Does Rethinking Text as Vision Allow us to Scale LLMs Better?

DeepSeek-OCR: ***Contexts Optical Compression***
Glyph: ***Scaling Context Windows via Visual Text Compression***

Authors:

- *Wei et al. @ DeepSeek*
- *Cheng et al. @ Tsinghua University*

Outline

1. Background: OCR and VLMs
2. Glyph: Introduction & Motivation
3. Glyph: Method & Novelty
4. Glyph: Results
5. DeepSeek OCR: Introduction & Motivation
6. DeepSeek OCR: Method & Novelty
7. DeepSeek OCR: Results
8. **Discussion and Open Questions**

Some Background

Long Context Problem

The Problem: **Long context problem.**

- OOD for the model. (performance drop).
- Inference cost increase.
- GPU CUDA memory limit.

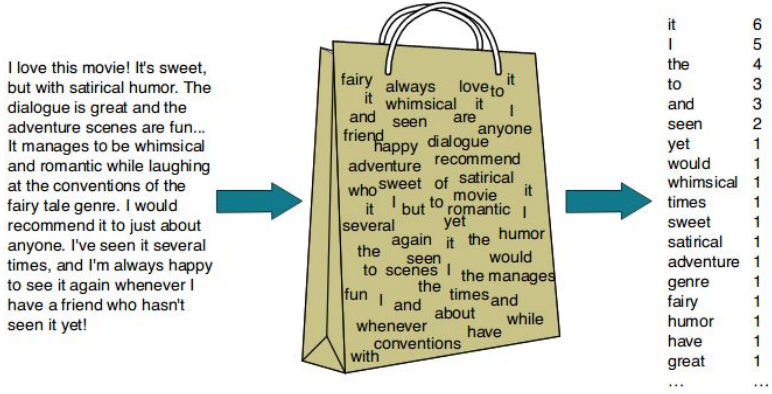
Solutions:

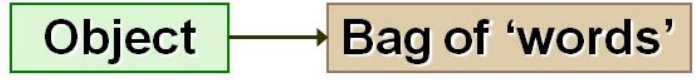
- Extend position encoding. e.g. YaRN
- KV cache eviction, sparse attention.
- OCR: compress text into visual tokens. ← Today's papers

Visual Tokens vs Text Tokens:

A picture have many patches, each patch has its own "local" semantic (i.e. a cat, a table edge), and have dependencies / connections in between. This is similar to tokens in natural language.

=> We can treat image as "bag of word"





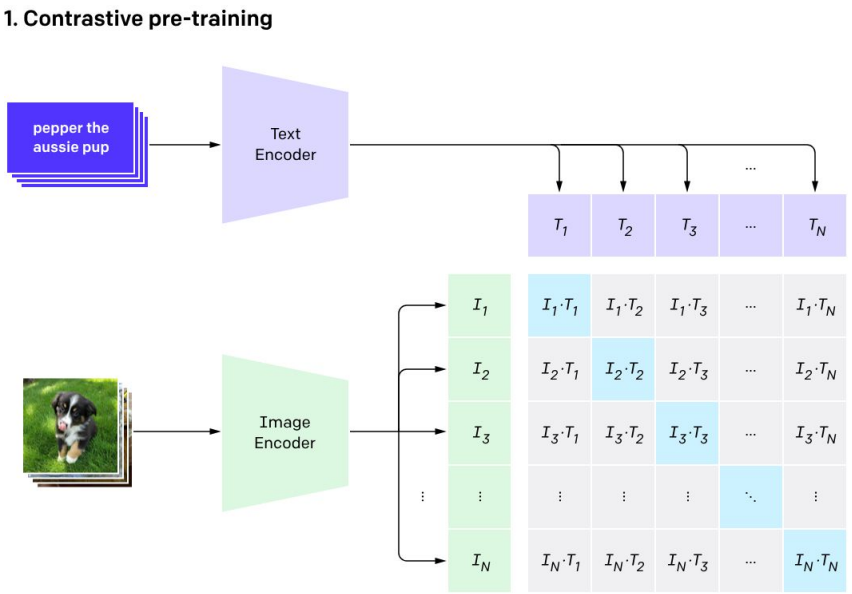
Background: VLM

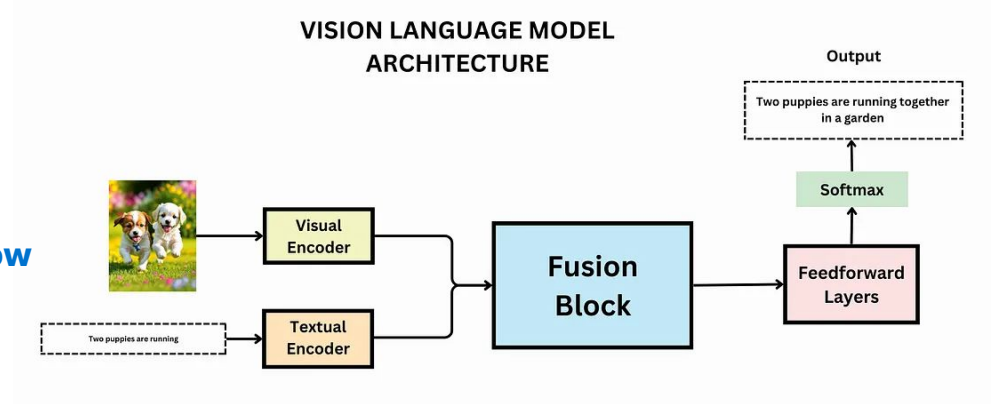
We have good text LLM already. Can we leverage natural language data to understand image?
Key: Representation can be shared shared.

Solution: Align visual token (of an image) to word token (of the caption) via contrastive pretraining. (=>CLIP)

Then we can fuze the visual encoder and the text encoder and just use a single LLM . (VLM)

Now with a VLM, what can we get?
=> Generic image understanding.
-> OCR ability!
-> Can we represent text as image? Just like how human read books.





Glyph: Scaling Context Windows via Visual Text Compression

Glyph: Introduction

Motivating the Problem

Reading isn't a character-by-character computation; the **visual stream** aggregates shapes into **word forms** that the **linguistic stream** maps to meaning.

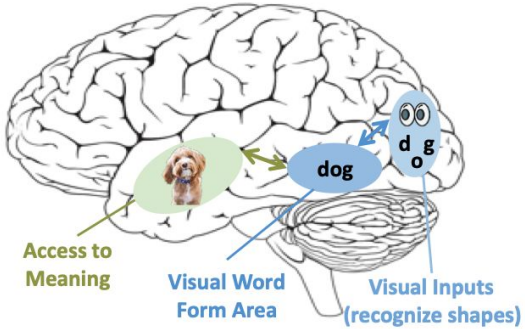
Pure text tokenizers ignore this pipeline and over-segment (especially across scripts and morphologies), inflating sequence length with units that are visually redundant.

If we let the model ingest the *visual* bundles directly, a few visual tokens can carry the same semantic payload as many subwords—compressing context without discarding meaning.

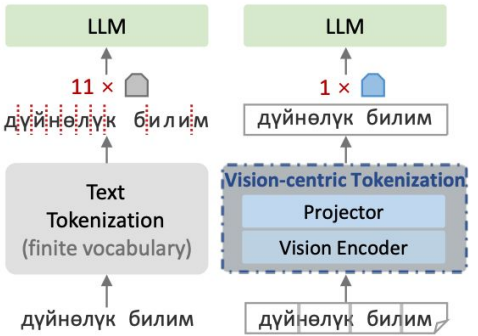
Why Current Solutions Don't Work!

