Vision Language Models for Spreadsheet Understanding: Challenges and Opportunities

Shiyu Xia[*][†]**, Junyu Xiong**[*][†]**, Haoyu Dong**[‡]**, Jianbo Zhao**[†]**, Yuzhang Tian**[†]**,**
Mengyu Zhou, Yeye He, Shi Han, Dongmei Zhang
Microsoft Corporation

Abstract

This paper explores capabilities of Vision Language Models on spreadsheet comprehension. We propose three self-supervised challenges with corresponding evaluation metrics to comprehensively evaluate VLMs on Optical Character Recognition (OCR), spatial perception, and visual format recognition. Additionally, we utilize the spreadsheet table detection task to assess the overall performance of VLMs by integrating these challenges. To probe VLMs more finely, we propose three spreadsheet-to-image settings: column width adjustment, style change, and address augmentation.

We propose variants of prompts to address the above tasks in different settings. Notably, to leverage the strengths of VLMs in understanding text rather than two-dimensional positioning, we propose to decode cell values on the four boundaries of the table in spreadsheet boundary detection. Our findings reveal that VLMs demonstrate promising OCR capabilities but produce unsatisfactory results due to cell omission and misalignment, and they notably exhibit insufficient spatial and format recognition skills, motivating future work to enhance VLMs' spreadsheet data comprehension capabilities using our methods to generate extensive spreadsheet-image pairs in various settings.

1 Introduction

Spreadsheets are widely-used for data management and analysis (Birch et al., 2018; Wu et al., 2023). However, they are designed to be "human-friendly, not "machine-friendly" 1. Cells are arranged on the grid and illustrated by various visual formats like borders, colors, and bold fonts. Unlike machines, humans naturally leverage these visual cues to understand the layouts and structures of spreadsheets,

such as the location of the table (e.g., "A2:N32") using borders, the headers (e.g., "A2:N3") using bold fonts, and aggregated rows and columns (e.g., rows 17, 19, and 20) using fill colors.

While LLMs have shown promising performance in serializing spreadsheets as text sequences (Chen et al., 2024; Li et al., 2024), representing spreadsheets in this manner loses critical visual signals. With the recent surge in Vision Language Models (VLMs) (Laurençon et al., 2024), we propose studying the capability of language models to leverage visual signals for spreadsheet understanding. Fortunately, a spreadsheet can be straightforwardly processed using third-party tools like Interop and converted into an image. This motivates us to construct spreadsheet-image pairwise data for self-supervised tasks. To this end, we propose three self-supervised tasks to comprehensively examine critical abilities of VLMs separately: Optical Character Recognition (OCR) of cells, two-dimensional spatial position perception, and visual format recognition. Finally, we use spreadsheet table detection (Dong et al., 2019), a fundamental and enabling task in Microsoft Excel and Google Sheets, to jointly examine the effectiveness of VLMs, as this task combines the challenges of all three self-supervised tasks.

Specifically, as shown in Figure. 1, spreadsheet images present the following challenges: 1) The rows and columns are very compact, even overlapping, which makes the OCR task difficult. Specifically, VLMs sometimes struggle to split multiple cells and mistakenly treat them as a single cell. 2) The absence of explicit cell addresses and clear boundaries between rows and columns makes it difficult to perceive spatial locations. 3) Spreadsheets often contain a variety of formats, making it hard to recognize all formats precisely at the pixel level. To address these issues, we propose three different spreadsheet-to-image settings to probe the VLMs' performance: column width adjustment,

---

[*] Equal contribution.
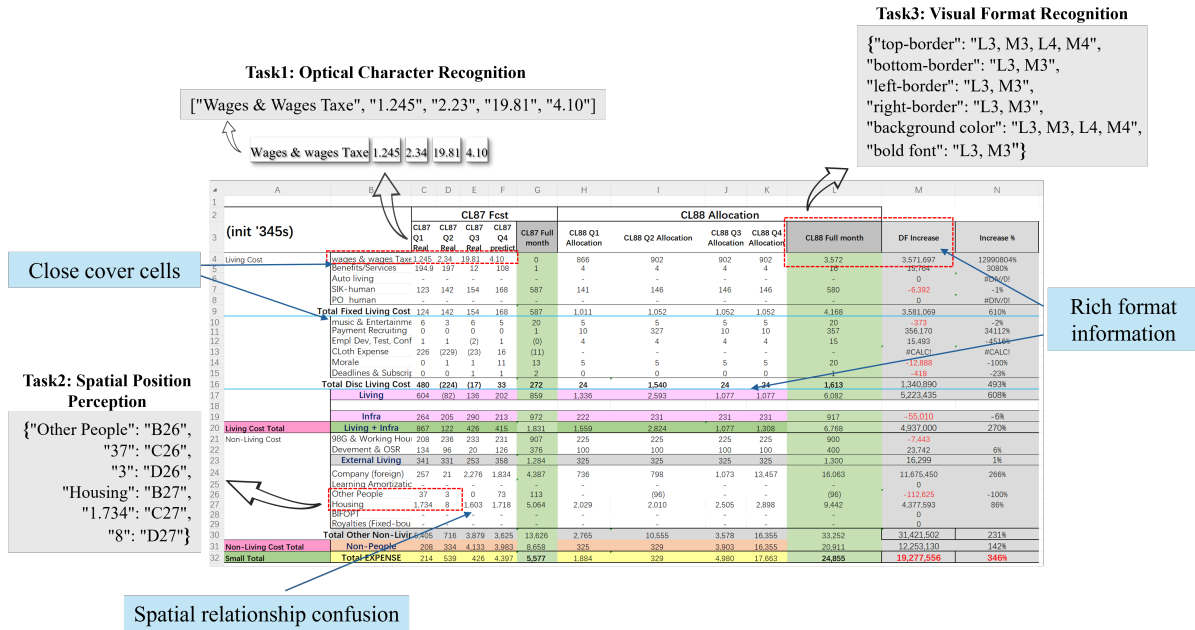[†] Work during internship at Microsoft.
[‡] Corresponding author.

