# Extraction Data From Graphs

**AGH University of Krakow**
Przemysław Maresz
Tomasz Makowski
Wojciech Neuman
maresz@student.agh.edu.pl
makowskit@student.agh.edu.pl
neuman@student.agh.edu.pl
Graph Chat GitHub Repository

June 2025

## Contents

# 1 Literature Review

Extracting numerical data from graphs embedded in scientific publications is a routine yet labor-intensive step in systematic reviews and meta-analyses. Despite the availability of several digitization tools, such as WebPlotDigitizer [5], PlotDigitizer [2], and GraphReader [3], the process often requires manual calibration of axes, point selection, and user supervision. This makes the workflow inefficient, particularly when applied to large-scale studies involving dozens or hundreds of figures.

WebPlotDigitizer is a widely-used semi-automated tool offering support for XY, bar, and even polar plots, but it relies on manual axis setup, which can be time-consuming and prone to user bias. PlotDigitizer offers improved automation through color detection and line/bar tracking but still struggles with low-resolution or cluttered plots. GraphReader introduces OCR support for text and axes, simplifying label interpretation, yet performs poorly on non-standard or artistic graphs [5, 2, 3].

To overcome these limitations, recent research has shifted towards deep learning and vision-language approaches. VizExtract [1], for instance, uses a deep learning pipeline trained on synthetic datasets to identify and extract relationships from scientific charts, achieving an accuracy of approximately 87%. However, the generalization of such systems to real-world figures remains limited due to dataset-specific training.

More recently, systems like ChatGraph [4] have demonstrated the potential of large language models (LLMs) integrated with chart-parsing capabilities. ChatGraph allows users to interact with plots via natural language queries, such as *"What is the y-value at x=3?"*, showcasing a step towards more intuitive and flexible data retrieval. Nevertheless, this tool is still in the research phase and lacks key functionalities such as precise numerical extraction, support for image uploads, and robust batch processing.

These advancements suggest a paradigm shift: from procedural digitization toward conversational and intelligent interpretation of visual data. Vision-language models (VLMs) like GPT-4V, BLIP-2, or Gemini Pro Vision open new possibilities for understanding chart layouts, identifying semantically meaningful elements, and interpreting user questions without predefined templates.

The project described in this report aims to leverage these technologies by fine-tuning a multimodal LLM capable of:

- Understanding chart structure from raw images, including axes, labels, and data points;
- Extracting exact coordinate values based on user queries expressed in natural language;
- Handling diverse chart types, including line plots, bar charts, and ROC curves;
- Providing a user interface to facilitate interactive, scalable, and reproducible data extraction.

This approach not only reduces manual effort but also improves accessibility and reproducibility in meta-analytical workflows. Ultimately, it builds upon the strengths of earlier tools while addressing their limitations through language-based, semantic interaction and enhanced visual understanding.

# 2 LLMs Comparison

## 2.1 Benchmark Dataset Description

To evaluate the performance of various multimodal and vision-language large language models (LLMs), we created a synthetic benchmark dataset composed of three chart types: bar charts, line/polyline plots, and scatter plots. Each chart was procedurally generated using Python scripts with randomized parameters, including axis ranges, styles, and noise levels.

Each image in the dataset is paired with a JSONL metadata file containing ground-truth (x, y) values for each visual element. Prompts were automatically generated to mimic real user queries, such as:

- "What is the value of the 5th bar from the left?",
- "What is the y-value when x = 9?",
- "Find the value of the peak point in the line chart.".

A total of 120 charts were used for evaluation, distributed across:

- **Bar charts** (20): randomized heights, gaps, categories,
- **Line/poly charts** (50): polynomials of degrees 2–4,
- **Scatter plots** (50): sinusoidal + noisy distributions.

The evaluation was carried out by prompting each model with chart-image + natural language question pairs and recording their numeric responses. The predictions were then compared to ground truth using metrics such as absolute error and relative error.

