Pseudocode Generation from Clinical Protocol Flowchart using Large Vision-Language Models

Le Zhou
Pace University
New York City, USA
lz25261n@pace.edu

Xiao Luo
Southern Methodist University
Dallas, USA
xnluo@smu.edu

Chinmay Chandra
Pace University
New York City, USA
cc55048n@pace.edu

Zhan Zhang
Pace University
New York City, USA
zzhang@pace.edu

***Abstract*—A key component of developing clinical decision support systems (CDSS) involves translating clinical protocols into executable code linking to the backend Electronic Health Record (EHR) systems – a challenging, time-consuming, and error-prone task, particularly for flowchart-based protocols with complex logic and role-specific actions. This study investigates the potential of large vision-language models (LVLMs) to automate pseudocode generation from clinical protocols as an initial step in this process. Using emergency medical services (EMS) as a case study, we evaluated state-of-the-art LVLMs and finetuned two LVLMs for generating pseudocode that can support the development of CDSS for future EHR database integration. The results indicate that model finetuning leads to a substantial performance improvement of at least 15% in terms of BLEU or CodeBLEU for pseudocode generation and up to 20% in terms of F1 for medical terminology extraction. Although these models show promise, challenges remain in the interpretation of complex protocols with conditional assessments tied to specific roles of care providers. The Pass@1 evaluation results show that human expert evaluation is necessary to assess both the semantic and logical correctness of code generation.**

Index Terms*—*Large Vision-Language Models*, *Clinical Protocol*, *Pseudocode Generation*, *Clinical Decision Support

I. Introduction

Clinical decision support systems (CDSS) are increasingly recognized as essential tools for advancing the quality, safety, and efficiency of healthcare delivery. By integrating patient-specific data with established clinical knowledge, these systems assist healthcare providers in making timely, accurate, and evidence-based decisions at the point of care [8], [23]. A fundamental step in building effective CDSS involves the formalization and computational translation of clinical protocols. These protocols, which encapsulate expert knowledge and standardized procedures for managing specific medical conditions, are typically documented in unstructured or semi-structured formats such as text narratives, tables, or flowcharts. Among these, flowchart-based protocols are especially prevalent, as they offer an intuitive visual representation of complex clinical workflows, decision trees, and condition-specific actions. However, converting flowchart-based protocols into machine-executable formats poses significant challenges. These visual diagrams often encode nuanced clinical logic, including nested decision branches, conditional assessments, temporal constraints, and role-specific tasks (e.g., actions to be performed by different types of providers).

The advent of artificial intelligence (AI), particularly large language and vision-language models (LLMs/LVLMs), offers a promising avenue to address these challenges. Recent work has demonstrated the capability of these models to interpret structured and semi-structured data, including text and graphical representations, making them suitable candidates for automating the translation of clinical protocols into executable pseudocode [31]. In particular, code-specialized LVLMs excel at translating natural language descriptions and structured flowchart into executable code across a wide range of domains [17], [29], [30]. Despite this potential, limited research has systematically evaluated their applicability and performance in converting clinical protocol flowchart into pseudocode that can be used for CDSS development.

In this study, we aim to address this research gap. More specifically, we use emergency medical services (EMS) as a case study due to the vital role EMS protocols play in guiding life-saving interventions and the inherent complexity of their flowchart-based structure. To assess the generalizability of our approach, we include EMS protocols from six U.S. states and evaluate several state-of-the-art LVLMs in converting these flowcharts into structured pseudocode. This work represents a foundational step toward building robust CDSS tools for EMS providers. The key contributions of this study are as follows:

- We evaluated four state-of-the-art LVLMs for generating pseudocode from clinical protocol flowcharts, identifying both their strengths and limitations.
- We assessed the accuracy of medical terminology extraction from clinical protocol flowcharts using these LVLMs with and without model finetuning.
- We lay the foundation for future research aimed at bridging the gap between clinical protocol representation and CDSS development.

II. Related Work

Research on generating pseudocode from clinical flowcharts explores methods to translate visual representations of clinical protocols into structured textual descriptions. This body of research primarily includes rule-based and deep learning-based approaches. For rule-based techniques, an exemplary study is Carlisle et al.'s work [4], which introduced RAPTOR, a rule-based modeling and simulation tool capable of automatically