- **Extractive prompt compression (LLMLingua family):** Removes "less important" tokens to hit budgets; helpful but **task-dependent**.
- **Context-extension via RoPE scaling (YaRN / NTK-aware):** Increases the **window size** but **doesn't reduce tokens**—compute and memory still grow with the raw sequence length.

Analogy from prior work: **Visual processing** can *aggregate shape/word* cues into **fewer** units: used here only to motivate Glyph's visual-text compression.



Reading Activity in the Brain



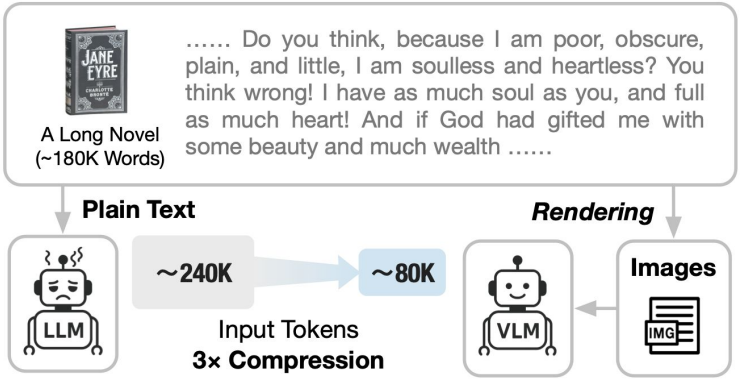
Glyph: Introduction & Motivation

How Glyph Presents the Problem

Modern LLMs need to reason over **long inputs,** but compute and memory grow with **sequence length**, not semantic content density!

Scaling to **million-token** contexts is impractical due to prohibitive compute and memory; existing approaches either extend windows or prune tokens with trade-offs.

Render → VLM: Convert long text to images so each **visual token aggregates many words**, attacking the bottleneck by changing the **representation**, not just the window.

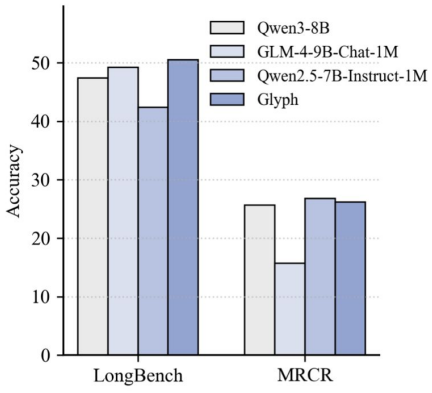


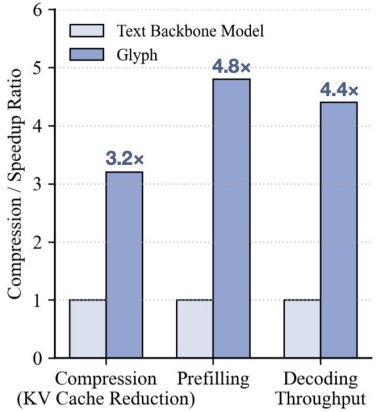
What it Proposes to Solve the Problem

Framework: A practical pipeline that **renders long text to images** and processes them with a VLM, enabling large effective context with fewer tokens and stable accuracy.

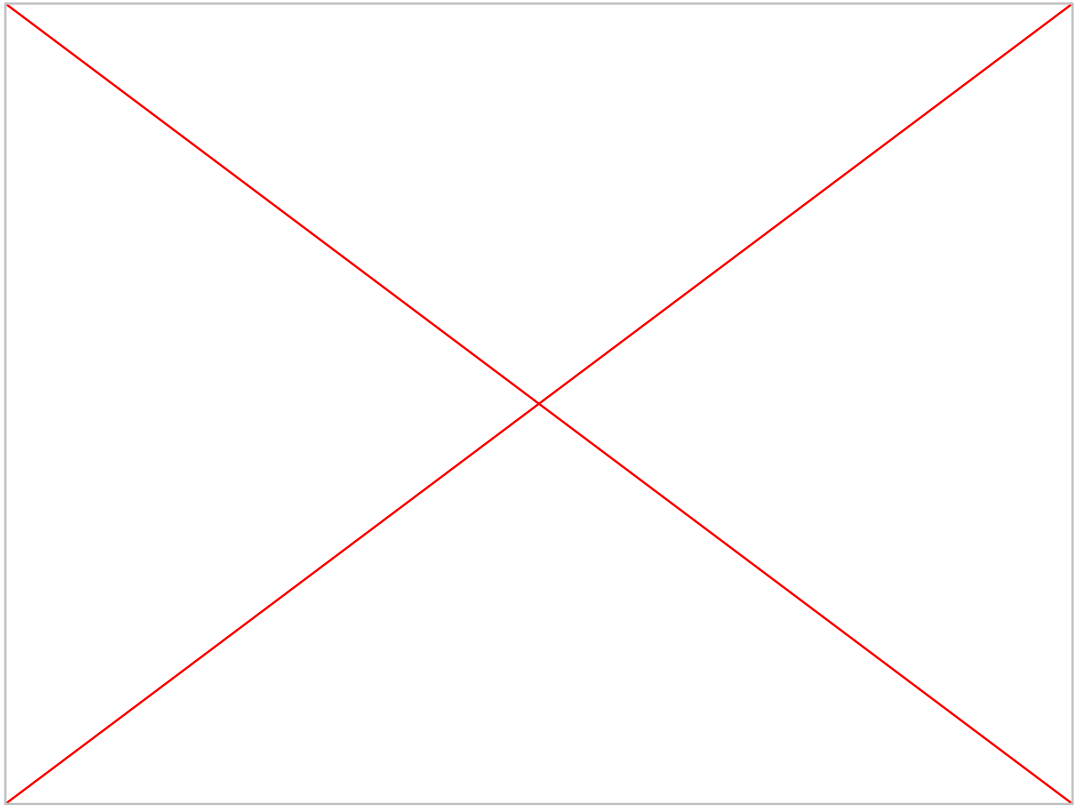
LLM-driven rendering search: A **genetic search** over fonts/layout/DPI to optimize the **readability ↔ compression** trade-off automatically.

Training recipe + scaling result: Continual pre-training on rendered corpora → **post-training (SFT/RL)** with an **auxiliary OCR-style loss**.





Glyph: Demo to Motivate



Some Basics

Any task is defined by the triplet: $(\mathcal{I}, \mathcal{C}, \mathcal{R})$ - **I** is a user instruction **that specifies** the core goal **C**, where $\mathcal{C} = \{c_1, \ldots, c_T\}$ is an **ultra-long** textual context.

