Self-Ensembling Vision-Language Models for Chart Data Extraction

Anonymous ACL submission

Abstract

Charts effectively convey quantitative information, but the underlying data are often locked in image form, hindering reuse and analysis. Manually digitizing charts is time-consuming and error-prone, motivating automatic chart-to-table extraction. Recent approaches use specialized vision-language models (VLMs), yet performance still lags on charts with many datapoints or substantial stylistic variation. We propose a VLM self-ensembling method that repeatedly samples multiple tabular outputs from the same VLM for a fixed chart image and aggregates them at the level of individual table cells. We align candidate tables and take per-cell medians over numerical values to produce a more accurate consensus table. Our method also includes convergence detection to stop sampling once the aggregated table stabilizes, and uncertainty estimation based on dispersion across samples to help users assess extraction reliability. Because existing chart extraction benchmarks contain relatively simple plots with limited room for improvement, we introduce WB-ChartExtract, a new benchmark built from World Bank data with more complex and stylistically diverse charts; on average, its charts contain 7× more datapoints than those in the ChartQA benchmark. Across both ChartQA and WB-ChartExtract, our approach improves extraction accuracy over single-pass VLM outputs, yielding up to 23% relative improvement on WB-ChartExtract after ensembling. More broadly, our method helps unlock tabular data previously siloed in chart images, enabling downstream analysis and reuse.

1 Introduction

Charts and plots are a primary medium for communicating quantitative information in scientific articles, policy reports, news stories, and online dashboards. Unlike tables, charts typically appear only as rasterized images in PDFs or on the web, making the underlying numbers difficult to access: they cannot be searched, joined with other sources, or reused for downstream analysis. Recovering these data at scale is increasingly important for meta-analysis, model validation, and real-time monitoring, where source data are often missing or inaccessible.

Manual digitization with tools such as WebPlotDigitizer (Rohatgi) is possible but slow and error-prone, especially for large corpora or visually complex figures. This has motivated automatic chart-to-table extraction, where a system reconstructs a structured table of datapoints from a chart image. Early systems (Leow et al., 2005; Savva et al., 2011) used engineered pipelines combining computer vision, optical character recognition (OCR), and heuristic reasoning about axes and legends. More recent work (Zhang et al., 2024; Chen et al., 2024; Xia et al., 2025) leverages vision-language models (VLMs) that can directly "read" charts and emit a textual table representation. While VLM-based methods simplify the stack and have improved performance and generalization on several benchmarks (Masry et al., 2022; Methani et al., 2020), they still struggle with dense datapoints, multiple overlaid series, or idiosyncratic labeling.

These failure modes reflect a broader pattern in generative models: a single forward pass can be brittle (Wang et al., 2023; Snell et al., 2024). When prompted multiple times on the same chart, a VLM can return noticeably different tables—dropping or adding points or entire series, or misreading values—yet these outputs are often partially correct and complementary. We verify this empirically on our benchmark: running a chart-specialized VLM (TinyChart) 20 times per chart on WB-ChartExtract, 99.4% of charts produced at least one differing extraction, and 49.0% had different table dimensions across runs. Existing chart extraction methods typically ignore this variability, relying on a single "best guess," and they provide little signal about extraction reliability, which is

problematic for realistic, cluttered figures.

We propose a complementary approach that embraces VLM stochasticity by ensembling outputs at the level of individual table cells. Our method is model-agnostic, can be layered on top of existing chart extraction systems, and applies to any chart type the underlying model can handle. Concretely, we repeatedly prompt a VLM on the same chart image, align candidate tables at the cell level, and compute per-cell medians over numerical values to form a consensus table. We also introduce a convergence detection mechanism to stop sampling once additional predictions are unlikely to change the ensemble, and an uncertainty estimate based on variability across samples to quantify extraction reliability.

Progress on chart extraction also depends on benchmarks with challenging, stylistically varied charts. Yet widely used datasets such as ChartQA offer increasingly limited room for improvement: state-of-the-art VLMs already score highly, in part because many plots are simple, with few datapoints, limited stylistic variation, and numbers printed directly on the chart. To provide a more demanding testbed, we introduce WB-ChartExtract, a benchmark of synthetic charts built from real World Bank data that contain on average 7× more datapoints than ChartQA charts, span four chart types and four rendering libraries, and include no printed value labels (§4.1).

We evaluate our self-ensembling approach on both ChartQA and WB-ChartExtract. Across datasets and underlying models, ensembling consistently improves accuracy over single-pass VLM extraction, with gains of up to 23% relative on WB-ChartExtract. We further analyze which error types our approach corrects and which remain challenging. Importantly, we show that convergence detection is well calibrated and supports a controllable trade-off between computational cost and marginal accuracy gains via practical early stopping. Finally, we demonstrate that our uncertainty estimate is empirically inversely correlated with extraction accuracy.

Our contributions are fourfold:

1. A model-agnostic self-ensembling method for chart data extraction.

2. A convergence detection mechanism that identifies when additional samples are unlikely to change the aggregated result.

3. Ensemble uncertainty estimates to help users gauge extraction reliability.

4. WB-ChartExtract, a new benchmark constructed from real-world data that is, to our knowledge, the most challenging to date.

Together, these contributions move chart-to-table extraction closer to real-world deployment.

2 Background and Related Work

Chart data extraction (also called *chart-to-table* or *derendering*) aims to recover the numerical values in a chart image along with their semantic structure (e.g., axis values and series names). Early systems decomposed the task into multi-stage pipelines combining computer vision, OCR, and heuristics to map pixels to values (Leow et al., 2005; Savva et al., 2011; Mishchenko and Vassilieva, 2011; Al-Zaidy and Giles, 2017). While effective when figures match anticipated templates, these pipelines are often brittle under modest stylistic variation (e.g., layout, font, or color).

Recent work instead uses VLMs to generate a textual table representation directly from the chart image (Liu et al., 2023a,b; Han et al., 2023; Meng et al., 2024; Chen et al., 2024; Zhang et al., 2024; Xia et al., 2025). However, VLMs often struggle with fine-grained visual distinctions (Liu et al., 2025; Chen et al., 2025), including in chart settings (Razeghi et al., 2024). Consequently, one-pass generation remains error-prone on complex charts: models may omit datapoints, hallucinate values, misread axes, or inconsistently name series across runs. Our approach is designed to sit atop either specialized chart extraction models or generalist VLMs, repeatedly querying the underlying model and aggregating outputs to improve robustness.

Benchmarks. Common chart-extraction datasets include ChartQA (Masry et al., 2022), PlotQA (Methani et al., 2020), and ChartX (Xia et al., 2025). ChartQA contains real-world charts, but many instances are visually simple (e.g., values printed directly on bars). PlotQA and ChartX provide large-scale synthetic charts with broader chart-type coverage, but limited stylistic diversity and complexity. In contrast, WB-ChartExtract does not print values directly on marks, includes longer series and multiple series per chart, spans four chart types and four rendering libraries, and exhibits substantial stylistic variance.