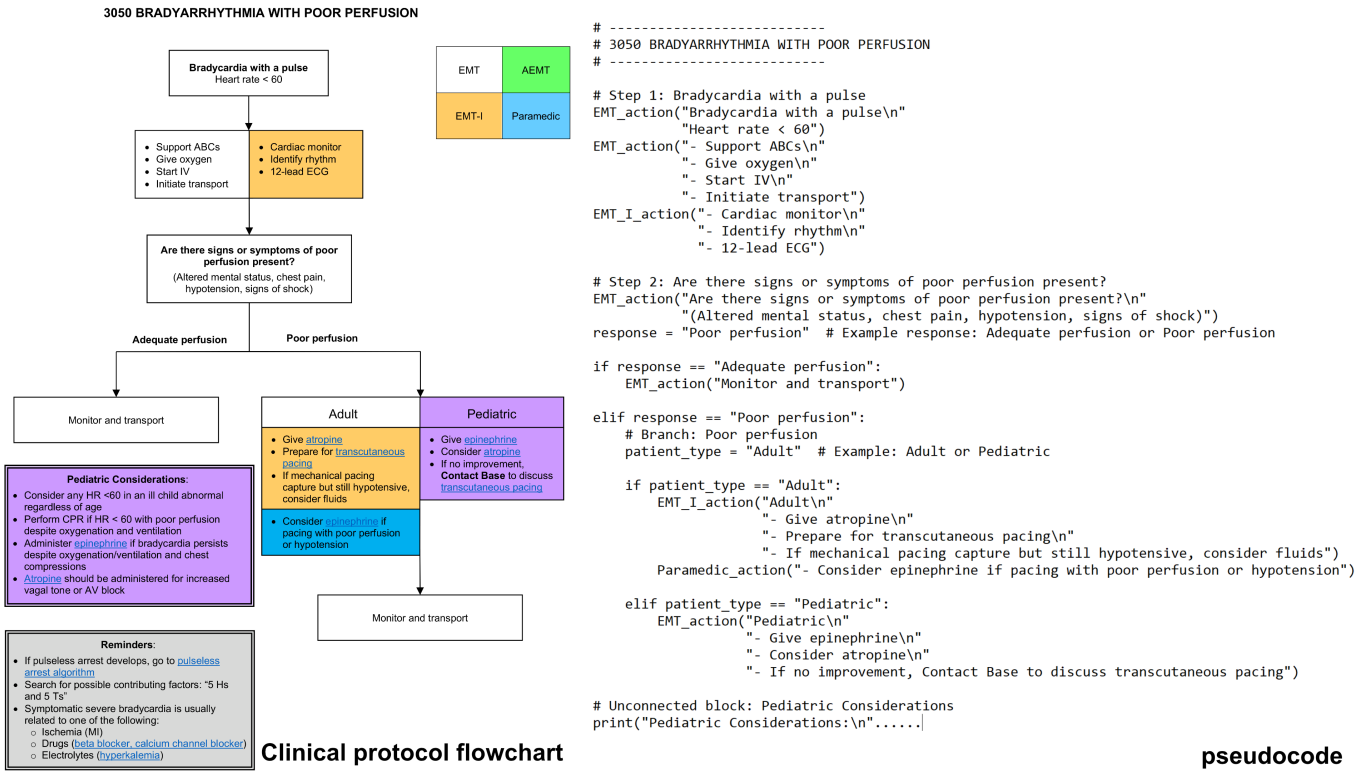

Fig. 1. An example of the clinical protocol flowchart and its corresponding pseudocode

generating code using user-defined selection and loop primitives. Similarly, Wu et al. [28] proposed a structure identification algorithm that detects loops and conditional selections based on the structural attributes of flowcharts. Wang et al. [26] extended these efforts by developing a method for handling semi-structured flowcharts, incorporating elements such as breaks and returns into the code generation process. Additional rule-based efforts include Ongenae et al. [19], who designed a semi-automatic verification and translation framework capable of turning manually constructed diagrams into ready-to-use programs. Liu et al. [13] noted that flowcharts differ from text by using graphical structures such as loops and conditionals. To address this, they proposed a deep learning–based structure recognition model that converts flowcharts into pseudocode using programming-style constructs like while and if statements. Recent studies have begun to explore vision-based and multimodal learning approaches and have demonstrated that state-of-the-art pretrained vision-language models worked better than traditional rule-based or algorithmic models [6], [25]. For example, Zhang et al. [6] introduced Flow2Code to show that the supervised fine-tuning technique contributes greatly to the LLMs' performance on code generation based on the given flowcharts.

In addition to research on converting flowcharts to pseudocode, there are studies investigating the transformation of natural language protocols into pseudocode using the state-of-the-art LLMs. For example, O'Donoghue et al. [18] employed GPT-4o to transform natural language protocols into pseudocode. Yi et al. [31] developed an automated framework leveraging GPT-4o to extract pseudocode from biological protocols. Their research demonstrated the effectiveness of GPT-4o and cohere as robust tools for formulating scientific protocols, underscoring the potential of advanced language models in scientific applications.

Despite these advances, there are still important gaps in applying those latest techniques in the clinical domain. Prior work has not addressed the unique challenges posed by flowchart-based representations of clinical protocols, which often encode nested conditions, iterative structures, and role-specific actions. To the best of our knowledge, this study is the first to systematically evaluate LVLMs for converting clinical protocol flowcharts into pseudocode to support CDSS development. This work aims to lay the foundation for seamlessly automating the integration of clinical protocols into CDSSs.

III. Dataset Creation

We constructed a dataset titled ClinicalP2Pseudocode (Clinical Protocol to Pseudocode) by collecting EMS protocols publicly released by 6 U.S. states: Alabama, Colorado, Maryland, Nebraska, Ohio, and Pennsylvania. This dataset comprises pairs of clinical protocol flowcharts, their corresponding ground-truth pseudocode, and annotated medical terms explicitly referenced in the flowcharts. Figure 1 (left) presents an example protocol published by the state of Colorado, which outlines the management of patients experiencing bradyarrhythmia with poor perfusion. The flowchart specifies the

interventions authorized for different EMS roles, such as Emergency Medical Technicians (EMT), Advanced Emergency Medical Technicians (AEMT), Emergency Medical Technicians-I (EMT-I), and Paramedics (the most advanced EMS provider level). The corresponding pseudocode, which translates these steps into structured logic, is shown in Figure 1 (right). The dataset development process consists of three key stages: (1) compiling a collection of clinical protocol flowcharts; (2) generating ground-truth pseudocode for each flowchart; and (3) manually annotating medically relevant terms found within the flowcharts (e.g., "heart rate < 60").

TABLE I
STATISTICS OF CLINICAL PROTOCOLS

	min	**max**	**mean**
# of steps	3	26	10.42
# of clinical conditions (CC)	0	8	2.01
# of involved EMS roles	0	6	2.025

A. ClinicalP2Pseudocode - Clinical Protocol Flowcharts

It is important to note that EMS protocols are presented in various formats, including both natural language descriptions and structured flowcharts. For this pilot study, we focused on 163 flowchart-based protocols sourced from six different states. These flowcharts were selected to represent a diverse range of protocol complexities—differing in the number of procedural steps, the clinical conditions (CC) influencing decision pathways, and the EMS personnel authorized to carry out specific actions. Table I presents an overview of their characteristics. Each flowchart includes a minimum of three steps, with some extending to as many as 26. Certain protocols address up to eight distinct CC, each linked to separate treatment pathways. While some flowcharts specify up to six EMS roles, others do not define specific roles, in which case all types of EMS providers are presumed eligible to conduct the procedures outlined in the protocol.

TABLE II
STATISTICS OF GRAPH STRUCTURES BY STATE

State	Tree	DAG	Cycle	Total
Alabama	19	18	2	39
Colorado	34	17	1	52
Maryland	16	0	1	17
Nebraska	24	0	2	26
Ohio	19	5	0	24
Pennsylvania	5	0	0	5

In addition to analyzing the content of the flowchart protocols, we examined their underlying graph structures. Because flowcharts vary considerably in form, we classified them into three categories: (1) tree structures, (2) directed acyclic graphs (DAGs) that are not a tree structure, and (3) graphs with cycles, including those containing directed loops. Table II presents the distribution of flowcharts across these categories by state, illustrating that our study accounts for different levels of structural complexity. For each category, the corresponding ground-truth pseudocode reflects different control flow, including linear sequences for trees, branching logic for DAGs, and loop constructs for cyclic graphs.

