

# Unveiling the Power of Integration: Block Diagram Summarization through Local-Global Fusion

Shreyanshu Bhushan<sup>1</sup>, Eun-Soo Jung<sup>1</sup> and Minho Lee<sup>1,2</sup>

<sup>1</sup>ALI Co., Ltd., South Korea

<sup>2</sup>Department of Artificial Intelligence, Kyungpook National University, South Korea  
{shreyanshubhushan, euns0000.jung, mholee}@gmail.com

## Abstract

Block Diagrams play an essential role in visualizing the relationships between components or systems. Generating summaries of block diagrams is important for document understanding or question answering (QA) tasks by providing concise overviews of complex systems. However, it's a challenging task as it requires compressing complex relationships into informative descriptions. In this paper, we present "BlockNet", a fusion framework that summarizes block diagrams by integrating local and global information, catering to both English and Korean languages. Additionally, we introduce a new multilingual method to produce block diagram data, resulting in a high-quality dataset called "BD-EnKo". In BlockNet, we develop "BlockSplit", an Optical Character Recognition (OCR) based algorithm employing the divide-and-conquer principle for local information extraction. We train an OCR-free transformer architecture for global information extraction using BD-EnKo and public data. To assess the effectiveness of our model, we conduct thorough experiments on different datasets. The assessment shows that BlockNet surpasses all previous methods and models, including GPT-4V, for block diagram summarization.<sup>1</sup>

## 1 Introduction

Document Artificial Intelligence (Document AI) is gaining momentum across industries for streamlining document processes, enhancing efficiency, and extracting insights from unstructured data. Downstream tasks span various domains and paradigms, such as document QA (Tanaka et al., 2021), layout detection (Zhong et al., 2019), classification (Harley et al., 2015), and more. Documents contain not only text, images, charts, or tables but also the most important block diagram, which is often

overlooked but plays a crucial role in illustrating systems, processes, or relationships.

Block diagrams are visual representations of systems, processes, or relationships using blocks or boxes to represent components or elements, and lines or arrows to show the relationships between them. They are particularly useful in technical documentation, engineering reports, business models, scientific papers, and more. Block diagrams can vary in complexity from simple representations of basic systems to more complex systems with numerous interconnected components. Summaries from block diagrams offer numerous advantages, including reducing manual effort, enhancing productivity, integrating with AI systems for analysis, Document AI, QA, and knowledge representation. They improve accessibility for visually impaired individuals, facilitate efficient information retrieval, support collaboration among team members, aid decision-making, and streamline workflows. These summaries also convert complex information into concise descriptions, aiding comprehension and saving time compared to manual analysis.

Despite its numerous advantages and potential applications, the block diagram summarization problem remains relatively overlooked within the natural language processing (NLP) community. We encountered a scarcity of literature on block diagram summarization. Previous approaches primarily focused on detecting different shapes within the diagrams (Julca-Aguilar and Hirata, 2018) or converting handwritten block diagrams into electronic formats (Schäfer and Stuckenschmidt, 2019; Schäfer et al., 2021; Schäfer and Stuckenschmidt, 2021). However, approaches failed to address the vital task of linking text phrases with shapes and arrows, crucial for effective summarization. To the best of our knowledge, only a few methods have effectively accomplished diagram summarization by combining object detection with pre-trained language models (Bhushan and Lee, 2022). Neverthe-

<sup>1</sup>The dataset, source code, and model checkpoints are publicly available at <https://github.com/shreyanshu09/BlockNet>.

less, their results are subject to several limitations such as hallucination problem, OCR errors, lack of large dataset, and support limited diagrams. Even the latest powerful vision language model, GPT-4V (Yang et al., 2023), struggles with efficiency, particularly in the task of converting block diagrams to text, often lacking detailed information. Addressing block diagram summarization encounters two main challenges: the absence of a large dataset and the lack of robust baselines.

In this paper, we propose a fusion-based framework “BlockNet”, comprising three main components: a local information extractor, a global information extractor, and an integrator. The local extractor, based on our algorithm “BlockSplit”, follows the divide-and-conquer principle to split block diagrams into subunits, each containing two connected nodes or symbols. These subunits are then processed using OCR to extract triplets (head, relation, tail). The global extractor utilizes an OCR-free visual document understanding (VDU) transformer model, trained on both English and Korean diagram datasets. Lastly, the integrator, a large language model (LLM), interprets visual information and combines outputs from both extractors. We also introduce a new multilingual data collection approach for block diagram understanding and generation, enabling flexibility and scalability. With this approach, we compile a large, high-quality dataset, “BD-EnKo”, containing 83,394 diagrams with their summaries in English and Korean, expandable to any language using our approach.

To summarize, our contributions are below:

- We propose BlockNet, a fusion framework that combines local and global information from block diagram to enhance summary generation.
- We introduce a novel multi-lingual data collection approach specifically designed for block diagram understanding and generation, offering flexibility and scalability.
- Using our innovative data collection methodology, we produce the diverse and high-quality benchmark dataset BD-EnKo in English and Korean languages.
- We conduct several automatic and human evaluations to check the performance of the proposed framework.

## 2 Related Work

**Visual Document Understanding** Several VDU methods have been developed to interpret semi-structured documents and extract vital information, such as receipts and tables (Davis et al., 2019; Huang et al., 2019; Hong et al., 2022). Early approaches aimed to utilize OCR-independent visual backbones (Kang et al., 2014; Harley et al., 2015; Afzal et al., 2015; Guo et al., 2019) but exhibited limited performance. Subsequent advancements in OCR (Baek et al., 2019a,b; Jung et al., 2021) led to the development of various OCR-dependent VDU models by combining with language models (Hwang et al., 2019; Bhushan and Lee, 2022). These methods typically address the task in two stages: 1) extracting text from document images using deep learning-based OCR; 2) comprehending the document as a whole. However, OCR-dependent techniques face challenges as OCR errors can negatively impact downstream processes (Hwang et al., 2019, 2021). To address this, Kim et al. (2022) introduced a Transformer-based OCR-free architecture (Devlin et al., 2019; Dosovitskiy et al., 2020; Kim et al., 2021) but still it faces problems particularly with non-English words. Considering these points, we integrate both OCR techniques (local information) and OCR-free models (global information) to address the respective limitations of each.

**Image to Data Generation** Earlier, Montalvo (1990) explored understanding symbolic descriptions in diagrams and suggested a method that links symbolic descriptions with visual attributes to enhance comprehension and analysis. Julca-Aguilar and Hirata (2018) introduced a handwritten block diagram detection system using Faster R-CNN. However, their method’s inability to recognize edges prevents it from identifying the symbols connected by the arrows, thus limiting its effectiveness in block diagram understanding tasks. Arrow R-CNN addresses this by employing an arrow key-point predictor to learn and predict arrow positions (Schäfer and Stuckenschmidt, 2019). Furthermore, Schäfer and Stuckenschmidt (2021) improves arrow detection by modeling arrows as relationships between shapes rather than as independent objects with bounding boxes. However, these systems only detect symbol categories without capturing key information conveyed by the block diagram. Bhushan and Lee (2022) introduced the BloSum

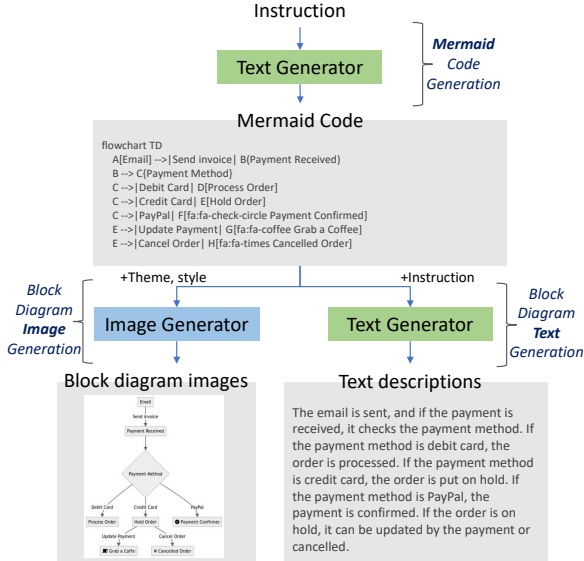


Figure 1: Pipeline of our data generation method.

framework, which employs a language model to generate textual descriptions for identified block diagram images, thereby improving tasks such as document summarization. However, language models face challenges such as the illusion problem, where the generated content may lack meaning or reliability compared to the original content. Moreover, the dataset that they used for training the model was very small. In contrast, our approach is to integrate the local information (summary of each sub-block) from the diagram as well as the global information (overall summary). Moreover, our model is trained on a significantly larger dataset (around 76K) compared to the BloSum framework (around 500).

### 3 Method

In this section, we divide into four subsections. The first explains our unique data generation method. The second discusses the large-scale dataset “BD-EnKo”. The third analyzes the dataset, and fourth presents “BlockNet”, our fusion-based framework.

#### 3.1 Data Generation

In this subsection, we detail our unique approach of data generation. It mainly comprises three parts as shown in Figure 1. These parts are detailed in the following paragraphs.

**Mermaid Code Generation** The first part is to collect different varieties of mermaid code. Mermaid code is a markup language for creating diagrams, charts, and flowcharts. It is concise and intuitive, allowing for the easy creation of complex

Types	Train		Validation	
	English	Korean	English	Korean
Flowchart	7,875	7,678	875	854
GraphLR	5,462	2,861	607	318
GraphTD	3,500	2,402	389	267
Journey	8,494	8,858	942	985
Sequence	2,708	3,539	301	394
Simple Graph	3,825	1,194	425	133
State Diagram	6,444	6,265	717	693
C4	3,549	0	395	0
Real World	76	304	15	50
Total	41,933	33,101	4,666	3,694

Table 1: Statistics of our BD-EnKo dataset comprising various types of block diagrams.

visual representations. It supports a wide range of diagrams, including flowcharts, sequence diagrams, Gantt charts, etc. We instruct GPT-3.5 (Ouyang et al., 2022) to generate mermaid codes encompassing various types and random themes of diagrams. This method guarantees the creation of diverse mermaid data with appropriate structure, essential for image generation. Additionally, by controlling these characteristics, we intentionally reduce bias, leading to a more balanced dataset.