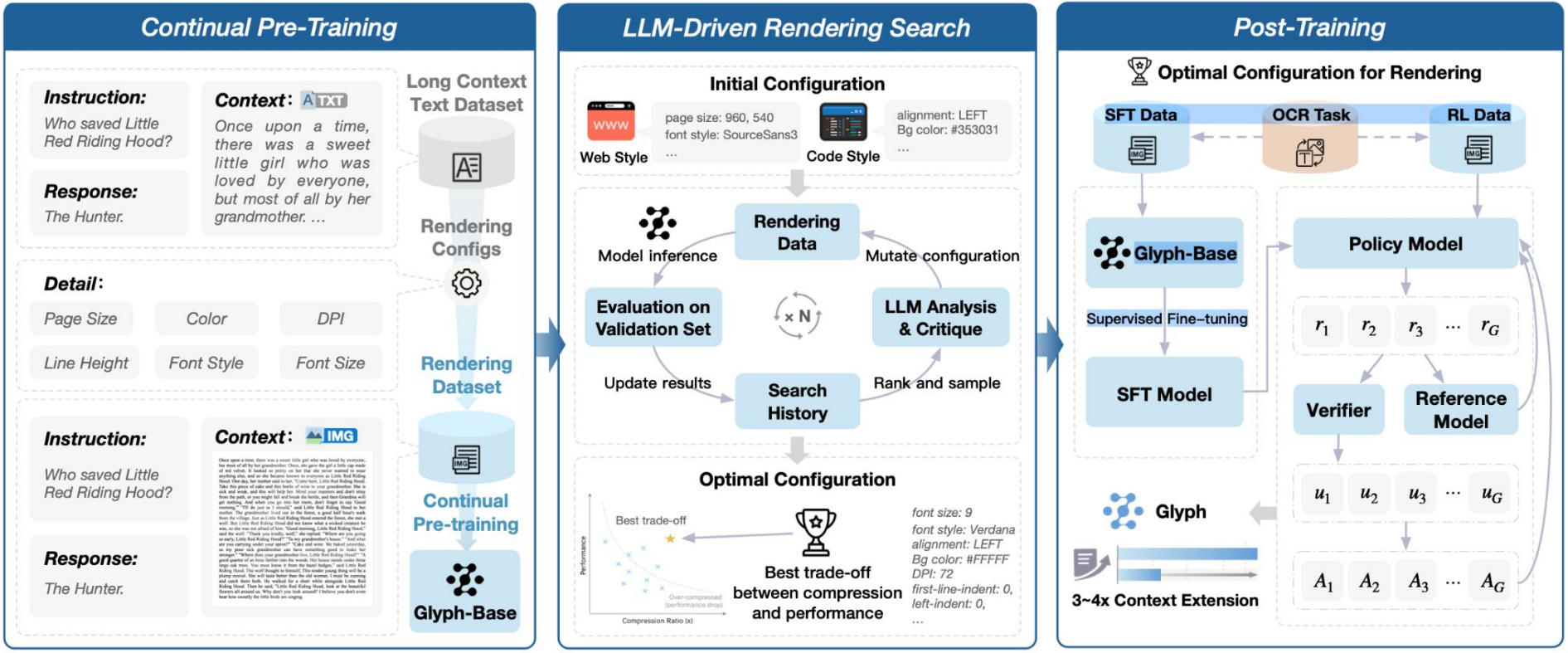
R is target response.

Traditionally, we would do: P(R | I, C).

GLYPH: Render C as visual pages $\mathcal{V} = \{v_1, \ldots, v_n\}$ each containing *glyph's* of multiple text segments. Hence, Glyph does:

P (R | I, V) and a task is defined by (I, V, R)

Glyph: Method



Glyph: Method

CONTINUAL PRE TRAINING

Goal: Transfer long-context from **text tokens** → **visual tokens** by exposing a VLM to massive **rendered long-text** with diverse typography/layout, so the model learns to read pages as compact carriers of text.

Three tasks:

- **OCR reconstruction:** read one/many rendered pages and **transcribe all text** verbatim.
- **Interleaved LM:** mix text and rendered spans and **predict next tokens** while **switching modalities**.
- **Generation/completion:** given partial pages (beginnings/ends), **complete the missing content**, exercising long-range dependencies.

$$\mathcal{L}_{\text{CPT}} = -\mathbb{E}_{(\mathcal{I}^*,\mathcal{V},\mathcal{R})} \sum_t \log P_\phi(r_t \mid \mathcal{I}^*, \mathcal{V}, r_{<t}),$$

$$\mathcal{L} = \text{CE}(\hat{y}, y) \quad \text{with} \quad \hat{y} = \text{VLM}_\phi(I, V_\theta)$$

Gives us **Glyph-Base.** Where does the 'theta' in V come from? Those are the configs of the 'page' -> Next step!

Glyph: Method

Rendering Search

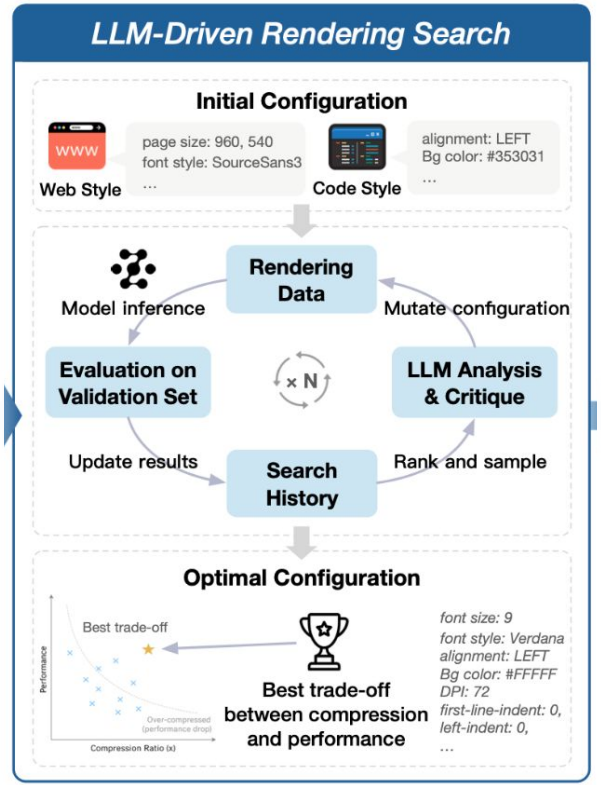
Why? Remember the photos are not 'real' but instead generated from text. Need configs for how the photo 'looks'.

Examples:

- DPI & Page Size
- Font Family and Size
- Borders and Spacing
- Colors

Parametrized by Θ. Controlling this **controls compression ratio from text to vision.**

$$\rho(\boldsymbol{\theta}) = \frac{|\mathcal{C}|}{\sum_{i=1}^{n} \tau(v_i)},$$



Genetic Algorithm Approach

What is being evolved. A *rendering genome* θ. Each θ renders the same long text into images Vθ, changing **compression** (visual tokens/page) vs **readability**.

Mutation-only evolutionary loop. Start from seed styles (web/code).

For each generation: **Render**, **external LLM critiques** outputs (where/why errors happen, readability issues), **Rank & sample** promising candidates, **Mutate** parameters.

Objective/selection signal. Keep candidates that **maximize task utility under high compression**, effectively searching the **Pareto front** between accuracy and compression ratio; the chosen $\theta^\star$ is the *best trade-off*.