Figure 1: A sample spreadsheet showing various challenging points in spreadsheet understanding task.

style change, and address augmentation respectively, as shown in Figure. 2.

We explore variants of prompts to address the above tasks in different settings. Notably, to leverage the strengths of VLMs in understanding text rather than two-dimensional positioning, we propose to decode cell values on the four boundaries of the table rather than decoding regions like "A2:E5" directly in the task of spreadsheet boundary detection. By analyzing the experiment results, we draw the following conclusions: Firstly, VLMs possess strong OCR capabilities, yet they often encounter issues of cell omission and prediction misalignment when dealing with spreadsheet images. Secondly, VLMs lack robust spatial perception in images because they need to infer the number of rows and columns implicitly on a large two-dimensional cell grid rather than reading it directly. It is highly noteworthy that their performance on recognizing visual formats on a cell grid is far from satisfactory; they are far from human-level in comprehending spreadsheet formats. Lastly, in the task of spreadsheet table detection, VLMs do not perform as well as the existing CNN-based TableSense (Dong et al., 2019), which is well-trained using a human-labeled dataset, indicating that there is still a long way to go in understanding spreadsheet images for VLMs.

2 Related Work

2.1 Table Representation

The advent of Large Language Models (LLMs) has significantly spotlighted the task of processing structured data (Jiang et al., 2023; Tang et al., 2023; Guo et al., 2023; Dong et al., 2022; Dong and Wang, 2024), particularly tabular data. In the quest to effectively communicate tabular data to LLMs, researchers have devised numerous formats, including HTML, JSON, Markdown, and XML, to represent such data. Studies by Sui et al. (Sui et al., 2023a) and Singha et al. (Singha et al., 2023) have underscored the efficacy of using Markdown and HTML for tabular data representation. However, these methods do not apply to spreadsheets since they have a single table assumption with an explicit region. Moreover, they do not leverage visual formats. (Deng et al., 2024) explored the usage of LLMs to evaluate representations of tables in image form, and Singh et al. (Singh et al., 2023) examined the capability of GPT-4 with vision(GPT-4V) (Achiam et al., 2023) on structured data, but they also focus on table-based input but not spreadsheet input that can include multiple tables and scattered notes. In contrast, there's a growing interest in exploring the vision perspective of spreadsheets to leverage the visual cues and take the whole spreadsheet rather than a single table as input. For instance, Dong et al. (Dong et al., 2019) uses CNN to capture spatial layouts of spreadsheet. (Wang et al.,

2021) uses transformer-based encoders to learn embeddings of cell values and formats in spreadsheets. However, our research diverges by focusing on exploring LLMs' ability to understand spreadsheet images. (Huang et al., 2023) proposed to model table boundaries as language sequences and use sequence decoder for table recognition.

2.2 Table-Related Tasks

Previous research has extensively explored tasks related to tables, encompassing table QA, table fact-checking, table-to-text, table manipulation, and table interpretation, etc (Pasupat and Liang, 2015; Novikova et al., 2017; Chen et al., 2020; Sui et al., 2023b; Li et al., 2023; Zhang et al., 2023). However, many of these tasks primarily revolve around understanding tables at the textual level. In reality, tables are often embedded within documents, images, and web pages, necessitating the exploration of related tasks such as table header detection, table structure recognition, and table recognition.

In recent studies, Fang et al. (Fang et al., 2012) identified tables within PDF documents using existing table extraction tools and employed machine learning algorithms to construct classifiers for identifying and categorizing table headers. Nassar et al. (Nassar et al.) introduced a novel table unit object detection decoder based on Transformer architecture to comprehend table structures. Ly et al. (Ly and Takasu, 2023) decomposed the table recognition task into two subtasks: table structure recognition and cell content recognition. They proposed an end-to-end multi-task learning model to address these subtasks.

However, our current study focuses more on the understanding of spreadsheet images by VLMs. This involves investigating the OCR capabilities of VLMs, their aptitude in capturing formatting information, their perception of spatial positioning, and their efficacy in detecting tables from spreadsheets (Dong et al., 2019).

3 Preliminary

3.1 Probing tasks

We design the following three probing tasks to evaluate the performance of VLMs on spreadsheet understanding.

Optical Character Recognition (OCR): A spreadsheet is a two-dimensional cell grid that differs from plain text. In OCR tasks for text, the output simply sequences the characters. However, OCR for spreadsheets not only involves recognizing characters but also requires organizing them in units of distinct cells as shown in Task1 of Figure. 1.

Understanding spatial position: The ability of VLMs to perceive the spatial position of images has been a long-standing challenge. Unlike ordinary images, spreadsheet images employ a precise two-dimensional coordinate system, where misalignment of rows and columns severely disrupts the understanding of information. Each cell's address corresponds to exact row and column coordinates, however, the images don't explicitly indicate the coordinate positions, so we define the top row in the image as the first row and the leftmost column as the first column. Consequently, the address of the cell located at the intersection of the first row and first column is defined as "1,1". Cell numbers increase from left to right and from top to bottom. As shown in Task2 of Figure. 1, the address for "Other People" is "B26." But for spreadsheet images without given coordinate positions, it should be recognized as "26,2".