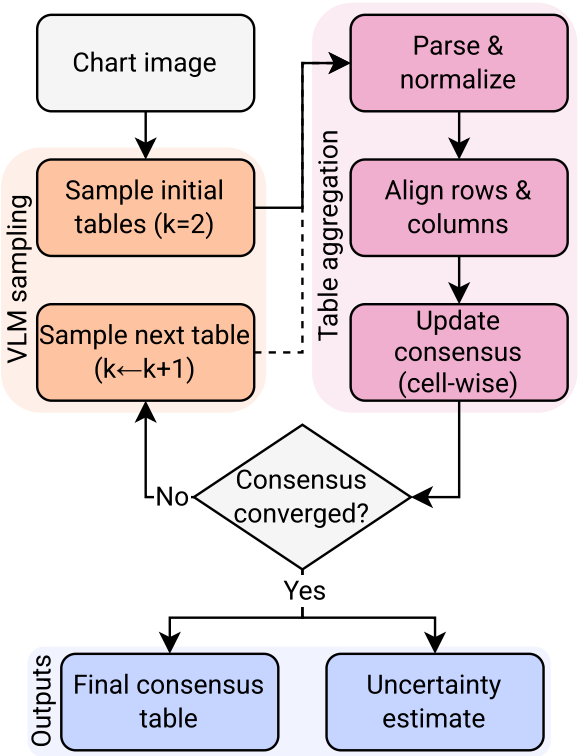

Figure 1: Iterative self-ensembling for chart-to-table extraction. We sample tables from a base VLM, parse/normalize and align rows/columns, and update a cell-wise consensus until convergence, producing a final table and uncertainty estimate based on the median absolute deviation (MAD).

3 Methodology

Given a chart image, our goal is to recover the underlying table of numeric datapoints. Our approach is *model-agnostic*: we repeatedly prompt a base VLM to produce tabular outputs, then align and ensemble these outputs at the level of individual table cells to obtain a consensus table. Compared to single-pass extraction, self-ensembling suppresses outlier predictions (e.g., hallucinated or missing datapoints) and yields a natural uncertainty signal. Figure 1 provides an end-to-end overview.

3.1 Base Model Sampling

For each chart image, we query a base VLM (by default, Llama 4 Scout (Meta AI, 2025)[1]) with the following prompt:

> Here is an image of a chart. Please extract the numerical data it represents and return it in TSV (tab-separated values) format with appropriate headers. Copy the headers exactly as they are in the image.
>
> IMPORTANT: For the TSV, use tab (`\t`) as the separator. Remember: The sole output should be the TSV table surrounded by "'tsv "'. Nothing else.

We sample at temperature T (a hyperparameter; Section 4.7). For each image, we generate two initial tables, parse, align, and aggregate them (as described below). We continue sampling and updating the ensemble until either: (i) for *patience*$= 2$ consecutive updates, at least *coverage*$= 95\%$ of aggregated cell values change by no more than a relative *tolerance*$= 1\%$ between successive ensemble updates (sensitivity analysis in Appendix D); or (ii) a maximum number of samples $K_{\max}$ is reached. We use $K_{\max} = 20$ by default based on our computational budget, but this can be adjusted (Section 4.6.2).

3.2 Parsing and Normalization

Each sample is a TSV-like text block and may contain formatting errors (e.g., inconsistent column counts). We first parse as TSV; if parsing fails due to ragged rows, we repair by padding or truncating rows to the modal column count.

After parsing, we treat (i) the first row as column headers, (ii) the first column as row headers (index), and (iii) remaining cells as numeric values. We convert value cells to numeric, stripping common artifacts (commas, currency symbols, percent signs); non-numeric strings (including empty cells) are treated as missing and set to `nan`. If needed, we transpose the table to enforce more rows than columns, which simplifies alignment for time-on-x charts with multiple series.

3.3 Row and Column Alignment via Clustering

Across samples, tables may differ in row/column order and labels may be noisy (e.g., spelling, abbreviations). We therefore align tables by clustering row and column labels (independently) across samples to form canonical row and column groups.

We measure label similarity with ANLS (Biten et al., 2019a, average normalized Levenshtein similarity), yielding $\mathrm{sim}(\ell_1, \ell_2) \in [0, 1]$ (Liu et al., 2023a; Biten et al., 2019b). Clusters must satisfy: (i) grouped labels are similar under sim, and (ii) no cluster contains two labels from the same sampled table (to avoid merging within-table duplicates).

We use greedy clustering with threshold $\tau = 0.5$ (following Liu et al. (2023a); Biten et al. (2019b)).

[1]We developed primarily with Llama 4 Scout due to its efficiency and low inference cost on Groq (Groq, Inc., 2025), but the method is model-agnostic.

Each cluster c is represented by $\mathrm{rep}(c)$, the first label assigned to c:

1. Initialize an empty set of clusters.
2. For each label ℓ, assign it to an existing cluster c if (i) $\mathrm{sim}(\ell, \mathrm{rep}(c)) \geq \tau$ and (ii) c contains no label from the same sampled table as ℓ. If multiple clusters qualify, choose the one maximizing $\mathrm{sim}(\ell, \mathrm{rep}(c))$.
3. If no cluster qualifies, create a new cluster containing ℓ.

We then prune clusters that appear in fewer than 20% of tables (default; ablation in Appendix E) to remove spurious labels that would otherwise introduce extra rows/columns. For each remaining cluster, we define its canonical label as the most frequent label string in the cluster (ties broken at random) and use these canonical labels as the aggregated table's row/column names.

3.4 Cell-Wise Aggregation

Let $\mathcal{R} = \{r_1, \ldots, r_{|\mathcal{R}|}\}$ and $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ be the retained row and column clusters. For each aligned cell $(r, c) \in \mathcal{R} \times \mathcal{C}$, we collect all numeric predictions from sampled tables whose row label falls in r and column label falls in c. Let $V_{r,c}$ denote this set (ignoring nan). We aggregate as $\hat{y}_{r,c} = \mathrm{median}(V_{r,c})$ (ablation in Appendix C); if $V_{r,c}$ is empty, we output nan. The final table orders row and column clusters by lexicographic sort of their canonical labels.

3.5 Uncertainty Estimation

Self-ensembling yields a natural uncertainty signal from disagreement across samples: consistent values imply reliable medians, while high variability indicates brittle extraction and additional verification (or more samples) may be warranted.

For each aligned cell (r, c), we compute uncertainty as the median absolute deviation (MAD) over $V_{r,c}$, normalized by the magnitude of the ensembled value:

$$u_{r,c} = \frac{\mathrm{MAD}_{r,c}}{|\hat{y}_{r,c}|},$$
$$\text{where} \quad \mathrm{MAD}_{r,c} = \mathrm{median}_{v \in V_{r,c}} |v - \hat{y}_{r,c}| \,.$$

To avoid division by zero, we exclude cells with $\hat{y}_{r,c} = 0$ when computing table-level uncertainty summaries. Relative MAD is scale-invariant, making uncertainty comparable across charts and across cells with different magnitudes.