## 2.2 Model Selection

The following vision-capable LLMs were selected for evaluation:

- **Gemini 1.5 Flash (Google)** - part of Google's family of multimodal large language models (LLMs), optimized for low-latency inference ("Flash" variant). It delivers fast performance while maintaining strong capabilities in image understanding, diagram interpretation, and layout reasoning. However, its precision in extracting and interpreting numerical data from visual sources (i.e., chart numeracy) remains limited, which affects its usefulness in tasks requiring accurate quantitative analysis.
- **Gemini 2.5 Flash (Google)** - the latest generation in the Gemini line (as of 2025), enhancing its predecessor with significantly improved optical character recognition (OCR), document structure parsing (layout understanding), and numeric grounding. This model achieves better accuracy in processing visual and semi-structured data while retaining high-speed inference suitable for production environments.
- **Gemma 3 12B IT (Open/Google)** - an instruction-tuned, open-weight large language model from Google with 12 billion parameters, designed for local deployment. It is a text-only model without native vision capabilities. In visual tasks, it was tested using pre-parsed image features (e.g., via BLIP or CLIP), passed as textual prompts. The model demonstrates strong performance in interpreting visual contexts through descriptive text, but lacks direct image comprehension.

- **Gemma 3 4B IT** - a smaller, 4-billion-parameter version of the Gemma 3 model, also instruction-tuned and suitable for local deployment on consumer-grade hardware (e.g., 8–16 GB VRAM GPUs). Like the 12B version, it lacks native vision processing and relies on textual representations of images. Its visual reasoning capabilities are therefore limited but sufficient for basic descriptive tasks.
- **Qwen2.5 VL 3B Instruct (Alibaba)** - a compact multimodal model from Alibaba's Qwen 2.5 series, with 3 billion parameters and native vision-language support. Despite its small size, it delivers surprisingly strong performance in tasks such as OCR, layout analysis, and rapid visual reasoning. Its primary limitation lies in lower regression and numeric precision (e.g., exact value extraction from charts), although this is balanced by its fast and efficient local inference capabilities.

## 2.3 Evaluation Table

Table 1: Model Performance Comparison on Synthetic Chart Dataset

| Model | MAE | Abs. Errors' Median | Rel. Error | Latency (s) |
|---|---|---|---|---|
| Gemini 1.5 Flash | 6.950 | 0.990 | 0.082 | 1.584 |
| Gemini 2.5 Flash | 5.666 | 2.310 | 0.138 | 5.582 |
| Gemma 3 12B IT | 16.866 | 3.425 | 0.134 | 84.050 |
| Gemma 3 4B IT | 128.953 | 8.275 | 0.585 | 44.420 |
| Qwen2.5 VL 3B Instruct | 88.520 | 6.300 | 0.480 | 82.951 |

## 2.4 Preliminary Observations

Based on the comparative results presented in Table 1, Gemini 1.5 Flash emerges as the top-performing model in terms of overall accuracy and efficiency. Despite not achieving the lowest mean absolute error (MAE), it records the lowest relative error (8.2%) and fastest inference time (1.58 seconds) among all evaluated models. This makes it the most practically useful model when precision and responsiveness are both required.

Interestingly, Gemini 1.5 Flash reports a higher MAE (6.95) compared to Gemini 2.5 Flash (5.67), yet it shows a much lower relative error. This apparent discrepancy can be attributed to the distribution of errors: while the MAE reflects the average magnitude of all absolute errors (including outliers), the relative error measures the proportional error with respect to ground truth values. The higher MAE is likely caused by a few large deviations, whereas the lowest median absolute error (0.99) confirms that the majority of predictions are accurate and only a few outliers — likely due to hallucinations — significantly impact the mean.

In contrast, local models such as Gemma 3 (12B, 4B) and Qwen2.5 VL demonstrate significantly higher inference times (44–84 seconds) and generally weaker accuracy metrics. These models were evaluated on a consumer-grade laptop without access to GPU clusters or high-performance cloud accelerators, which explains the performance gap when compared to the Gemini models running in a cloud environment.