Understanding visual format information: Spreadsheets contain rich formatting details that enhance comprehension and processing. If VLMs could "read" format information in images, it would perceive the images much like human do. Although spreadsheets contains a variety of format, we primarily focus on top border, bottom border, left border, right border, bold font, and fill color as shown in Task3 of Figure. 1.

3.2 Spreadsheet Table Detection Task

Spreadsheet table detection (Dong et al., 2019), involves identifying all tables within a given spreadsheet and determining their respective ranges. The spreadsheet will feature a visually rich design containing several tables scattered throughout, each potentially featuring a unique structure. Variability in the layout and structure of multiple tables contains rich visual information greatly complicating the task by obscuring table boundaries. Spreadsheet table detection is a horizontal and enabling task benefiting various intelligent features in spreadsheet softwares. Therefore, We employ this critical task in our work to assess the extent to how visual information influences the ability of VLMs to comprehend spreadsheets.
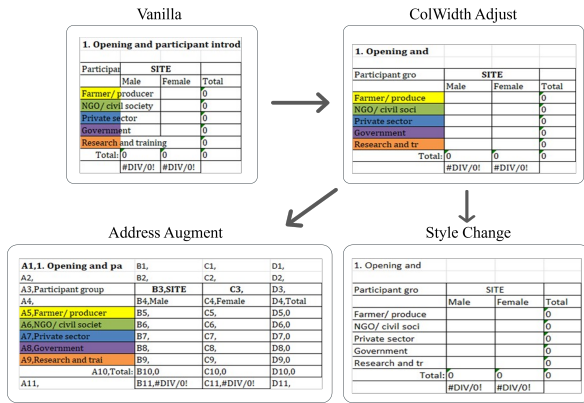
Figure 2: Illustration of spreadsheet-to-image settings.

4 Methodology

4.1 Dataset Construction

In order to study the spreadsheet understanding capabilities of VLMs such as GPT-4V and Gemini (Team et al., 2023), we convert the spreadsheet dataset (Dong et al., 2019) into images using the Microsoft Office Interop Excel library [1] without any human labeling efforts. Then we can simply reverse the dataset to get image-spreadsheet pairs.

Next, to probe the differences for VLMs to understand image spreadsheets under various image settings, we propose three processing methods on the input spreadsheet shown as Figure. 2. They are column width adjustment, style change, and address augment, respectively.

Column Width Adjustment: Since the column width effect the maximum number of characters displayed in each cell, if the column width is too small, the content between multiple cells will be very compact, making it difficult for the model (or even humans) to distinguish it. If the column width is too large, space will be wasted. Therefore, we come up with a setting that adjusts the column width based on the text length, but if the text length is too long, we limit it to the first 15 characters.

Style Change: Spreadsheet style attributes mainly include background color and various font properties such as bold, italic, fill color, and size. These styling elements serve specific functions, for instance, background color often groups similar data, while font color and bolding emphasize important details. These styles provide distinct visual cues within the spreadsheet. To minimize the influence of these stylistic elements on the under-

Figure 3: The prompt of OCR task.

standing of VLMs, it's necessary to standardize these attributes: removing background colors and bold formatting from each cell, setting font color to black, and using a consistent font type and size, such as Calibri at 12pt, etc.

Address Augment: In spreadsheets, cell contents typically serve the sole purpose of storing data. However, a comprehensive understanding of the spreadsheet requires grasping the spatial relationships and format correspondences between cells. Existing VLMs may struggle to robustly capture these precise spatial relationships. To address this, we propose a new setting that incorporates cell address information alongside the cell content. That is, we explicitly concatenate the cell address (e.g., "A1") with its value (e.g., "day"), using a comma to separate them. This results in a fashion like "A1, day."

4.2 Optical Character Recognition

We instruct the VLM to sequentially decode the text of each cell in the spreadsheet image, moving from top to bottom and left to right, while omitting cells that contain null values. Figure. 3 provides a prompt example.

Evaluation Method: We adopt two kinds of evaluation method, Strict and longest common substring (LCS). As shown in Figure. 4, the LCS argorithm is uesd to find the longest common subsequence between the predicted sequence and the ground truth sequence. It helps to effectively alleviate the problem of poor performance caused by missing some cells in the output and can test the OCR ability of the VLMs to the greatest extent.

Figure 4: The difference between LCS matching and Strict matching.

---

[1] https://github.com/microsoft/Windows-Packaging-Samples/tree/master/OfficeInterop/Excel.Interop

Figure 5: The prompt for vanilla experiment of spatial position perception task.

Figure 6: The prompt for vanilla experiment of visual format recognition task.

4.3 Spatial Position Perception

We prompt the VLMs to recognize the spatial positions of specified cells ensuring that each cell value and its address correspond uniquely. Figure. 5 provides a prompt for vanilla experiment, other prompts see Appendix A.

Specifically, we input a spreadsheet image along with a list of randomly shuffled cell values into the VLMs. Then, we prompt the VLMs to output the address corresponding to each value. It is important to note that for the vanilla, colwidth adjust, and style change experiments, the input image does not contain cell addresses. Therefore, the addresses output by the VLMs should be composed of the row and column indices of the cell, in the form "2,3". In contrast, the address augment experiment outputs addresses in the form "C2".

4.4 Visual Format Recognition

We have defined six specific cell formats: top border, bottom border, left border, right border, bold font, and fill color. For each format, we instruct the VLMs to identify and output the addresses of all cells that exhibit the specified format. Figure. 6 provides a prompt for vanilla experiment, other prompts see Appendix A.