To obtain a single uncertainty score per chart, we summarize $\{u_{r,c}\}$ over cells with non-missing predictions and $\hat{y}_{r,c} \neq 0$ using: U_{med} (median), U_{mean} (mean), and U_{max} (maximum). These capture complementary failure modes: U_{med} reflects typical uncertainty, U_{mean} reflects overall dispersion, and U_{max} flags catastrophic disagreement in any cell.

4 Experiments

4.1 Datasets

ChartQA. We evaluate on ChartQA, the most widely used benchmark for chart data extraction. Its test set contains 1,509 real-world chart images, but has notable limitations: charts are often visually simple, with relatively few datapoints, limited stylistic diversity, and values frequently printed directly on the chart, allowing models to rely on text rather than chart geometry and axes. As a result, the benchmark is somewhat saturated. ChartQA's ground-truth tables are also often noisy, reducing evaluation reliability.

WB-ChartExtract. To address these limitations, we introduce WB-ChartExtract, a new benchmark constructed from World Bank Data Bank time series (The World Bank) spanning 52 indicators, 218 countries, and 65 years. We prune series with more than half missing values or where there are missing values in the interior (i.e., not just at the ends) of the series. We then create 1,000 datasets by repeatedly sampling one indicator and 2–3 countries at random and collecting the corresponding series, using each underlying series at most once. Each dataset is rendered as one of four chart types—line, area, grouped bar, or stacked bar—using one of four plotting libraries: Matplotlib (Hunter, 2007), Seaborn, Plotly, or Bokeh. Chart type and library are assigned uniformly at random, yielding 62–63 charts per combination. Within each chart, we randomize font family and size, color palette, grid presence and style, line styles, markers, transparency, and figure size. This yields 1,000 charts with diverse visuals, chart types, and rendering styles, along with clean ground-truth tables. Its difficulty arises from several complementary factors rather than any single one: higher data density (on average 7× more datapoints than ChartQA, particularly for line and area charts), the absence of printed value labels (removing the OCR shortcut available on many

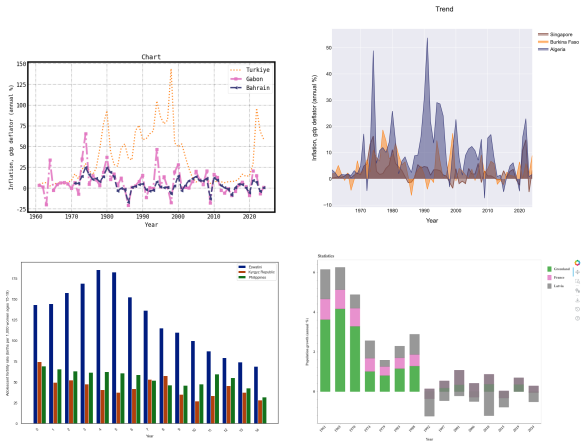

Figure 2: Examples of charts in WB-ChartExtract, showing four chart types rendered by four different libraries (clockwise from top-left: line/Matplotlib, area/Plotly, stacked bar/Bokeh, grouped bar/Seaborn).

ChartQA plots), multiple overlaid series, chart-type diversity requiring different visual inference strategies, and cross-library stylistic variation. Figure 2 shows example charts.

Synthetic generation is a deliberate choice: it yields clean ground-truth tables at scale, which is otherwise difficult to obtain for real-world charts since the underlying data is rarely available—reflected in the noisy ChartQA annotations noted above. Existing synthetic benchmarks do not fill this gap, as PlotQA (Methani et al., 2020) and FigureQA (Kahou et al., 2017) target question answering rather than structured extraction, with fewer series, shorter series lengths, and fewer datapoints per chart.

To ensure full reproducibility, we will release the complete benchmark—all chart images and clean ground-truth tables (under CC BY 4.0; World Bank data is CC BY 4.0)—together with our chart-generation, extraction, ensembling, and evaluation code and configuration files.[2]

4.2 Evaluation Metric

We evaluate using *Relative Mapping Similarity* (RMS_{F1}) (Liu et al., 2023a), which treats each table as an unordered set of (row header, column header, value) triples and computes a minimum-cost bipartite matching between predicted and ground-truth entries. Per-entry similarity multiplies a header score (one minus thresholded normalized Levenshtein distance, threshold τ) with a value score (one minus clipped relative error, threshold θ). We report the harmonic mean of mapping-level precision and recall, set $\tau = 0.5$ and $\theta = 0.1$ following Liu et al. (2023a), and take the maximum over the predicted table and its transpose for transposition invariance. Full definition in Appendix A.

4.3 Main Results

We compare against single-pass predictions from three chart-specialized models (OneChart (Chen et al., 2024), TinyChart (Zhang et al., 2024), DePlot (Liu et al., 2023a)), three open-source VLMs (Qwen3-VL 235B A22B Instruct (Bai et al., 2025), Llama 4 Scout (Meta AI, 2025), Seed 1.6 Flash (ByteDance Seed Team, 2025)), and three closed-source VLMs (GPT-5.1 (OpenAI, 2025), Claude Opus 4.6 (Anthropic, 2026), Gemini 2.5 Pro (Comanici et al., 2025)). To demonstrate model-agnosticism, we apply self-ensembling on top of TinyChart, DePlot, Qwen3-VL, Llama 4 Scout, and Seed 1.6 Flash, a spread that spans chart-specialized models and general-purpose open-source VLMs. The closed-source frontier models (GPT-5.1, Claude Opus 4.6, Gemini 2.5 Pro) serve as strong single-pass reference points that characterize the difficulty ceiling of each benchmark. For convergence detection, we use default parameters with a tolerance of 1%.

Table 1 reports RMS_{F1} on ChartQA and WB-ChartExtract. Single-pass performance on WB-ChartExtract spans a wide range (23.06–87.83), reflecting the benchmark's difficulty and diversity. Gemini 2.5 Pro is the strongest general-purpose model on WB-ChartExtract (TinyChart leads ChartQA at 95.20 thanks to in-domain fine-tuning, as discussed below; Qwen3-VL leads among general-purpose models on ChartQA at 91.43), achieving 87.83 on WB-ChartExtract, while weaker models show a large gap between the two.

The reversal between TinyChart and the general-purpose models across the two benchmarks reflects training-data alignment: TinyChart is fine-tuned on ChartQA's training split, inflating its in-distribution score, whereas the general-purpose models have no exposure to either benchmark. On WB-ChartExtract, where no model has seen the underlying data and the benchmark spans four chart types rendered by four libraries, the strongest general-purpose models' broader visual reasoning yields the best single-pass results. This further supports WB-ChartExtract as a complementary bench-

[2]Code and data are anonymized for review and will be released publicly upon acceptance.