B. ClinicalP2Pseudocode - Ground-truth Pseudocode

To create the ground-truth pseudocode for each protocol, two human annotators independently generated pseudocode based on the corresponding flowcharts. In this process, they considered code optimization through function calls, ensured logical consistency, and verified the accuracy of each function's content. Any discrepancies between their versions were resolved through discussion until a consensus was reached. Table III presents the statistical properties of the pseudocode derived from the flowcharts. The finalized ground-truth pseudocode contains up to 16 conditional statements and 41 statements within the main function.

TABLE III
STATISTICS OF THE PSEUDOCODE

	min	**max**	**mean**
# of if conditions	0	16	3.02
# of statements in main	2	41	9.58

TABLE IV
STATISTICS OF THE ANNOTATED MEDICAL TERMINOLOGIES IN THE FLOWCHARTS

	Example	**Total**	**Avg**
Vitals	180 bpm, SBP>90	421	2.5828
Medication	Methylprednisolone 125mg IV/IO	396	2.4294
Treatment	Treat hypotension using Shock Protocol	1226	7.5215

C. ClinicalP2Pseudocode - Medical Terminology Annotation

The objective of annotating the medical terminologies in both the flowchart and the ground-truth pseudocode was to evaluate the readiness of the generated pseudocode for CDSS, which often needs real-time analysis of clinical terms against a patient's current status. For example, the oxygen level "SpO2 94%" specified in a flowchart should be correctly converted and embedded in the pseudocode so that the measured value can be compared with the protocol threshold to determine the next clinical step. Two human annotators independently annotated all flowchart protocols by marking content that belongs to three main EMS fields: vitals, medication, and treatment. All disagreements between the annotators were resolved by discussion with a third annotator. Table IV provides examples of each category along with the corresponding count of annotated terms.

IV. METHOD

A. Pseudocode Generation using LVLMs

We conducted experiments with selected LVLMs through their respective Application Programming Interfaces (API), which allowed us to provide instructions via two role variables. The models were chosen based on their reported performance

in recent literature [20], including Step-1V-32K [1], GPT-4o [2], Gemini-2.0-flash [24], and Claude 3.5 [21]. We conducted experiments both with and without LVLM fine-tuning. Our prompt was structured as follows:

- System: Defines task instructions for the LVLM, specifying the desired role. This variable provides guidance to the AI, ensuring it functions as a pseudocode generator by processing the input flowchart.
- User: Provides an image of the input flowchart for pseudocode generation.

Based on the definitions above, the user input consists of a message containing a flowchart, with the assistant tasked to generate the corresponding pseudocode. Table V outlines our prompt design, which balances sophistication with model generalizability. After experimenting with several strategies (e.g., asking the model to extract medical terminology before generating pseudocode), we finalized an optimized prompt that instructs the LVLMs to convert the input flowchart into pseudocode in a structured format. The output includes role-specific functions and statements when the protocol specifies role-dependent actions. Specifically, the prompt asks the model to first identify any EMS roles mentioned in the flowchart and then generate corresponding role-specific functions, so that the resulting logic can be invoked based on roles while remaining generalizable across state protocols.

Since patient care instructions frequently include treatments or medications, we also instructed LVLMs to produce these as print statements so they can be displayed to EMS providers within a CDSS. In addition, we explicitly defined the required output format: LVLMs were instructed to generate only Python code in the specified format, with no additional explanations.

B. LVLM Finetuning

Recent studies have shown that fine-tuning using curated datasets can significantly improve performance in downstream tasks, including visual question answering, instruction follow-up, and medical image captioning [11], [12]. Fine-tuning enables models to better align with EMS vocabulary, the structural patterns of clinical protocol flowcharts, and the specific requirements of pseudocode generation, thereby enhancing both precision and reliability. For this study, we used 70% of the total flowcharts for fine-tuning and the remaining 30% for evaluation. The fine-tuning set included flowcharts of varying complexity (e.g., different numbers of steps, CC, and EMS roles) with corresponding ground-truth pseudocode. We fine-tuned two models—GPT-4o and Gemini-2.0-flash—while Step-1V-32K and Claude 3.5 were excluded because they do not support API-based model fine-tuning.

C. Extracting Medical Terminology from the Generated Pseudocode

To evaluate how effectively LVLMs retain critical medical terms in the generated pseudocode, we developed a structured extraction pipeline that compares the terms in the ground-truth code to their counterparts in the model-generated output. The process begins by locating the segment of code in the reference pseudocode where each medical term appears, along with its surrounding lines, to define a contextual window for comparison. This window serves as a guide for identifying semantically relevant portions in the generated pseudocode. We then apply a two-stage matching approach to extract the best-matching expression for each term. In the first stage, we use Levenshtein distance [7] to perform approximate string matching and identify top candidate n-grams. In the second stage, we calculate semantic similarity using the Sentence-BERT model [14], embedding both the reference term and candidate phrases and computing cosine similarity. The phrase with the highest combined score across both stages is selected as the corresponding medical term for evaluation.

V. Experimental Setting and Results

A. Evaluation Metrics

Drawing on the literature on code generation, BLEU and CodeBLEU were used for performance comparison and evaluation. As pseudocode has no standard runtime behavior, functional execution metrics are not applicable. While BLEU and CodeBLEU are imperfect proxies for logical correctness, they provide a reproducible and language-agnostic basis for comparison in pseudocode generation tasks, particularly under heterogeneous medical flowchart standards.