This experiment is similar to the spatial position perception experiment. The addresses output by the vanilla and colwidth adjust experiments should be composed of the row and column indices of the cell in the form "1,2", while the address augment experiment outputs addresses in the form "B1".

4.5 Spreadsheet Table Detection

We instruct VLMs to detect all table ranges from spreadsheet images. Figure. 7 provides a prompt for vanilla experiment, other prompts see Appendix A.

By convention, contiguous cell ranges are represented by the addresses of the upper left and lower right cells, separated by ":", and cells are referenced by their column and row indices, e.g., "A4:D120". However, when presenting a spreadsheet as an image input to the VLMs, the image may lack the ability to deduce the cell addresses. To address this challenge, we propose a novel approach where the VLMs directly decode the contents of the four boundaries of the table. Subsequently, these decoded contents are mapped to a conventional addresses using our proposed method as introduced in the follow paragraph.

Specifically, except for the address augment experiment, which can directly output a range in the form "A4:D120," the other experiments output the result by decoding the four boundaries.

Mapping Algorithm: Consider a spreadsheet S comprising m rows and n columns, where each cell is represented by $c_{i,j}$, with i and j denoting its row and column index, respectively, within the spreadsheet.

$$S = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{bmatrix} \quad (1)$$

By allowing the model to decode the four boundaries of all tables, we obtain the model's prediction result denoted as $Predict = [T_1, T_2, ...]$. Among them, T_i means the predicted four boundaries of the i-th table, that is,

$$T_i = \left\{ \begin{array}{l} B_t : [c_1, c_2, \ldots, c_t], \\ B_b : [c_1, c_2, \ldots, c_b], \\ B_l : [c_1, c_2, \ldots, c_l], \\ B_r : [c_1, c_2, \ldots, c_r] \end{array} \right\} \quad (2)$$

Among them, B_t, B_b, B_l and B_r represent the contents of top_border, $bottom\_border$, $left\_border$ and $right\_border$ respectively. For top_border and bottom_border, we can map it to the most likely row index in the spreadsheet through

algorithm 1. For left_border and right_border, we only need to transpose them and do the same. Finally, after we obtain the row/column index corresponding to each predicted border, we process it into a region such as "A1:D9".

Algorithm 1: Map the content of a specific row to the corresponding row index.

Input : The border content B predicted by the model and the contents S of the spreadsheet.

Initialize the origin confidence $Conf$ to 0.8.;

Initial the result index res to -1.;

for $i = 1$ **to** $|S|$ **do**
   if $|S[i]| \geq |B|$ **then**
      $tList \leftarrow S[i]$;
      $sList \leftarrow B$;
   end
   else
      $tList \leftarrow B$;
      $sList \leftarrow S[i]$;
   end
   $sCnt \leftarrow |sList|$;
   $tCnt \leftarrow |tList|$;
   for $j = 1$ **to** $tCnt$ **do**
      if $j + shortCnt > tCnt$ **then**
         Break;
      end
      $cConf \leftarrow \frac{\sum_{k=1}^{sCnt}(sList[k]==tList[k])}{sCnt}$;
      if $cConf \geq Conf$ **then**
         $res \leftarrow i$;
         $Conf \leftarrow cConf$;
      end
   end
end

Output : the result index res.

---

5 Experiment Setting

We conducted experiments using GPT-4V (*2024-02-15 preview*) (Achiam et al., 2023) and Gemini (*1.5-pro lateset until 2024-05-16*) (Team et al., 2023). To ensure consistent experimental parameters, we set the generation temperature for both GPT-4V and Gemini-pro to 0.7, top_p to 0.95, and max_output_token to 4096.

Due to GPT-4V's input image restrictions—specifically, that the image file size must be less than 4MB and the resolution must

Figure 7: Zero-shot prompt for vanilla images decoding four boundaries on spreadsheet table detection task.

be between 50×50 and $10,000 \times 10,000$ pixels—we filtered 76 images from the test dataset of TableSense (Dong et al., 2019) to meet these criteria. For each experiment, we evaluate the models using precision, recall, and F1, repeating each experiment three times and taking the average results.

6 Experiment Result

6.1 Performance of Optical Character Recognition

Table. 1 presents the OCR task results of GPT-4V and Gemini-pro, calculated using both Strict and LCS matching methods. From the table, we can observe:

1) Both GPT-4V and Gemini-pro are generally capable of accurately recognizing the content in spreadsheet images. Specifically, the best performance of GPT-4V and Gemini-pro can reach F1 scores of 79.59% and 81.85%, respectively, demonstrating their strong ability to recognize content in a two-dimensional grid.

2) For both GPT-4V and Gemini-pro, the performance of LCS matching far exceeds that of Strict matching, indicating that they tend to miss some cells or predict misalignments during performing OCR task, causing almost all predictions to be incorrect from the first missed cell in Strict matching. Specifically, GPT-4V's F1 scores under LCS matching are higher than Strict matching by 54.98%, 62.63%, and 52.98% and Gemini-pro's F1 scores under LCS matching are higher by 66.51%, 65.16%, and 66.9% for the three different inputs, respectively.