Method	ChartQA	WB-ChartExtract
OneChart	35.93	26.49
DePlot	88.32	23.06
+ Self-ens. (ours)	89.02	24.57
TinyChart	95.20	28.61
+ Self-ens. (ours)	95.28	29.37
Qwen3-VL	91.43	52.91
+ Self-ens. (ours)	93.21	56.23
Seed 1.6 Flash	81.62	35.08
+ Self-ens. (ours)	87.04	43.17
Llama 4 Scout	75.13	30.41
+ Self-ens. (ours)	77.10	33.14
GPT-5.1	84.80	51.26
Claude Opus 4.6	87.71	60.99
Gemini 2.5 Pro	88.23	87.83

Table 1: Main results (RMS_{F1}; higher is better). "Self-ens." denotes our self-ensembling procedure applied on top of a base model's direct predictions.

Method	ChartQA	WB-ChartExtract
OneChart	$0.00	$0.00
TinyChart	$0.00	$0.00
+ Self-ens.	$0.00	$0.00
DePlot	$0.00	$0.00
+ Self-ens.	$0.00	$0.00
Qwen3-VL	$0.42	$0.63
+ Self-ens.	$1.78	$9.03
Seed 1.6 Flash	$0.76	$1.06
+ Self-ens.	$5.30	$14.63
Llama 4 Scout	$0.24	$0.28
+ Self-ens.	$2.29	$4.47
GPT-5.1	$2.93	$4.55
Claude Opus 4.6	$32.13	$39.17
Gemini 2.5 Pro	$7.04	$11.09

Table 2: Total API inference cost (USD) per benchmark.

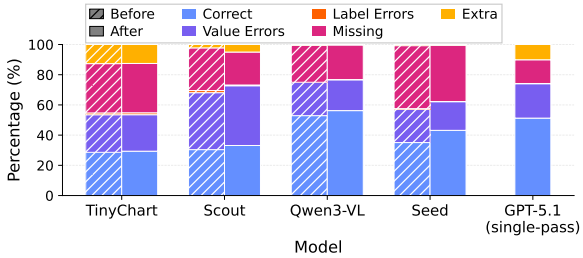

Figure 3: Error-type breakdown on WB-ChartExtract for each base model before vs. after self-ensembling.

mark that probes general chart understanding rather than benchmark-specific fit.

Self-ensembling consistently improves over single-pass on WB-ChartExtract across all five base models: Qwen3-VL improves from 52.91 to 56.23 (+3.32), Seed 1.6 Flash from 35.08 to 43.17 (+8.09), Llama 4 Scout from 30.41 to 33.14 (+2.73), DePlot from 23.06 to 24.57 (+1.51), and TinyChart from 28.61 to 29.37 (+0.76). The relative gains span roughly 2.7% (TinyChart) to 23.1% (Seed 1.6 Flash), meaningful improvements given the benchmark's difficulty. On ChartQA, self-ensembling improves the three general-purpose VLMs (Seed 1.6 Flash 81.62→87.04, Llama 4 Scout 75.13→77.10, Qwen3-VL 91.43→93.21) as well as the DePlot specialist (88.32→89.02), while the gain is essentially zero for TinyChart (95.20→95.28, +0.08), which is already saturated on its in-domain training distribution and leaves no headroom for ensembling to recover. These improvements are corroborated by two additional value-centric metrics, RNSS and RD, reported in Appendix F.

Cost–accuracy tradeoff. Table 2 reports API cost per benchmark. Self-ensembling increases cost by roughly 4–16× depending on the model and dataset, but absolute costs remain modest—under $15 for an entire benchmark in every ensembled configuration. The convergence detection mechanism (§4.6.1) further mitigates this by stopping early for easier samples. Appendix G (Figure 5) visualizes this accuracy–cost tradeoff, showing that self-ensembling shifts every base model up and to the right, trading a higher per-image cost for consistent accuracy gains.

4.4 Error Analysis

To understand how self-ensembling improves RMS_{F1}, we decompose each model's output into **Correct** (example-level F1) plus four error categories that together sum to 100%: **Value Errors** (numeric mismatch under the relative error distance), **Label Errors** (imperfect label matching under ANLS), **Missing** (ground-truth datapoints with no matched prediction), and **Extra** (predicted datapoints with no matched ground-truth). Figure 3 shows the average breakdown before and after ensembling on WB-ChartExtract for TinyChart, Llama 4 Scout, Qwen3-VL, and Seed 1.6 Flash, together with the single-pass breakdown for GPT-5.1.

Label Errors are negligible (≤1.4%), and errors are dominated by **Value Errors** (19–40%) and **Missing** (22–42%). Ensembling improves **Correct**

for all four base models shown (by 0.8–8.1 pp), but the mechanism varies by model. For Qwen3-VL and Seed 1.6 Flash, gains come primarily from reduced **Value Errors** (1.6–3.1 pp) together with fewer **Missing** datapoints (1.5–4.5 pp). For Llama 4 Scout, the gain is instead driven by recovering **Missing** structure (28.3→21.8%), at the cost of a slight rise in **Value Errors** and **Extra** datapoints as more candidate rows are retained. **Extra** stays small for Qwen3-VL and Seed 1.6 Flash (<1%), is moderate for Scout (≈5%), and is substantially larger for TinyChart (≈13%), reflecting frequent spurious predicted datapoints when the chart-specialized model is applied to out-of-distribution charts. Even a strong frontier model like GPT-5.1 is bottlenecked by **Value Errors** (22.8%) and tends to over-generate datapoints (**Extra** 10.1%). Across both weak and strong models, then, the dominant failure mode is precise numeric reading rather than chart structure—especially on area and line charts, where values must be inferred from curves rather than discrete bar heights.

Breakdown by chart type. Table 3 reports single-pass RMS_{F1} on WB-ChartExtract by chart type (known by construction; see Section 4.1). Bar charts are the easiest categories and area charts the hardest. Grouped bar charts have the highest scores across all models (e.g., 95.18 for Gemini), likely because the subsampled year axis yields fewer datapoints with discrete, easy-to-read bar heights. Area charts are hardest (e.g., 79.20 for Gemini, 13.94 for Llama 4 Scout), as overlapping or stacked filled regions make it difficult to decompose individual series values. Weaker models show a particularly steep drop from bar to area charts: Llama 4 Scout falls from 48.39 on grouped bars to 13.94 on area, while Gemini's drop is more modest (95.18 to 79.20). This suggests that chart-type diversity in WB-ChartExtract exposes genuine differences in visual reasoning ability that a single chart type would miss. The analogous breakdown on ChartQA is in Appendix B.

4.5 Empirical Validation of Uncertainty Estimation

We assess whether ensemble disagreement predicts extraction quality by correlating each table-level uncertainty summary with final RMS_{F1}. On WB-ChartExtract, relative MAD is significantly anti-correlated with RMS_{F1} (Spearman $\rho = -0.34$ for U_{med}, $\rho = -0.37$ for U_{mean}, and $\rho = -0.30$ for U_{max}; all $p < 0.001$). Thus, higher ensemble disagreement reliably indicates lower extraction accuracy, making relative MAD a useful uncertainty estimate for downstream decisions.