**Block Diagram Image Generation** The next part is to transform the generated mermaid code into visual block diagrams using the popular diagram generation tool, Mermaid<sup>2</sup>. We provide further details on enhancing aspects such as color and theme schemes. To broaden the variety, we also applied random noise, blur effects, and grayscale effects to the images. This yielded a diverse collection of block diagram images, each accurately depicting connections and symbols while being visually appealing, thereby demonstrating the effectiveness of our approach.

**Block Diagram Text Generation** The final part involves generating text descriptions for each block diagram image. Leveraging the structure of mermaid code, which GPT-3.5 turbo understands well, makes it particularly straightforward to generate descriptions based on this code. To accomplish this, we utilize the previously generated mermaid code from mermaid code generator. We then instruct GPT-3.5 turbo once more to generate coherent and meaningful descriptions based on this mermaid code structure. The prompts utilized at each part and the pipeline explanation with examples are detailed in Appendix A.1.

<sup>2</sup><https://github.com/mermaid-js/mermaid>

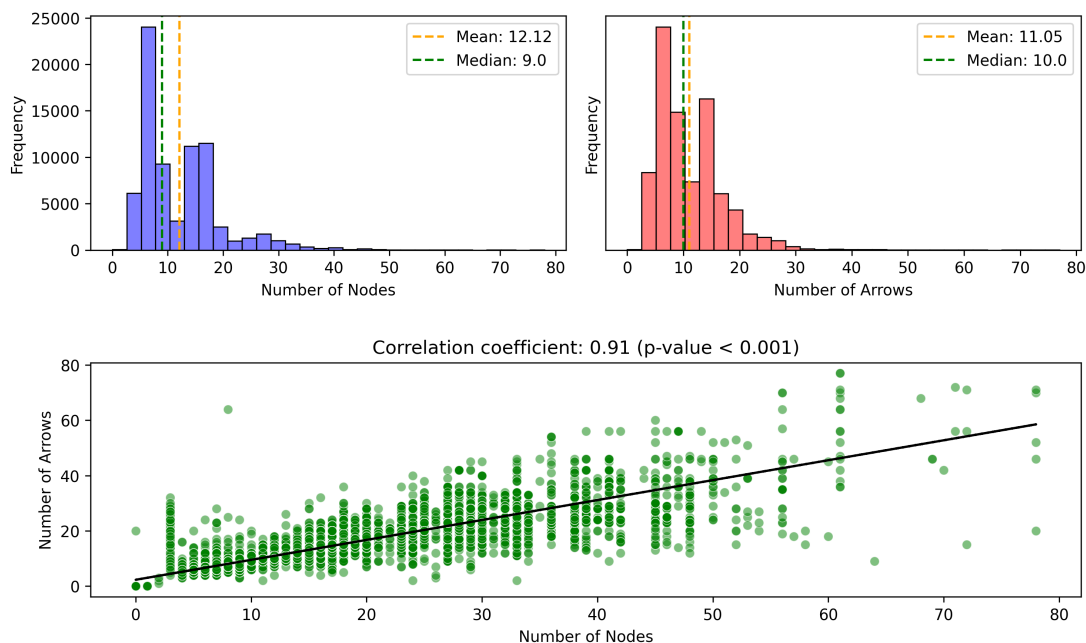


Figure 2: Visualization of BD-EnKo dataset statistics depicting the relationship between ‘Number of Nodes’ and ‘Number of Arrows’.

### 3.2 BD-EnKo Dataset

Our method for generating block diagram data resulted in a large-scale dataset comprising 83,394 samples. These samples consist of both English (46,599 samples) and Korean (36,795 samples) languages, featuring a diverse range of block diagram images with their high-quality summaries. As depicted in Table 1, our method generated eight distinct types of block diagrams, each possessing its unique style and significance. We supplemented our dataset by collecting a few real block diagram images from publicly available sources on the Google search engine, consisting of 91 English samples and 354 Korean samples. The decision to collect more samples for the Korean language stemmed from the availability of the public dataset “CBD” for English cases, as collected by [Bhushan and Lee \(2022\)](#) from various search engines. For the real-world samples, we engaged Korean native speakers to provide meaningful summaries for the given diagrams in Korean, while English native speakers performed the same task for the English samples. Dataset samples of various categories can be found in the Appendix A.2.

### 3.3 Dataset Analysis

Figure 2 presents the statistical insights from the training data of the BD-EnKo dataset. The top-left histogram, colored in blue, shows the distribution of the number of nodes, with a high frequency of

occurrences around 0 to 20 nodes, peaking at approximately 25,000 occurrences, and tapering off significantly beyond 50 nodes. The top-right histogram, colored in red, illustrates the distribution of the number of arrows, similarly showing a high frequency around 0 to 10 arrows, peaking at around 29,000 occurrences, and diminishing beyond 30 arrows. The bottom scatter plot, with green dots, depicts the relationship between the number of nodes and the number of arrows, showing a positive correlation. The linear regression line fitted to the scatter plot indicates a strong positive correlation with a correlation coefficient of 0.91 and a p-value less than 0.001, suggesting a statistically significant relationship between the number of nodes and the number of arrows in the dataset. Descriptive statistics of the dataset are in Appendix A.3.

### 3.4 BlockNet

This subsection details BlockNet’s architecture, shown in Figure 3, with three main blocks: local extractor, global extractor, and integrator. Local information alone misses overall structure, while global information alone overlooks details. BlockNet integrates both for better diagram summarization. Each block of BlockNet is elaborated in the following paragraphs.

**Local Information Extractor** We construct this module following the divide-and-conquer princi-



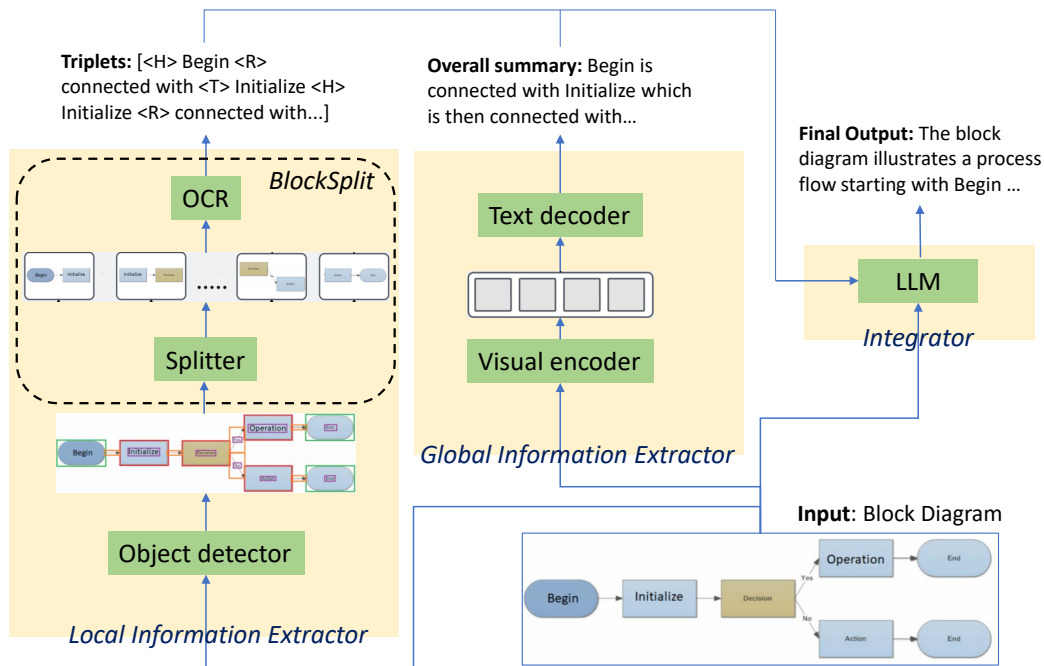


Figure 3: Schematic of BlockNet: Local connections are captured as triplets, while a transformer-based encoder-decoder generates a global summary. The integrator merges local and global data from the image.

ple, emphasizing the division of block diagrams into smaller pairs to improve efficiency and prevent oversight of detailed information. In this block, the very first step is to detect all symbols and arrows within the diagram, which we classify as an object detection task. Unlike previous work (Bhushan and Lee, 2022), we train the object detection model based on YOLOv5 (Jocher et al., 2020) that provides fundamental abilities crucial for detecting different objects in an image. We train this model with CBD, FC\_A, and FC\_B dataset which provides the annotations of different shapes and arrows in a diagram. We train the model for six labels: arrow, terminator, process, decision, data, and text. Additionally, we utilize the Pororo OCR<sup>3</sup> that supports both English and Korean to extract text from the images. Subsequently, we developed an algorithm “BlockSplit” to divide the block diagram image into sub-diagrams, each containing two nodes connected by an arrow, detailed in Algorithm 1.

This algorithm takes three inputs: shapes detected by the object detection module, texts recognized by the OCR, and the head and tail keypoints of arrows. To identify the head and tail keypoints of arrows, each arrow is divided into two parts, and the number of white pixels is counted. A higher count indicates the arrow’s head, while a lower count in-

dicates its tail. For each head and tail keypoint, the algorithm first identifies the closest shapes, then determines the associated shape’s name. Subsequently, it locates text within the shape; if text is present, it is labeled as either Head or Tail, otherwise, the shape’s name is used. Regarding relations, the algorithm calculates the distance between the arrow and all text bounding boxes written outside shapes. Texts within a threshold distance (set at 5) are labeled as Relations. In the absence of texts within this threshold, the relation is denoted as ‘connected with’. Each sub-diagram is represented as a triplet (<H> Head, <R> Relation, <T> Tail). The output of this block consists of a collection of all triplets found within the diagrams.

