Two-Stage Detection of Block Diagrams using Large Language Model

Shunichi Ejima[1] and Masanobu Koga[1]*

[1]Department of Intelligent and Contorl Systems, Kyushu Institute of Technology, Fukuoka, Japan(ejima.shunichi154@mail.kyutech.jp, koga@ics.kyutech.ac.jp)* Corresponding author

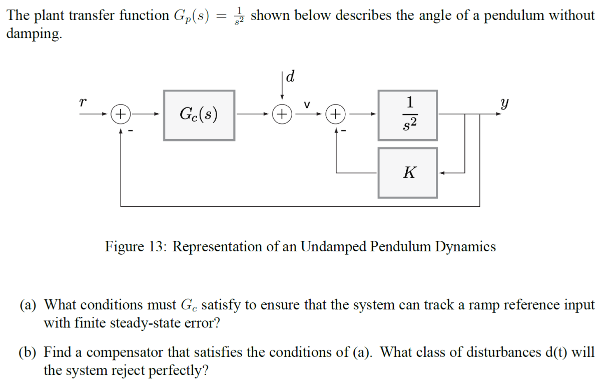
Abstract: In this study, we introduce two novel two-stage detection approaches to enhance the performance of block diagram recognition using large language models (LLMs), by integrating visual inference and text generation capabilities. The first approach, termed Two-Stage by Double LLM (TSD), employs one LLM to generate LaTeX code from a block diagram image and a second LLM to interpret the code. The second approach, termed Two-Stage by Single LLM (TSS), similarly generates LaTeX code from the image but uses the same LLM for both generation and interpretation. We evaluate these methods on the dataset of block diagram images. Experimental results confirm that while ChatGPT o4-mini attains the accuracy of 35%, the TSD and TSS methods achieve significantly higher accuracies of 87% and 96%, respectively.

Keywords: LLM, ChatGPT, Block Diagram, Simulink Model, Detection

1. INTRODUCTION

In recent years, generative artificial intelligence has transcended discrete tasks, including image recognition and natural language processing, evolving into platforms capable of holistically processing multimodal data [1]. Notably, large language models (LLMs) with integrated visual processing capabilities, exemplified by ChatGPT o4-mini [2], have extended their functionality to encompass the interpretation and reasoning of charts and diagrams domains which have been traditionally considered challenging [3], [4]. Consequently, these models are expected to find applications across diverse disciplines, such as medicine and the natural sciences [5], education [6], cognitive science and psychology [7], security [8], and traffic engineering [9].

The performance of LLMs within the domain of control engineering has been investigated [3]. Preliminary studies indicate that the problems with graphical elements, such as block diagrams (e.g., Fig.1), exhibit lower accuracy relative to text-based problem types[3]. In control engineering, block diagrams play a pivotal role in both industrial and academic fields by facilitating rapid model-based design and simulation [3], [10], [11], thereby substantially reducing cycle times of design. Nevertheless, the preliminary studies evaluating the capability of large language models (LLMs) to recognize block diagrams have reported lower accuracy when confronted with diagrams which have multiple feedback loops and disturbances.



The plant transfer function $G_p(s) = \frac{1}{s^2}$ shown below describes the angle of a pendulum without damping.

Figure 13: Representation of an Undamped Pendulum Dynamics

(a) What conditions must G_c satisfy to ensure that the system can track a ramp reference input with finite steady-state error?

(b) Find a compensator that satisfies the conditions of (a). What class of disturbances d(t) will the system reject perfectly?

Fig. 1.: (4.4 Tracking a Ramp Reference) benchmark problem in [2]

In order to overcome these challenges, this paper introduces two novel two-stage methodologies [13] for enhancing LLM-based block diagram recognition. The first approach, **Two-Stage by Double LLM (TSD)**, employs the first LLM to convert block diagram images into TikZ-formatted LaTeX code [12][14][15][16], followed by the second LLM that interprets the LaTeX output to generate MATLAB scripts for constructing Simulink models. The second approach, **Two-Stage by Single LLM (TSS)**, similarly translates the block diagram image into TikZ LaTeX via a LLM, then makes the same LLM to interpret the LaTeX to produce the requisite MATLAB code. By integrating visual inference with structured text transformation, both methods aim to substantially enhance recognition performance. To validate their effectiveness, we constructed a dataset of block diagram images and conducted comprehensive evaluations.