Despite their limitations, local models offer distinct advantages, especially regarding privacy, deployment flexibility, and potential for fine-tuning. Being open-weight and instruction-tuned, they can be further adapted to specific chart types or domain-specific numeric reasoning tasks. In particular, Qwen2.5 VL, as a compact multimodal model with strong OCR capabilities, represents a promising baseline for future lightweight deployments.

Moreover, these findings strongly suggest that fine-tuning can play a critical role in closing the performance gap between open local models and closed-source commercial systems. A targeted fine-tuning

procedure using a curated dataset of synthetic and real-world scientific charts — with rich variation in styles, axis formats, and numeric ranges — could enable models like Qwen2.5 VL to internalize chart semantics, axis decoding, and visual-to-symbolic alignment far more effectively. In this context, even modest-sized models could reach state-of-the-art accuracy for narrow but practically important tasks such as extracting exact values from visual data.

Finally, we note that chart understanding remains a challenging task in multimodal AI, particularly when precise numerical extraction is required. Top-performing models successfully combine layout awareness, text recognition, and grounded numeric reasoning. Furthermore, the discrepancy between mean and median errors highlights the importance of reporting multiple error metrics, as large outliers can distort average performance and obscure the model's typical behavior.

This analysis reinforces the need for continued research in visually grounded numerical reasoning, as well as the value of domain-specific benchmarks and fine-tuning pipelines that reflect the complexity of real-world use cases.

# 3 Model Fine-Tuning

## 3.1 Motivation and Constraints

To enable precise extraction of numerical data from scientific charts via natural language prompts, we experimented with fine-tuning a vision-language model (VLM) on synthetic chart datasets. However, the initial phase of this project was constrained by the lack of access to GPU-enabled infrastructure. Despite attempts to obtain computational resources via academic platforms such as AGH Cyfronet, no allocation was granted. As a result, the initial experiments were limited to CPU-only environments.

## 3.2 Initial Setup: CPU-based Fine-Tuning

We selected `Salesforce/blip2-flan-t5-xl`, a multimodal encoder-decoder model capable of handling both image and text inputs. This model was chosen due to its compatibility with CPU-based fine-tuning, and its support for vision-language inputs without requiring GPU-based optimizations such as quantization.

The training pipeline was built using Hugging Face Transformers and PyTorch. Approximately 180 synthetic charts were generated across three types: bar plots, line/polyline plots, and scatter plots. Each example included a `.png` image and a corresponding `.jsonl` metadata file with the chart values. The prompts followed a templated structure (e.g., *"What is the y-value for $x = 7$ in the bar chart?"*). Fine-tuning was performed over 3 epochs with a batch size of 1.

Despite this effort, results were suboptimal. The model tended to memorize a constant output (e.g., always predicting 76.5), likely due to:

- a small dataset size (only 180 samples),
- low variance in y-values,
- and an abstract prompt format that failed to guide attention to relevant chart regions.

## 3.3 GPU-based Testing and Alternative Models

Later in the project, we conducted comparative tests on GPU-enabled machines using two lightweight models: `gemma-3-4b-it` and `Qwen2.5-VL-3b`. Although these models offered faster inference and more efficient vision-language encoding, their performance remained similar to the base (pretrained) models. This was likely due to insufficient training data and a limitations concerning the computing power.

### 3.4 Findings and Recommendations

The results indicate that fine-tuning large multimodal models requires:

- Substantial computational resources (ideally GPU with CUDA),
- A significantly larger dataset (3,000–5,000 examples),
- Better-designed prompts leveraging visual semantics (e.g., *"3rd bar from the left"* rather than *"x = 3"*),
- Possibly modular architectures (e.g., CLIP encoder + Gemma decoder or OCR + LLM).

Future efforts should focus on using GPU-backed environments and exploring newer models specialized in document understanding such as LLaVA or MiniGPT-4. With improved data diversity and computational capacity, meaningful gains in accuracy and generalization are expected.

# 4 Human vs. LLM - results comparison

In this section, we compare the performance of human predictions against the **Gemini 1.5 Flash** model (best performing one) in extracting values from different types of charts. The comparison focuses on two key metrics: mean absolute error (MAE) and mean relative error (expressed as a percentage).