**Global Information Extractor** We train the transformer encoder and decoder model based on the configuration specified in Donut (Kim et al., 2022) to extract the global information of the block diagram. The straightforward architecture comprises a visual encoder module and a text decoder module based on the Transformer architecture (Vaswani et al., 2017). This model does not rely on any OCR-related modules but instead utilizes the visual encoder to extract features from the given image. We utilize Swin Transformer (Liu et al., 2021) as the visual encoder. Initially, the visual encoder divides the input image into non-overlapping patches, applying a window-based

<sup>3</sup>[https://github.com/yunwoong7/korean\\_ocr\\_using\\_pororo](https://github.com/yunwoong7/korean_ocr_using_pororo)

multi-head self-attention module and a two-layer multi-Layer Perceptron (MLP) to these patches. Subsequently, patch merging layers are applied to the patch tokens at each stage. The resulting output is then fed into the text decoder.

For the text decoder architecture, we employ BART (Lewis et al., 2020). The text decoder takes the outputs of the visual encoder and generates a sequence of tokens. Specifically, we initialize the decoder model weights using a publicly available pre-trained multilingual BART model (Parikh et al., 2020). We train the end-to-end model using our BD-EnKo dataset, that contains both English and Korean diagrams, as well as the CBD dataset collected by Bhushan and Lee (2022), FC\_A handwritten block diagram dataset collected by Awal et al. (2011) and FC\_B handwritten block diagram datasets collected by Bresler et al. (2016). These datasets comprise block diagram images and their corresponding text summaries, which aid in the end-to-end training process. The pre-trained Donut model is capable of reading the texts in the image according to the reading order, typically from top-left to bottom-right. However, applying the model directly to our task is suboptimal due to the complexity of block diagram layouts and relationships.

**Integrator** After obtaining both the local and global information from the diagram, the final step involves integrating them to mitigate the errors or limitations of each. To address these limitations, we utilize an LLM GPT-4V to seamlessly integrate both local and global information based on the provided block diagram image. This module receives inputs from two sources: the local information extractor, which produces triplets as outputs, and the global information extractor, which provides an overall summary. Additionally, it takes the input block diagram image. The output is a final summary that integrates information from all these sources. The prompt utilized for integration is outlined in the Appendix A.4.

## 4 Experiments

We first describe the experimental setup in Section 4.1, followed by a discussion of the metrics employed to measure performance in Section 4.2, and at last present the main results in Section 4.3.

### 4.1 Experimental Setup

**Datasets** We utilize three datasets to evaluate and compare the performances of various models. Our

---

### Algorithm 1 BlockSplit Algorithm

---

**Input:** Shape  $\mathbf{s} \in \mathcal{S}$ , Texts  $\mathbf{t} \in \mathcal{T}$ , Head & Tail Arrow Keypoints  $\mathbf{a} \in \mathcal{A}$  ( $a^{head}, a^{tail}$ )

**Output:**  $\langle H \rangle$  head,  $\langle R \rangle$  relation,  $\langle T \rangle$  tail

```

1: for  $a_i$  in  $\mathcal{A}$ :
2:    $\langle H \rangle_i = \operatorname{argmin}_s(\operatorname{dist}(a_i^h, \mathbf{s})), \mathbf{s} \in \mathcal{S}$ 
3:    $\langle T \rangle_i = \operatorname{argmin}_s(\operatorname{dist}(a_i^t, \mathbf{s})), \mathbf{s} \in \mathcal{S}$ 
4:   if any  $\mathbf{t} \in \mathcal{T}$  in  $\langle H \rangle_i$  then
5:      $\langle H \rangle_i = \mathbf{t}, \mathbf{t} \in \mathcal{T}$ 
6:   end if
7:   if any  $\mathbf{t} \in \mathcal{T}$  in  $\langle T \rangle_i$  then
8:      $\langle T \rangle_i = \mathbf{t}, \mathbf{t} \in \mathcal{T}$ 
9:   end if
10:  if  $\min(\operatorname{dist}(a_i, \mathbf{t})), \mathbf{t} \in \mathcal{T} < 5$  then
11:     $\langle R \rangle = \operatorname{argmin}_t(\operatorname{dist}(a_i, \mathbf{t})), \mathbf{t} \in \mathcal{T}$ 
12:  else
13:     $\langle R \rangle = \text{'connected with'}$ 
14:  end if

```

---

dataset, BD-EnKo, is partitioned into two subsets: BD-En (4,666 samples) for English diagrams and BD-Ko (3,694 samples) for Korean diagrams. This division is necessary as some models do not support the Korean language, ensuring a fair comparison across different languages. Additionally, we employ CBD (106 samples) (Bhushan and Lee, 2022), an English real-world dataset, and FC\_B (196 samples) (Bresler et al., 2016), consisting of English handwritten diagrams.

**BlockNet** For symbol detection within the local information extractor task, we employ a pre-trained YOLOv5 model (Jocher et al., 2020) as the backbone network. Although the diagram images differ greatly from real-world images in the COCO dataset (Chen et al., 2015), we fine-tune the model using the CBD dataset due to its provision of annotations for symbol detection tasks. Within the global information extractor, we train the transformer encoder and decoder model using a combined dataset including BD-EnKo (75,034 samples), CBD (396 samples), FC\_A (357 samples), and FC\_B (476 samples).

Additional details regarding the experimental setup for BlockNet and various other models used for comparison are provided in the Appendix A.5.

### 4.2 Evaluation Metrics

Text quality can be assessed through both human evaluation and automatic evaluation metrics. Automatic evaluation metrics fall into two categories:

Models	BD-En (English)				BD-Ko (Korean)			
	BLEU ↑	ROUGE-1 ↑	BLEURT ↑	PPL ↓	BLEU ↑	ROUGE-1 ↑	BLEURT ↑	PPL ↓
mPlug	6.2	20.7	-0.34	<b>5.25</b>	-	-	-	-
Faster R-CNN + BART_Large	13.01	31.26	-0.11	13.25	-	-	-	-
Faster R-CNN + T5_Large	17.95	38.21	-0.04	11.37	-	-	-	-
BloSum + BART_Large	22.29	39.4	-0.17	11.15	-	-	-	-
BloSum + T5_Large	24.62	45.36	0.11	10.13	-	-	-	-
Donut	55.77	69.91	0.17	21.26	33.76	52.22	-0.12	13.97
GPT-4V	50.65	67.04	0.29	7.79	20.72	33.61	-0.46	<b>4.3</b>
<b>BlockNet (ours)</b>	<b>58.61</b>	<b>74.25</b>	<b>0.38</b>	7.22	<b>62.41</b>	<b>78.61</b>	<b>0.27</b>	4.34

Table 2: Automatic evaluation results on our dataset (BD-EnKo) from different models. Up arrow ↑ shows, higher is better. Down arrow ↓ shows, lower is better. Bold numbers indicate the best score.

Models	CBD				FC_B			
	BLEU ↑	ROUGE-1 ↑	BLEURT ↑	PPL ↓	BLEU ↑	ROUGE-1 ↑	BLEURT ↑	PPL ↓
mPLUG	3.74	15.18	-0.73	<b>5.46</b>	3.51	14.81	-0.48	<b>4.68</b>
Faster R-CNN + BART_Large	17.29	31.16	-0.69	17.93	22.19	41.63	-0.1	13.58
Faster R-CNN + T5_Large	22.11	40.1	0.1	12.06	27.81	50.47	0.08	10.91
BloSum + BART_Large	33.47	75.24	0.11	8.33	15.49	35.39	-0.13	14.1
BloSum + T5_Large	42.18	80.78	0.2	7.55	20.04	40.85	0.06	13.46
Donut	54.73	75.5	-0.24	38.35	69.29	73.37	-0.18	26.89
GPT-4V	69.5	85.33	<b>0.48</b>	5.95	62.17	89.06	0.52	4.79
<b>BlockNet (ours)</b>	<b>72.31</b>	<b>88.9</b>	0.41	6.87	<b>67.74</b>	<b>90.63</b>	<b>0.59</b>	5.13

Table 3: Automatic evaluation results on CBD and FC\_B dataset from different models. Up arrow ↑ shows, higher is better. Down arrow ↓ shows, lower is better. Bold numbers indicate the best score.

those not requiring training and those learned via neural networks. Metrics that do not require training, such as BLEU (Post, 2018) and ROUGE (Lin, 2004), are commonly used due to their convenience. The overall BLEU score is calculated by averaging BLEU n-grams (n=1 to 4) while considering the brevity penalty. Additionally, we use the F1 score of ROUGE-1. However, experiments show that these metrics, based only on n-gram overlap and lacking semantic understanding, often inadequately reflect text quality compared to human evaluation. Hence, for a more accurate alignment with human evaluation, we employ learned metrics like BLEURT (Sellam et al., 2020) (specifically BLEURT-large-512). Moreover, we utilize Perplexity (PPL) (Radford et al., 2019) using a pre-trained GPT-2 Large to evaluate the fluency and readability of the generated text descriptions derived from block diagram images.

### 4.3 Experimental Results

**Automatic Evaluation** Table 2 presents the results of automatic evaluation conducted on our dataset separately for English and Korean, providing clearer insights. In the English dataset (BD-En), our proposed framework, BlockNet, achieves the highest scores for BLEU (58.61), ROUGE-1

Criteria	BlockNet		GPT-4V	
	English	Korean	English	Korean
Accuracy	85	80	75	30
Completeness	82	78	67	28
Readability	88	83	80	36
Semantic	80	75	71	22

Table 4: Human evaluation scores for English and Korean diagrams from BlockNet and GPT-4V.

