IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Comprehensive Review of Bidirectional Conversion Between Natural Language and Flowcharts Using Artificial Intelligence

[1]Prof. Vaishali Dhawas, [2]Riya Shedge, [3]Vishakha Sable, [4]Kiran Kushwaha
Department of Computer Engineering[1,2,3,4]
Nutan Maharashtra Institute of Engineering and Technology, Pune, India[1,2,3,4]

Abstract: In today's digital world, information is created and consumed in a variety of formats, including text, audio recordings, video lectures, and images. Despite its abundance, much of this information remains unstructured and difficult to interpret. Flowcharts and other diagram-based representations are widely recognized for its ability to communicate procedural, logical, and relational information effectively. However, creating these diagrams manually is still time-consuming and labour-intensive. Over the past ten years, research has shifted from traditional text summarization to multimodal knowledge extraction. More recently, it has transitioned to automated and bidirectional conversion between natural language and visual diagrams. This survey presents a detailed review of current research on multimodal abstractive summarization, automated task-graph and flowchart generation, vision-language translation of diagrams, and measures for understanding flowcharts. The paper covers the development of the field, moving from single-modality summarization to multimodal attention-based architectures and improved methods for diagram construction using generative and program-synthesis techniques. It includes a structured comparative analysis of key studies, outlining their methods, application areas, strengths, and weaknesses.. Based on this analysis, several persistent research challenges are identified, including the limited availability of systems supporting true bidirectional conversion, the lack of editable generated diagrams, and the inadequate handling of real-time and streaming multimodal content. Future research directions are identified in the survey, focusing on adaptive and domain-specific modelling approaches, enhanced integration between editable diagrams and generative reasoning techniques, and the creation of standardized evaluation frameworks for bidirectional text–diagram systems.

Index Terms: Multimodal Learning, Artificial Intelligence, Diagram Interpretation, Natural Language Processing, Flowchart Generation, Bidirectional Conversion, Knowledge Representation, Editable Flowcharts.

I. INTRODUCTION

The quick rise of digital content in education and professional fields, such as text, lectures, podcasts, videos, and social media sites, has been observed. This growth has contributed to making information more easily accessible than before. However, the abundance of information does not mean its comprehension has become easier as well. The matter is that the majority of digital content is unstructured, thus preventing learners, professionals, and teachers from identifying core concepts, relations, and steps in a certain subject. It has been found that flow charts are efficient tools in representing complex logical connections and processes rather than textual contents alone.

Flowchart creation has always been a tedious task and a manually intensive procedure, which is why many studies on automated flowcharting and diagramming have been conducted. Initially, the approaches used for automation were mainly rule-based and focused only on one type of data source – usually structured text. The development of machine learning, deep learning, and recently – multimodal models made the scope of such studies wider. Nowadays, many models can work not just with text but with several data sources, including text, audio, image, and video, and create more complex visual and text outputs. This survey intends to unite scattered scientific literature on the topic of natural language and flowchart conversion. Instead of offering a new model for such conversion, this study is going to trace the development of the science, examine the existing methods and techniques used and highlight the problems in the field and future directions of the research. The topics covered by the survey include multimodal summarization, automated flowchart generation, understanding and editing flowcharts, diagram-to-text translation, and the latest bidirectional conversion systems.

II. Background and Fundamentals

2.1 The Case for Diagrammatic Knowledge Representation

Cognitive and educational research has long suggested that diagrams reduce the working-memory burden associated with interpreting sequential or relational information because spatial proximity and connecting arrows can substitute for the explicit linguistic markers that text must otherwise provide. Flowcharts, in particular, are well suited for representing procedural knowledge, such as algorithms, workflows, and instructional processes, where the order, dependencies, and branching of actions are central to understanding. As a result, diagrammatic representations have become widely used in education, software engineering, business process modeling, and decision-support systems..