- BLEU [16], [20], [22]: A widely used word-overlap metric for evaluating the quality of machine-translated text. It compares n-grams in the generated output with those in the reference (ground-truth) code, counting the number of matches to provide an indication of how closely the output mirrors the intended text.
- CodeBLEU [29]: A specialized metric designed to assess the quality of generated code. It evaluates both syntactical and logical correctness, while also considering the code structure as represented in its abstract syntax tree and data flow. Additionally, it compares n-grams to measure the similarity between the generated code and the reference (ground-truth) code, offering a comprehensive evaluation framework. Equation 1 shows the calculation of CodeBLEU, where w_1, w_2, w_3, and w_4 are set to be 0.25.
- Pass@1: A metric for evaluating functional and logical correctness by executing the generated code against unit tests. Pass@1 represents the probability that the correct code is ranked first for each flowchart [6]. We first convert the pseudocode to Python code and then evaluate the generated Python code using Pass@1 on the test dataset after finetuning the model.

$$\begin{aligned}\text{CodeBLEU} = {} & w_1 \times \text{ngram\_match} \\ & + w_2 \times \text{weighted\_ngram\_match} \\ & + w_3 \times \text{syntax\_match} \\ & + w_4 \times \text{dataflow\_match}\end{aligned} \tag{1}$$

In addition to the evaluation of the code generation, we also assess the accuracy of medical terminology extraction using character-level metrics: Recall, Precision, and F1 score. To

TABLE V
DETAILED DESIGN OF THE PROMPT

LVLM Role	Content

System	You are an AI assistant specialized in converting medical flowcharts into Python code.
	Your task:
	1) Carefully analyze the flowchart to detect any color codes, text labels, or legends that represent EMS roles or certification levels (e.g., "Red = Paramedic", "B = EMT").
	2) If colors are used to indicate roles, infer the corresponding EMS role from the legend and define one function per role.
	3) For each identified role or label, define a corresponding Python function. Use the exact role name or label as the function name.
	4) If the flowchart includes no role indicators, assume all steps apply universally and use a general-purpose function.
	Function definition format:
	`def ROLE(description):`
	`  print("ROLE:{{description}}")`
	Examples:
	• If legend says "Green = EMT": `def EMT(description):`
	`  print("EMERGENCY MEDICAL TECHNICIAN (EMT):{{description}}")`
	• If legend says "Red = Medical Direction": `def ON_LINE_MEDICAL_DIRECTION(description):`
	`  print("ON-LINE MEDICAL DIRECTION:{{description}}")`
	• If labels are used: `def A(description):`
	`  print("A: Advanced EMT{{description}}")`
	`def Paramedic_action(description):`
	`  print("Paramedic Action:{{description}}")`
	Instruction blocks:
	• For each instruction box, use the appropriate role function.
	• Use `if`, `elif`, and `else` for branching logic.
	• For general steps with no role, use `print(...)` or `All_Levels(...)`.
	Additional rules:
	• Include floating comments or notes using `print(...)`.
	• Maintain the original logical structure and branching flow.
	Output requirements:
	• Output only raw Python code.
	• Do not include markdown, comments, explanations, or `if __name__ == "__main__"` blocks.

User	*input flowchart*

measure performance, we apply a strict matching approach, which aligns with the evaluation used in prior string match research. Under this method, all extracted medical terms are tokenized into individual characters, and character-level precision and recall are computed. A character is classified as a True Positive (TP) if it appears in both the annotated ground-truth and the LVLM-generated output. Characters present in the LVLM output but absent from the ground-truth are labeled as False Positives (FP), while characters in the ground-truth but missing from the LVLM output are treated as False Negatives (FN). Equations 2 to 4 define the calculation of Recall (R), Precision (P), and F1 score.

$$R = \frac{TP}{TP + FN} \tag{2}$$

$$P = \frac{TP}{TP + FP} \tag{3}$$

$$F1 = \frac{2 \times R \times P}{R + P} \tag{4}$$

B. Performance Evaluation and Comparison

1) Pseudocode generation evaluation: Table VI presents the performance comparison of the four evaluated LVLMs on generating pseudocode without model finetuning on the overall dataset. We calculated performance metrics based on the complexity of the clinical protocols, considering that some treatments can only be performed by certain EMS roles and the CC assessments required for different treatments. The results indicate that GPT-4o achieves the best performance in terms of average BLEU and CodeBLEU scores. Claude 3.5 performs slightly better than the other two models in terms of average CodeBLEU.

Claude 3.5 demonstrates relatively good performance in terms of BLEU score when the clinical protocols involve multiple roles and multiple CC linked to different treatments; in other words, when there are no if-else conditions in the generated pseudocode. Across all four models, performance is generally better on protocols that do not involve multiple CC.

Among the four LVLMs evaluated, only GPT-4o and Google Gemini-2.0-Flash support model fine-tuning. Table VII com-

TABLE VI
PERFORMANCE COMPARISON ON PSEUDOCODE GENERATION WITHOUT LVLM FINETUNING ON THE OVERALL DATASET

Clinical protocol type	Count	GPT-4o		Gemini-2.0-flash		Step-1V-32K		Claude 3.5	
		BLEU	CodeBLEU	BLEU	CodeBLEU	BLEU	CodeBLEU	BLEU	CodeBLEU
Multiple Roles and Multiple CC	94	0.388	0.591	0.424	0.475	0.313	0.462	0.467	0.468
Multiple Roles and No Multiple CC	35	0.496	0.731	0.559	0.631	0.474	0.628	0.496	0.515
Single Role and Multiple CC	28	0.397	0.586	0.463	0.520	0.256	0.428	0.429	0.440
Single Role and No Multiple CC	6	0.609	0.779	0.452	0.625	0.430	0.599	0.459	0.460
Macro-average	163	0.473	0.672	0.474	0.563	0.368	0.529	0.463	0.471

Note: Bold numbers indicate the highest performance of the category.

TABLE VII
PERFORMANCE COMPARISON ON PSEUDOCODE GENERATION W AND W/O LVLM FINETUNING ON THE EVALUATION DATASET

Clinical protocol type	Count	GPT-4o				Gemini-2.0-flash			
		w/o Fine-tuning		w Fine-tuning		w/o Fine-tuning		w Fine-tuning	
		BLEU	CodeBLEU	BLEU	CodeBLEU	BLEU	CodeBLEU	BLEU	CodeBLEU
Multiple Roles and Multiple CC	38	0.385	0.581	0.658	0.756	0.415	0.465	0.686	0.680
Multiple Roles and No Multiple CC	3	0.502	0.734	0.849	0.887	0.600	0.651	0.762	0.710
Single Role and Multiple CC	8	0.385	0.553	0.528	0.638	0.503	0.525	0.691	0.647
Single Role and No Multiple CC	2	0.476	0.688	0.667	0.834	0.345	0.587	0.714	0.802
Macro-average	51	0.437	0.639	0.675	0.779	0.466	0.557	0.713	0.710

Note: Bold numbers indicate the highest performance of the category.