Glyph: Method

Post Training

Two tasks to improve "**Glyph-Base**": Supervised FT and RL & Auxiliary OCR Alignment Task

SFT: Similar to what you would do with a Language Model **except C is now V**. Loss function is written as:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(\mathcal{I},\mathcal{V},\mathcal{R})} \sum_{t} \log P_\phi(r_t \mid \mathcal{I}, \mathcal{V}, r_{<t}),$$

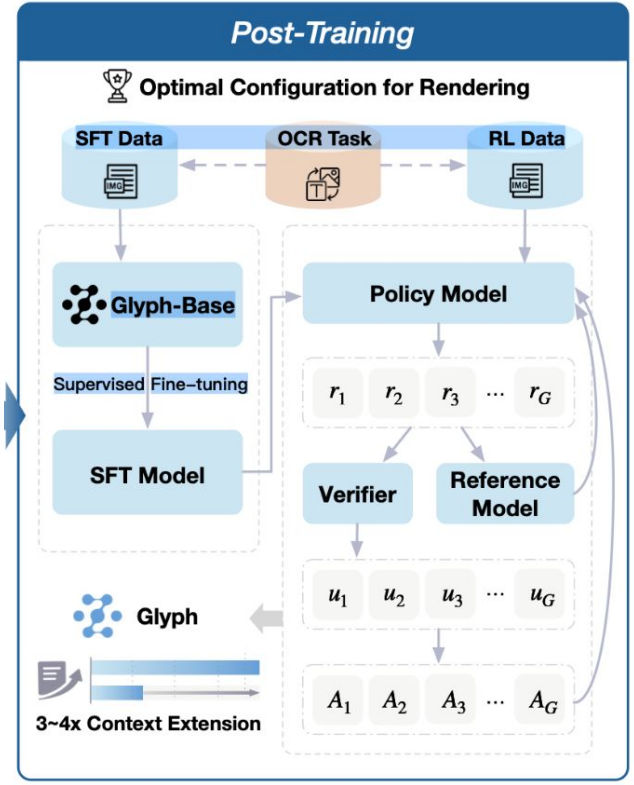
RL: Again, like basic LLM training, they **use GRPO** (C is now V). Importance sampling weight defined as:

$$w_i = \frac{\pi_\phi(r_i \mid \mathcal{I}, \mathcal{V})}{\pi_{\phi_{\text{old}}}(r_i \mid \mathcal{I}, \mathcal{V})}.$$

Reward r is {0,1} integrating **Verifiable Rewards** and **Format Rewards.**

$$A_i = \frac{u(r_i) - \text{mean}(\{u(r_j)\}_{j=1}^{G})}{\text{std}(\{u(r_j)\}_{j=1}^{G})},$$

$$\mathcal{J}_{\text{GRPO}}(\phi) = \mathbb{E}_{(\mathcal{I},\mathcal{V})\sim P,\, \{r_i\}_{i=1}^{G}\sim\pi_{\phi_{\text{old}}}} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \Big(\min\left(w_i A_i,\ \text{clip}(w_i,\, 1-\epsilon_l,\, 1+\epsilon_h)\, A_i\right) - \beta\, D_{\text{KL}}(\pi_\phi \,\|\, \pi_{\text{SFT}}) \Big) \Bigg],$$



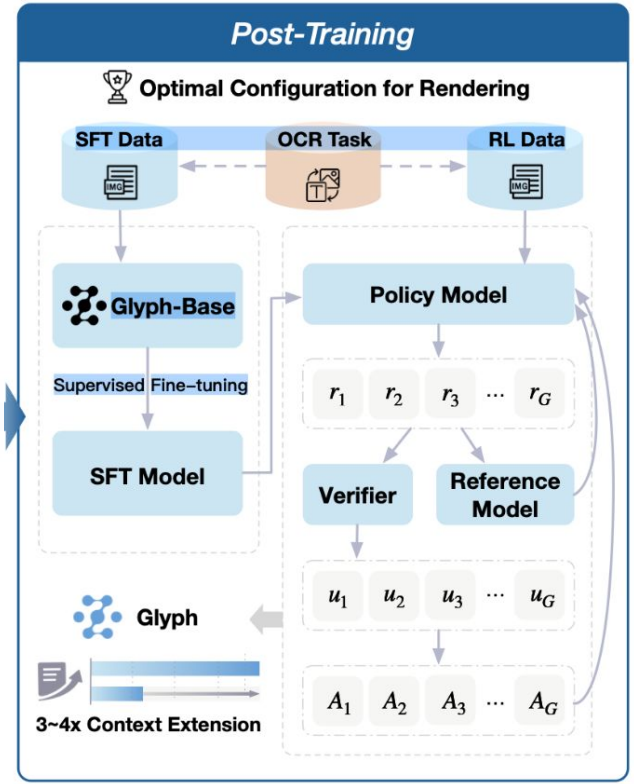
Glyph: Method

Post Training OCR Alignment

Reconstruct all text from a generated 'V' and fine tune to improve on that task - similar to **Continual Pre-Training Step.**

They do this in both the SFT and RL Step:

1. For SFT - **Its same** as Pre-Training. Just minimize CE.
2. For RL - **Its** basic GRPO but reward is Levenshtein Distance.



Glyph: Controllable Factors and Sampling

Factor	Specification / Sampling Strategy
`dpi`	Mixture of sets: *lowest* (45–59), *low* (60–71), *medium* (72–119), *normal* ({72,80,96,100,110,120,144,150,300}), *high* (over 300); favor normal/medium with small probability spikes to extremes.
`page_size`	(i) Fixed paper sizes (A4, Letter, Legal, A5, B5, A3, B4, Tabloid) with priors; (ii) common aspect ratios (e.g., 1.414, 1.333, 1.5, 1.778); (iii) fully random aspect via piecewise distribution (narrow → tall).
`font_family`	Pooled and deduplicated families across serif/sans/mono/pixel; italics sampled by filename heuristics (suffixes, `italic`/`oblique`).
`font_size`	$\{7, 7.5, 8, 9, 9.5, 10, 11, 12, 14\}$; `line_height` tied as `font_size` + $\{0, \ldots, 3\}$.
`alignment`	LEFT/JUSTIFY (dominant) with small-prob. RIGHT/CENTER.
`margins`	Three patterns: all-equal; vertical-larger; horizontal-larger; values in 10–40pt ranges.
`indent`	Modes: none; first-line indent (≈1–2.5 em); block/hanging with left/right indents.
`spacing`	`space-before`/`space-after` use a multi-mode prior (none, small, large).
`h_scale`	Horizontal glyph scaling (0.75–1.0) with decaying probabilities.
`colors`	Page/background/font palettes for light/dark themes; document/web/code styles inherit coherent triplets (page, paragraph, font).
`borders`	Optional paragraph borders with width/padding; disabled by default.
`newline_markup`	With small probability, explicit markers (e.g., \n, tags, or tokens) inserted to preserve structure.
`auto_crop`	Optional white-margin cropping and last-page trimming.

Experiments and Results

Glyph: Experimental Setup

Benchmarks:

- **LongBench**: a bilingual , multitask long context understanding benchmark
- **MRCR**: A multi-document / “needle in haystack” style long-context retrieval QA benchmark.
- **Ruler**: long-context benchmark with various sub-tasks including UUID recognition etc (where they ignore the UUID recognition.
- **MMLongBench-Doc**: A long document understanding benchmark (multimodal PDF documents with images/layout) used for cross-modal generalisation.

Backbone VLM: GLM-4.1V-9B-Base

Glyph: Results

Main performance on LongBench => Matches GPT-4.1

(And in Appendix the rest results)

More degradation in Single-Doc QA (Glyph has a more blurry understanding?)

and summarization.

Model	Single-Doc QA		Multi-Doc QA		Summarization		Few-shot		Synthetic		Code		Avg
	QP	NQA	HQA	2QA	QSUM	GovRep	TREC	TriQA	PR Zh	PR En	RB	LCC	
GPT-4.1	51.60	35.73	69.10	74.15	23.50	33.36	77.00	93.36	100.00	100.00	67.94	68.43	56.03
LLaMA-3.1-8B-Instruct	44.56	26.34	56.88	46.67	**23.28**	**32.36**	19.25	89.12	62.20	99.50	42.81	46.35	41.34
Qwen2.5-7B-Instruct-1M	**45.29**	25.61	60.70	40.51	22.95	29.97	59.37	86.93	98.5	**100.00**	29.80	21.72	42.42
Qwen3-8B	44.67	26.13	65.83	**73.92**	19.60	26.85	70.50	87.98	**100.00**	97.26	40.89	44.87	47.46
GLM-4-9B-Chat-1M	43.75	26.72	58.98	50.89	22.84	27.60	61.50	**90.07**	**100.00**	99.50	55.64	**59.54**	49.27
Glyph	40.64	**28.45**	**66.42**	72.98	19.78	25.53	**82.62**	88.54	89.03	99.50	**60.80**	48.85	**50.56**

Table 1: Performance comparison of Glyph with leading LLMs on LongBench (%). Our model achieves competitive results in the overall average score. Best results are **bolded**, and second-best are underlined. Refer to Table 10 for the rest of the results.