The experimental results indicate that the recognition accuracy of block diagrams using ChatGPT o4-mini was limited to approximately 35%, whereas TSD and TSS achieved high accuracies of approximately 87% and 96%, respectively. These results confirm the effectiveness of our methods in accurately recognizing complex block diagrams containing feedback loops and disturbances.

This paper is organized as follows: Chapter 2 describes the problem setting of having a generative AI recognize block diagrams and then generate Simulink models. Chapter 3 presents the proposed methods for improving recognition accuracy. Chapter 4 evaluates the proposed methods. Chapter 5 offers a discussion, and Chapter 6 concludes the paper.

2. PROBLEM-FORMULATION

2.1 Large-scale language model (LLM)

In this study, we employ the large language models provided by OpenAI via ChatGPT. ChatGPT offers various models with different performance profiles; in this study, we utilize o4-mini, the most recent model available as of April 17, 2025, distinguished by its superior image recognition and visual inference capabilities.

2.2 Block Diagram Model

Simulink's block diagram model is used to evaluate the performance on block diagram recognition by LLM.MATLAB and Simulink are numerical analysis and simulation environments provided by MathWorks, and are widely used in the fields of control engineering, signal processing, robotics, and machine learning, among others.The version of MATLAB used in this study is "R2022b".

2.3 Recognition of block diagrams

In the experiment for generating Simulink model from block diagram images, the user is supposed to provide a block diagram image with the following message to ChatGPT. The message reads, "Please generate the MATLAB code to construct the Simulink model corresponding to the block diagram which will be uploaded the MATLAB version is R2022b." Upon execution of the MATLAB code produced by ChatGPT, a Simulink model is generated. The generated block diagram is then evaluated by comparing the connectivity of the model produced by the code with that of the original block diagram. Since the primary objective of the generated Simulink model is analysis and simulation of the control system, discrepancies in layout or graphical notation are permissible; only the equivalence of interconnections is required for correctness.

Examples of the block diagram images supplied to ChatGPT appear in Fig. 2, the instructions presented to ChatGPT are illustrated in Fig. 3, and the MATLAB script invoked to generate the Simulink model is provided in Fig. 4. The Simulink model

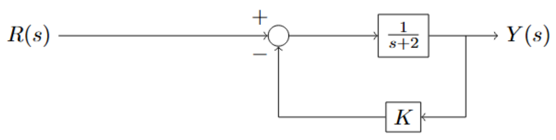
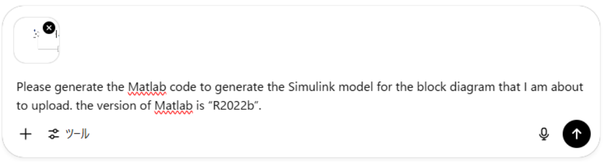


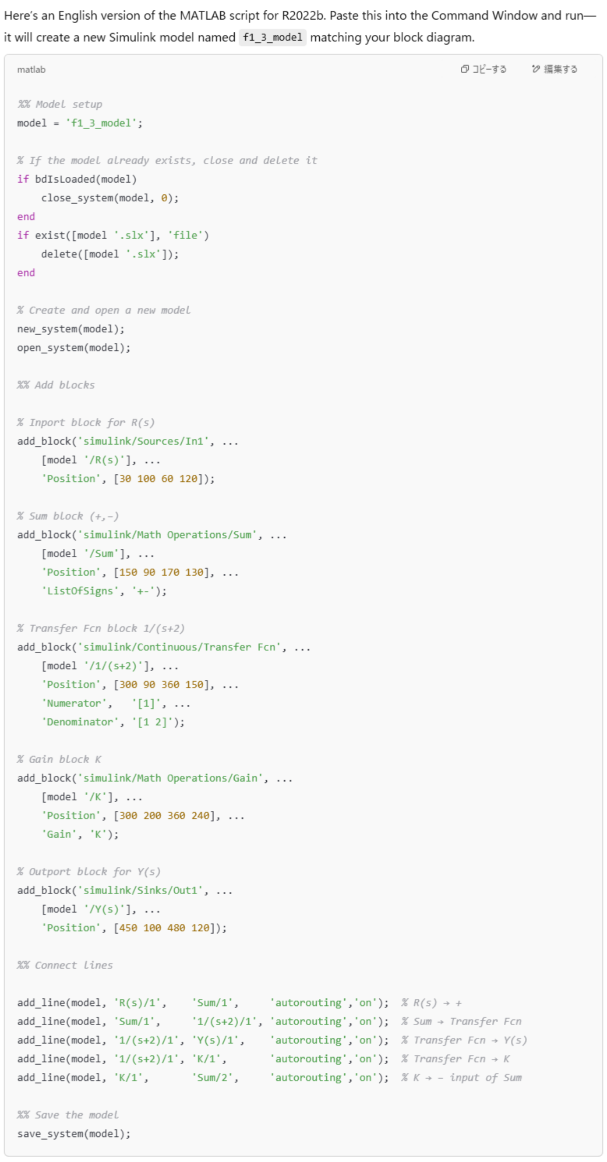
Fig. 2.: Example of block diagram image with single feedback loop