TABLE VIII
PERFORMANCE COMPARISON ON PSEUDOCODE GENERATION FOR GRAPH STRUCTURES W FINETUNED LVLM ON EVALUATION DATA

Graph Structure	Count	BLEU	CodeBLEU	Pass@1
Tree	36	0.6459	0.7455	0.2778
DAG	12	0.8264	0.8799	0.5000
Cycle	3	0.4459	0.5664	1.0000

pares their performance before and after fine-tuning. The results indicate a substantial improvement (over 15%) in overall performance after fine-tuning. Specifically, GPT-4o achieves a CodeBLEU score of 0.754 in complex clinical protocols that involve multiple roles and CC. Although Gemini-2.0-Flash demonstrates comparable performance to GPT-4o in protocols characterized by a single role but multiple CC, GPT-4o consistently outperforms in more structurally complex scenarios. Our Pass@1 evaluation further demonstrates that the GPT-4o model's performance improves after fine-tuning.

Beyond protocol-level evaluation, we assessed model performance in pseudocode generation as well as logical correctness using Pass@1 across different protocol graph structures (Table VIII). Since GPT-4o performed best in earlier experiments, we focused on its fine-tuned variant. Results show consistently strong performance across trees, DAGs, and cycles, with DAGs slightly outperforming others in BLEU and CodeBLEU scores. The lower scores on cycles may reflect fewer examples in the dataset. However, the Pass@1 results indicate that the generated code exhibits stronger logical correctness on DAGs and cyclic graphs than on trees, demonstrating the necessity of considering both evaluation metrics for a comprehensive assessment. Overall, the fine-tuned model generalizes well across structural complexities, demonstrating robustness in handling diverse protocol representations.

2) Medical terminology extraction evaluation: Table IX presents the results of medical term extraction from clinical protocols embedded within pseudocode. Among the models evaluated, Step-1V-32K and GPT-4o achieved the strongest overall performance. Step-1V-32K was particularly effective in identifying treatment and medication terms, while Claude 3.5 outperformed others in extracting vital sign terms. Gemini-2.0-flash, on the other hand, lagged behind the other three LVLMs in overall performance. Table X displays the results following the fine-tuning of GPT-4o and Gemini-2.0-flash. After fine-tuning, Gemini 2.0 flash showed a substantial improvement - exceeding a 20% increase in both the F1 score and recall - and ultimately surpassed the fine-tuned GPT-4o in overall extraction performance. However, GPT-4o maintained a slight edge in extracting medication-related terms.

3) Computational setting and efficiency: The best performance was achieved by using a fine-tuned GPT-4o model (ft:gpt-4o-2024-08-06) with a temperature of 0.2 and a maximum output length of 3,000 tokens. Based on 51 inference runs on the test set, the mean response latency was 16.53 seconds (median: 10.62 s), with a standard deviation of 14.70 seconds; the 95th percentile latency was 40.80 seconds, and observed latencies ranged from 4.30 to 86.05 seconds. All experiments were executed on a local Windows 10 machine running Python 3.12.10, equipped with an Intel-based processor (6 logical CPUs) and 47.9 GB of RAM.

C. Case Study

Based on the performance of these LVLMs, we analyzed specific cases to identify the challenges associated with generating pseudocode from EMS protocols using LVLMs with and without fine-tuning.

Figure 2 shows the "BURN" protocol from the Ohio Adult EMS Guidelines and the 2024 Procedure Manual, which yielded low evaluation scores across all four LVLMs when generating pseudocode. This protocol outlines various clinical

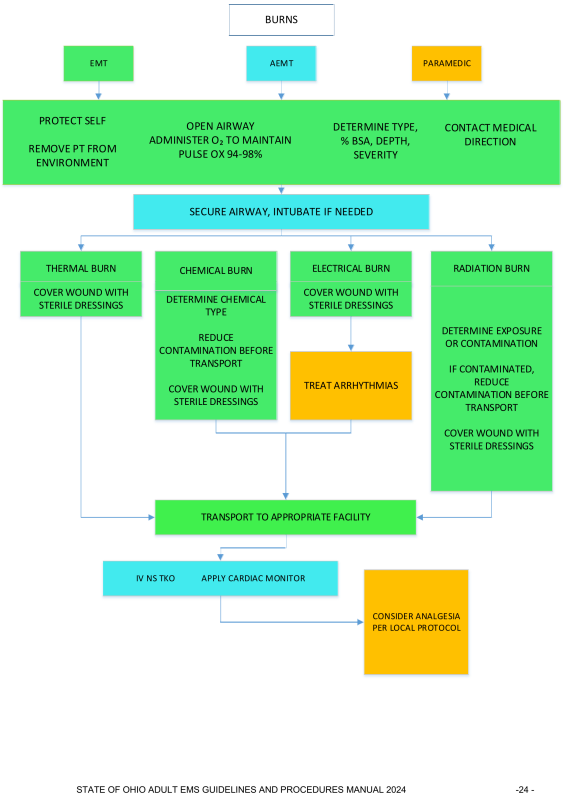

Fig. 2. Protocol "Burn"

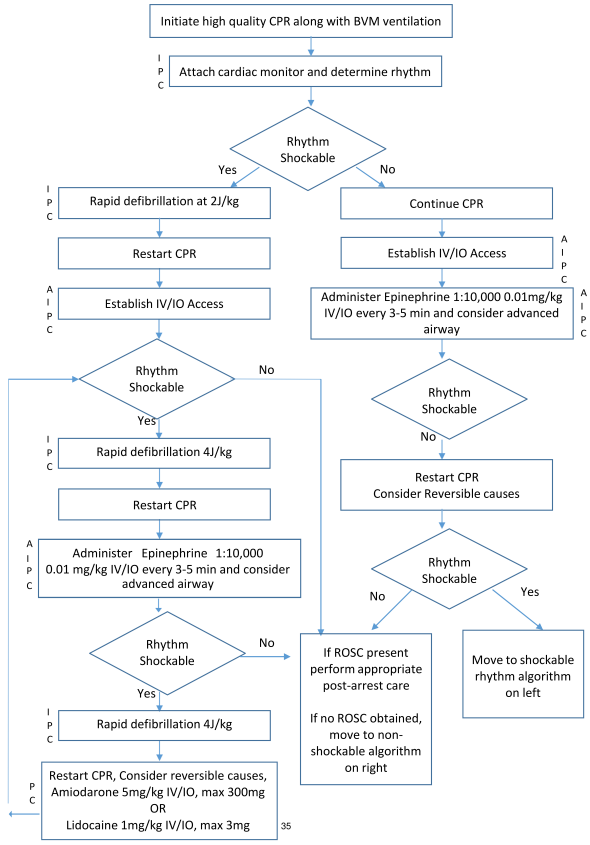

Fig. 3. Protocol "Cardiac Arrest - Pediatric"