(74.25), and BLEURT (0.38), indicating its effectiveness. However, mPLUG (Ye et al., 2023) obtains the best PPL score (5.25), indicating its ability to generate fluent sentences. Nonetheless, its BLEU (6.2), ROUGE-1 (20.7), and BLEURT (-0.34) scores are significantly lower compared to BlockNet, suggesting that while mPLUG generates fluent sentences, it may not accurately capture facts. Similar trends are observed in the Korean dataset (BD-Ko), where BlockNet achieves the highest scores for BLEU (62.41), ROUGE-1 (78.61), and BLEURT (0.27). Notably, there is a substantial performance gap between BlockNet and GPT-4V in the Korean dataset, underscoring the superiority of our model over GPT-4V. Figure 4 illustrates the comparison of outputs between BlockNet and GPT-4V. It’s evident that BlockNet better captures the structure and flow of the diagram compared to

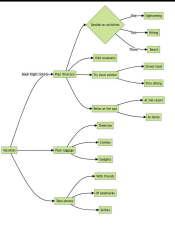
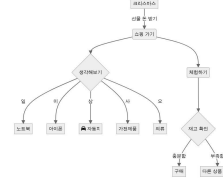
Image	BlockNet	GPT-4V
	The block diagram outlines a comprehensive approach to planning a vacation. It begins with the initial step of booking flight tickets, which leads into planning the itinerary. The itinerary planning includes deciding on activities, which are categorized into three options: sightseeing, hiking, and going to the beach. Additionally, the itinerary involves visiting museums and trying local cuisine, with the choice between street food and fine dining. Another aspect of the itinerary is to relax at the spa, with options to do so either at the resort or at home. Before embarking on the vacation, one must pack luggage, ensuring to include toiletries, clothes, and gadgets. The final step in the vacation planning process is to take photos, capturing memories with friends, of landmarks, and taking selfies.	The block diagram outlines the planning and activities associated with a vacation. It begins with two main tasks: "Decide on activities" and "Pack luggage." Under "Decide on activities," there are three options: "Sightseeing," "Hiking," and "Beach." From "Sightseeing," the activities branch out to "Visit museums," "Try local cuisine," and "Relax at the spa," with "Try local cuisine" further dividing into "Street food" and "Fine dining," and "Relax at the spa" offering the choice of doing so "At the resort" or "At home." The "Plan itinerary" step is connected to "Book flight tickets" and feeds into the decision-making process for activities. The "Pack luggage" task leads to three categories of items: "Toiletries," "Clothes," and "Gadgets." Additionally, "Take photos" is another activity with three subcategories: taking photos "With friends," "Of landmarks," and "Selfies." Each of these elements is interconnected, illustrating the comprehensive process of organizing and enjoying a vacation.
	이 이미지는 크리스마스 시즌에 선물로 받은 돈으로 쇼핑을 가는 과정을 나타내는 블록 다이어그램입니다. 크리스마스에 선물 돈을 받은 후, 쇼핑을 가기로 결정합니다. 쇼핑을 가기 전에 생각해보기 단계를 거치게 되는데, 이 단계에서는 일상생활에서 필요한 다양한 물품들에 대해 고려합니다. 구체적으로는 노트북, 아이폰, 자동차, 가전제품, 의류 등을 사용하는 상황을 생각해봅니다. 쇼핑을 하러 간 후에는 체험하기 단계로 넘어가게 되며, 이 단계에서는 매장에 가서 원하는 상품의 재고를 확인합니다. 재고 확인 결과에 따라 두 가지 경로로 나뉩니다. 만약 재고가 충분하다면 구매를 진행하고, 재고가 부족하다면 다른 상품을 선택하는 과정을 거치게 됩니다. 이러한 과정을 통해 쇼핑 경험이 완성됩니다.	이 블록 다이어그램은 '생각해보기'라는 중심 개념에서 출발하여 다양한 요소들과의 관계를 나타냅니다. '생각해보기'는 네 가지 하위 요소로 나뉘며, 이는 '논술', '아이디어', '직무지', '가치제품'입니다. '논술'은 '글쓰기'로 연결되며, '글쓰기'는 다시 '책 읽기'와 연결됩니다. '아이디어'는 '크리에이션'으로 연결되고, '크리에이션'은 '새로운 발견'과 '제품화하기'로 이어집니다. '직무지'는 '재료 원인'으로 연결되며, '재료 원인'은 '구매'와 '부족함'으로 나뉘어집니다. '가치제품'은 '위험'으로 연결되고, '위험'은 '재료 원인'으로 다시 연결됩니다. '제품화하기'와 '재료 원인'은 각각 '품질'과 '부족함'으로 연결되며, 이 두 요소는 '구매'와 '다른 상품 선택'으로 이어집니다. 이 다이어그램은 복잡한 의사결정 과정과 아이디어의 발전, 제품화 과정을 시각적으로 표현하고 있습니다.

Figure 4: Sample outputs from BlockNet and GPT-4V for the provided diagrams. The sentences highlighted in red represent incorrect predictions. The first row is for the English, and the second row is for the Korean diagram.

No.	Local Information	Global Information	Integrator (GPT-4V)	Integrator (LLaVa)	BLEU
1	✓	X	✓	X	54.73
2	✓	X	X	✓	31.61
3	X	✓	✓	X	45.19
4	X	✓	X	✓	22.07
5	✓	✓	X	✓	36.45
6	✓	✓	✓	X	<b>60.68</b>

Table 5: Average BLEU scores on the BD-EnKo dataset with different modules. The highest score is highlighted in bold. “✓” indicates that particular module is used in the framework, while “X” indicates that the module is not used.

GPT-4V. While GPT-4V performs poorly on Korean diagrams and its performance on English is also lacking, BlockNet overcomes these limitations and captures finer details more effectively.

Table 3 displays the results of automatic evaluation conducted on computerized (CBD) and handwritten (FC\_B) datasets, both comprising English diagrams. In the CBD dataset, BlockNet achieves higher BLEU (58.61) and ROUGE-1 (74.25) scores, while GPT-4V attains a better BLEURT (0.48) score, indicating its proficiency in generating grammatically correct sentences. However, the difference in BLEURT scores between BlockNet (0.41) and GPT-4V is minimal. Similar to the results for BD-EnKo, the Perplexity (PPL) scores for both CBD (5.46) and FC\_B (4.68) datasets are higher for mPLUG, as PPL assesses sentence fluency rather than factual accuracy. In the FC\_B dataset, BlockNet again outperforms all models, achieving the highest BLEU (67.74), ROUGE-1 (90.63), and BLEURT (0.59) scores. Additional experimental results and sample outputs are detailed under Appendix A.6.

**Human Evaluation** There are various ways to describe block diagrams, and no single automatic evaluation metric is perfect. To assess the quality of text descriptions generated from these diagrams, we enlisted the input of 30 individuals, comprising 15 native English speakers and 15 native Korean speakers. They were tasked with rating the descriptions based on four criteria: (1) Accuracy, which evaluates if the converted text faithfully represents the original diagram’s content; (2) Completeness, assessing if all aspects of the diagrams were adequately translated; (3) Readability, gauging the ease of understanding and clarity of the converted text; and (4) Semantic correctness, determining if the converted text effectively conveys the intended meaning of the original diagram. Each criterion was rated on a scale of 0 to 100, where 0 signified “strongly disagree” and 100 indicated “strongly agree”. We randomly selected 60 English block diagrams from the BD-En dataset and 60 Korean diagrams from the BD-Ko dataset. Each examiner received corresponding textual descriptions generated by the proposed BlockNet model and GPT-4V.



The average scores provided by the reviewers for the generated text descriptions are illustrated in Table 4. Based on the table results, it is evident that BlockNet outperforms GPT-4V across all four evaluation criteria for English and Korean diagrams.

## 5 Ablation Study

As shown in Table 5, we further investigate the effectiveness of our method across various modules, including the substitution of the GPT-4V integrator module with LLaVA-1.6 “liuhaotian/llava-v1.6-34b” (Liu et al., 2023). Notably, integrating both local and global information with GPT-4V yields superior performance (60.68), highlighting the significance of each module: local and global information extractors. Remarkably, when one module, either local or global information, is omitted, there is a significant decrease in accuracy, underscoring the indispensability of each component. However, it appears that the performance degradation is more pronounced without Local Information. Therefore, we can infer that GPT-4V and LLaVa models tend to exhibit weaker performance in capturing details. These findings underscore the holistic approach required for optimal performance in our framework, emphasizing the interplay between local and global information for enhanced results.

Furthermore, we investigated the feasibility of automating the generation of Mermaid code directly from block diagrams extracted from images, with subsequent summarization using GPT-3.5. Initially, we trained a transformer encoder-decoder model for this task but encountered considerable challenges. The strict syntax requirements of Mermaid code, particularly regarding arrow and text arrangement, posed significant hurdles, as even minor deviations could drastically alter the code’s meaning. Despite our efforts, the model frequently produced inaccurate outputs due to difficulties in capturing diagram structure accurately. Additionally, generating coherent summaries using GPT-3.5 from inaccurately formatted Mermaid code proved problematic, leading to errors.

## 6 Conclusion

In conclusion, this paper introduces the “BlockNet” framework, which effectively integrates local and global information from block diagrams to enhance summary generation. Our innovative approach to multi-lingual data collection, designed for block diagram understanding and generation, not only

provides flexibility and scalability but also facilitates the creation of the diverse and high-quality benchmark dataset “BD-EnKo” in English and Korean languages. Through rigorous automatic and human evaluations, our framework showcases state-of-the-art performance on both English and Korean diagrams. These findings underscore the significance of emphasizing the challenges that even advanced models like GPT4V face in comprehending block diagrams. Moreover, integrating GPT4V as a component in BlockNet underscores the crucial role of “local and global” fusion in achieving the superior performances observed. Overall, these contributions drive advancements in various fields, including block diagram understanding, Document AI, and QA, laying a solid foundation for future research across diverse domains.