Chart type	n	Gemini	GPT	Llama	TinyChart
Grouped bar	248	95.18	67.91	48.39	54.23
Stacked bar	248	89.39	69.67	31.52	29.28
Line	252	87.68	38.53	28.10	18.32
Area	252	79.20	29.48	13.94	13.02
Overall	1000	87.83	51.26	30.41	28.61

Table 3: Single-pass RMS_{F1} on WB-ChartExtract by chart type. Bar charts are easiest (fewer subsampled years, discrete heights); area charts are hardest (overlapping filled regions).

4.6 Convergence Detection

4.6.1 Convergence Detection Ablation

We ablate convergence detection using Llama 4 Scout. With convergence detection *off*, we always sample a fixed budget of $K_{\max} = 20$ tables per image. With early stopping *on*, we stop once the ensemble meets our convergence criterion, up to the same $K_{\max}$.

Table 4 shows that early stopping achieves most of the ensemble gain over single-pass while using fewer samples on average. On WB-ChartExtract, fixed-budget ensembling improves RMS_{F1} from 30.41 to 33.17 (+2.76), while early stopping reaches 33.14 (+2.73), retaining **99%** of the ensembling boost with **1.24× fewer** samples (20.0→16.11). The modest speedup on WB-ChartExtract reflects the benchmark's difficulty: with chart-type diversity and complex plots, the ensemble stabilizes slowly and early stopping triggers only for a subset of images (27.7% convergence rate).

These results suggest convergence detection is practical for deployment, substantially reducing sampling cost while preserving most ensemble gains. Crucially, convergence is evaluated *independently for each chart*: every image stops as soon as its own consecutive aggregates agree, so the method adaptively spends more samples on hard charts and fewer on easy ones, with no preset stopping point or per-dataset tuning required. Unless otherwise noted, we report results with early stopping on.

Early stopping	RMS_{F1}	$\bar{S}$
Off (fixed budget)	33.17	20.00
On (early stopping)	33.14	16.11

Table 4: Effect of early stopping on WB-ChartExtract using Llama 4 Scout. $\bar{S}$ is the average number of samples per image (lower is better).

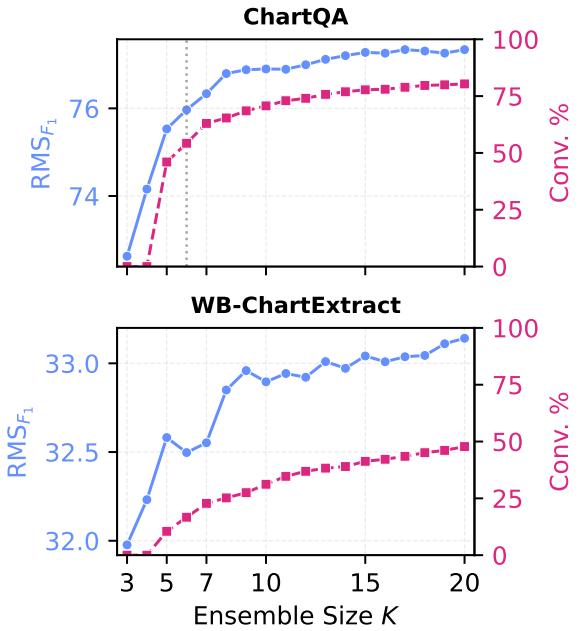

Figure 4: Effect of ensemble size K on RMS_{F1} and the percentage of examples that have met the convergence criterion by size K. Gray dashed lines mark the median stopping iteration under early stopping.

4.6.2 Ensemble Size

Figure 4 shows how RMS_{F1} and convergence evolve with ensemble size K (Llama 4 Scout). On ChartQA (T=2.0), RMS_{F1} rises sharply from K=3 to K=7 and largely saturates thereafter, with 80% of images converged by K=20 and a median stopping iteration of K=6 (average samples per image $\bar{S}$=9.34). On WB-ChartExtract (T=0.0), gains are more gradual and convergence is slower: at K=20 only 27.7% of images have converged, reflecting the benchmark's greater difficulty with diverse chart types. The median stopping iteration here is K=20 because fewer than half of the images converge before the cap, so the 50th-percentile image simply hits $K_{\max}$; as noted above, each chart still stops at its own convergence point, so this lower convergence rate reflects more genuinely hard charts rather than a uniform cutoff.

4.7 Sampling Temperature

Sampling temperature trades per-sample accuracy against ensemble diversity. Figure 6 (Appendix H) plots RMS_{F1} vs. ensemble size for $T \in \{0.0, 1.0, 2.0\}$ using Llama 4 Scout: the optimum is benchmark-dependent. On ChartQA, T=0.0 wins for very small ensembles ($K \leq 4$) but T=2.0 takes over for $K \geq 5$ (best ensemble 77.10) as added diversity outweighs reduced determinism; we therefore use T=0.0 for direct-prediction baselines and T=2.0 for self-ensembling. On WB-ChartExtract, T=0.0 is strictly best at every K (best ensemble 33.14)—the harder charts cannot afford reduced per-sample accuracy—so we use T=0.0 for both. We tune temperature only on Llama 4 Scout and reuse the resulting choices (T=0 for direct-prediction baselines and on WB-ChartExtract self-ensembling; T=2 for ChartQA self-ensembling) across the other base models without per-model retuning, so reported gains for Qwen3-VL, Seed 1.6 Flash, DePlot, and TinyChart are not optimized for those models.

5 Conclusion

We introduced a model-agnostic self-ensembling method for chart data extraction that exploits VLM stochasticity across repeated runs to produce more accurate consensus tables than a single pass. The method also provides ensemble uncertainty estimates and a practical convergence-based early-stopping strategy. To better stress-test chart extraction, we presented WB-ChartExtract, a new benchmark derived from World Bank data with four chart types rendered by four plotting libraries, providing chart-type and cross-library diversity alongside higher data density. Across ChartQA and WB-ChartExtract, self-ensembling improves over single-pass VLM baselines in nearly every model-benchmark combination, with gains of up to 23% relative.

6 Limitations

Our approach inherits several limitations from both the underlying VLM and our ensembling procedure. First, self-ensembling is bounded by base-model failure modes: it can only aggregate among the candidate tables the VLM produces. When the model makes systematic errors, aggregation may not correct them and can even reinforce consistent mistakes.

Second, our early-stopping criterion focuses on numeric stability and may miss "semantic" non-convergence. Because convergence is defined by relative changes in aggregated numeric cells, the ensemble may stabilize while the table structure remains incorrect, particularly when such structural mistakes are consistent across samples.

Third, our uncertainty estimate is heuristic and incomplete. Relative MAD provides a useful disagreement signal, but it is not a calibrated probability of correctness. In particular, uncertainty can be low even when the model is consistently wrong due to systematic errors, so it should be interpreted as a diagnostic rather than a formal confidence measure.