TABLE IX
PERFORMANCE COMPARISON ON MEDICAL TERMINOLOGY EXTRACTION WITHOUT LVLM FINETUNING ON THE OVERALL DATASET

Categories of the Medical Terminology	**GPT-4o**			**Gemini-2.0-flash**			**Step-1V-32K**			**Claude 3.5**		
	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Vitals	0.645	0.937	0.585	0.560	0.931	0.497	0.625	0.939	0.558	**0.702**	**0.950**	**0.643**
Medication	0.607	**0.973**	0.543	0.490	0.943	0.420	**0.646**	0.959	**0.594**	0.545	0.947	0.469
Treatment	0.740	0.965	0.700	0.611	0.960	0.504	**0.747**	**0.975**	**0.700**	0.731	0.975	0.683
Average	0.707	0.962	0.661	0.586	0.953	0.522	**0.714**	**0.967**	**0.663**	0.700	0.967	0.647

Note: Bold numbers indicate the highest performance of the category.

TABLE X
PERFORMANCE COMPARISON ON MEDICAL TERMINOLOGY EXTRACTION W AND W/O LVLM FINETUNING ON THE EVALUATION DATASET

Categories of the Medical Terminology	**GPT-4o**						**Gemini-2.0-flash**					
	w/o Fine-tuning			**w Fine-tuning**			**w/o Fine-tuning**			**w Fine-tuning**		
	F1	P	R	F1	P	R	F1	P	R	F1	P	R
Vitals	0.671	0.929	0.614	0.737	0.956	0.686	0.609	0.932	0.549	**0.800**	**0.957**	**0.752**
Medication	0.613	0.967	0.548	**0.731**	**0.983**	**0.676**	0.470	0.935	0.395	0.722	0.971	0.667
Treatment	0.738	0.975	0.694	0.780	**0.997**	0.685	0.736	0.963	0.623	**0.808**	0.982	**0.770**
Average	0.706	0.964	0.657	0.764	**0.977**	0.682	0.605	0.943	0.510	**0.794**	0.975	**0.715**

Note: Bold numbers indicate the highest performance of the category.

procedures tied to specific EMS roles and includes condition assessments such as "thermal burn" and "chemical burn", each requiring corresponding if-else logic in the pseudocode. Additionally, it involves three distinct EMS roles, each authorized to perform specific actions. As illustrated in Figure 4, even the best-performing model prior to fine-tuning, GPT-4o, failed to correctly identify all CC assessments, especially those linked to particular roles. All generated functions were indiscriminately assigned to every EMS role, and the pseudocode lacked conditional structures that mirrored the decision paths in the original flowchart, such as differentiating types of burns. After fine-tuning, as shown in Figure 5, the model more

```
def EMT(description):
  print(f"EMERGENCY MEDICAL TECHNICIAN (EMT):\n{description}\n")

def AEMT(description):
  print(f"ADVANCED EMERGENCY MEDICAL TECHNICIAN (AEMT):\n{description}\n")

def PARAMEDIC(description):
  print(f"PARAMEDIC:\n{description}\n")

def All_Levels(description):
  print(f"{description}\n")

EMT("Protect self\nRemove pt from environment")
AEMT("Open airway\nAdminister O2 to maintain pulse ox 94-98%")
PARAMEDIC("Determine type, % BSA, depth, severity")
PARAMEDIC("Contact medical direction")
All_Levels("Secure airway, intubate if needed")
All_Levels("Thermal Burn\nCover wound with sterile dressings")
All_Levels("Chemical Burn\nDetermine chemical type\nReduce contamination before
    transport\nCover wound with sterile dressings")
All_Levels("Electrical Burn\nCover wound with sterile dressings\nTreat arrhythmias")
All_Levels("Radiation Burn\nDetermine exposure or contamination\nIf contaminated,
    reduce contamination before transport\nCover wound with sterile dressings")
All_Levels("Transport to appropriate facility")
All_Levels("IV NS TKO\nApply cardiac monitor")
All_Levels("Consider analgesia per local protocol")
```

Fig. 4. Generated Pseudocode for Protocol "Burn" using GPT-4o without finetuning

accurately associated actions with their corresponding EMS roles and correctly separated them into distinct functions. The only notable error was the introduction of an incorrect role, ON_LINE_MEDICAL_DIRECTION, which was mistakenly assigned to the step "CONSIDER ANALGESIA PER LOCAL PROTOCOL"—a task appropriate for the paramedic role.

Conversely, we identified several challenging cases that could not be accurately processed to generate pseudocode, even after model fine-tuning. For example, the protocol "Cardiac Arrest - Pediatric" (illustrated in Figure 3), from the 2022 Alabama EMS Patient Care Protocols, requires repeated rhythm evaluation to determine whether the rhythm is shockable. After each evaluation, if the rhythm is shockable, the model must account for interventions such as "rapid defibrillation at 2J/kg", "restart CPR", and "establish IV/IO access", all of which depend on the patient's condition. This structure creates a loop of treatments based on ongoing rhythm assessments. While fine-tuning improved the model's ability to recognize distinct roles and associated actions, it still struggled to represent this loop structure—capturing the need for continuous evaluation and action based on the shockable rhythm outcome. A likely contributing factor is the limited number of training examples in the original dataset that involve iterative clinical decision-making processes requiring loop constructs in the generated code.

In summary, our findings highlight the significant challenges LVLMs face in interpreting complex clinical protocols involving multiple roles and condition assessments, particularly when these assessments are linked to different treatments administered by various roles.

VI. Discussion

The results of this study underscore the potential of LVLMs to automate the translation of clinical flowchart protocols into pseudocode, addressing a critical challenge in developing CDSS. By leveraging LVLMs, healthcare organizations could significantly reduce the time and effort required to digitize clinical protocols, enabling more efficient integration of evidence-based guidelines or protocols into real-world decision-making tools.

Compared with traditional rule-based CDSS development, which relies on experts and engineers to manually encode clinical logic, LVLM-based automation offers a more efficient and scalable alternative. Manual methods are time-consuming, prone to inconsistent standards, and require repeated revisions when guidelines are updated. In contrast, LVLMs can generate executable pseudocode from flowcharts rapidly, reducing human error and circumventing the limitations of manual development. Future work could include expert-based evaluation of the semantic fidelity and correctness of the generated code.