## Limitations

After conducting several automatic and human evaluations, two primary limitations of this work have been identified. Firstly, despite proposing a multilingual data generation approach, we only generated data in English and Korean. Secondly, not many LLM options were explored for the Integrator module within our framework. Future efforts will concentrate on creating a smaller and faster LLM to enhance efficiency and accessibility.

Additionally, a limitation was discovered with the BD-EnKo dataset: its diagrams lack additional graphics akin to real-world images, although their structure remains consistent. In the future, our attention will be directed towards curating a more diverse dataset, inclusive of richer graphical content and encompassing multiple languages. This initiative aims to advance the development of a genuinely multilingual model.

## Acknowledgment

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A5A7026673)(50%) and by the NRF grant funded by the Korea government (MSIT) (No. NRF-2021R1A2C3011169)(50%).

## References

Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M Breuel, Andreas Dengel, and Marcus

- Liwicki. 2015. Deepdocclassifier: Document classification with deep convolutional neural network. In *2015 13th international conference on document analysis and recognition (ICDAR)*, pages 1111–1115. IEEE.
- Ahmad-Montaser Awal, Guihuan Feng, Harold Mouchere, and Christian Viard-Gaudin. 2011. First experiments on a new online handwritten flowchart database. In *Document Recognition and Retrieval XVIII*, volume 7874, pages 81–90. SPIE.
- Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoon Yun, Seong Joon Oh, and Hwalsuk Lee. 2019a. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4715–4723.
- Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoon Yun, and Hwalsuk Lee. 2019b. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9365–9374.
- Shreyanshu Bhushan and Minhoo Lee. 2022. Block diagram-to-text: Understanding block diagram images by generating natural language descriptors. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2022*, pages 153–168.
- Martin Bresler, Daniel Prusa, and Vaclav Hlavac. 2016. Online recognition of sketched arrow-connected diagrams. *International Journal on Document Analysis and Recognition (IJDA)*, 19(3):253–267.
- Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. **Microsoft coco captions: Data collection and evaluation server**. *ArXiv preprint*, abs/1504.00325.
- Brian Davis, Bryan Morse, Scott Cohen, Brian Price, and Chris Tensmeyer. 2019. Deep visual template-free form parsing. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 134–141. IEEE.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- He Guo, Xiameng Qin, Jiaming Liu, Junyu Han, Jingtuo Liu, and Errui Ding. 2019. Eaten: Entity-aware attention for single shot visual text extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 254–259. IEEE.
- Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. 2015. Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE.
- Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2022. Bros: A pre-trained language model focusing on text and layout for better key information extraction from documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10767–10775.
- Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. 2019. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE.
- Wonseok Hwang, Seonghyeon Kim, Minjoon Seo, Jinyeong Yim, Seunghyun Park, Sungrae Park, Junyeop Lee, Bado Lee, and Hwalsuk Lee. 2019. Post-ocr parsing: building simple and robust parser via bio tagging. In *Workshop on Document Intelligence at NeurIPS 2019*.
- Wonseok Hwang, Hyunji Lee, Jinyeong Yim, Geewook Kim, and Minjoon Seo. 2021. Cost-effective end-to-end information extraction for semi-structured document images. *arXiv preprint arXiv:2104.08041*.
- Glenn Jocher, Alex Stoken, Jirka Borovec, Liu Changyu, Adam Hogan, et al. 2020. ultralytics/yolov5: v3. 1-bug fixes and performance improvements. *Zenodo*.
- Frank D Julca-Aguilar and Nina ST Hirata. 2018. Symbol detection in online handwritten graphics using faster r-cnn. In *2018 13th IAPR international workshop on document analysis systems (DAS)*, pages 151–156. IEEE.
- Eun-Soo Jung, HyeongGwan Son, Kyusam Oh, Yongkeun Yun, Soonhwan Kwon, and Min Soo Kim. 2021. Duet: detection utilizing enhancement for text in scanned or captured documents. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5466–5473. IEEE.
- Le Kang, Jayant Kumar, Peng Ye, Yi Li, and David Doermann. 2014. Convolutional neural networks for document image classification. In *2014 22nd international conference on pattern recognition*, pages 3168–3172. IEEE.
- Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoon Yun, Dongyoon Han, and

- Seunghyun Park. 2022. Ocr-free document understanding transformer. In *European Conference on Computer Vision*, pages 498–517. Springer.
- Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. [BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*.
- Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.
- Fanya S Montalvo. 1990. Diagram understanding. In *Visual languages and applications*, pages 5–27. Springer.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. [ToTTo: A controlled table-to-text generation dataset](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.
- Matt Post. 2018. [A call for clarity in reporting BLEU scores](#). In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Bernhard Schäfer, Margret Keuper, and Heiner Stuckenschmidt. 2021. Arrow r-cnn for handwritten diagram recognition. *International Journal on Document Analysis and Recognition (IJ DAR)*, 24(1):3–17.
- Bernhard Schäfer and Heiner Stuckenschmidt. 2019. Arrow r-cnn for flowchart recognition. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 1, pages 7–13. IEEE.
- Bernhard Schäfer and Heiner Stuckenschmidt. 2021. Diagramnet: hand-drawn diagram recognition using visual arrow-relation detection. In *International Conference on Document Analysis and Recognition*, pages 614–630. Springer.
- Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. [BLEURT: Learning robust metrics for text generation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.
- Ryota Tanaka, Kyosuke Nishida, and Sen Yoshida. 2021. Visualmrc: Machine reading comprehension on document images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13878–13888.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.
- Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of llms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1):1.
- Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.
- Xu Zhong, Jianbin Tang, and Antonio Jimeno Yebes. 2019. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE.

## A Appendices

### A.1 Dataset Generator

Pipeline of our data generation approach involves three steps: first, generating mermaid codes using GPT-3.5 turbo; next, utilizing the generated code to draw diagrams using the Mermaid tool; and finally, employing GPT-3.5 turbo once more to generate text based on the provided mermaid code.

Figure 5 showcases the prompt utilized for generating mermaid code, which constitutes the first step in our data generation pipeline. Within this prompt, there are three variables highlighted in red. The first variable pertains to the block diagram type, which is initially set as “Graph TD”. There exist a total of seven types of block diagrams, and this variable can be adjusted to any one of them as needed. The second variable is the language, which defaults to “English” but can be modified to any desired language. We utilized both English and Korean languages for generating the dataset. The third variable serves as an example of the given block diagram type, enabling GPT-3.5 turbo to comprehend the code’s structure and produce output in that specific format.

Figure 6 illustrates the transformation process conducted by the Mermaid tool, which seamlessly converts the provided mermaid code into an image representation. Notably, the Mermaid tool offers users the flexibility to customize the appearance of the generated diagrams by incorporating various colors and themes. By incorporating diverse colors and themes, the Mermaid tool ensures the creation of a rich and varied dataset. This variability in visual presentation enhances the dataset’s versatility, accommodating different aesthetic preferences and catering to a wide range of visual styles.

Figure 7 illustrates the prompt utilized for generating text from mermaid code, constituting step three in our data generation pipeline. There are two variables involved in this prompt. The first variable is the language, initially set as “English”, yet it can be altered to accommodate any desired language. The second variable is the mermaid code generated from step one. GPT-3.5 turbo is capable of generating descriptive text based on the provided mermaid code.

### A.2 BD-EnKo

The BD-EnKo dataset comprises nine distinct categories: Flowchart, GraphLR, GraphTD, Journey, Sequence Diagram, Simple Graph, State Diagram,

C4, and Real world. Each category represents a specific type of diagram or visualization commonly encountered in various domains. Flowchart is typically used to represent processes or workflows, where each step is connected by arrows indicating the flow of control or information. GraphLR is a type of directed graph where nodes are arranged from left to right, commonly used to represent hierarchical structures or dependency graphs. GraphTD is similar to GraphLR, but nodes are arranged from top to bottom, often used to represent hierarchical structures or decision trees. Journey represents a series of steps or stages in a process, commonly used in customer journey mapping or user experience design. Sequence Diagram illustrates the interactions between objects or components in a system over time, commonly used in software engineering to visualize system behaviors. C4 represents software architecture at different levels of abstraction, including Context, Container, Component, and Code views. Real world represents real-world scenarios or situations, such as floor plans, network diagrams, or organizational charts. Each category serves a distinct purpose and is characterized by specific structural and visual elements tailored to its intended use case. Figure 8 showcases samples of each category from our dataset BD-EnKo, illustrating the diverse range of diagrams encompassed by this dataset.