Model	Single-Doc QA		Multi-Doc QA		Summarization		Few-shot		Synthetic
	QA Zh	QA En	Mus	Dur	News	VcSum	Sam	Lsht	Pa C
GPT-4.1	63.90	51.27	55.63	24.58	23.70	14.66	41.25	50.00	26.5
LLaMA-3.1-8B-Instruct	62.20	**54.98**	31.61	**33.75**	**24.21**	**16.23**	7.61	0.00	7.13
Qwen3-8B	60.98	49.78	45.54	16.69	18.55	12.08	36.47	42.00	12.81
GLM-4-9B-Chat-1M	**63.17**	52.88	39.14	28.27	23.90	16.21	36.15	**47.38**	2.39
Qwen2.5-7B-Instruct-1M	62.98	53.62	34.72	21.85	21.02	12.20	**39.17**	28.68	3.50
Glyph	37.23	45.89	**56.18**	26.87	21.52	12.43	32.49	44.43	**30.50**

Table 10: The rest of the results on LongBench benchmark (%), which encompasses Single-Document QA, Multi-Document QA, Summarization, Few-shot Learning, and Synthetic task.

Glyph: Results

Main performance on MRCR => small gap below GPT-4.1 match open-source.

Model	4 Needle						8 Needle					
	0k-8k	8k-16k	16k-32k	32k-64k	64k-128k	Avg	0k-8k	8k-16k	16k-32k	32k-64k	64k-128k	Avg
GPT-4.1	50	38	29	42	38	39.4	33	26	17	22	19	23.4
LLaMA-3.1-8B-Instruct	33.42	25.97	22.73	**26.97**	12.68	24.35	23.80	17.69	**19.85**	17.72	11.79	**18.17**
Qwen2.5-7B-Instruct-1M	25.96	20.13	19.93	24.25	17.29	21.51	17.64	19.48	12.41	14.80	14.24	15.71
Qwen3-8B	29.34	22.67	20.34	23.63	**19.11**	23.02	18.75	19.69	16.81	**17.86**	**15.00**	17.62
GLM-4-9B-Chat-1M	15.17	13.78	9.18	20.27	15.05	14.69	14.55	9.65	9.34	9.47	8.97	10.40
Glyph	**35.44**	**26.82**	**24.15**	25.69	16.37	**25.81**	**25.12**	**21.22**	16.43	13.91	13.51	18.14

Table 2: Performance comparison of our model against leading LLMs on the 4-needle and 8-needle sub-tasks of the MRCR benchmark (%). Our method consistently ranks first or second across most settings while preserving about 3× compression ratio. Performance on the 2-needle task is deferred to the Appendix.

2 needle is less performant (so they defer to appendix and didn't explain why)

Model	2 Needle					
	0k-8k	8k-16k	16k-32k	32k-64k	64k-128k	Avg
GPT-4.1	83	72	67	62	59	68.6
LLaMA-3.1-8B-Instruct	54.27	**53.21**	**51.05**	29.81	24.98	42.66
Qwen3-8B	**58.95**	41.18	36.18	24.99	20.89	36.44
GLM-4-9B-Chat-1M	39.77	15.87	18.42	18.63	18.42	22.22
Qwen2.5-7B-Instruct-1M	45.92	51.07	46.97	**34.67**	**37.57**	**43.24**
Glyph	41.51	40.78	39.58	29.67	22.41	34.85

Table 9: Performance of various models on the MRCR task (%) with the 2 Needle setting across different context length intervals (0k–8k, 8k–16k, 16k–32k, 32k–64k, 64k–128k) and the average score.

Compression Ratio

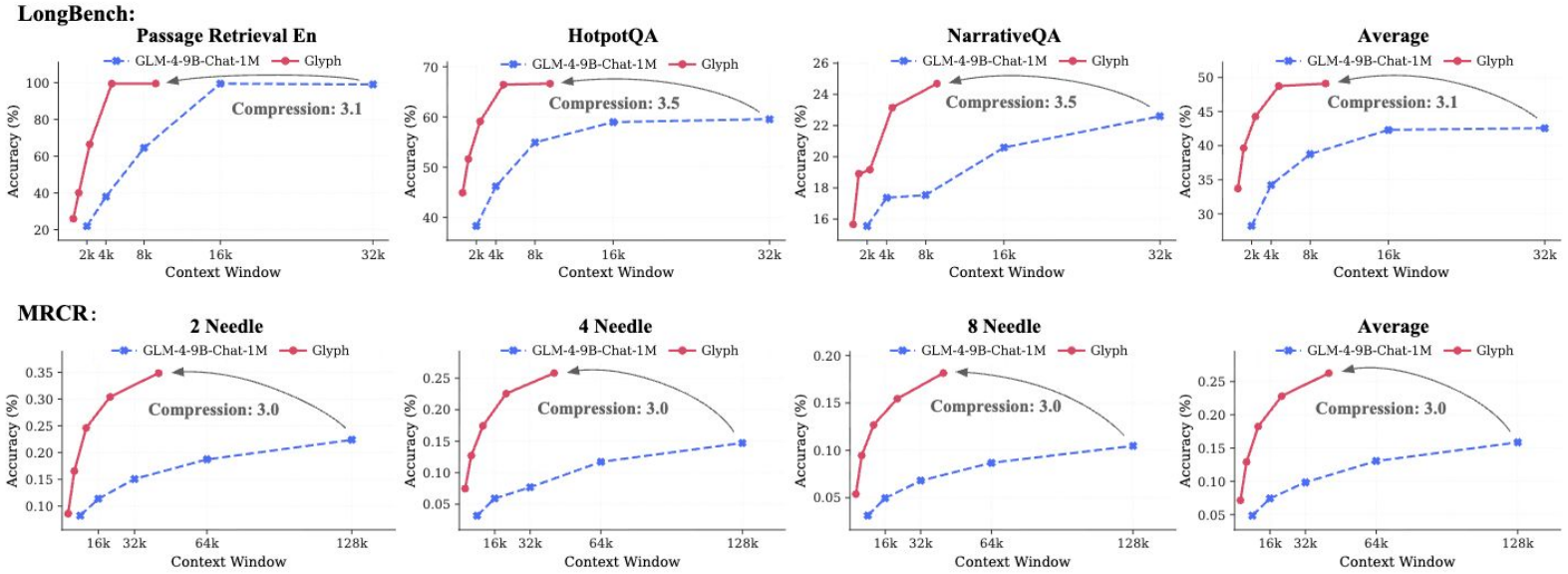


Figure 3: Performance comparison of Glyph and the baseline across different context windows, demonstrating that Glyph achieves performance equivalent to longer contexts with substantially shorter context windows.

Varying Compression Ratio.

Higher DPI → larger rendered doc → More 14*14 pixel patch → More visual token.