The methodologies explored in this study have broader implications beyond EMS. They could be adapted to evaluate LVLMs in other medical domains, such as critical care, oncology, and ED, where flowchart-based protocols are equally prevalent and critical to clinical workflows. For instance, ED protocols frequently use flowcharts to prioritize and manage patient triage and treatments. Automating the conversion of these protocols or guidelines could support the development of ED-specific clinical decision support systems, enhancing triage efficiency, diagnostic accuracy, and treatment workflows.

```
# Define functions for each certification level
def EMT(description):
    print(f"EMERGENCY MEDICAL TECHNICIAN (EMT):\n{description}\n")
def AEMT(description):
    print(f"ADVANCED EMERGENCY MEDICAL TECHNICIAN (AEMT):\n{description}\n")
def PARAMEDIC(description):
    print(f"PARAMEDIC:\n{description}\n")

def ON_LINE_MEDICAL_DIRECTION(description):
    print(f"ON-LINE MEDICAL DIRECTION:\n{description}\n")
# ----------------------------
# BURNS
# ----------------------------
# Initial steps for all EMS levels
EMT("PROTECT SELF\nREMOVE PT FROM ENVIRONMENT")
EMT("OPEN AIRWAY\nADMINISTER O2 TO MAINTAIN PULSE OX 94-98%")
EMT("DETERMINE TYPE, % BSA, DEPTH, SEVERITY")
EMT("CONTACT MEDICAL DIRECTION")
# Secure airway if needed
AEMT("SECURE AIRWAY, INTUBATE IF NEEDED")
# Determine burn type
burn_type = "THERMAL"  # Example: THERMAL, CHEMICAL, ELECTRICAL, RADIATION
if burn_type == "THERMAL":
    EMT("THERMAL BURN\nCOVER WOUND WITH STERILE DRESSINGS")
elif burn_type == "CHEMICAL":
    EMT("CHEMICAL BURN\nDETERMINE CHEMICAL TYPE\nREDUCE CONTAMINATION BEFORE
        TRANSPORT\nCOVER WOUND WITH STERILE DRESSINGS")
elif burn_type == "ELECTRICAL":
    EMT("ELECTRICAL BURN\nCOVER WOUND WITH STERILE DRESSINGS")
    PARAMEDIC("TREAT ARRHYTHMIAS")
elif burn_type == "RADIATION":
    EMT("RADIATION BURN\nDETERMINE EXPOSURE OR CONTAMINATION\nIF CONTAMINATED, REDUCE
        CONTAMINATION BEFORE TRANSPORT\nCOVER WOUND WITH STERILE DRESSINGS")
# Transport and additional care
EMT("TRANSPORT TO APPROPRIATE FACILITY")
AEMT("IV NS TKO\nAPPLY CARDIAC MONITOR")
ON_LINE_MEDICAL_DIRECTION("CONSIDER ANALGESIA PER LOCAL PROTOCOL")
```

Fig. 5. Generated Pseudocode for Protocol "Burn" using GPT-4o with finetuning

Despite these promising results, the challenges identified in this study highlight the need for further advancements in LVLM capabilities. A key limitation is the models' difficulty in interpreting intricate conditional assessments tied to specific care provider roles or complex clinical situations which are essential for ensuring reliability in real-world applications. Addressing these challenges could involve several strategies. For instance, adapting LVLMs that were trained using biomedical data [11], [15], or enhancing LVLMs' fine-tuning by including a larger and domain-specific dataset to improve their understanding of clinical flowchart protocols, especially those involving complex logic and role-specific tasks [16]. LVLMs pretrained on the medical domain outperform general models on domain-specific tasks. BioMistral [10], a domain-adaptive pretrained model [5], significantly outperforms Mistral [9], the base model, demonstrating the effectiveness of domain adaptation strategies. Additionally, retrieval-augmented generation (RAG) has shown potential for improving code generation by incorporating relevant external context at inference time. For example, models that leverage the retrieval of curated programming resources can outperform those that rely purely on parametric knowledge, especially in unfamiliar or open-domain coding tasks [27]. Additionally, the embedding of medical ontologies or structured domain knowledge in LVLM could enhance their contextual understanding of clinical concepts [3], improving the accuracy of pseudocode generation.

Furthermore, while the fine-tuned LVLM showed improved performance on EMS protocols, its applicability to protocols from different regions or institutions remains uncertain. Although our dataset includes flowcharts from multiple states with varying standards and conventions, the prompts used in reasoning cannot explicitly encode all protocol standards or reliably adapt to unknown regional differences, leaving the generalizability of the approach to be further investigated.

A few limitations should be noted. First, the small ground-truth dataset used for evaluation may not capture all possible representations of clinical protocols. Second, the pseudocode generation process does not account for integration with EHR back-end databases or connections to external devices for data input. Finally, the dataset was sourced from only six U.S. states. Future work should expand its size and geographic diversity to ensure broader applicability, as LVLMs typically

require large, heterogeneous datasets for robust evaluation.

VII. Conclusions

This study investigated the potential of LVLMs to automate the translation of clinical flowchart protocols into structured pseudocode, using EMS protocols as a representative use case. We evaluated four state-of-the-art LVLMs—including two fine-tuned models—and demonstrated that these models can capture procedural logic and differentiate role-specific actions with reasonable accuracy. Fine-tuning notably improved performance, yielding more than a 15% gain in both CodeBLEU scores and medical term extraction accuracy. Despite these promising results, significant challenges remain. Current LVLMs struggle with complex protocol elements such as nested conditions, iterative structures, and ambiguous or inconsistently defined clinical roles. Nonetheless, this work represents an important first step toward leveraging LVLMs to streamline the digitization of clinical protocols.

Acknowledgment

This study was supported by the National Science Foundation (grant# 2237097) and the National Institute of Health (grant#1R15LM014556-01).

References

[1] Step 1v 32k. https://lobechat.com/discover/model/step-1v-32k, 2024.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, and S. Anadkat. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] B. Boecking, N. Usuyama, S. Bannur, D. C. Castro, A. Schwaighofer, S. Hyland, M. Wetscherek, T. Naumann, A. Nori, and J. Alvarez-Valle. Making the most of text semantics to improve biomedical vision–language processing. In *European conference on computer vision*, 2022.