### A.3 Additional Dataset Analysis

We have analyzed the descriptive statistics of our BD-EnKo dataset. The number of nodes in the training dataset has a mean of 12.15 and a standard deviation of 7.53, indicating moderate variability around the mean. The distribution of nodes shows that 25% of the data points have 7 or fewer nodes, 50% have 9 or fewer nodes, and 75% have 16 or fewer nodes. Similarly, the number of arrows has a mean of 11.07 and a standard deviation of 6.03, suggesting slightly less variability compared to nodes. The minimum number of arrows is 0, and the maximum is also 196. The distribution of arrows reveals that 25% of the data points have 6 or fewer arrows, 50% have 10 or fewer arrows, and 75% have 15 or fewer arrows. For the validation set, the number of nodes in the dataset has a mean of 12.11 and a standard deviation of 7.23, indicating moderate variability around the mean. The distribution of nodes shows that 25% of the data points have 7 or fewer nodes, 50% have 10 or fewer nodes, and 75%





### GPT-3.5 for Mermaid Code Generation

Your task is to generate different `graph TD` as shown below as an example by for different random texts and names used in different applications in English language and add more data to it.

```
graph TD;
  Start[New Project] --> Requirements[Collect & Validate Requirements];
  Start --> DataRetrieval[Retrieve Data];
  Requirements --> Transformation[Transform Data];
  DataRetrieval --> DatabaseQuery[Query Database];
  DataRetrieval --> ExternalDataFetch[Fetch External Data];
  DatabaseQuery --> RecordsFound[Database Records Found];
  ExternalDataFetch --> DataReceived[Received Data];
  DataReceived --> Storage&Utilization[Store & Utilize Data];
  RecordsFound --> DataAnalysis[Data Analysis];
  Rejected --> Requirements;
  Integration --> FinalOutput[Generate Final Output];
  DataAnalysis --> FinalOutput;
  Storage&Utilization --> FinalOutput;
  FinalOutput --> End[Project Completion];
```

#### Output from GPT-3.5

```
graph TD;
  Start[New Blog] --> Idea[Generate Blog Idea];
  Start --> Research[Conduct Research];
  Idea --> ContentCreation[Create Content];
  Research --> DataCollection[Collect Data];
  Research --> CaseStudy[Analyze Case Studies];
  DataCollection --> DataAnalysis[Analyze Data];
  DataAnalysis --> ContentCreation;
  ContentCreation --> Editing[Edit Blog];
  Editing --> Review[Review and Proofread];
  Review --> Publish[Publish Blog];
  Publish --> Promotion[Promote Blog];
  Promotion --> End[Blog Completion];
```

Figure 5: The prompt template is designed for generating mermaid codes. Utilizing this template, assist GPT-3.5 in generating a variety of raw mermaid codes in both English or Korean languages.

have 16 or fewer nodes. Similarly, the number of arrows has a mean of 11.06 and a standard deviation of 5.63, suggesting slightly less variability compared to nodes. The minimum number of arrows is 0, and the maximum is 70. The distribution of arrows reveals that 25% of the data points have 7 or fewer arrows, 50% have 10 or fewer arrows, and 75% have 15 or fewer arrows.

We have produced block diagrams encompassing a diverse array of topics. The graph presented in Figure 9 provides a glimpse into the top twenty topics extracted from our dataset. This visualization offers a concise overview of the most prevalent themes identified through our analysis, serving as a condensed representation of the breadth and scope of the data under examination.

#### A.4 BlockNet: Integrator

Figure 10 illustrates the prompt utilized to integrate outcomes from both the local information extractor and the global information extractor. This prompt comprises three variables that can be adjusted based on the specific use case. The first variable is “lang”, which can be set to any language since GPT-4V is proficient in handling various languages. The second variable represents the output from the global information extractor, denoted as “global”, providing an overall summary of the diagram. Lastly, the third variable represents the output from the local information extractor, referred

Dataset	Train		Validation	
	English	Korean	English	Korean
BD-EnKo	41,933	33,101	4,666	3,694
CBD	396	0	106	0
FC_A	357	0	0	0
FC_B	476	0	196	0
Total	43,162	33,101	4,968	3,694

Table 6: Dataset statistics for each split.

to as “local”, which consists of a collection of all potential triplets depicting the connections within the diagram.

#### A.5 Experimental Setup Details

**BlockNet** We train the YOLOv5 model utilizing the SGD optimizer with parameters lr=0.01 and weight decay=5e-5, maintaining a batch size of 128 and 300 fixed training steps. The training of the transformer encoder and decoder model is conducted according to the configuration outlined in Donut (Kim et al., 2022). For the visual encoder, we employed Swin-B with 10 layers and window sizes of 2, 2, 14, and 2 respectively. Additionally, we utilized the initial four layers of BART as the decoder. The model was optimized using the Adam optimizer with a learning rate of 3e-05. The input resolution of the model was set to 1280×960, with a maximum decoder length of 768. Splitting of dataset is shown in Table 6.

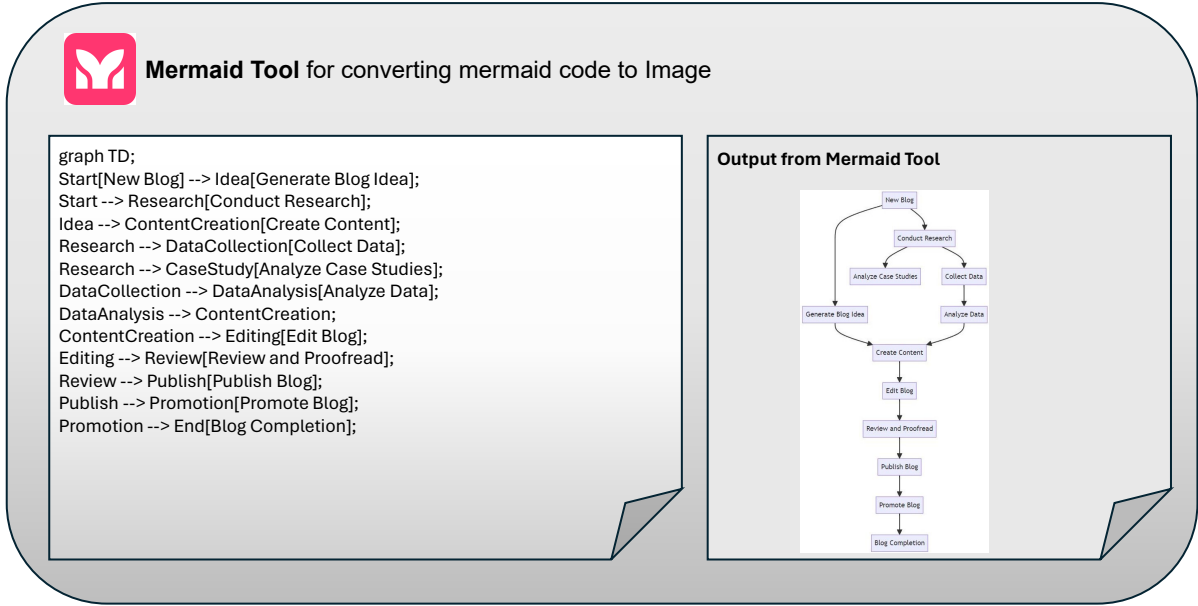


Figure 6: Mermaid tool used for generating block diagram images based on the given mermaid code.

Class	Precision	Recall	mAP50
Arrow	0.956	0.918	0.949
Terminator	0.986	1.000	0.995
Process	0.990	0.974	0.992
Decision	0.993	1.000	0.995
Text	0.992	0.842	0.965
Data	0.959	0.979	0.993
Average	0.979	0.952	0.982

Table 7: Symbol detection result on CBD dataset

**mPLUG** We compare our model with mPLUG (Ye et al., 2023), specifically using the “MAGAer13/mpplug-owl-llama-7b” model with a maximum token limit of 512 and retaining all settings unchanged. mPLUG, trained on a large dataset of image-text pairs with discriminative and generative objectives, achieves state-of-the-art performance across various vision-language tasks. Its efficient architecture incorporates cross-modal skip-connections for accelerated information processing. As this model lacks support for the Korean language, our testing was conducted exclusively with English diagrams.

**Donut** We employ the pre-trained Donut-base model from Hugging Face and fine-tune it with a combination of datasets used in training BlockNet to ensure a fair comparison. We configure the max epochs to 100 and warm-up steps to 20, with a maximum sequence length of 768. All other hyper-

parameters remain consistent with those utilized in the pre-training of Donut (Kim et al., 2022).

**BloSum/Faster R-CNN + T5/BART** We adopt the methodology outlined in the BloSum/Faster R-CNN and T5/BART papers (Bhushan and Lee, 2022), maintaining consistency by adhering to their specified settings and procedures. We conduct experiments exclusively with English data, as training T5 or BART models for the Korean language is not feasible due to the unavailability of appropriate open-source Korean triplet datasets for fine-tuning. Our focus on English datasets ensures comparable evaluations and comparisons as outlined in the original papers.

## A.6 Additional Experimental Results

We conduct symbol detection evaluation specifically on the CBD dataset, which possesses labeled data for symbol detection. Table 7 illustrates the precision, recall, and mAP50 (mean Average Precision at 50 IoU threshold) score values of the object detection module used in the local information extractor of BlockNet. Precision refers to the ratio of correctly predicted positive observations to the total predicted positives, indicating the model’s accuracy in identifying relevant symbols. Recall, on the other hand, represents the ratio of correctly predicted positive observations to the actual positives in the dataset, indicating the model’s ability to correctly identify all relevant instances. mAP50 measures the average precision across different IoU (In-



## GPT-3.5 for Block Diagram Text Generation

Your task is to convert the given mermaid code as shown below into meaningful sentence by capturing every detail information in **English** language.

```
flowchart TD;
  subgraph Main Process
    A[Start] -->|Initialize| B[Process data]
    B -->|Condition 1| C{ }
    C -->|Yes| D[Subroutine]
    C -->|No| E[End]
    B -->|Loop| F[Loop]
    F -->|Done| G[End]
    B -->|Parallel| H[Process A]
    H -->|Process B| I[ ]
    I -->|Process C| J[ ]
  end
  A --> K[Input]
  K --> L[Output]
  B -->|Database| M[Database]
  B -->|Subgraph| N[Subgraph]
```

### Output from GPT-3.5

The process of creating a new blog starts with generating a blog idea. Once the idea is generated, research is conducted to gather relevant information. The next step is to create the content for the blog. In parallel, data collection is done to gather necessary data. The collected data is then analyzed, and the findings are used in the content creation process. The created content is then edited, followed by a review and proofreading. After the review, the blog is published. Once published, the blog is promoted to reach a wider audience. Finally, the blog is considered complete.

Figure 7: The prompt template is designed for generating texts from mermaid codes. Utilizing this template, assist GPT-3.5 in generating a meaningful description in both English or Korean languages.

tersection over Union) thresholds up to 50%, providing a comprehensive evaluation of the model's detection performance. The experimental findings indicate that following fine-tuning with Yolov5m, achieves an impressive average precision of 0.979 for all categories on the CBD dataset.