3) Preprocessing spreadsheets by adjusting column width significantly enhances the OCR capabilities of VLMs on spreadsheet images, but further preprocessing with style change does not improve

%		Strict match			LCS match		
		Precision	Recall	F1	Precision	Recall	F1
GPT-4V	Vanilla	14.78	12.68	13.65	74.32	63.74	68.63
	ColWidth Adjust	17.87	15.24	16.96	83.87	75.74	79.59
	Style Change	27.44	25.54	26.46	82.40	76.69	79.44
Gemini-pro	Vanilla	9.26	8.08	8.63	80.61	70.39	75.14
	ColWidth Adjust	16.13	13.03	14.42	89.03	71.94	79.58
	Style Change	16.40	13.74	14.95	89.80	75.20	81.85

Table 1: Precision, recall and F1 results of GPT-4V and Gemini-pro on OCR task. Among them, colwidth adjust is the processing operation of column width adjustment.

the OCR performance of VLMs. Specifically, adjusting column width can increase GPT-4V's F1 scores by 3.31% and 10.96% in the Strict match and LCS match methods, respectively, and increase Gemini-pro's F1 scores by 5.79% and 4.44% in the Strict match and LCS match methods, respectively.

4) Gemini-pro's OCR capability on spreadsheet images is slightly stronger than that of GPT-4V. Specifically, in the vanilla and style change experiments, Gemini-pro's F1 scores are 6.51% and 2.41% higher than those of GPT-4V, respectively.

Finally, we analyze the results of GPT4-V on a case in detail in Appendix B.1.

%		Precision	Recall	F1
GPT-4V	Vanilla_{number}	12.44	12.37	12.41
	$\text{ColWidth Adjust}_{number}$	13.45	13.35	13.39
	$\text{Style Change}_{number}$	12.16	12.14	12.15
	$\text{Address Augment}_{address}$	48.87	49.09	48.97
Gemini-pro	Vanilla_{number}	16.72	18.00	17.33
	$\text{ColWidth Adjust}_{number}$	14.29	15.40	14.82
	$\text{Style Change}_{number}$	16.75	18.13	17.41
	$\text{Address Augment}_{address}$	83.66	87.53	85.55

Table 2: Precision, recall and F1 results of GPT-4V and Gemini-pro on spatial position perception task.

%		Precision	Recall	F1
GPT-4V	Vanilla_{number}	24.97	11.69	15.79
	$\text{ColWidth Adjust}_{number}$	24.31	11.07	14.77
	$\text{Address Augment}_{address}$	28.88	13.28	17.83
Gemini-pro	Vanilla_{number}	35.27	13.28	17.53
	$\text{ColWidth Adjust}_{number}$	35.09	12.69	16.93
	$\text{Address Augment}_{address}$	41.93	16.78	22.19

Table 3: Precision, recall and F1 results of GPT-4V and Gemini-pro on visual format recognition task.

6.2 Performance of Spatial Position Perception

Table 2 shows the results of GPT-4V and Gemini-pro in performing spatial position perception tasks. Analyzing the results in Table 2, we first observe that GPT-4V and Gemini-pro perform poorly in the vanilla, colwidth adjust, and style change experiments. This underperformance is attributed to the three types of experiments demanding that the VLMs count the rows and columns in the spreadsheet. However, the boundaries of rows and columns in the spreadsheet are often unclear due to the lack of borders or the presence of line breaks that cause content overlap (e.g., "A5", "C3", etc. in Figure. 1).

Secondly, we noted that although preprocessing spreadsheets with address augment can significantly enhance the performance of both GPT-4V and Gemini-pro, since address augment allows VLMs to fully utilize their OCR capabilities, GPT-4V does not achieve the same level of OCR performance as Gemini-pro. This suggests that GPT-4V may not understand the task prompts as thoroughly as Gemini-pro.

In addition, we observe that in the four types of experiments, Gemini-pro outperform GPT-4V in F1 scores by 4.92%, 1.43%, 5.26%, and 36.58%, respectively, indicating that Gemini-pro has a stronger spatial position perception capability in spreadsheet image tasks.

Finally, we analyze the results of GPT4-V on a case in detail in Appendix B.2.

6.3 Performance of Visual Format Recognition

Table 3 presents the results of GPT-4V and Gemini-pro in testing their ability to recognition the visual

%		Zero-Shot			One-Shot			Trained		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
GPT-4V	Vanilla$_{four}$	52.38	10.23	17.11	49.29	8.66	14.68	-	-	-
	ColWidth Adjust$_{four}$	49.43	17.49	25.79	48.72	15.10	23.05	-	-	-
	Address Augment$_{range}$	9.26	14.85	11.41	14.60	13.86	14.22	-	-	-
Gemini-pro	Vanilla$_{four}$	25.96	18.40	21.53	35.82	6.98	11.67	-	-	-
	ColWidth Adjust$_{four}$	26.66	22.40	24.33	26.93	7.03	11.15	-	-	-
	Address Augment$_{range}$	9.08	19.94	12.47	7.62	15.55	10.00	-	-	-
TableSense	Text Input	-	-	-	-	-	-	80.21	76.24	78.17

Table 4: Precision, recall and F1 results of GPT-4V, Gemini-pro and TableSense (Dong et al., 2019) on spreadsheet table detection task.

format information in spreadsheet images. The results indicate that the best F1 scores for GPT-4V and Gemini-pro across multiple experiments are only 17.83% and 22.19%, respectively. This demonstrates that their ability to comprehend format information in images is quite poor and that they cannot deeply understand images by combining format information as humans do. Therefore, this is an area where VLMs need improvement in the future. Additionally, in two types of experiments, Gemini-pro's F1 scores are higher than GPT-4V's by 1.74%, 2.16% and 4.36%, respectively, indicating that Gemini-pro again has a slight edge over GPT-4V in this aspect.

Then, we analyze the results of GPT4-V on a case in detail in Appendix B.3.

6.4 Performance of Spreadsheet Table Detection

The results of GPT-4V and Gemini-pro for the spreadsheet table detection task are shown in Table 4. Firstly, we can see that the F1 scores obtained by having the VLMs decode the four boundaries and then applying our proposed mapping algorithm are significantly higher than those obtained by directly outputting the address range (e.g., "A1:C10"). Specifically, GPT-4V's zero-shot performance is 5.7% and 14.38% higher, and Gemini-pro's is 9.06% and 11,86% higher,respectively, which can be attributed to their excellent OCR capabilities.