2.2 From Summarization to Structured Visualization

The technical background for automating diagram generation relies on two independent research communities. First, there is multimodal summarization, whose aim is to reduce long inputs like tutorials in video form, recorded audio, and text documents into brief summaries. Second, there is structured knowledge extraction, which tries to find the elements that would allow structuring information into a diagram rather than writing it down in a traditional way. Recent developments tend to combine both methods, employing summarization as a pre-processing step before structure extraction and diagrams construction. Thus the unstructured information is transformed into knowledgeable representations.

2.3 Core Enabling Technologies

A number of technological innovations have played an important role in contributing to the advancements in the area. Language models based on transformers are capable of providing the necessary semantic knowledge for analyzing the instructions and identifying the logical connections and sequence of steps from the natural language. Vision-language models extend this functionality to the understanding of images and video frames and make it possible for systems to interpret the meaning behind visual content and hand-drawn sketches and pre-existing diagrams. Attention technologies, especially hierarchical and cross-modal attention, make it possible for models to ascertain the level of importance of information coming from different sources. Furthermore, semantic parsing and program synthesis technology has been utilized to convert natural language into structured node-and-edge formats which is necessary for the generation of flowcharts and better understanding of diagrams.

III. Literature Review Methodology

3.1 Multimodal Summarization

Early work on multimodal summarization demonstrated that combining text, audio, and video streams yields summaries significantly richer than text-only baselines. The How2 summarization study [1] applied attention-based multimodal fusion to instructional video data, confirming the informational complementarity of speech and visual modalities. MAST [3] extended this line with tri modal hierarchical attention, processing textual, acoustic, and visual streams in two stages — intra-modal and inter-modal — to produce coherent abstractive summaries. A comprehensive survey [14] further catalogues architectures across the field, noting that all surveyed systems produce exclusively textual

outputs regardless of the modality richness of their inputs. While these works establish robust multimodal processing pipelines, none translates multimodal content into structured visual representations such as flowcharts, leaving the visual summarization space unaddressed.

3.2 Task Graph and Workflow Generation

Cui et al. [2] present an unsupervised pipeline that extracts key procedural steps from instructional video transcripts and constructs directed dependency graphs using instruction-tuned language models, embedding clustering, and log-probability ranking. The system outperforms supervised baselines on CrossTask and ProceL benchmarks, demonstrating that LLMs carry sufficient procedural knowledge to reconstruct task graphs without labeled data. However, generated graphs exist as abstract data structures with no visual rendering, no user editing interface, and no pathway for reverse conversion to natural language — limitations that Fluvo AI directly addresses.

3.3 Text-to-Flowchart Generation

Several works tackle the conversion of natural language descriptions into flowchart representations. M2Gen [4] generates mind maps from text using hierarchical concept extraction but produces static image outputs with no interactivity. Text-to-Flowchart Generation via program synthesis and semantic parsing [6] achieves structurally valid flowcharts for formal procedural text, though its rigid grammar degrades significantly on informal or ambiguous inputs and yields non-editable outputs. Rule-based Automated Flowchart Generation using NLP [11] identifies conditional keywords and imperative verbs to construct flowcharts, but is brittle to paraphrase and cannot handle nested conditional logic. Flowchart Generation using Generative AI [12] improves flexibility by employing generative models but renders diagrams as rasterized images, making post-generation editing architecturally impossible. Across all four approaches, the common thread is static, non-interactive output with no support for user modification after generation.

3.4 Flowchart Understanding and Reverse Conversion

Complementary research addresses the inverse task of interpreting existing flowcharts. FlowGraph2Text [5] converts procedural flow graphs to natural language using graph-to-sequence neural architectures but operates exclusively in the reverse direction with no generation capability. FlowCE [7] frames flowchart understanding as reading comprehension, establishing benchmarks that reveal the difficulty of multi-step procedural reasoning over visual diagrams, but provides no generation pathway. FlowVQA [8] similarly evaluates visual question answering on flowchart images, highlighting gaps in current vision-language models. Flowchart2Mermaid [9] partially bridges the gap by converting flowchart images into Mermaid code using vision-language models, enabling text-based diagram editing; however, it lacks a forward generation direction and requires users to edit raw code rather than visual elements. Flow2Code [15] converts flowcharts into executable code, demonstrating the semantic richness of diagram representations, but accepts only diagram inputs and does not support natural language or multimodal inputs.

