Panigrahi, U.; Sahoo, P.K.; Panda, M.K.; Panda, G. A ResNet-101 deep learning framework induced transfer learning strategy for moving object detection. *Image and Vision Computing* **2024**, *146*, 105021.
157. Sanfeliu, A.; Fu, K.S. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics* **2012**, *3*, 353–362.
158. Pan, L.; Xue, Z.; Zhang, K. GAML-YOLO: A Precise Detection Algorithm for Extracting Key Features from Complex Environments. *Electronics* **2025**, *14*, 2523.
159. Arun, A.; Patil, S.S.; Poduval, G.M.; Shah, A.; Patil, R.P. Hand-Drawn Electronic Component Detection and Simulation Using Deep Learning and Flask. *Cureus Journals* **2025**, *2*.
160. Gody, M. Hand-drawn electric circuit schematic components. *URL: https://www. kaggle. com/datasets/moodrammer/handdrawn-circuit-schematic-components.(accessed: 10.14. 2022)* **2022**.
161. Cao, W.; Chen, Z.; Wu, C.; Li, T. A Layered Framework for Universal Extraction and Recognition of Electrical Diagrams. *Electronics* **2025**, *14*, 833.
162. Uzair, W.; Chai, D.; Rassau, A. ElectroNet: An Enhanced Model for Small-Scale Object Detection in Electrical Schematic Diagrams. *Research Square* **2023**. Available as preprint at https://www.researchsquare.com/article/rs-3137489/v1, https://doi.org/10.21203/rs.3.rs-3137489/v1.
163. Cui, C.; Sun, T.; Lin, M.; Gao, T.; Zhang, Y.; Liu, J.; Wang, X.; Zhang, Z.; Zhou, C.; Liu, H.; et al. Paddleocr 3.0 technical report. *arXiv preprint arXiv:2507.05595* **2025**.
164. Bi, R.; Xu, T.; Xu, M.; Chen, E. Paddlepaddle: A production-oriented deep learning platform facilitating the competency of enterprises. In Proceedings of the 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys). IEEE, 2022, pp. 92–99.
165. He, J.; Ničković, D.; Bartocci, E.; Grosu, R. TD-Magic: From Pictures of Timing Diagrams To Formal Specifications. In Proceedings of the 2023 60th ACM/IEEE Design Automation Conference (DAC), 2023, pp. 1–6. https://doi.org/10.1109/DAC56929.2023.10247685.
166. Li, Y.; Qu, Q.; Wang, M.; Yu, L.; Wang, J.; Shen, L.; He, K. Deep learning for digitizing highly noisy paper-based ECG records. *Computers in biology and medicine* **2020**, *127*, 104077.
167. Mehendale, N.; Mishra, S.; Shah, V.; Khatwani, G.; Patil, R.; Sapariya, D.; Parmar, D.; Dinesh, S.; Daphal, P. Ecg paper record digitization and diagnosis using deep learning. *Available at SSRN 3646902* **2020**.
168. Peli, T.; Malah, D. A study of edge detection algorithms. *Computer graphics and image processing* **1982**, *20*, 1–21.
169. Rapaport, W.J. Implementation is semantic interpretation. *The Monist* **1999**, *82*, 109–130.
170. Thompson, N. Strict partial order. In *The Routledge handbook of metaphysical grounding*; Routledge, 2020; pp. 259–270.
171. He, J.; Kenbeek, V.T.W.; Yang, Z.; Qu, M.; Bartocci, E.; Ničković, D.; Grosu, R. TD-Interpreter: Enhancing the Understanding of Timing Diagrams with Visual-Language Learning. *arXiv preprint arXiv:2507.16844* **2025**.
172. Liu, H.; Li, C.; Wu, Q.; Lee, Y.J. Visual instruction tuning. *Advances in neural information processing systems* **2023**, *36*, 34892–34916.
173. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. Lora: Low-rank adaptation of large language models. *ICLR* **2022**, *1*, 3.