Secondly, both GPT-4V and Gemini-pro fall significantly short when compared to TableSense, with the closest F1 result still being 52.38% lower. However, it is worth noting that TableSense inputs inputs serialized text from the spreadsheet, whereas the VLMs we are exploring take images as input. This indicates that there is a long way to go in

continuously improving VLMs to achieve results comparable to text input.

Moreover, we observed an anomalous result: the one-shot results of GPT-4V and Gemini-pro are generally worse than their zero-shot results. This might be due to the complex structure of spreadsheets, where providing an example can lead VLMs to favor outputs with structures similar to the example, resulting in misjudgments.

Finally, we analyze the results of GPT4-V on a case in detail in Appendix B.4.

7 Conclusion and Future Work

In this paper, we develop a suite of probing tasks aimed at evaluating the critical capabilities of VLMs in OCR, comprehension of formatting details, and recognition of spatial positioning within spreadsheet images. Our findings demonstrate that while VLMs possess strong OCR capabilities, they are prone to cell omission and prediction misalignment during OCR tasks on spreadsheet images. Furthermore, their spatial perception is insufficient, as they struggle to accurately determine the row and column numbers of cells in a two-dimensional spreadsheet grid. Surprisingly, VLMs cannot comprehend visual formats well like humans. Additionally, we introduce a spreadsheet table detection task designed to thoroughly assess the ability of VLMs to interpret spreadsheet images effectively. However, the performance of this task falls short of that achieved by existing SOTA method, indicating that processing and comprehending spreadsheets remains a significant challenge.

Future research could focus on handling larger spreadsheet images and segmenting these spreadsheets without compromising the integrity of their format and spatial relationships. Despite these challenges, the potential benefits of treating spread-

sheets as images are substantial. In this paper, we have proposed methods that can massively generate spreadsheet-image pairs, and under our proposed settings, we can control various challenges. Utilizing these methods to generate large amounts of data, we train open-source large models to enhance their understanding of structured data on grids, further advancing the comprehensive capabilities of vision language models.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

David Birch, David Lyford-Smith, and Yike Guo. 2018. The future of spreadsheets in the big data era. *arXiv preprint arXiv:1801.10231*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. *Preprint*, arXiv:1909.02164.

Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, and Jianye Hao. 2024. Sheetagent: A generalist agent for spreadsheet reasoning and manipulation via large language models. *arXiv preprint arXiv:2403.03636*.

Naihao Deng, Zhenjie Sun, Ruiqi He, Aman Sikka, Yulong Chen, Lin Ma, Yue Zhang, and Rada Mihalcea. 2024. Tables as images? exploring the strengths and limitations of llms on multimodal representations of tabular data. *arXiv preprint arXiv:2402.12424*.

Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. 2022. Table Pretraining: A survey on model architectures, pretraining objectives, and downstream tasks. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*.

Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. 2019. Tablesense: Spreadsheet table detection with convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 69–76.

Haoyu Dong and Zhiruo Wang. 2024. Large language models for tabular data: Progresses and future directions. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2997–3000.

Jing Fang, Prasenjit Mitra, Zhi Tang, and C Lee Giles. 2012. Table header detection and classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 599–605.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*.

Yongshuai Huang, Ning Lu, Dapeng Chen, Yibo Li, Zecheng Xie, Shenggao Zhu, Liangcai Gao, and Wei Peng. 2023. Improving table structure recognition with visual-alignment sequential coordinate modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11134–11143.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and ZHAO-XIANG ZHANG. 2024. Sheetcopilot: Bringing software productivity to the next level through large language models. *Advances in Neural Information Processing Systems*, 36.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*.

Nam Tuan Ly and Atsuhiro Takasu. 2023. An end-to-end multi-task learning model for image-based table recognition. *arXiv preprint arXiv:2303.08648*.

A Nassar, N Livathinos, M Lysak, and PWJ Staar. Tableformer: table structure understanding with transformers. corr abs/2203.01017 (2022).

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *Preprint*, arXiv:1706.09254.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *Preprint*, arXiv:1508.00305.

Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, and Gust Verbruggen. 2023. Assessing gpt4-v on structured reasoning tasks. *arXiv preprint arXiv:2312.11524*.

Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. 2023. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. *arXiv preprint arXiv:2310.10358*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2023a. Gpt4table: Can large language models understand structured table data? a benchmark and empirical study. *arXiv preprint ArXiv:2305.13062*.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2023b. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. *arXiv preprint arXiv:2312.09039*.

Xiangru Tang, Yiming Zong, Yilun Zhao, Arman Cohan, and Mark Gerstein. 2023. Struc-bench: Are large language models really good at generating complex structured data? *arXiv preprint arXiv:2309.08963*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Xindong Wu, Hao Chen, Chenyang Bu, Shengwei Ji, Zan Zhang, and Victor S Sheng. 2023. Huss: A heuristic method for understanding the semantic structure of spreadsheets. *Data Intelligence*, 5(3):537–559.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023. Tablellama: Towards open large generalist models for tables. *arXiv preprint arXiv:2311.09206*.

A Prompt Examples

Figure. 8 shows the prompt for address augment experiment in spatial position perception task; Figure. 9 shows the prompt for address augment experiment in visual format recognition task; Figure. 10 shows the prompt for spreadsheet table detection task under different experiment setting and output format.