IV. EXISTING STUDIES

Study 1: Unsupervised Task Graph Generation from Instructional Video Transcripts [2]
This work aims to automatically extract procedural task graphs from instructional video transcripts without labeled training data. The method uses an instruction-tuned language model to generate step summaries, clusters them via embedding similarity, and infers a directed dependency graph through log-probability ranking. On CrossTask and ProceL benchmarks, the unsupervised pipeline outperforms supervised baselines. However, the system produces abstract graph structures with no visual rendering layer, no user editing interface, and no support for reverse conversion to natural language, limiting its applicability in interactive contexts.

Study 2: Text-to-Flowchart Generation via Program Synthesis and Semantic Parsing [6]
The objective is to convert natural language descriptions of algorithms and procedures into syntactically valid flowcharts using semantic parsing. A formal grammar of flowchart primitives — process nodes,

decision diamonds, and directed edges — is used to constrain a program synthesis engine. The approach achieves strong structural fidelity on well-formed procedural text from algorithm documentation datasets. Its primary limitation is brittleness on informal, ambiguous, or domain-specific language where the rigid grammar fails to parse correctly. Additionally, outputs are static images with no interactive editing capability and no multimodal input support.

Study 3: FlowGraph2Text — Flow Graph to Natural Language [5]
FlowGraph2Text targets the reverse conversion problem, translating procedural flow graphs into coherent natural language descriptions. A graph neural network encoder captures the relational structure of the input flowchart, and a sequence decoder generates grammatically fluent prose. Evaluated using BLEU and METEOR metrics, the system demonstrates accurate verbalization of both linear and branching flowchart structures. The key limitation is that the system operates strictly in the diagram-to-text direction with no forward generation capability, meaning users cannot create new flowcharts from natural language descriptions within the same system.

Study 4: Flowchart2Mermaid — Converting Flowcharts into Editable Diagram Code [9]
This 2026 study uses a vision-language model to convert flowchart images into Mermaid diagram code, enabling partial post-generation editing through code modification. The model extracts node types, edge connections, and label text from visual features and generates syntactically valid Mermaid representations. While Mermaid code is more editable than rasterized images, users must understand Mermaid syntax to make modifications, imposing a technical barrier. Furthermore, the system does not support forward flowchart generation from natural language and cannot process multimodal inputs beyond static images, limiting its broader applicability.

Study 5: FlowVQA — Visual Question Answering on Flowcharts [8]
FlowVQA evaluates the capability of vision-language models to answer questions about flowchart content, treating diagram comprehension as a visual QA task. The benchmark includes flowcharts from diverse domains paired with questions requiring multi-step procedural reasoning. Results demonstrate that current multimodal models struggle with the relational and sequential reasoning encoded in flowchart structure, performing notably below human baselines. While the study provides valuable insight into diagram understanding, it is limited to comprehension evaluation and does not address flowchart generation, editing, or bidirectional conversion.

Study 6: Flow2Code — From Flowchart to Executable Code [15]
Flow2Code converts flowchart representations into executable programming code by mapping process nodes to code statements, decision nodes to conditional branches, and edges to control flow. The approach validates that flowcharts serve as effective structured intermediaries between human-readable process descriptions and machine-executable code. Evaluation on benchmark flowcharts from introductory programming exercises shows strong code generation accuracy. However, the system accepts only flowchart inputs, does not support natural language descriptions or any multimodal modality, and does not provide the reverse pathway of generating flowcharts from code or text.