generated by executing the code in Fig. 4 is presented



Please generate the Matlab code to generate the Simulink model for the block diagram that I am about to upload. the version of Matlab is "R2022b".

Fig. 3.: Instructions to ChatGPT



Here's an English version of the MATLAB script for R2022b. Paste this into the Command Window and run—it will create a new Simulink model named `f1_3_model` matching your block diagram.

```matlab
%% Model setup
model = 'f1_3_model';

% If the model already exists, close and delete it
if bdIsLoaded(model)
    close_system(model, 0);
end
if exist([model '.slx'], 'file')
    delete([model '.slx']);
end

% Create and open a new model
new_system(model);
open_system(model);

%% Add blocks

% Inport block for R(s)
add_block('simulink/Sources/In1', ...
    [model '/R(s)'], ...
    'Position', [30 100 60 120]);

% Sum block (+,-)
add_block('simulink/Math Operations/Sum', ...
    [model '/Sum'], ...
    'Position', [150 90 170 130], ...
    'ListOfSigns', '+-');

% Transfer Fcn block 1/(s+2)
add_block('simulink/Continuous/Transfer Fcn', ...
    [model '/1/(s+2)'], ...
    'Position', [300 90 360 150], ...
    'Numerator',   '[1]', ...
    'Denominator', '[1 2]');

% Gain block K
add_block('simulink/Math Operations/Gain', ...
    [model '/K'], ...
    'Position', [300 200 360 240], ...
    'Gain', 'K');

% Outport block for Y(s)
add_block('simulink/Sinks/Out1', ...
    [model '/Y(s)'], ...
    'Position', [450 100 480 120]);

%% Connect lines

add_line(model, 'R(s)/1',    'Sum/1',     'autorouting','on');  % R(s) → +
add_line(model, 'Sum/1',     '1/(s+2)/1', 'autorouting','on');  % Sum → Transfer Fcn
add_line(model, '1/(s+2)/1', 'Y(s)/1',    'autorouting','on');  % Transfer Fcn → Y(s)
add_line(model, '1/(s+2)/1', 'K/1',       'autorouting','on');  % Transfer Fcn → K
add_line(model, 'K/1',       'Sum/2',     'autorouting','on');  % K → - input of Sum

%% Save the model
save_system(model);
```

Fig. 4.: MATLAB code generated by ChatGPT

in Fig. 5. Despite differences in layout and graphical representation between Fig. 2 and 5, the interconnection of the blocks remains identical. Therefore, we conclude that the executed MATLAB script successfully generated an correct Simulink model. The Simulink model generated from the block diagram image shown in Fig.6 is shown in Fig.7. The inner feedback loop is missing in the Simulink model in Fig.7, and the connection relation is different from the given block diagram image;therefore, we determine that the generated Simulink model is incorrect.