Fourth, WB-ChartExtract is synthetically rendered. While based on real World Bank time series and rendered across four plotting libraries (Matplotlib, Seaborn, Plotly, Bokeh) for stylistic diversity, the charts may not capture all real-world artifacts such as overlapping annotations, low-resolution scans, or non-standard layouts.

References

Rabah A. Al-Zaidy and C. Lee Giles. 2017. A machine learning approach for semantic structuring of scientific charts in scholarly documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4644–4649. AAAI Press.

Anthropic. 2026. Introducing claude opus 4.6. https://www.anthropic.com/news/claude-opus-4-6.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusiñol, Ernest Valveny, C. V. Jawahar, and Dimosthenis Karatzas. 2019a. Scene text visual question answering. *Preprint*, arXiv:1905.13648.

Ali Furkan Biten, Rubèn Tito, Andrés Mafla, Lluis Gomez, Marçal Rusiñol, Minesh Mathew, C.V. Jawahar, Ernest Valveny, and Dimosthenis Karatzas. 2019b. Icdar 2019 competition on scene text visual question answering. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1563–1570.

ByteDance Seed Team. 2025. Introduction to techniques used in Seed1.6. https://seed.bytedance.com/en/blog/introduction-to-techniques-used-in-seed1-6. Accessed: 2026-04-09.

Jinyue Chen, Lingyu Kong, Haoran Wei, Chenglong Liu, Zheng Ge, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024. Onechart: Purify the chart structural extraction via one auxiliary token. *Preprint*, arXiv:2404.09987.

Shiqi Chen, Tongyao Zhu, Ruochen Zhou, Jinghan Zhang, Siyang Gao, Juan Carlos Niebles, Mor Geva, Junxian He, Jiajun Wu, and Manling Li. 2025. Why is spatial reasoning hard for vlms? an attention mechanism perspective on focus areas. *Preprint*, arXiv:2503.01773.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Google. 2026. Gemini 3.1 flash lite. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-flash-lite/. Accessed: 2026-05-04.

Groq, Inc. 2025. Groq is fast, low cost inference.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. Chartllama: A multimodal llm for chart understanding and generation. *Preprint*, arXiv:2311.16483.

Peter J. Huber. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Akos Kadar, Adam Trischler, and Yoshua Bengio. 2017. Figureqa: An annotated figure dataset for visual reasoning. *Preprint*, arXiv:1710.07300.

Wee Kheng Leow, Weihua Huang, and Chew Lim Tan. 2005. Associating Text and Graphics for Scientific Chart Understanding . In *Proceedings. Eighth International Conference on Document Analysis and Recognition*, pages 580–584, Los Alamitos, CA, USA. IEEE Computer Society.

Chengzhi Liu, Zhongxing Xu, Qingyue Wei, Juncheng Wu, James Zou, Xin Eric Wang, Yuyin Zhou, and Sheng Liu. 2025. More thinking, less seeing? assessing amplified hallucination in multimodal reasoning models. *Preprint*, arXiv:2505.21523.

Fangyu Liu, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin

Altun. 2023a. DePlot: One-shot visual language reasoning by plot-to-table translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10381–10399, Toronto, Canada. Association for Computational Linguistics.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. 2023b. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *Preprint*, arXiv:2212.09662.

Junyu Luo, Zekun Li, Jinpeng Wang, and Chin-Yew Lin. 2021. ChartOCR: Data extraction from charts images via a deep hybrid framework. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1917–1925.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *Preprint*, arXiv:2203.10244.

Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Chartassistant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. *Preprint*, arXiv:2401.02384.

Meta AI. 2025. The llama 4 herd: The beginning of a new era of natively multimodal intelligence. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`.

Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. *Preprint*, arXiv:1909.00997.

Ales Mishchenko and Natalia Vassilieva. 2011. Chart image understanding and numerical data extraction. In *Proceedings of the 2011 Sixth International Conference on Digital Information Management (ICDIM)*, pages 115–120. IEEE.

OpenAI. 2025. Gpt-5.1: A smarter, more conversational chatgpt. `https://openai.com/index/gpt-5-1/`.

Yasaman Razeghi, Ishita Dasgupta, Fangyu Liu, Vinay Venkatesh Ramasesh, and Sameer Singh. 2024. Plot twist: Multimodal models don't comprehend simple chart details. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5922–5937, Miami, Florida, USA. Association for Computational Linguistics.

Ankit Rohatgi. Webplotdigitizer.

Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. 2011. Revision: automated classification, analysis and redesign of chart images. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 393–402, New York, NY, USA. Association for Computing Machinery.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

The World Bank. World development indicators | databank: Gdp growth (annual %) (ny.gdp.mktp.kd.zg).

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Peng Ye, Min Dou, Botian Shi, Junchi Yan, and Yu Qiao. 2025. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *Preprint*, arXiv:2402.12185.

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. 2024. TinyChart: Efficient chart understanding with program-of-thoughts learning and visual token merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1898, Miami, Florida, USA. Association for Computational Linguistics.

A RMS Metric Details

We evaluate chart data extraction using *Relative Mapping Similarity* (Liu et al., 2023a, RMS), a table-matching metric that treats a table as an unordered set of mappings from a (row header, column header) pair to a cell value. We represent the predicted table as $P = \{p_i\}_{i=1}^{N}$ and the ground-truth table as $T = \{t_j\}_{j=1}^{M}$, where each entry is a triple $p_i = (p_i^r, p_i^c, p_i^v)$ and $t_j = (t_j^r, t_j^c, t_j^v)$. This set representation is invariant to row/column permutations.

RMS compares *keys* (row/column headers) using a thresholded normalized Levenshtein distance $\mathrm{NL}_\tau(\cdot,\cdot)$, clipping distances above τ to 1. Keys are formed by concatenating headers: $k(p_i) = p_i^r \| p_i^c$ and $k(t_j) = t_j^r \| t_j^c$. Numeric values are compared by a clipped relative error

$$D_\theta(p,t) = \min\Big(1, \frac{\|p-t\|}{\|t\|}\Big),$$

and combined into an entry-level similarity

$$s(p_i,t_j) = \Big(1 - \mathrm{NL}_\tau(k(p_i), k(t_j))\Big)\Big(1 - D_\theta(p_i^v, t_j^v)\Big),$$

which is high only when both headers and values agree.

Chart type	n	TinyChart	Llama 4 Scout	GPT-5.1
Vertical bar	673	99.16	83.71	93.53
Horizontal bar	380	97.35	76.09	87.13
Stacked bar	167	91.78	67.25	71.13
Line	211	86.59	65.73	75.91
Pie	78	81.32	38.64	51.43
Overall	1509	95.20	75.13	84.80

Table 5: Single-pass RMS_{F1} on ChartQA by chart type. Pie and line charts are the hardest categories across all models; the specialist TinyChart leads uniformly.