Overall, the mean absolute error for human predictions was only **1.88**, compared to the MAE of **6.95** for Gemini 1.5 Flash. Similarly, the mean relative error for Gemini 1.5 Flash was **8.2%**, while for human observations it was just **3.65%**. This outcome is understandable: when charts are well-designed, extracting precise information is relatively straightforward for human observers.

Table 2 presents the detailed breakdown of errors across different chart types.

| Chart Type | MAE (LLM) | MAE (Human) | Rel. Error (LLM, %) | Rel. Error (Human, %) |
|---|---|---|---|---|
| Bar | 5.03 | 0.35 | 9.34 | 0.69 |
| Line | 13.63 | 3.82 | 13.86 | 6.97 |
| Scatter | 1.34 | 0.67 | 2.24 | 1.72 |

Table 2: Comparison of human and Gemini 1.5 Flash (LLM) performance on different chart types

As shown, the largest discrepancy between human and LLM performance was observed for **bar charts**. The mean relative error for Gemini 1.5 Flash on bar charts was over **13 times higher** than that of human predictions (9.34% compared to 0.69%). This likely comes from the fact that bar charts have a fixed x-axis, which makes it very easy for humans to read corresponding y-axis values. The LLM, however, appears to struggle more in these scenarios.

In the case of **line charts**, both human and LLM errors were higher, with relative errors of 6.97% and 13.86% respectively. This can be attributed to the scale of the y-axis being significantly larger than in other chart types, making it more challenging to accurately estimate where a point lies in relation to the x-axis.

For **scatter plots**, the LLM performed comparably to humans, with a mean relative error of 2.24% versus 1.72%. This suggests that in scenarios where individual data points are isolated and clear, the LLM can approach human-level accuracy.

Importantly, the use of an LLM like Gemini 1.5 Flash offers the possibility of automating the process of extracting values from charts. While humans may achieve higher accuracy in most of the cases, the automation enabled by the LLM means that large volumes of data can be processed rapidly and without manual effort. This can be particularly valuable in applications requiring extraction from large numbers of charts or in real-time settings where human input would not be feasible.

# 5 User Interface

This section describes the user interface and functionalities of the Graph Chat application, which is a Streamlit-based tool designed to extract information from various types of charts using the Gemini Large Language Model. It allows users to upload image files of charts and then interact with the LLM to ask questions about their content or to perform automated data extraction.

## Key Features:

- **Gemini API Key Configuration:** Users must first configure their Gemini API Key in the sidebar. This key is essential for the application to communicate with the Gemini model.

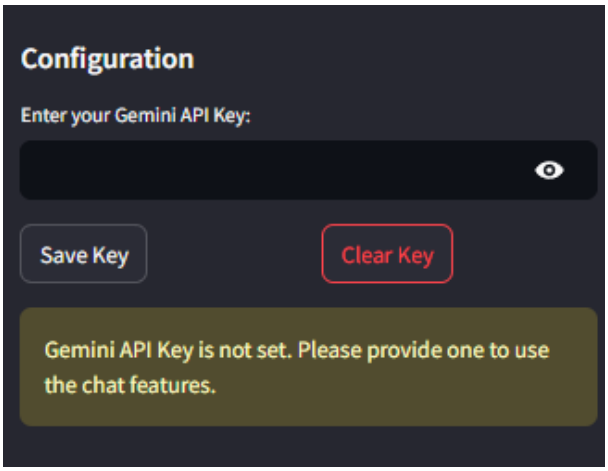


Figure 1: Gemini API Key Configuration

- **Chart Upload:** The application provides a user-friendly interface to upload chart images. Supported formats include PNG, JPG, JPEG, and WEBP, with a limit of 200MB per file.