Model	Niah-S1	Niah-S2	Niah-M1	Niah-M2	Niah-V	Niah-Q	VT	CWE	FWE	QA-1	QA-2	Avg
GPT-4.1	100.0	98.85	100.0	100.0	99.67	100.0	100.0	97.87	98.66	86.82	77.47	96.30
LLaMA-3.1-8B-Instruct	99.33	99.33	99.33	**99.00**	98.17	99.67	87.07	57.30	81.85	**84.00**	58.00	87.55
Qwen2.5-7B-Instruct-1M	**100.00**	99.67	99.67	**99.00**	93.83	98.75	85.40	72.10	85.67	80.00	60.67	88.61
Qwen3-8B	**100.00**	**100.00**	95.33	84.67	97.42	99.33	98.47	74.67	86.67	70.33	53.33	87.29
GLM-4-9B-Chat-1M	**100.00**	**100.00**	92.67	**99.00**	95.00	**100.00**	98.20	49.50	83.22	72.67	56.67	86.08
DPI: 72 / Compression rate: average 4.0, up to 7.7												
Glyph	73.33	64.67	67.33	56.00	73.42	71.42	77.93	94.40	92.67	59.33	63.33	72.17
DPI: 96 / Compression rate: average 2.2, up to 4.4												
Glyph	98.00	95.33	95.67	85.00	96.33	95.83	94.93	94.80	98.00	79.00	70.67	91.23
DPI: 120 / Compression rate: average 1.2, up to 2.8												
Glyph	99.67	99.00	**100.00**	93.67	**99.00**	99.58	**99.33**	**98.97**	**99.11**	79.00	**74.00**	**94.67**

Table 3: Performance on the Ruler benchmark (%). We demonstrate the impact of different DPI settings on our model's performance and the resulting compression ratios. For each configuration, the table includes both the average compression ratio across all sub-tasks and the maximum compression achieved for specific sub-task types.

Efficiency

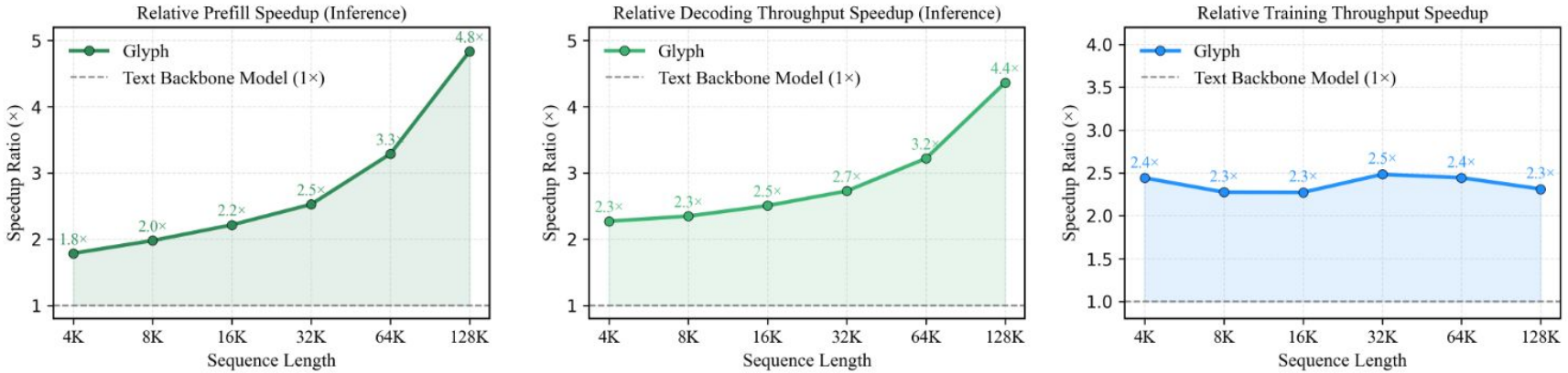


Figure 4: Speedup ratios of Glyph over the text backbone model for prefill, decoding, and training across different sequence lengths.

As context grow…

Glyph has slower degradation.

(what about making axis the number of token? Is the degradation tied deeply to the attention mechanism - some retrieval efficiency upper bound)

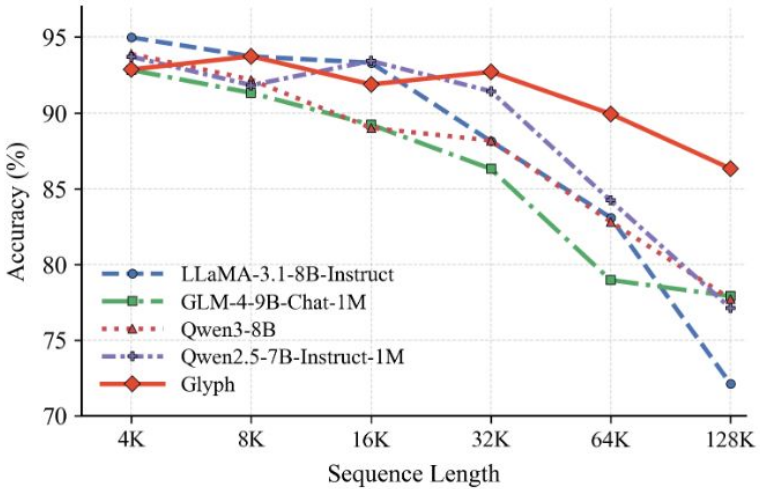


Figure 5: Model performance degradation across different sequence lengths on the Ruler benchmark.

Better than base VLM

On MMLongBench-Doc (multimodal document understanding)

The full Glyph version outperform Base VLM.

→ Cross modality post-training/RL actually improves model's ability.

Model	SP	CP	UA	Acc	F1
GLM-4.1V-9B-Base	36.76	23.41	21.52	29.18	28.78
Glyph-Base	47.91	22.24	14.80	32.48	34.44
Glyph	**57.73**	**39.75**	**27.80**	**45.57**	**46.32**

Table 4: Results on MMLongBench-Doc (%). SP, CP, UA, and Acc denote Single-page, Cross-page, Unanswerable, and Overall Accuracy, respectively.

Ablation: Is Genetic Search necessary

Configuration	LongBench	MRCR	Ruler	Avg.
Random Config	41.78	15.82	65.13	40.91
Manual Config	**43.45**	19.33	68.09	43.62
Search-based Config	**43.45**	**22.10**	**71.24**	**45.60**

Table 5: Ablation study comparing randomly combined, manually designed, and search-based configurations on three benchmarks under SFT setting. The search-based configuration achieves the best overall performance.

Ablation: Is Aux. Objective necessary

Model	LongBench	MRCR	Ruler
Glyph	50.56	26.27	72.17
– w/o OCR (in RL)	-1.40	-2.00	-0.35
– w/o RL	-7.11	-4.17	-0.93
– w/o OCR (in SFT)	-8.12	-8.42	-1.23

Table 6: Ablation study showing the performance drop (%) relative to the final Glyph model when components are progressively removed.

Limitations

Sensitivity to rendering parameters.

- Model's performance noticeably affected by rendering configurations such as resolution, font, and spacing.

OCR-related challenges.