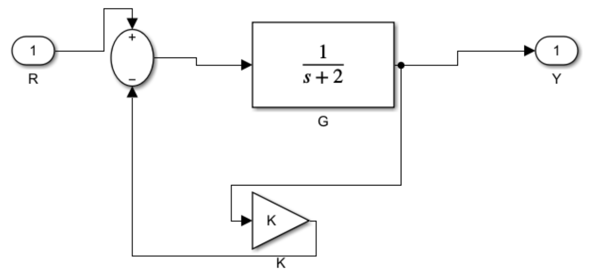


Fig. 5.: Simulink model generated from the block diagram image in Fig. 1

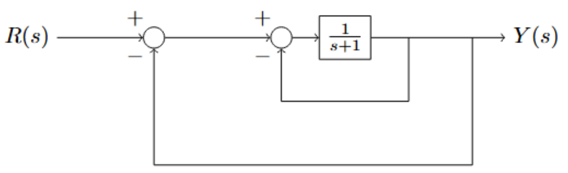


Fig. 6.: Example of block diagram image with double feedback loop

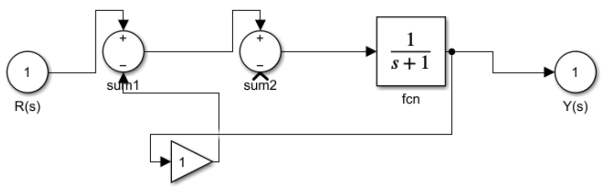


Fig. 7.: Simulink model generated from block diagram image in Fig.6

2.4 Data Set 1

In order to evaluate the performance of block diagram recognition, we created Data Set 1, comprising 18 block diagram images. The categorization of images within Data Set 1 is summarized in Table 1. The block diagrams may have one of three feedback loop configurations: No Loop (N.L.), Single Loop (S.L.), and Double Loop (D.L.), each with either one disturbance or no disturbance yielding six distinct diagram types as detailed in the Appendix. For each configuration, the dataset includes three images.

Table 1.: Components of Data Set 1

	No Loop	Single Loop	Double Loop
No Disturbance	3	3	3
With Disturbance	3	3	3

3. EVALUATION OF o4-mini

Table 2 presents the accuracy (ACC) results for the problems in Data Set 1, obtained by generating each block diagram image three times using the o4-mini model. Table 2 shows that the correct response rate for o4-mini was 41%. We observed that o4-mini frequently mistakes when interpreting diagrams containing feedback loops. Various types of errors were identified, including failure to detect the presence of feedback loops, failure to recognize blocks within the feedback loops, and misconnection of feedback loops to the summing junction for the disturbance input. Examples of block diagram provided to o4-mini are shown in Figs. 8 and 9, while Figs. 10 and 11 depict the resulting Simulink models generated by o4-mini.

Table 2.: Performance evaluation of o4-mini on block diagram recognition

	o4-mini
Topics	ACC
N.L./N.D.	89% (8/9)
N.L./W.D.	78% (7/9)
S.L./N.D.	11% (1/9)
S.L./W.D.	33% (3/9)
D.L./N.D.	22% (2/9)
D.L./W.D.	11% (1/9)
Total	41% (22/54)

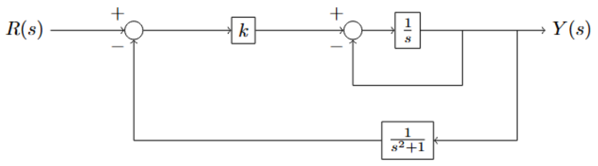


Fig. 8.: Block diagram 1 given to o4-mini

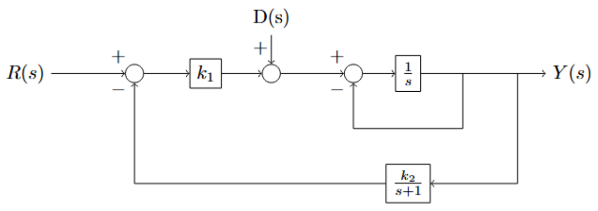


Fig. 9.: Block diagram 2 given to o4-mini

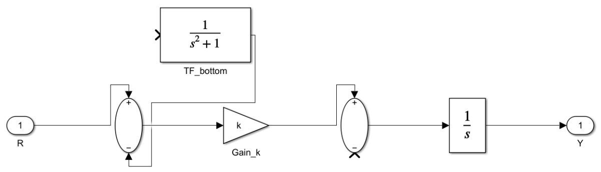


Fig. 10.: Simulink model output by o4-mini1

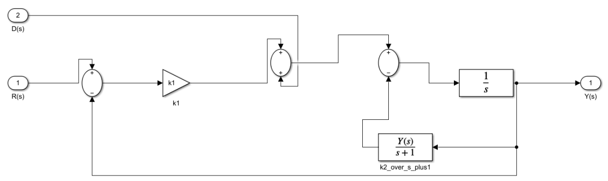


Fig. 11.: Simulink model 2 output by o4-mini

4. PROPOSED METHOD

In this work, we propose two two-stage detection methods [13] to enhance LLM-based block diagram recognition performance. We employ customized GPTs—specialized GPT tailored to specific tasks or domains—to instruct the LLMs. These GPTs leverage the TikZ package [12][14][15][16] to represent the recognized block diagrams. TikZ (TikZ ist kein Zeichenprogramm) is a LaTeX library for creating high-quality graphics.