Aggregator	WB-ChartExtract
Mean	30.49
RANSAC	31.18
Weighted Confidence	31.41
Huber	32.82
Median	**33.14**

Table 6: Aggregation method ablation using Llama 4 Scout on WB-ChartExtract (RMS_{F1}).

RMS then finds a minimum-cost bipartite matching between predicted and target entries (key-based), yielding an assignment matrix $X \in \mathbb{R}^{N \times M}$, and computes mapping-level precision and recall:

$$\text{RMS}_{\text{prec}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} X_{ij}\, s(p_i, t_j),$$

$$\text{RMS}_{\text{rec}} = \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{M} X_{ij}\, s(p_i, t_j).$$

We report RMS_{F1}, the harmonic mean of RMS_{prec} and RMS_{rec}. Following Liu et al. (2023a), we set $\tau = 0.5$ and $\theta = 0.1$. Because predictions may be transposed, RMS also scores both the predicted table and its transpose and takes the higher RMS_{F1}, yielding transposition invariance.

B ChartQA Chart-Type Breakdown

Table 5 reports single-pass RMS_{F1} on ChartQA broken down by chart type (classified using Gemini 3.1 Flash Lite (Google, 2026)). All models exhibit the same qualitative pattern: pie charts are by far the hardest category, followed by line charts and stacked bar charts, while vertical bar charts are easiest. TinyChart leads in every category, consistent with its ChartQA-specific fine-tuning providing a uniform advantage rather than benefiting only particular chart types. Pie charts stand out as especially challenging for the general-purpose models: Llama 4 Scout scores only 38.64, and even GPT-5.1 reaches just 51.43. These categories require inferring values from angles or slopes rather than aligned bar endpoints, explaining the larger gap between the specialist and generalist models.

C Aggregation Method Ablation

We ablate the cell-wise aggregation function using Llama 4 Scout on WB-ChartExtract, comparing five strategies (Table 6): mean, Huber estimator (Huber, 1964), median, *Weighted Confidence* (retaining the top 60% most consistent extractions based on similarity to the ensemble median), and *RANSAC-style* (removing outliers beyond 2×MAD from the median). Median performs best so we use it in all experiments.

D Sensitivity to Convergence Hyperparameters

Our early-stopping rule declares convergence when, between two consecutive ensemble updates, at least a fraction *coverage* of aggregated cells change by no more than a relative *tolerance*, and this holds for *patience* consecutive updates. Increasing patience or coverage, or decreasing tolerance, makes the criterion stricter, typically requiring more samples but yielding marginal accuracy gains. The hyperparameter ablations in this section, as well as those for early stopping (Section 4.6.1), aggregation (Appendix C), and cluster pruning (Appendix E), are conducted with Llama 4 Scout on WB-ChartExtract, and the resulting defaults are then applied uniformly to all other base models without per-model retuning.

Tables 7–9 examine this trade-off on WB-ChartExtract using Llama 4 Scout. **Patience** has the largest compute swing (Table 7): increasing it from 1 to 3 raises $\bar{S}$ from 12.51 to 18.70, while RMS_{F1} rises only marginally (32.88→33.14→33.23) with clearly diminishing gains beyond patience=2. We use patience=2 as a default because it captures most of the accuracy gain over patience=1 (+0.26 pp) while avoiding the disproportionate compute cost of patience=3 (+2.59 samples for only +0.09 pp). Varying **coverage** (90%→97.5%) modestly increases $\bar{S}$ with negligible change in RMS_{F1} (≤0.06 pp; Table 8). **Tolerance** offers a similar cost–accuracy trade-off (Table 9): tighter tolerance (0.1%) increases $\bar{S}$ with negligible improvement in RMS_{F1} (+0.05 pp), while looser tolerance (10%) stops much earlier ($\bar{S}$

Patience	RMS_{F1}	$\bar{S}$
1	32.88	12.51
2 (default)	33.14	16.11
3	33.23	18.70

Table 7: Sensitivity to *patience*, fixing coverage=95% and tolerance=1%. $\bar{S}$ is the average samples per image. Llama 4 Scout on WB-ChartExtract.

Coverage	RMS_{F1}	$\bar{S}$
90%	33.18	15.24
95% (default)	33.14	16.11
97.5%	33.20	16.58

Table 8: Sensitivity to *coverage*, fixing patience=2 and tolerance=1%. Llama 4 Scout on WB-ChartExtract.

Tolerance	RMS_{F1}	$\bar{S}$
0.1%	33.19	16.70
1% (default)	33.14	16.11
10%	32.90	12.11

Table 9: Sensitivity to *tolerance*, fixing patience=2 and coverage=95%. Llama 4 Scout on WB-ChartExtract.

Pruning Threshold	WB-ChartExtract
0.0 (no pruning)	32.00
0.1	32.84
0.2 (default)	**33.14**
0.3	32.93
0.4	32.27
0.5	31.89

Table 10: Effect of the cluster pruning threshold on RMS_{F1} using Llama 4 Scout on WB-ChartExtract.

drops 25%) at a small cost in RMS_{F1} (0.24 pp).

These hyperparameters therefore tune the cost–accuracy frontier. Unless otherwise noted, we use patience= 2, coverage= 95%, and tolerance= 1%.

E Cluster Pruning Threshold

We ablate the cluster pruning threshold, which sets the minimum fraction of sampled tables in which a row/column label must appear to be retained after clustering. Higher thresholds remove low-support (often spurious) labels, while lower thresholds retain more labels but risk noise. Using Llama 4 Scout on WB-ChartExtract, we test thresholds in $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, where 0.0 denotes no pruning.

Table 10 reports RMS_{F1}. The best threshold is 0.2 (33.14), with 0.1–0.3 forming a plateau within 0.3 pp; performance degrades on both sides, with no pruning (0.0) at 32.00 and aggressive pruning (0.5) at 31.89. We use 0.2 as the default and apply this single value across all base models rather than tuning per model, since per-model tuning would risk cherry-picking and complicate cross-model comparisons. The optimum may differ for Qwen3-VL, Seed 1.6 Flash, DePlot, and TinyChart, which we did not separately ablate; for deployment, we recommend any value in the 0.1–0.3 range.

F Additional Metrics: RNSS and RD

RMS_{F1} jointly scores headers and values through a key-aware matching. To check that our gains are not an artifact of this particular metric, we additionally report two value-centric, header-agnostic metrics on WB-ChartExtract.

RNSS. Relative Number Set Similarity (Luo et al., 2021) treats each table as a multiset of numeric values (headers and indices excluded). Given predicted values $\{p_i\}_{i=1}^{N}$ and ground-truth values $\{g_j\}_{j=1}^{M}$, we form the relative-distance matrix $d_{ij} = \min(1, |p_i - g_j|/|g_j|)$ (with g_j=0 scored as 0 iff p_i=0, else 1), pad the smaller set with dummy entries of distance 1, and compute a minimum-cost bipartite matching. Then

$$\text{RNSS} = 1 - \frac{\sum_{\text{matched}} d_{ij} + |N - M|}{\max(N, M)},$$

which penalizes both inaccurate values and cardinality mismatches (higher is better).