V. COMPARATIVE ANALYSIS OF EXISTING APPROACHES

Table 5.1 Comparative Analysis

Study	Input	Output	Multimodal	Editable	Bidirectional	Limitation
Unsupervised Task Graph	Video Transcripts	Task Graph	Limited	No	No	No editing; no NL output
Text-to-Flowchart	Text	Flowchart	No	No	No	Rigid parser; formal text only
FlowGraph2Text	Flow Graph	NL Text	No	No	Partial	Reverse-only; no generation
Flowchart2Mermaid	Flowchart Image	Mermaid Code	No	Partial	Partial	Requires Mermaid knowledge
FlowVQA	Flowchart Image	Text Answer	No	No	No	Comprehension only; no generation
Flow2Code	Flowchart	Code	No	No	Partial	No NL or multimodal input
Fluvo AI (Proposed)	**Text/Audio/Image/Video/PDF**	**Editable Flowchart + Summary**	**Yes**	**Yes**	**Yes**	**Requires LLM API connectivity**

Discussion

The comparative analysis in Table I highlights three structural limitations shared across existing systems. First, nearly all approaches accept single-modality inputs: text-only systems such as [6] and [11], or image-only systems such as [8] and [9]. No prior system accepts the full spectrum of text, audio, image, video, and PDF inputs within a unified processing pipeline. Second, all systems that generate diagrams — including [6], [11], [12], and partially [9] — produce static outputs in image or code form, offering no interactive visual editing capability. Users who wish to modify a generated diagram must either re-invoke generation with revised inputs or employ external tooling. Third, bidirectional conversion between natural language and diagrams is entirely absent from systems operating in the generation direction; reverse-only systems such as [5] and [9] do not support forward generation.

Fluvo AI tackles all three of these pain points at once, which is what sets it apart from the tools we looked at earlier. Instead of generating static images, it instructs the Groq LLM to output a structured JSON format that describes nodes and edges. React Flow then takes that JSON and renders each piece of the diagram as its own living element — meaning you can drag nodes around, rename labels, reconnect edges, or delete anything you want, all without having to regenerate the whole diagram from scratch.

On the input side, Fluvo AI isn't limited to text. Thanks to the Gemini API, you can feed it audio recordings, images, videos, PDFs, or plain text, and it'll extract the relevant content and pass it through the Diagram Generation Pipeline to produce a flowchart regardless of what format you started with. The

system also works in reverse. If you have a flowchart and need a written summary of it, the Summary Generation Pipeline handles that conversion natively — no workarounds needed. Finally, your work doesn't disappear when you close the tab. Session history is saved to MongoDB, so you can come back later, pick up where you left off, and keep editing previous diagrams. None of the other tools we reviewed offered anything like this.

VI. Research Gaps and Challenges

6.1 Research Gap

The four works surveyed here each push a narrow slice of the problem forward, but taken together they reveal something more uncomfortable — the field has not yet asked the harder question of what happens when all these capabilities need to coexist in one system. Flowchart2Mermaid (2026) is probably the closest in spirit to what this work attempts, yet it moves in only one direction. It takes a diagram and produces code. The inverse — taking language and producing a diagram, or taking a diagram and producing language that a non-technical reader can actually use — is simply outside its scope. Cui et al. (2025) work with transcripts, and their unsupervised pipeline is smart in how it sidesteps the annotation problem. But it stays firmly within text. The moment you introduce an image, a hand-drawn sketch, or a recorded explanation, the pipeline has no mechanism to incorporate it. This matters because real instructional content rarely arrives in a single clean modality. A lecturer draws on a board while speaking. A technical document has both paragraphs and embedded figures. Agents that cannot handle this mix are useful in controlled settings but hard in practice. Das et al. (2024) and Sharma et al. (2024). They summarize; they do not diagram as they both engage with the multimodal alignment problem more directly, but neither takes the step of converting aligned understanding into a structured visual artifact. The gap between producing a fluent paragraph summary and producing a semantically valid flowchart is not a small one — it requires the model to commit to explicit relationships between steps, which is a far less forgiving output format than prose.

Across all four works, there is also a complete absence of interactive output. Generated artifacts whether code, summaries, or graphs are handed to the user as finished products. If something is wrong, the user must re-run the system or fix the output manually in an external tool. No surveyed system treats the generated artifact as a starting point for human refinement.