4.1 Two-Stage Double LLM method

In this study, we introduce the Two-Stage by Double LLM (TSD) method for enabling ChatGPT to interpret block diagram images and subsequently generate Simulink models. As depicted in the flowchart in Fig. 12, the process begins with a customized GPT which generates TikZ-formatted LaTeX code from the block diagram image. Thereafter, this LaTeX code is submitted to ChatGPT (o4-mini), which produces MATLAB code for constructing the corresponding Simulink model.

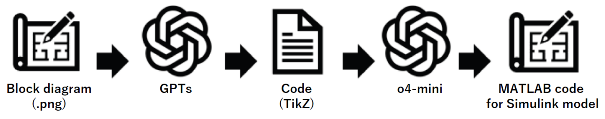


Fig. 12.: Flowchart of TSD

GPTs are provided with block diagram images and receive no direct user prompts. The instructions to the GPTsfor block diagram recognition are outlined in Fig. 13, and the instructions for generating MATLAB code are detailed in Fig. 14. The instructions were calibrated using Data Set 1 to obtain better of TikZ-formatted LaTeX code from the block diagram images. For details, please refer to "Prompt.txt" in https://github.com/ppp-00/ChatGPT.git.

In the second stage of the TSD method, the user supplies ChatGPT with LaTeX code, with the following prompt: "Please generate MATLAB code to construct a Simulink model from the block diagram defined in the TikZ-formatted LaTeX code that will be provided. The MATLAB version is R2022b."

This modular design enables the substitution of the second LLM with a higher-performing LLM.

4.2 Two-Stage Single LLM method

In this work, we propose the Two-Stage Single LLM (TSS) method to enable ChatGPT to generate Simulink models directly from block diagram images. As illustrated in Fig. 16, we adapt customized GPTs to produce MATLAB code that yields Simulink models via TikZ-formatted LaTeX code for block diagram images. We augment the instruction set of TSD by appending the instruction shown in Fig. 15.

- For a given block diagram, analyze the characteristics of the number, type, location, and connection of the blocks.

- Clarify the starting and ending points of the feedback loop when analyzing images.

- After outputting the code, the number of recognized feedback loops and the presence of disturbances are output.—(A1)

- Temporarily lists and outputs the position of each element in the image (coordinates, order, connection relations, summing and take off points). —(A2)

- When the feedback loop has no block, add a block of "1" . —(A3)

Fig. 13.: Instruction for block diagram recognition

- Take the time to analyze the given block diagram and be sure to convert it into a form that can be compiled directly into LaTeX code using the TikZ package.

- If there is a statement in the given image, it should also be described in the code.

Fig. 14.: Instruction for code output

- Output MATLAB code for the generated LaTeX code to create a Simulink model.

Fig. 15.: Additional instructions for TSS

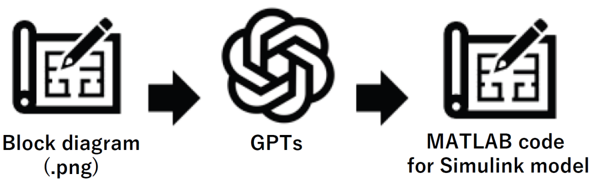


Fig. 16.: Flowchart of TSS

This approach offers the advantage, owing to its reduced requirement for user intervention compared to TSD.

Table 3 shows the auuracy(ACC) results for the problems in Data Set 1, obtained by generating each Simulink model three times using the proposed methods. For reference, the accuracy of o4-mini is also included for comparison.