> **Prompt:** Given an input that consists of an image containing the spreadsheet and a list which elements are cell values in that spreadsheet. Please output the address where it appears in the spreadsheet for each cell value in the list in the order in the list. Specifically:
> 1. Each cell in the spreadsheet contains a cell address and its corresponding value, separated by a comma, for example, "A1,Year". The value may be empty, resulting in cell content like "A1,".
> 2. The cell values contained in the list are arranged in order from top to bottom and from left to right. If the cell value is empty, it does not appear in the list.
> 3. The output should be strictly in the following JSON format(Don't output anything else).
> {
>   "Cell-1":["Address-1"],
>   "Cell-2":["Address-2"],
>   "Cell-3":["Address-3"],
>   ...
> }
> Note that the above data such as "Address-1" is replaced with the real cell address and the data such as "Cell-1" is replaced with the real cell value.
> Here are the image and the list:
> <image> <values list>

Figure 8: The prompt for address augment experiments of spatial position perception task.

> **Prompt:** The provided image depicts the spreadsheet where each cell contains a cell address and its corresponding value, separated by a comma, for example, "A1,Year". The value may be empty, resulting in cell content like "A1,". Each cell may have six formats. Specifically, the "top-border" indicates whether the cell has a top border, the "bottom-border" indicates whether the cell has a bottom border, the "left-border" indicates whether the cell has a left border, and the "right-border" indicates whether there is a right border, the "bold font" indicates whether the font in the cell is bold, and the "background color" indicates whether the cell is filled with the background color (white represents no filling). For each format, your task is to output the addresses of all cells with this format, arranged from left to right and top to bottom. The output should strictly adhere to the following JSON format(Don't output anything else):
> {
>   "top-border": ["Address-1", "Address-2", ...],
>   "bottom-border": ["Address-1", "Address-2", ...],
>   "left-border": ["Address-1", "Address-2", ...],
>   "right-border": ["Address-1", "Address-2", ...],
>   "font bold": ["Address-1", "Address-2", ...],
>   "background color": ["Address-1", "Address-2", ...]
> }
> In this structure, "Address" represents the cell address.
> Here's the image:
> <image>

Figure 9: The prompt for address augment experiments of visual format recognition task.

B Case Study

B.1 A case of OCR task

In order to deeply explore the impact of different reconstructed spreadsheet images on GPT-4V's OCR capabilities, we will analyze the case shown in Figure. 12 in detail.

First, by comparing the results of Figure. 12a and Figure. 12b, we can clearly find that not adjusting the column width in the spreadsheet will cause the OCR capability of GPT-4V to drop significantly. This is due to the fact that the cell content in many spreadsheets will not be fully displayed when the column width is not adjusted, and there may be overlap or coverage between adjacent cells, as shown in Figure. 12a.

Secondly, by observing these three pictures, we will find that GPT-4V has insufficient positioning capabilities when performing OCR, resulting in

> **Prompt:** The image I provide is a spreadsheet. The address of the first cell in the upper left corner of the spreadsheet is "1,1", which is the first row and first column. The number of columns increases from left to right, and the number of rows increases from top to bottom. I need you to find the range of each table in the spreadsheet. For each table, find: (1) the number of top row, (2) the number of bottom row, (3) the number of leftmost column and (4) the number of rightmost column. Specifically:
> 1. There may be one or more tables in the spreadsheet, so all RANGE should be returned strictly in the list form, for example ["number_top, number_bottom, number_left, number_right",...] (don't output anything else).
> 2. Keep in mind that tables don't contain titles and comments, but may contain header rows (head or left), which help you find the correct table.
> Here's the input image:
> <image>

(a) The zero-shot prompt for outputting ranges in vanilla experiments.

> **Prompt:** The provided image depicts the spreadsheet where each cell contains a cell address and its corresponding value, separated by a comma, for example, "A1,Year". The value may be empty, resulting in cell content like "A1,". I need you to find the range of each table in the spreadsheet and output addresses of the cells in the top-left and bottom-right corners separated by ":", like "Address-1: Address-2". Specifically:
> 1. There may be one or more tables in a spreadsheet, so you should output all the RANGE in list form: ["Address_1:Address_2", "Address_3:Address_4", ...], don't output anything else.
> 2. Keep in mind that tables don't contain notes, captions and comments, but may contain table headers and row headers, which help you find the correct table.
> Here's the input image:
> <image>

(b) The zero-shot prompt for outputting ranges in address augment experiments.

> **Prompt:** The picture I provide is a spreadsheet. I need your help to find the number of tables in the spreadsheet and the boundaries of each table. Specifically, for each table boundary, find: (1) the top-most-row, (2) the bottom-most-row, (3) the left-most-column and (4) the right-most-column. Please output the results according to the following rules.
> 1. Please strictly follow the format to return the number of tables you find and each cell value on the table boundaries. The cell values must be output accurately and in order (just output the values shown on the picture).
> 2. If some cell values are empty, do not output these cells;
> 3. The values of the vertex cell in the upper left corner of the table should appear on both the top-most row and left-most column, and the value of the vertex cell in the lower right corner of the table appear on both the right-most column and bottom-most-row;
> 4. Keep in mind that tables don't contain titles and comments, but may contain header rows (head or left), which help you find the correct table.
> If you have two tables in an Excel spreadsheet, the output format should be as follows (don't output anything else):
> [
>   {
>   "top-most-row": ["cell-1", "cell-2", ...],
>   "bottom-most-row": ["cell-1", "cell-2", ...],
>   "left-most-column": ["cell-1", "cell-2", ...],
>   "right-most-column": ["cell-1", "cell-2", ...]
>   },
>   {
>   "top-most-row": ["cell-1", "cell-2", ...],
>   "bottom-most-row": ["cell-1", "cell-2", ...],
>   "left-most-column": ["cell-1", "cell-2", ...],
>   "right-most-column": ["cell-1", "cell-2", ...]
>   }
>]
> I have given two pictures containing the spreadsheet, the first is an example I gave, which will be introduced below; the second is the picture I need you to give answers according to the requirements above.
> Example:
> Input: The first image
> Output:
> [
>   {
>   "top-most-row": ["Industry Group", "Project Name", "Responsible Pa", "Project Type", "$$ Exposure", "Assessment of R", "Updated"], "Update/Comments"],
>   "bottom-most-row": ["Terradyne Group", "-", "Sold?", "1/8/2000"],
>   "left-most-column": ["Industry Group", "Coal", "Exploration / E", "Acquisition & D", "Oil Field Servi", "Pulp & Paper", "Steel", "Other"],
>   "right-most-column": ["Update/Comments"]
>   }
>]
> Input: The second image
> Output:

(c) The one-shot prompt for decoding four boundaries in vanilla experiments

> **Prompt:** The provided image depicts the spreadsheet where each cell contains a cell address and its corresponding value, separated by a comma, for example, "A1,Year". The value may be empty, resulting in cell content like "A1,". I need you to find the range of each table in the spreadsheet and output addresses of the cells in the top-left and bottom-right corners separated by ':', like "Address-1: Address-2". Specifically:
> 1. There may be one or more tables in a spreadsheet, so you should output all the RANGE in list form: ["Address_1:Address_2", "Address_3:Address_4", ...], don't output anything else.
> 2. Keep in mind that tables don't contain titles and comments, but may contain header rows (head or left), which help you find the correct table.
> I have given two pictures containing the spreadsheet, the first is an example I gave, which will be introduced below; the second is the picture I need you to give answers according to the requirements above.
> Example:
> Input: The first image
> Output:
> ["A3:H57"]
> Input: The second image
> Output:

(d) The one-shot prompt for outputting ranges in address augment experiments.

Figure 10: The prompt of spreadsheet table detection task.

some cells being missed or misplaced during the prediction process. For example, GPT-4V's prediction results for the three pictures in Figure. 12 ignore the first two lines of the spreadsheet, and in both Figure. 12b and Figure. 12c, GPT-4V predicts "Aronowitz, Alan" to "713-858-7795" After, but actually it should be in front.

B.2 A case of spatial position perception task

Figure. 11 presents a tangible example evaluating GPT-4V's proficiency in spatial position awareness within spreadsheet environments. Upon scrutiny, it's apparent that even in relatively straightforward scenarios, both vanilla and style change experi-

(a) Vanilla

(b) Style Change

(c) Address Augment

Figure 11: An example of GPT-4V on spatial position perception task. The content marked in red indicates the LCS match error prediction

ments reveal GPT-4V's inadequate performance in accurately predicting position. While GPT-4V effectively forecasts column positions for most cells, it consistently struggles with row positions, consistently displaying an offset. This issue becomes more pronounced in the presence of empty rows, leading to inaccuracies in subsequent cell position predictions.

In contrast, the address augment experiment showcases a comparatively better performance by GPT-4V. This improvement can be attributed to its impressive OCR capabilities, allowing it to accurately identify and pair cell addresses with their corresponding values within a single cell.

B.3 A case of visual format recognition task

The case depicted in Figure. 14 presents the outcomes of GPT-4V's analysis under vanilla and address augment experiments. Examination of these results reveals GPT-4V's limited grasp of format information in both scenarios, indicating its potential inability to comprehend spreadsheet format details

akin to humans, likely due to image encoding constraints. Upon meticulous scrutiny of GPT-4V's outputs, a discernible trend emerges: it tends to follow imaginary rules to identify locations featuring specific formats. For instance, under the vanilla experiment, GPT-4V consistently identifies the three-line area spanning from "1,1: 1,6", "2,1:2,6", and "6,1:6,6" for bottom borders. Similarly, under the address agument condition, it consistently outputs areas such as "A1:F1", "A5:F5", and "A7:F7" representing top borders.

B.4 A case of spreadsheet table detection task

In order to explore the performance of GPT-4V on the spreadsheet table detection task, We will analyze the case in detail.

First, by analyzing Figure. 14a, we can find that the reason why one-shot effect is worse than zero-shot effect is that the example we give always have inevitable biases, which will induce the VLMs to make wrong judgments, even worse than the VLMs' own judgment under zero-shot setting. Furthermore, VLMs have serious hallucination problems so in one-shot experiments settings, there is always a tendency to output example answers as part of the results.

Second, by comparing predictions in Figure. 14b, we can find that the GPT-4V makes an error to directly output the address of the table range, while GPT-4V correctly output the values on the four boundaries of the table. According to the previous experiment results, we learn that the VLMs has poor spatial perception of spreadsheet images, so it's hard for them to infer the address of table ranges accurately. In contrast, VLMs has quite strong OCR capabilities, which allow to decode the cell values on the table boundaries.

(a) Vanilla

(b) ColWidth Adjust

(c) Style Change

Figure 12: An example of GPT-4V on OCR task. Due to space limitations, only the contents of the first 15 cells are shown. The content marked in red indicates the LCS match error prediction.

(a) Vanilla

(b) Address Augment

Figure 13: An example of GPT-4V on visual format recognition task. The content marked in red indicates the wrong predictions.

(a) zeroshot vs. oneshot

(b) four vs. range

Figure 14: An example of GPT-4V on spreadsheet table detection task. The red color represents the wrong predictions.