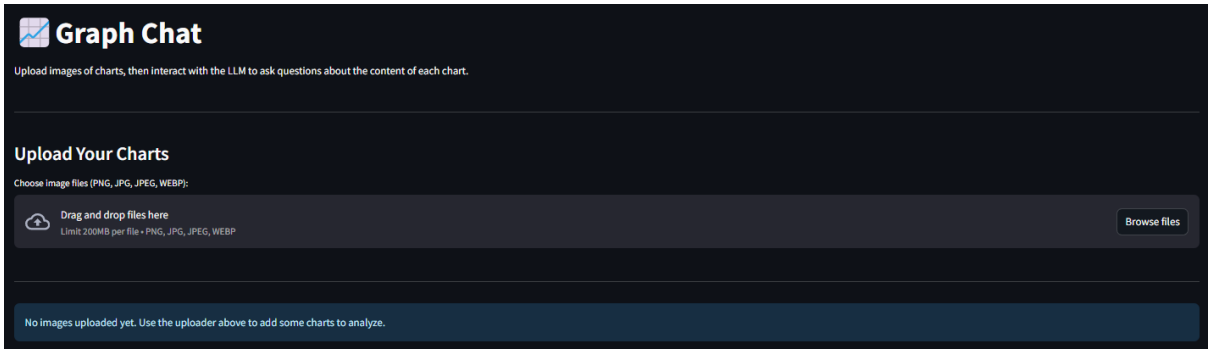


Figure 2: Chart Upload Interface

- **Interactive Analysis:** Once a chart is uploaded, it appears in an expandable section, allowing for detailed analysis. Users can interact with the Gemini model in two primary ways:
  - **Manual Question:** Users can type in custom questions about the uploaded chart, and Gemini will provide a direct response based on the chart's content.
  - **Automated Analysis Methods:** The application offers pre-defined analysis methods for specific chart types:

* **Line Chart: Point Detection:** Extracts a specified number of (x, y) coordinates from line charts, evenly distributed along the X-axis.
* **Bar Chart: Value Extraction:** Identifies bar labels and their corresponding numerical values from bar charts.
* **Scatter Plot: Point Extraction:** Extracts a specified number of (x, y) coordinates from scatter plots.

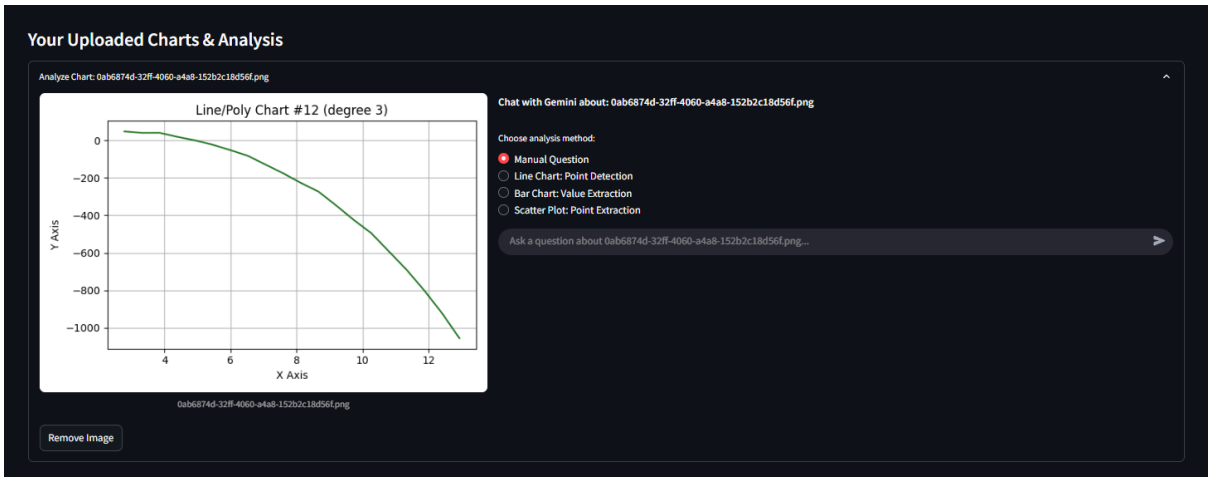


Figure 3: Analysis Methods Selection

- **Chat History:** The application maintains a chat log for each uploaded image, displaying both user queries and Gemini's responses, providing a clear history of the analysis performed.