Figure 11 depicts a side-by-side comparison of sample outputs generated by our model, BlockNet, and GPT-4V. A clear observation emerges: while GPT-4V faces challenges in accurately generating relations for block diagrams, BlockNet demonstrates a superior ability to adeptly capture all the intricate details and relationships within the diagrams. This stark contrast underscores the effectiveness and precision of our BlockNet model in generating coherent and contextually relevant outputs compared to GPT-4V. In Figure 12, we provide a glimpse of a sample output generated by our BlockNet model using the intuitive interface of the Gradio app. This visualization underscores the effectiveness and user-friendliness of our model within an accessible environment. Additionally, we have expanded the functionality of our method to include QA capabilities, as illustrated in Figure 13. This exciting enhancement showcases the adaptability and versatility of our approach, seamlessly transitioning between different tasks with minimal adjustments. While we have made significant strides, further research and experimentation are needed to fully solidify its robustness, which

will be our focus for future work.

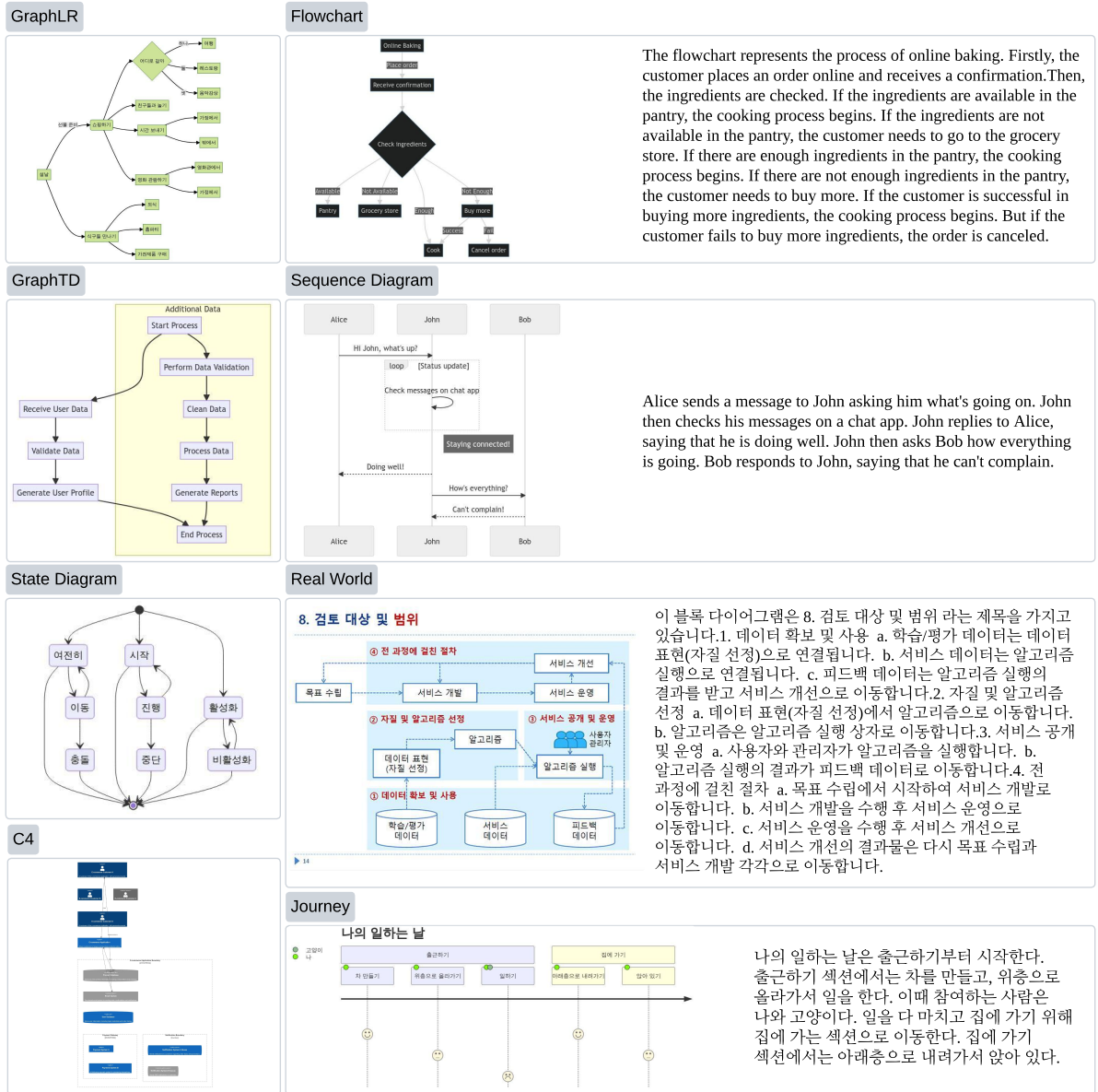


Figure 8: Sample diagrams representing various categories from the BD-EnKo dataset.



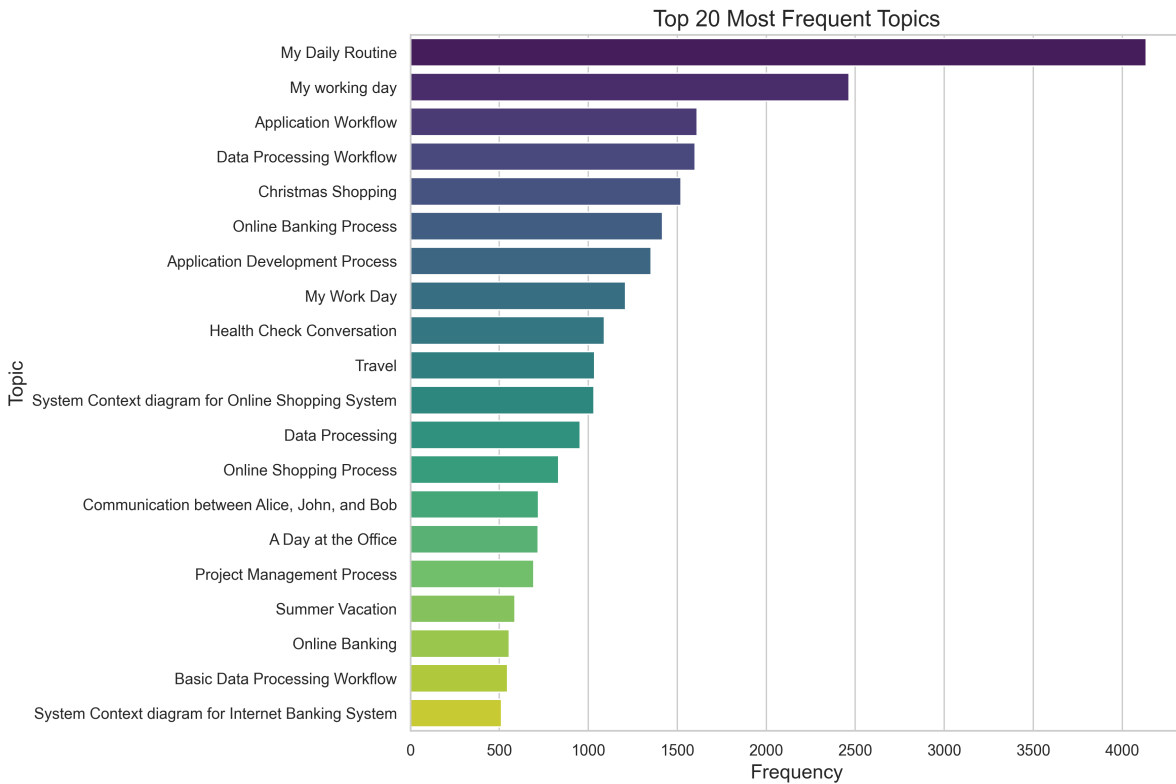


Figure 9: Top twenty topics from BD-EnKo dataset

### GPT-4V for Integration of Local and Global Information

“Your task is to generate the Description in detail of the given block diagram image with the help of Reference summary and Reference triplets only in `{lang}` Language without mentioning about these helps in the Output. \nReference summary: `{global}` \nReference triplets: `{local}`”

global: “Start is connected with Initialize which is then connected with Take Values which is further connected with Input Rate. Input Rate is connected with Compute Interest=  
Amount\*Years\*Rate/100 which is then connected with Output Interest which is finally connected with End.”

local: '<H> Start <R> connected with <T> Initialize', '<H> Initialize <R> connected with <T> Take Values', '<H> Take Values <R> connected with <T> Input Rate', '<H> Input Rate <R> connected with <T> Compute Interest Amount\*Years\*Rate/100', '<H> Compute Interest Amount\*Years\*Rate/100 <R> connected with <T> Output Interest', '<H> Output Interest <R> connected with <T> End'

Figure 10: The prompt template is designed to integrate the local and global information of the block diagram image. Utilizing this template, assist GPT-4V in generating a meaningful description in both English or Korean languages.