Table 3.: Evaluation of the proposed method with Data Set 1

	TSD+o4-mini	TSS	o4-mini
Topics	ACC	ACC	ACC
N.L./N.D.	100% (9/9)	100% (9/9)	89% (8/9)
N.L./W.D.	100% (9/9)	100% (9/9)	78% (7/9)
S.L./N.D.	100% (9/9)	100% (9/9)	11% (1/9)
S.L./W.D.	100% (9/9)	100% (9/9)	33% (3/9)
D.L./N.D.	46% (5/9)	100% (9/9)	22% (2/9)
D.L./W.D.	89% (8/9)	89% (8/9)	11% (1/9)
Total	91% (49/54)	98% (53/54)	41% (22/54)

Table 3 shows that the correct response rates for TSD and TSS were 91% and 98%, respectively. When generating Simulink models from block diagram images, the accuracy improved by translating to LaTeX code compared to using o4-mini alone. Moreover, using a single LLM to both generate and interpret LaTeX code further increased the accuracy rate.

To evaluate the effect of instructions (A2) and (A3) in Fig. 13, we conducted an evaluation on Data Set 1 using GPTs without the instructions the correct respons rates are is shown in Table 4. Comparing these results to those reported for TSD in Table 3, we find that the omission of instruction (A2) and (A3) yields a lower accracy. These findings confirm that the instruction influences recognition accuracy.

Table 4.: Accuracy by TSD without instructions (A2) and (A3)

	TSD without (A2) and (A3)
Topics	ACC
N.L./N.D.	100% (9/9)
N.L./W.D.	100% (9/9)
S.L./N.D.	89% (8/9)
S.L./W.D.	89% (8/9)
D.L./N.D.	56% (5/9)
D.L./W.D.	67% (6/9)
Total	81% (44/54)

5. EVALUATION

5.1 Evaluation with Data Set 2

Data Set2 was constructed to evaluate the performance of the proposed methodology. The parameters (numerator, denominator, and gain values) of all blocks in the block diagrams from Data Set 1 were modified to generate 18 new problem instances, while preserving the original interconnection. The evaluation results for Data Set 2 are shown in Table 5.

Table 5.: Performance evaluation using Data Set 2

	TSD+o4-mini	TSS	o4-mini
Topics	ACC	ACC	ACC
N.L./N.D.	100% (9/9)	100% (9/9)	100% (9/9)
N.L./W.D.	100% (9/9)	100% (9/9)	89% (8/9)
S.L./N.D.	89% (8/9)	100% (9/9)	22% (2/9)
S.L./W.D.	67% (6/9)	100% (9/9)	0% (0/9)
D.L./N.D.	78% (7/9)	78% (8/9)	0% (0/9)
D.L./W.D.	89% (8/9)	89% (8/9)	0% (0/9)
Total	87% (47/54)	96% (52/54)	35% (19/54)

Table 5 illustrates that the correct response rates for TSD, TSS, and o4-mini were 87%, 96%, and 35%, respectively. Consistent with the result from Data Set 1, the proposed methodologies demonstrated superior recognition performance. We will construct a dataset that contains the block diagrams in which the position of blocks are modified from that of the block diagrams in Data Set 1 and conduct further evaluations in the future.

6. DISCUSSION

Table 3 compares the accuracy obtained by TSD and TSS with those achieved using o4-mini. Although o4-mini exhibits strong visual reasoning capabilities and can extract certain information directly from block diagrams, our results suggest that providing diagrammatic content in a text-based format, such as TikZ-formatted LaTeX code, enhances the recognition performance of the LLM.

We think that TSS achieved a high accuracy rate not only due to the intermediate LaTeX representation but also owing to its direct access to and analysis of the original block diagram images.

providing the instructions (A1) and (A2) appears to force the LLM to follow the instructions and improved recognition accuracy. Moreover, the inclusion of a code template within the prompt play the role to encourage the LLM to emulate the template’ s structure.

7. CONCLUSION

In this study, we proposed two two-stage detection methods to enhance block diagram recognition by large language models. The first method, Two-Stage by Double LLM (TSD), uses one LLM to generate TikZ-formatted LaTeX code from block diagram images and another LLM to interpret this code into MATLAB scripts that construct Simulink models. The second method, Two-Stage by Single LLM (TSS), similarly produces LaTeX code from images but employs the same LLM for both code generation and interpretation. To evaluate the effectiveness of these methods, we created a dataset of block diagram images for recognition tasks. Experimental results demonstrate that both TSD and TSS significantly outperform the standalone o4-mini model, yielding substantially higher recognition accuracies.Future research is needed to verify whether this method is also effective for block diagram images with distortion or warping.