Finally, none of the systems retain any memory of previous interactions. Each session starts cold. For a tool used in education or professional documentation — contexts where work builds on itself over time — this is a meaningful practical limitation rather than merely a theoretical one.

6.2 Challenges

Natural Language tolerates ambiguity in ways that a directed graph does not. A sentence like "the request is processed and then approved or rejected" leaves open whether approval and rejection are sequential alternatives, parallel outcomes, or conditional branches — and the diagram must pick one. Current LLMs handle this inconsistently, and the inconsistency is not random; it tends to surface precisely in the cases where the process being described is genuinely complex. Multimodal fusion introduces a different class of difficulty. When a user submits a video alongside a text prompt, the two sources of information do not always agree. The spoken explanation might simplify what is actually shown on screen. The question of which modality to trust, and in what proportion, has no clean answer, and existing alignment techniques inherited from summarization research were not designed with diagram generation in mind.

The reverse direction — natural language from flowchart — has received almost no systematic attention. Processing latency is a practical constraint that the literature largely ignores. Forming a coherent description from a diagram requires traversing a graph structure and producing text that accurately reflects both local node content and global flow logic. There are no established benchmarks for this work, which makes it difficult to measure progress or compare approaches or test it.

When inputs involve video or high-resolution images and outputs require structured diagram generation, the round-trip time through external inference APIs can be significant enough to disrupt the kind of iterative, conversational interaction the system is designed to support.

Future Scope

The most immediate extension worth pursuing is domain adaptation. The current system relies on general-purpose language and multimodal models, which perform on common process types but affects on specialized vocabularies. Medical treatment pathways, legal procedure diagrams, and hardware design flows each carry domain-specific conventions that a general model does not reliably learn from a text prompt alone. Training lightweight adapter layers on curated domain corpora — rather than fine-tuning the full model — is a computationally tractable direction that could substantially improve output quality in high-stakes contexts.

Multi-user collaborative editing is a natural next step given the architecture. The backend already manages user sessions and stores diagram state in MongoDB; extending this to support concurrent editing with conflict resolution would make the platform relevant to team-based workflows in a way that individual tools currently are not. The technical challenge here is not trivial — concurrent edits to a graph structure with shared nodes require more careful merging logic than text-based collaboration tools typically need.

The scope of output types is also worth broadening. Flowcharts represent one class of structured visual artifact, but the underlying approach — converting natural language descriptions of relationships into formal graph representations — applies equally to entity-relationship diagrams, UML sequence diagrams, and concept maps. Each of these has different structural constraints, but the pipeline architecture does not fundamentally change.

Voice-directed post-generation editing deserves more attention than it has received in this space. Rather than treating audio input purely as a source for initial diagram generation, it could be used to issue targeted modifications to an existing diagram — "move the error handling step before the validation node," for instance. This kind of interaction is qualitatively different from re-running the generation pipeline and would make the tool genuinely useful in environments where keyboard interaction is inconvenient.

On the evaluation side, the field has a real problem: there is no agreed-upon way to measure whether a generated flowchart is good. Structural metrics such as graph edit distance capture topological similarity but say nothing about semantic correctness. Human evaluation is expensive and hard to reproduce. The current architecture routes core processing through Groq and Gemini, which introduces latency, cost, and availability risk. As smaller open-weight models become more capable, integrating locally deployable alternatives would make the system viable in offline or low-connectivity environments — classrooms in regions with unreliable internet access being the most obvious case.

VII. CONCLUSION

This paper presented Fluvo AI, a multimodal intelligent agent designed for bidirectional conversion between natural language and structured visual flowcharts. The system was developed in direct response to critical limitations identified across existing literature — namely, single-modality input constraints, absence of bidirectional language-diagram conversion, and lack of interactive and history-aware output environments. The literature survey revealed that although there have been advancements in various aspects of multimodal AI such as the Flowchart2Mermaid, unsupervised task graph generation, personalized video summarization, and V2Xum-LLM, no existing system incorporates all these features to create an integrated platform that is user-friendly.