Image	BlockNet	GPT-4V
	<p>The block diagram begins with a "Start" process, which leads to an "Edit Video" function. From there, the flow splits into two paths based on the type of video: a "Short video" goes directly to the "Click to upload" process, while other types of videos do not follow this path. If the user has a short video, they confirm by selecting "YES" at the "Click to upload" decision point, which then leads to a "VIP user?" check. If the user is not uploading a short video, they are not presented with the "Click to upload" option. At the "VIP user?" decision point, if the user is not a VIP, indicated by selecting "NO," they are directed to "Sign Up." If the user is a VIP, as indicated by selecting "YES," they proceed to "Login." After logging in, the user is taken to "Browse to upload," which is the final step before reaching the "Finish" process. This completes the workflow depicted in the block diagram.</p>	<p>The block diagram outlines a process starting with "Start," where a user begins by editing a video. Once the video is edited, it is evaluated to determine if it is a short video. If the video is short, the user can proceed to "Click to upload." After clicking to upload, the process checks if the user is a "VIP user." If the user is a VIP, they are prompted to "Sign Up," and after logging in, they can "Browse to upload" their video, leading to the "Finish" of the process. However, if the video is not short, or if the user is not a VIP after clicking to upload, the user is directed to "Sign Up." After signing up, the user can then become a VIP user, log in, browse to upload, and finish the process. The diagram uses decision diamonds to represent the points where the process diverges based on the video's length and the user's VIP status.</p>
	<p>The block diagram presents a structured flow of decisions and options related to travel planning. At the top, "Travel" is the starting point, leading to "Book tickets" as the first action. Following this, one must "Choose destination," which is a critical step in the planning process. After selecting a destination, the next step is to "Check budget," which is represented by a diamond shape, indicating a decision point with three possible outcomes: "Within budget," "Limited budget," and "Luxury budget." Depending on the budget, different accommodation options are available: "Hotel" for within budget, "Hostel" for limited budget, and "Cruise Ship" for luxury budget. From the accommodation options, the diagram branches out further. The "Hotel" and "Hostel" lead to "Tourist attractions," while the "Cruise Ship" leads to "Relaxation" and then to "Spa." Additionally, the "Cruise Ship" option also connects to "Excursions," which in turn leads back to "Tourist attractions." The "Tourist attractions" branch splits into two paths: "Museums" and "Historical sites." Under "Museums," there are two further distinctions: "Art museum" and "Science museum." The "Historical sites" branch leads to "Castle" and "Temple." The "Castle" connects to "Medieval history" and "Royal residence," while the "Temple" connects to "Worship rituals" and is associated with "Ancient civilization." The "Ancient civilization" theme is expanded with connections to "Pyramids," "Ruins," and "Temples" in various countries such as "Egypt," "Greece," and "India." Each of these connections highlights different aspects of ancient civilizations, such as the "Pyramids" in Egypt, the "Ruins" in Greece, and "Temples" in India, which are associated with "Worship rituals."</p>	<p>The block diagram outlines a structured approach to planning a travel experience, beginning with the initial step of booking tickets, followed by choosing a destination. Once the destination is selected, the traveler checks their budget, which determines the type of accommodation and activities they can afford. The budget is categorized into three types: within budget, limited budget, and luxury budget. For those within budget, a hotel is the suggested accommodation. Those with a limited budget may opt for a hostel, while travelers with a luxury budget might choose a cruise ship. The diagram then branches out to tourist attractions, which are divided into excursions and relaxation activities, with the latter leading to a spa experience. Excursions further branch out to museums and historical sites. Museums are subdivided into art and science museums, while historical sites are categorized into castles and temples. Castles are associated with medieval history or being a royal residence, and temples are linked to ancient civilizations and worship rituals. The diagram then specifies examples of ancient civilizations, with pyramids in Egypt, ruins in Greece, and temples in India, each representing a unique cultural and historical significance.</p>
	<p>이 블록 다이어그램은 소프트웨어 개발 과정의 여러 단계를 나타냅니다. 가장 왼쪽에는 '요구사항 분석'이 위치하며, 이는 '인수 테스트'로 직접 연결됩니다. '요구사항 분석'은 또한 '시스템 설계'와 연결되며, '시스템 설계'는 '시스템 테스트'로 이어집니다. 다음 단계인 '아키텍처 설계'는 '통합 테스트'와 연결되어 있습니다. 마지막으로, '모듈 설계' 단계는 '단위 테스트'와 연결되어 있습니다. 이 다이어그램의 중앙에는 각 단계를 연결하는 화살표가 있는데, 이는 각 단계가 서로 연결되어 있음을 나타냅니다. 맨 아래에는 '구현'이라는 단계가 있으며, 이는 모든 설계 단계가 최종적으로 구현으로 통합되는 과정을 의미합니다.</p>	<p>이 블록 다이어그램은 네 가지 유형의 테스트와 그들 간의 관계를 보여줍니다. "구현"은 중앙 블록에서 시작하며, 네 가지 다른 테스트 유형으로 분기됩니다: "인수 테스트", "시스템 테스트", "통합 테스트", 그리고 "단위 테스트". 각 테스트 유형은 구매 과정의 다른 단계를 나타내며, 이들은 서로 연결되어 있지 않고 독립적으로 구매 프로세스에 기여합니다. 이 다이어그램은 특정 시스템이나 제품을 구매할 때 고려해야 할 다양한 검증 절차를 시각적으로 표현한 것으로 해석될 수 있습니다.</p>
	<p>이 블록 다이어그램은 채용 프로세스를 나타내고 있습니다. 상단에는 "경영관리 업무 프로세스 구축"이라는 제목이 있습니다. 프로세스는 "채용요청서 승인"에서 시작하여 "채용요청서 접수"로 이어지며, 접수처에 관련 정보를 받습니다. 이후 "채용 공고" 단계로 넘어가며, 이는 헤드헌팅 업체 담당자와 연결됩니다. 다음 단계는 "신규 채용이 필요한" 부분으로, 여기서는 해당 부서 팀장이 관련 정보를 확인합니다. 이어서 "1차 합격자 선발" 과정이 진행되며, 이 과정은 임원 및 해당 부서 팀장의 일정 확인과 연결됩니다. 선발된 후보자들은 "면접 일정 확정" 단계로 넘어가고, 이는 면접과 연결됩니다. 면접을 통과한 후보자들은 "최종 합격자 합격 통보" 단계로 이어서 교육 단계가 있으며, 여기서는 근로계약서 및 보안서약서 작성이 이루어집니다. 이 모든 과정은 관리자의 담당하에 진행됩니다. 마지막으로, 다이어그램 하단에는 "회사명 부서명 or 회사 로고"라는 텍스트가 있어, 이 다이어그램이 특정 회사의 채용 프로세스를 설명하기 위한 것임을 나타냅니다.</p>	<p>이 블록 다이어그램은 '채용 프로세스'라는 경영관리 업무 프로세스 구축을 설명하고 있습니다. '채용요청서 승인'은 '채용요청서 접수'와 '채용 공고'로 이어지며, '채용 공고'는 '면접'과 연결되어 있습니다. '면접'은 '최종합격자 발표'로 이어지며, '최종합격자 발표'는 '채용공고 (채용공고 승인)'로 연결되어 있습니다. '채용공고 (채용공고 승인)'는 '면접'과 연결되어 있습니다. '면접'은 '최종합격자 발표'로 이어지며, '최종합격자 발표'는 '채용공고 (채용공고 승인)'로 연결되어 있습니다. '채용공고 (채용공고 승인)'는 '면접'과 연결되어 있습니다. '면접'은 '최종합격자 발표'로 이어지며, '최종합격자 발표'는 '채용공고 (채용공고 승인)'로 연결되어 있습니다. '채용공고 (채용공고 승인)'는 '면접'과 연결되어 있습니다. '면접'은 '최종합격자 발표'로 이어지며, '최종합격자 발표'는 '채용공고 (채용공고 승인)'로 연결되어 있습니다. '채용공고 (채용공고 승인)'는 '면접'과 연결되어 있습니다. '면접'은 '최종합격자 발표'로 이어지며, '최종합격자 발표'는 '채용공고 (채용공고 승인)'로 연결되어 있습니다.</p>

Figure 11: Output from our BlockNet model (second column) and GPT-4V (third column) for the images provided in the first column. Red highlights signify incorrect interpretations of the diagram.

**Block Diagram Assistant**

Block Diagram Image

→ Indicates flow of instructions and data  
 ..... Indicates the control exercised by the control unit

output

The block diagram illustrates the architecture of a computer system, highlighting the various units and their interconnections. At the top, there is a "Storage Unit" that encompasses both "Secondary Storage" and "Primary Storage." These storage components are responsible for holding data and programs.

Below the storage units, there is a "Control Unit," which is central to the diagram and is connected to both the "Primary Storage" and the "Secondary Storage." The control unit manages the flow of instructions and data within the system, as indicated by the solid arrows.

Adjacent to the control unit, on the left side, is the "Input Unit," which is connected to the "Program and Data" source. This unit is responsible for receiving input from the user or from another system. On the right side of the control unit is the "Output Unit," which is connected to the "Information (Results)" destination. This unit is responsible for sending processed data out of the system.

Below the control unit is the "Arithmetic Logic Unit," which performs mathematical and logical operations. This unit is also connected to the control unit, indicating that the control unit directs its operations.

At the bottom of the diagram, the "Central Processing Unit (CPU)" is labeled, which encompasses the control unit and the arithmetic logic unit, indicating that these two units are part of the CPU.

The diagram also includes annotations that explain the meaning of the arrows: a solid arrow "Indicates flow of instructions and data," while a dotted arrow "Indicates the control exercised by the control unit." These annotations clarify the functions of the control unit in directing both the data flow and the operations of the system.

Select Task

Long Description

Select Language

English

Enter Question (QA)

Type your question here only for QA Task

Clear Submit

Flag

Figure 12: Sample output from the Gradio app demonstrating the Block Diagram to Text task. The model supports two tasks: Short Description, providing concise description, and Long Description, offering descriptive responses. The current model supports both English and Korean languages.

**Block Diagram Assistant**

Block Diagram Image

input\_image

Structural ecodesign of a composite wind turbine blade

Initial database

Initial eco-coefficients

Software tool for structural analysis

Assessment

Good

Bad

Optimisation

Update result

New database

New eco-coefficients

Display result

output

If the assessment is good, the next step according to the block diagram is to "Display result." This indicates that the outcome of the structural analysis is satisfactory, and the results can be presented or reported as they are.

Flag

Select Task

Long QA

Select Language

English

Enter Question (QA)

What to do if assessment is good?

Clear

Submit

Figure 13: Sample output from the Gradio app demonstrating the Block Diagram QA task. The model supports two QA tasks: Short QA, providing concise answers, and Long QA, offering descriptive responses. The current model supports both English and Korean languages.