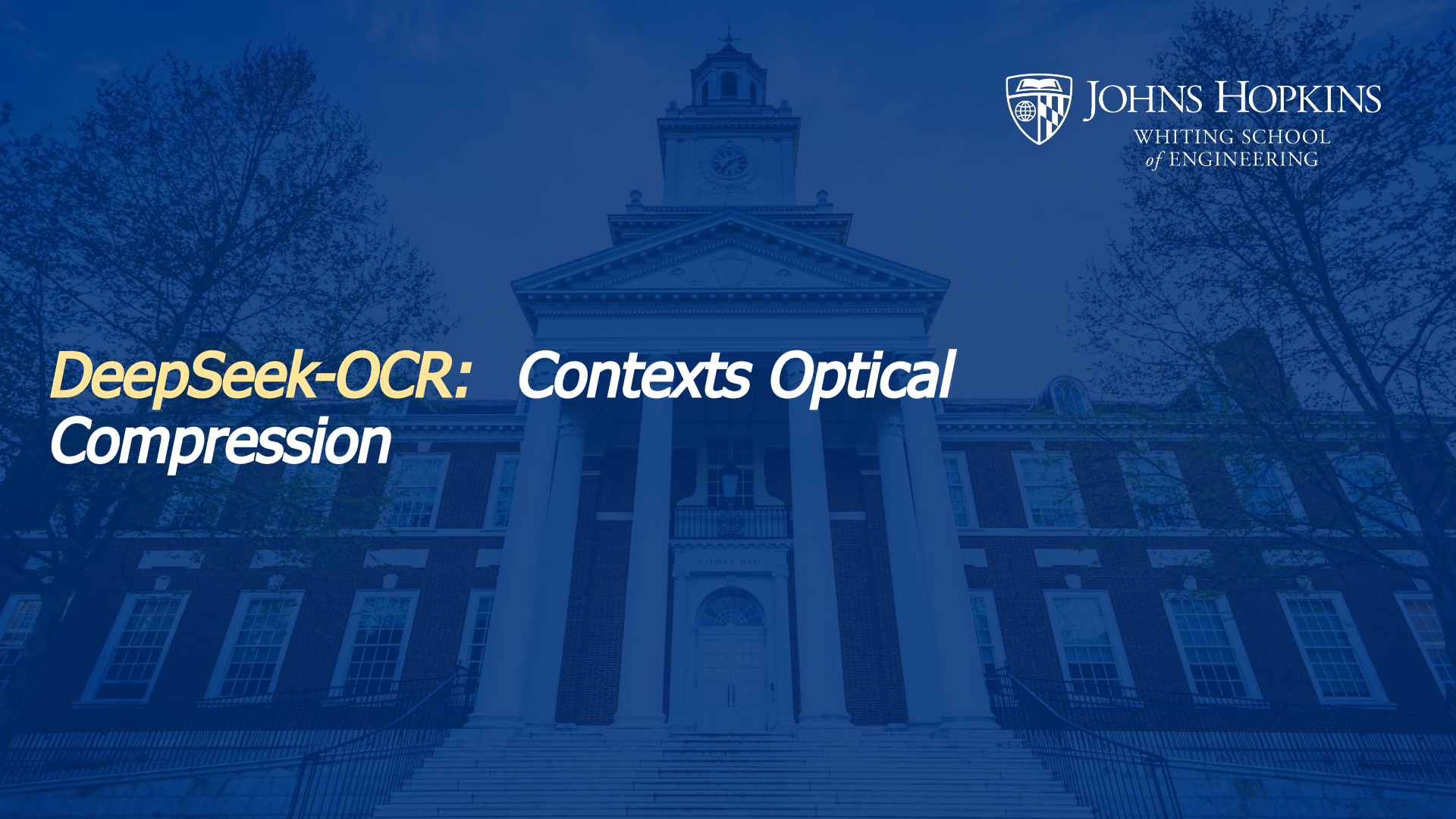
- UUID task especially challenging. (quote - even the strongest models [e.g., Gemini-2.5-Pro] often fail to reproduce them correctly.)

Task diversity.

- More tasks besides long-context understanding.
- e.g. agentic, reasoning.
- "We also observe that, compared with textual models, the visual-text model tends to generalize less effectively across tasks"

Questions?

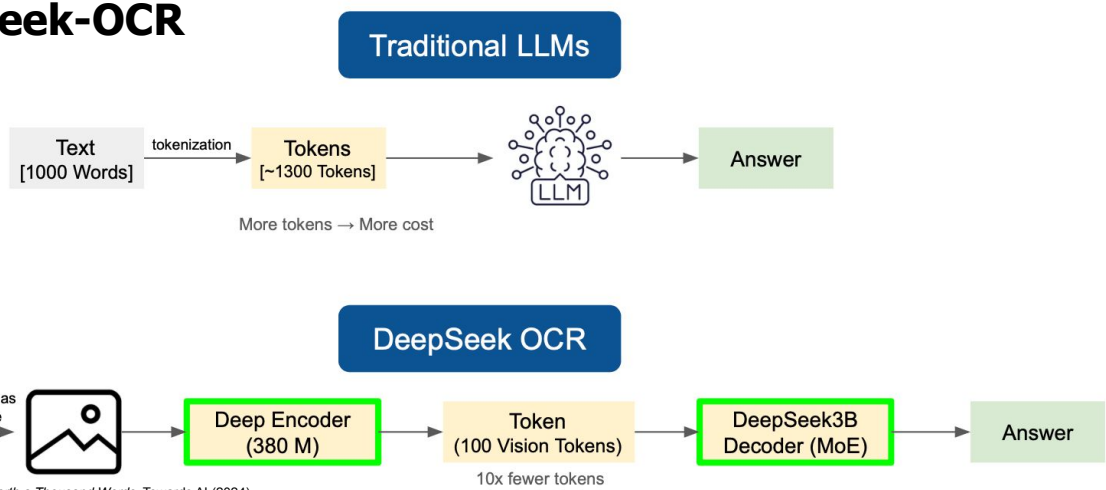
Is patch based visual token necessary or sufficient? (What are we actually gaining through using VLM, is it just a dense tokenization that encompass multiple token.)



DeepSeek-OCR: Contexts Optical Compression

DeepSeek-OCR: Introduction & Motivation

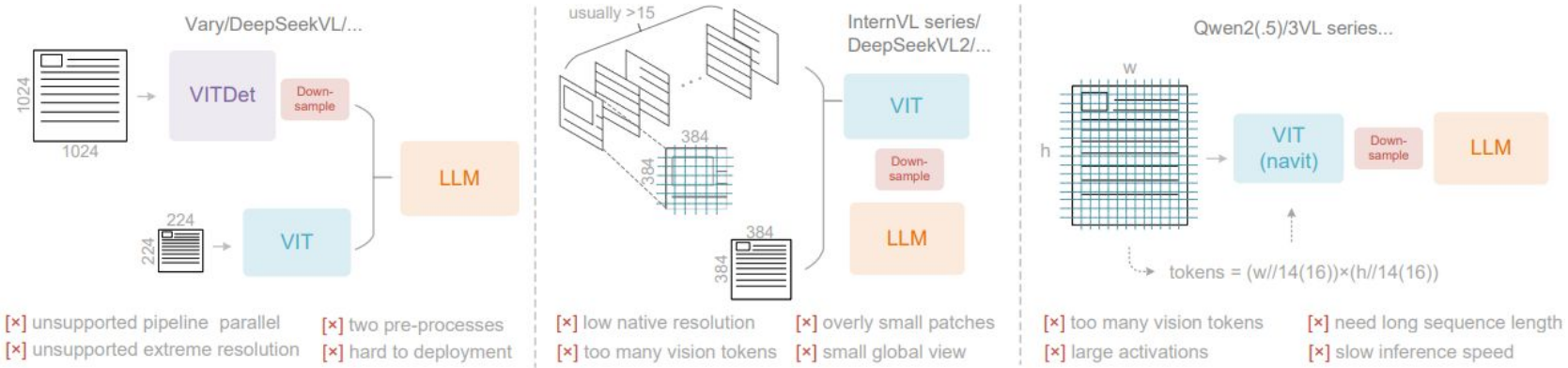
- Significant compute challenges with long textual content.
  - One solution: use single image containing document text → rich information while using substantially fewer tokens.
- OCR tasks establish natural compression-decompression mapping between visual and textual representations
  - Motivated **DeepSeek-OCR**