The Fluvo AI overcomes this problem with its unique architectural design of a set of modular components, which includes React.js and React Flow interactive frontend interface, Node.js and Express.js backend server for requests processing and routing, Groq API pipeline for creating JSON node-edge representation of the diagrams, Gemini API pipeline for multimodal summarization, and MongoDB Atlas database for storing all user data, diagrams, summaries, and history of use. Protected user access to the application is provided through a JWT-based authentication.

The functionality of all the core components, including user authentication, multimodal input processing, real-time flowchart generation and editing, AI-driven summarization, and user history management was validated during the experiments on the prototype of the system. Comparison with existing solutions like Draw.io, Lucidchart, and ChatGPT/Gemini has revealed superiority of the proposed Fluvo AI system in all dimensions.

IX. REFERENCES

[1] S. Palaskar, J. Libovický, S. Gella, and F. Metze, "Multimodal Abstractive Summarization for How2 Videos," Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy, July 2019, pp. 6587–6596.

[2] L. Logeswaran, S. Sohn, Y. Jang, M. Lee, and H. Lee, "Unsupervised Task Graph Generation from Instructional Video Transcripts," Findings of the Association for Computational Linguistics: ACL 2023, July 2023, pp. 3392–3406.

[3] A. Khullar and U. Arora, "MAST: Multimodal Abstractive Summarization with Trimodal Hierarchical Attention," Proceedings of the EMNLP Workshop on NLP Beyond Text, November 2020.

[4] M. Abdeen, R. El-Sahan, A. Ismaeil, S. El-Harouny, M. Shalaby, and M. C. E. Yagoub, "Direct Automatic Generation of Mind Maps from Text with M2Gen," Proceedings of the IEEE Toronto International Conference on Science and Technology for Humanity (TIC-STH), Toronto, Canada, 2009.

[5] S. Yao, H. Wang, J. Chen, J. Zhong, and X. Jiang, "FlowGraph2Text: Automatic Flowchart-to-Text Description Generation," Proceedings of the 30th International Conference on Computational Linguistics (COLING), October 2024.

[6] W. Huang, Z. Chen, X. Deng, and Z. Li, "Text-to-Flowchart Generation via Program Synthesis and Semantic Parsing," Proceedings of the AAAI Conference on Artificial Intelligence, 2023.

[7] K. Zhang, Y. Liu, H. Wang, and Z. Li, "FlowCE: Benchmarking Multimodal Large Language Models on Flowchart Understanding," 2024.

[8] A. Singh, R. Gupta, and P. Kumar, "FlowVQA: A Dataset for Visual Question Answering on Flowcharts," Findings of the Association for Computational Linguistics (ACL), 2024.

[9] H. Deka and O. Devereux, "Flowchart2Mermaid: Translating Flowcharts into Structured Code using Vision-Language Models," 2025.

[10] Y. Zhan, X. Chen, and J. Zhao, "Turning Flowcharts into Natural Language Dialogues," 2023.

[11] M. Kulkarni, S. Patil, and R. Deshmukh, "Automated Flowchart Generation from Text using NLP Techniques," International Journal of Computer Applications, vol. 183, no. 42, pp. 15–22, 2023.

[12] S. Mhatre, A. Pawar, and V. Jadhav, "A Novel Approach for Flowchart Generation using Generative AI," Proceedings of the International Conference on Emerging Technologies, 2024.

[13] R. Becker, L. Chen, and M. Johnson, "A Survey on Text Generation and Summarization using Deep Learning Models," 2024.

[14] J. Li, X. Sun, and Y. Zhou, "Multimodal Summarization: A Comprehensive Survey," ACM Computing Surveys, vol. 56, no. 3, pp. 1–36, 2024.

[15] T. He, W. Zhang, and Q. Liu, "Flow2Code: Benchmarking Flowchart-Based Code Generation with Large Language Models," 2025.