[4] M. C. Carlisle, T. A. Wilson, J. W. Humphries, and S. M. Hadfield. Raptor: introducing programming to non-majors with flowcharts. *Journal of Computing Sciences in Colleges*, 19(4):52–60, 2004.

[5] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks, 2020.

[6] Mengliang He, Jiayi Zeng, Yankai Jiang, Wei Zhang, Zeming Liu, Xiaoming Shi, and Aimin Zhou. Flow2code: Evaluating large language models for flowchart-based code generation capability. *arXiv preprint arXiv:2506.02073*, 2025.

[7] R. Y. Huang, C. Zhang, and H. L. Lim. Ribotyping staphylococcus epidermidis using probabilistic sequence analysis and levenshtein distance algorithm. *Current Microbiology*, 82(2):1–22, 2025.

[8] P. Jia, L. Zhang, J. Chen, P. Zhao, and M. Zhang. The effects of clinical decision support systems on medication safety: An overview. *PloS One*, 11(12):e0167683, 2016.

[9] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[10] Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. Biomistral: A collection of open-source pretrained large language models for medical domains, 2024.

[11] C. Li, C. Wong, S. Zhang, N. Usuyama, H. Liu, J. Yang, T. Naumann, H. Poon, and J. Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. In *Advances in Neural Information Processing Systems*, volume 36, pages 28541–28564, 2023.

[12] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, 2023.

[13] Z. Liu, X. Hu, D. Zhou, L. Li, X. Zhang, and Y. Xiang. Code generation from flowcharts with texts: A benchmark dataset and an approach. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.

[14] Xiao Luo, Le Zhou, Kathleen Adelgais, and Zhan Zhang. Assessing the effectiveness of automatic speech recognition technology in emergency medicine settings: a comparative study of four ai-powered engines. *Journal of Healthcare Informatics Research*, pages 1–19, 2025.

[15] Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Yash Dalmia, Jure Leskovec, Cyril Zakka, Eduardo Pontes Reis, and Pranav Rajpurkar. Med-flamingo: a multimodal medical few-shot learner. In Stefan Hegselmann, Antonio Parziale, Divya Shanmugam, Shengpu Tang, Mercy Nyamewaa Asiedu, Serina Chang, Tom Hartvigsen, and Harvineet Singh, editors, *Proceedings of the 3rd Machine Learning for Health Symposium*, volume 225 of *Proceedings of Machine Learning Research*, pages 353–367. PMLR, 10 Dec 2023.

[16] V. Nath, W. Li, D. Yang, A. Myronenko, M. Zheng, Y. Lu, Z. Liu, H. Yin, Y. Tang, and P. Guo. Vila-m3: Enhancing vision-language models with medical expert knowledge. *arXiv preprint arXiv:2411.12915*, 2024.

[17] Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen tau Yih, Sida I. Wang, and Xi Victoria Lin. Lever: Learning to verify language-to-code generation with execution, 2023.

[18] O. O'Donoghue, A. Shtedritski, J. Ginger, R. Abboud, A. Ghareeb, and S. Rodriques. Bioplanner: Automatic evaluation of llms on protocol planning in biology. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[19] Femke Ongenae, Femke De Backere, Kristof Steurbaut, Kirsten Colpaert, Wannes Kerckhove, Johan Decruyenaere, and Filip De Turck. Towards computerizing intensive care sedation guidelines: design of a rule-based architecture for automated execution of clinical guidelines. *BMC medical informatics and decision making*, 10:1–22, 2010.

[20] H. Pan, Q. Zhang, C. Caragea, E. Dragut, and L. J. Latecki. Flowlearn: Evaluating large vision-language models on flowchart understanding. In *ECAI 2024*, pages 73–80. IOS Press, 2024.

[21] S. A. A. Safavi-Naini, S. Ali, O. Shahab, Z. Shahhoseini, T. Savage, S. Rafiee, J. S. Samaan, R. A. Shabeeb, F. Ladak, and J. O. Yang. Vision-language and large language model performance in gastroenterology: Gpt, claude, llama, phi, mistral, gemma, and quantized models. *arXiv preprint arXiv:2409.00084*, 2024.

[22] S. Shukla, P. Gatti, Y. Kumar, V. Yadav, and A. Mishra. Towards making flowchart images machine interpretable. In *International Conference on Document Analysis and Recognition*, 2023.

[23] R. T. Sutton, D. Pincock, D. C. Baumgart, D. C. Sadowski, R. N. Fedorak, and K. I. Kroeker. An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ Digital Medicine*, 3(1):17, 2020.

[24] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, and A. Hauth. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[25] Hanbin Wang, Xiaoxuan Zhou, Zhipeng Xu, Keyuan Cheng, Yuxin Zuo, Kai Tian, Jingwei Song, Junting Lu, Wenhui Hu, and Xueyang Liu. Code-vision: Evaluating multimodal llms logic understanding and code generation capabilities. *arXiv preprint arXiv:2502.11829*, 2025.

[26] J. Wang, L. Cao, X. Luo, Z. Zhou, J. Xie, A. Jatowt, and Y. Cai. Enhancing large language models for secure code generation: A dataset-driven study on vulnerability mitigation. *arXiv preprint arXiv:2310.16263*, 2023.

[27] Zora Zhiruo Wang, Akari Asai, Xinyan Velocity Yu, Frank F. Xu, Yiqing Xie, Graham Neubig, and Daniel Fried. Coderag-bench: Can retrieval augment code generation?, 2025.

[28] X.-H. Wu, M.-C. Qu, Z.-Q. Liu, and J.-Z. Li. Research and application of code automatic generation algorithm based on structured flowchart. *Journal of Software Engineering and Applications*, 4(9):534–539, 2011.

[29] S. Xu, Z. Li, K. Mei, and Y. Zhang. Core: Llm as interpreter for natural language programming, pseudo-code programming, and flow programming of ai agents. *arXiv e-prints, arXiv:2405.06907*, 2024.

[30] Y. Xu, W. Li, P. Vaezipoor, S. Sanner, and E. B. Khalil. Llms and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *arXiv preprint arXiv:2305.18354*, 2023.

[31] S. Yi, J. Lim, and J. Yoon. Protocollm: Automatic evaluation framework of llms on domain-specific scientific protocol formulation tasks. *arXiv preprint arXiv:2410.04601*, 2024.