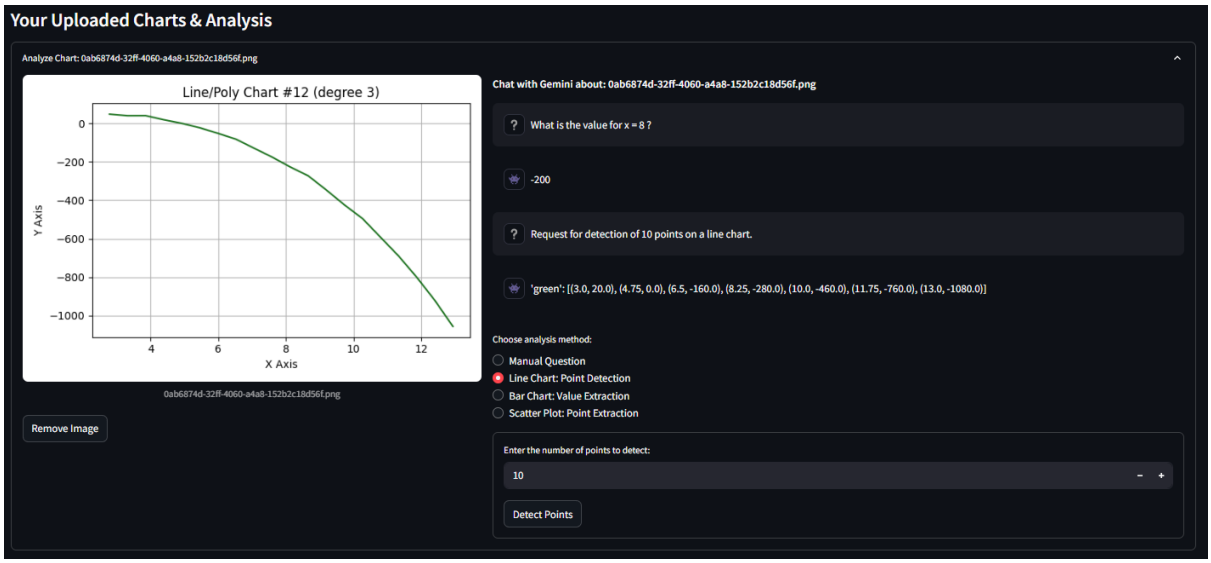


Figure 4: Chat History and Point Detection Example

## How to Use:

1. **Set API Key:** On the left sidebar, enter your Gemini API Key in the "Configuration" section and click "Save Key."
2. **Upload Charts:** In the main content area, use the "Upload Your Charts" section to drag and drop or browse for your chart image files.
3. **Analyze Charts:** Each uploaded chart will appear in a new expandable section.
   - To ask a custom question, select "Manual Question" and type your query in the chat input field.
   - To use automated extraction, select the appropriate analysis method (e.g., "Line Chart: Point Detection"), provide any necessary input (like the number of points), and click the corresponding detection/extraction button.
4. **Review Responses:** Gemini's responses will appear in the chat log specific to each chart.
5. **Remove Images:** If desired, charts can be removed from the analysis view by clicking the "Remove Image" button within each chart's section.

# References

[1] D. Decatur et al. “VizExtract: Automatic Relation Extraction from Data Visualizations”. In: *arXiv preprint arXiv:2112.03485* (2021). URL: https://arxiv.org/abs/2112.03485.

[2] DeveloperX. *PlotDigitizer*. https://plotdigitizer.com. Accessed: 2025-06-22.

[3] *GraphReader*. https://www.graphreader.com. Accessed: 2025-06-22.

[4] Fareed Khan et al. “Talking Graphs: Your Data Speaks Up”. In: *arXiv preprint arXiv:2401.12672* (2024). URL: https://arxiv.org/abs/2401.12672.

[5] Ankit Rohatgi. *WebPlotDigitizer*. https://apps.automeris.io/wpd/. Accessed: 2025-06-22.