Although future models may exhibit enhanced visual recognition capabilities, highly specialized domains such as control engineering—characterized by limited training data—will likely continue to present challenges for figure recognition. In these cases, our proposed two-stage detection methods offer a means to mitigate the shortcomings of existing LLMs. By integrating visual inference with structured text representations, the proposed approach can enhance response accuracy. We anticipate that this methodology will facilitate the adoption of generative AI within control engineering applications.

REFERENCES

[1] Alfa Faridh Suni, Aryo Baskoro Utomo, Khoiruddin Fathoni, Budiandra Yusuf Mahendra, Izzati Gemi Seinsiani, Ahmad Fashiha Hastawan, Text-to-Speech User Interface for ChatGPT, 2024 4th International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS), 2024

[2] OpenAI, ChatGPT, https://chatgpt.com/, 2025.

[3] Darioush Kevian, U. Syed, Xing-ming Guo, Aaron J. Havens, G. Dullerud, Peter J. Seiler, Lianhui Qin, Bin Hu, Capabilities of Large Language Models in Control Engineering: A Benchmark Study on GPT-4, Claude 3 Opus, and Gemini 1.0 Ultra, arXiv:2404.03647, 2024.

[4] Sai H. Vemprala, Rogerio Bonatti, Arthur Bucker, Ashish Kapoor, ChatGPT for Robotics: Design Principles and Model Abilities, IEEE Access, Vol. 12, pp. 55682-55696, 2024.

[5] Jiaxing Huang, Jingyi Zhang, A Survey on Evaluation of Multimodal Large Language Models, arXiv:2408.15769, 2024

[6] Bingyu Dong, Jie Bai, Tao Xu, Yun Zhou, Large Language Models in Education: A Systematic Review, 2024 6th International Conference on Computer Science and Technologies in Education (CSTE), 2024

[7] Marcel Binz, Eric Schulz, Turning large language models into cognitive models, arXiv:2306.03917, 2023

[8] Arun Iyengar, Ashish Kundu, Large Language Models and Computer Security, 2023 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), 2023

[9] Usman Syed, Ethan Light, Xingang Guo, Huan Zhang, Lianhui Qin, Yanfeng Ouyang, Bin Hu, Benchmarking the Capabilities of Large Language Models in Transportation System Engineering: Accuracy, Consistency, and Reasoning Behaviors, arXiv:2408.08302, 2024

[10] Itsuki Ogo, Masanobu Koga, Can ChatGPT Pass Classical Control Theory Exam?, In 2024 SICE Festival with Annual Conference, Kochi, Japan, 2024.

[11] Itsuki Ogo, Masanobu Koga, Can ChatGPT Pass Modern Control Theory Exam?, The 24th International Conference on Control, Automation and Systems, Jeju, Korea, 2024.

[12] Jin Bo, Apply LaTeX to Make Courseware of Engineering Mechanics, Computer Programming Skills & Maintenance, 2024

[13] Isabel Cachola, Silviu Cucerzan, Allen Herring, Vuksan Mijovic, Erik Oveson, S. Jauhar, Knowledge-Centric Templatic Views of Documents, arXiv:2401.06945, 2024

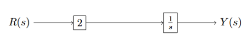
[14] Hackl J., TikZ-network manual, arXiv:1709.06005, 2017

[15] Dohse M., TikZ-FeynHand: Basic User Guide, arXiv:1802.00689, 2018

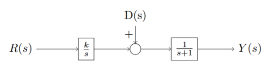
[16] Jonas Belouadi, Anne Lauscher and Steffen Eger, AutomaTikZ: Text-Guided Synthesis of Scientific Vector Graphics with TikZ, ArXiv:2310.00367, 2023

APPENDIX

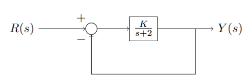
The six block diagrams included in Data Set 1 are shown below.



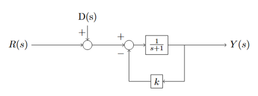
N.L./N.D.



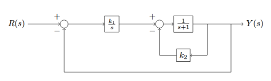
N.L./W.D.



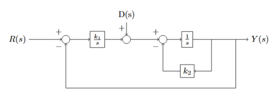
S.L./N.D.



S.L./W.D.



D.L./N.D.



D.L./W.D.