RD. Relative Deviation isolates numeric closeness from cardinality: it is the mean of d_{ij} over the matched real value pairs (lower is better), capturing how close matched numbers are irrespective of how many were matched.

Table 11 reports both. **RNSS** improves with self-ensembling for every base model (from +0.5 for TinyChart and DePlot up to +4.7 for Llama 4 Scout), mirroring the RMS_{F1} trend and confirming that ensembling recovers more correct numbers. **RD** is more mixed: it decreases (improves) for four of the five base models—TinyChart, DePlot, Qwen3-VL, and Seed 1.6 Flash—but rises slightly for Llama 4 Scout (19.96 → 20.70). This is consistent with the error analysis (§4.4): Llama 4 Scout also posts the largest RNSS gain (+4.7), so ensembling recovers many additional matched datapoints, but the newly drawn-in matches are lower-precision

	RNSS (↑)		RD (↓)	
Model	Single	+Self-ens.	Single	+Self-ens.
TinyChart	36.71	37.25	22.26	21.48
DePlot	30.95	31.48	26.86	25.93
Qwen3-VL	62.56	64.38	10.57	10.38
Seed 1.6 Flash	56.34	59.94	17.59	16.18
Llama 4 Scout	51.28	55.95	19.96	20.70

Table 11: RNSS and RD on WB-ChartExtract, single-pass vs. self-ensembled. RNSS (higher is better) improves for every model; RD (lower is better) improves for all base models except Llama 4 Scout, whose RD rises slightly.

and dilute per-match closeness, nudging RD upward.

G Accuracy–Cost Tradeoff

Figure 5 visualizes the accuracy–cost tradeoff discussed in §4: it places each model's single-pass and self-ensembled $\mathrm{RMS}_{\mathrm{F1}}$ (Table 1) against its per-image API cost (Table 2) on a log scale, making the upward-and-rightward shift induced by self-ensembling explicit.

H Sampling Temperature Ablation

Figure 6 reports the sampling-temperature ablation discussed in §4.7, plotting $\mathrm{RMS}_{\mathrm{F1}}$ against ensemble size for $T \in \{0.0, 1.0, 2.0\}$ on both benchmarks.

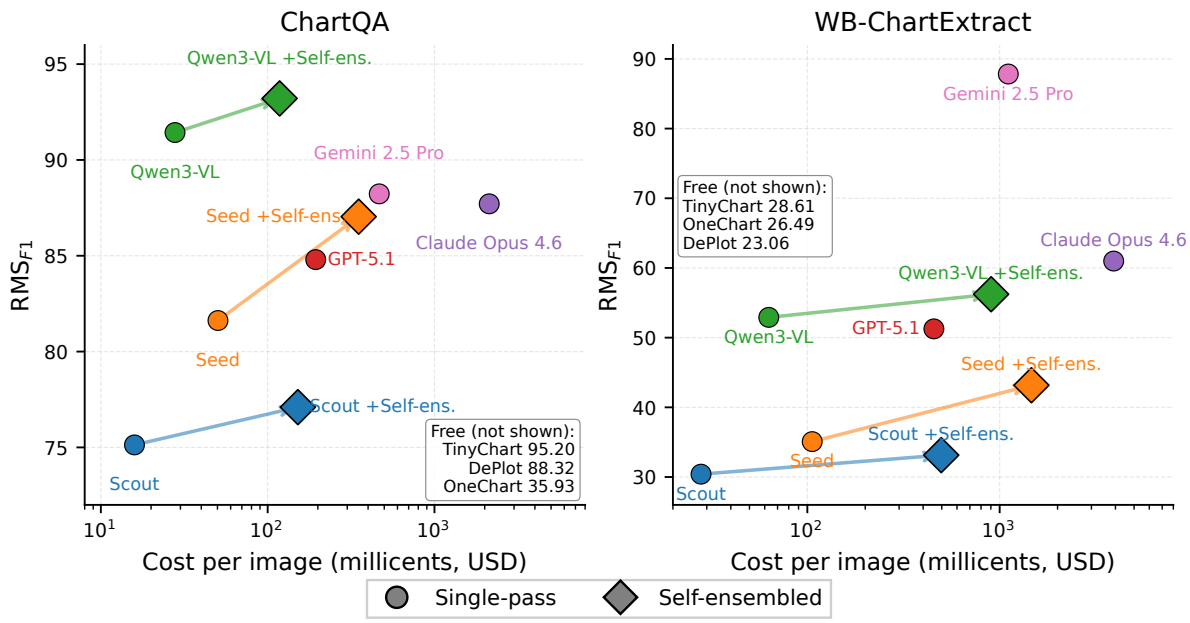

Figure 5: Accuracy–cost tradeoff on ChartQA (left) and WB-ChartExtract (right). Each base model's single-pass result (circle) is connected by an arrow to its self-ensembled variant (diamond); the x-axis is per-image API cost on a log scale. Self-ensembling consistently increases RMS_{F1} at a higher per-image cost. TinyChart, OneChart, and DePlot run locally at zero API cost and are annotated separately, as they cannot be placed on the log-cost axis.

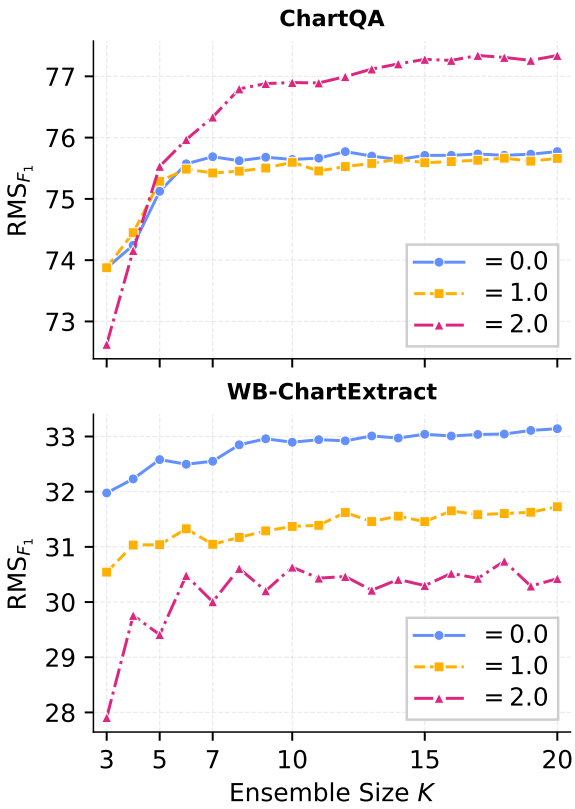

Figure 6: Effect of sampling temperature on ensemble performance using Llama 4 Scout on ChartQA (top) and WB-ChartExtract (bottom).