DeepSeek-OCR: A Picture is Worth a Thousand Words, Towards AI (2024)
https://pub.towardsai.net/deepseek-ocr-a-picture-is-worth-a-thousand-words-e2a8b9d74c7f

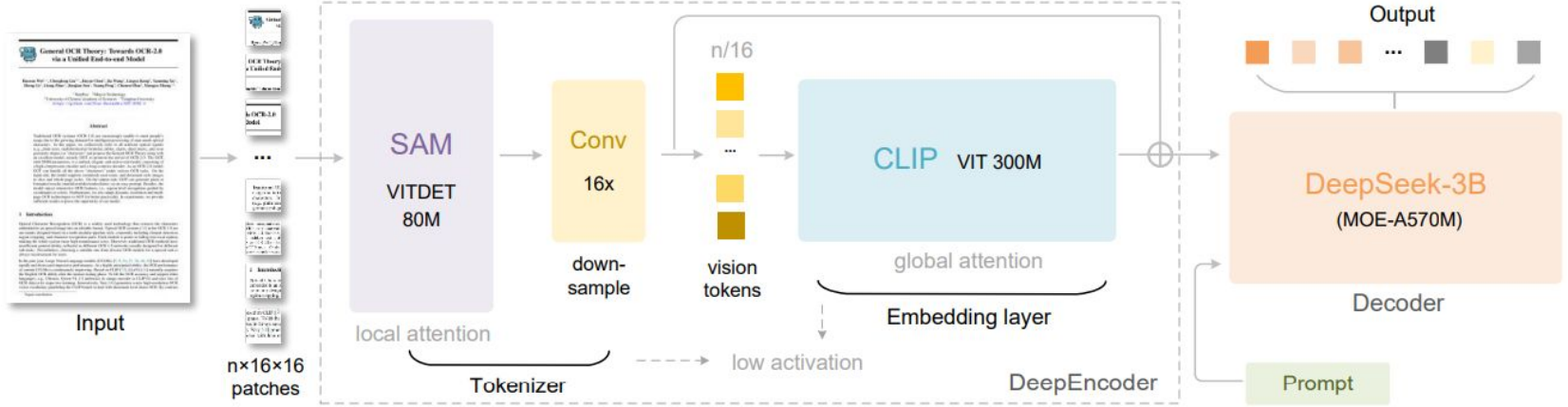
DeepSeek-OCR: Introduction & Motivation

- Types of Vision Encoders in current VLMs:
  - Dual-tower architecture: two parallel encoders
  - Tile-based method: split image into small tiles
  - Adaptive-resolution encoder: process full image using patch segmentation
- One question not addressed: What is the least amount of vision tokens we need to decode a document of x words?



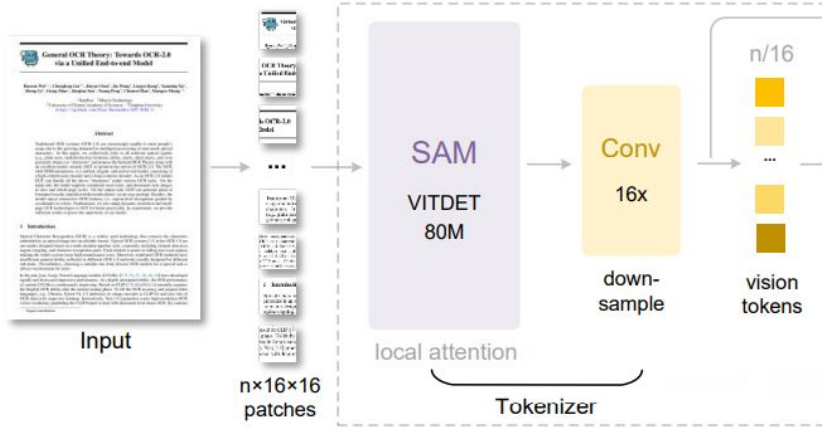
DeepSeek-OCR: Methodology

- Encoder three part architecture:
  - SAM base
  - 2 layer convolutional model (downsampler)
  - CLIP
- DeepSeek-3B MoE Decoder



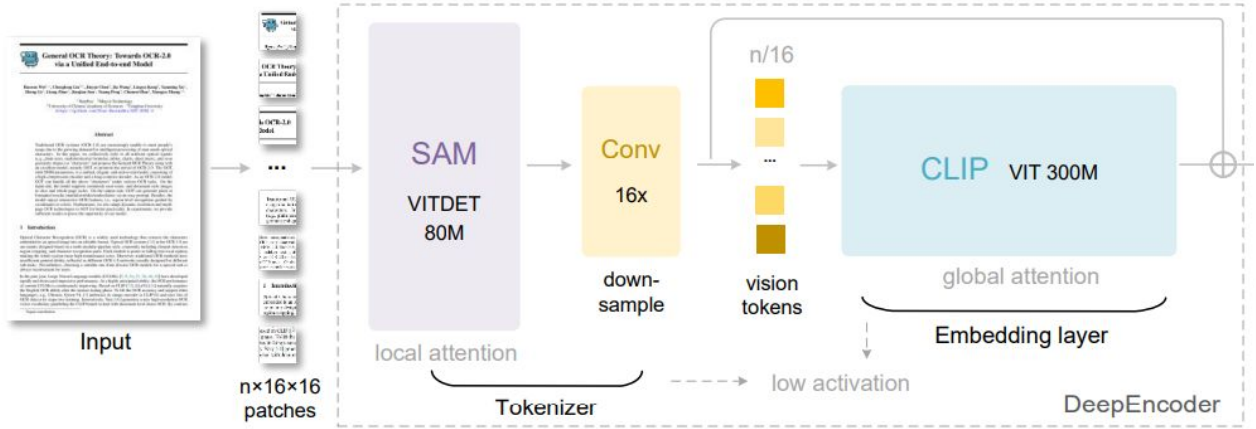
DeepSeek-OCR: Methodology

- Segment Anything Model (SAM) base: vision model designed to separate/segment regions or objects in an image and extract visual perception features.
- Breaks image into 16x16 patches and creates local feature tokens to then be sent into downsampling module → Downsampler reduces number of vision tokens



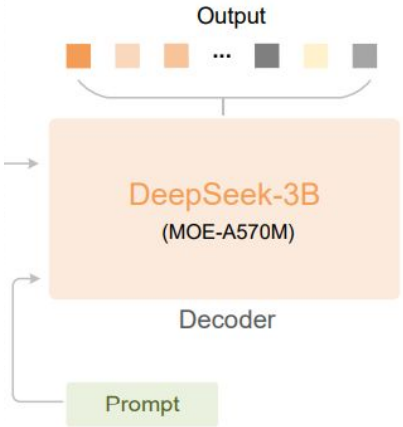
DeepSeek-OCR: Methodology

- CLIP: Embedding layer with global attention
  - Vision Encoder that learns to connect images and text.
  - The downsampled tokens are fed into CLIP which applies dense global attention, how pieces fit together as “text”



DeepSeek-OCR: Methodology

- During inference, model activates 8 out of 64 experts (specialized sub-networks)
- Z are the compressed latent (vision) tokens from DeepEncoder
- $\hat{X}$ hat is the reconstructed text representation.
- f_{dec} is a non linear mapping that can be learned.

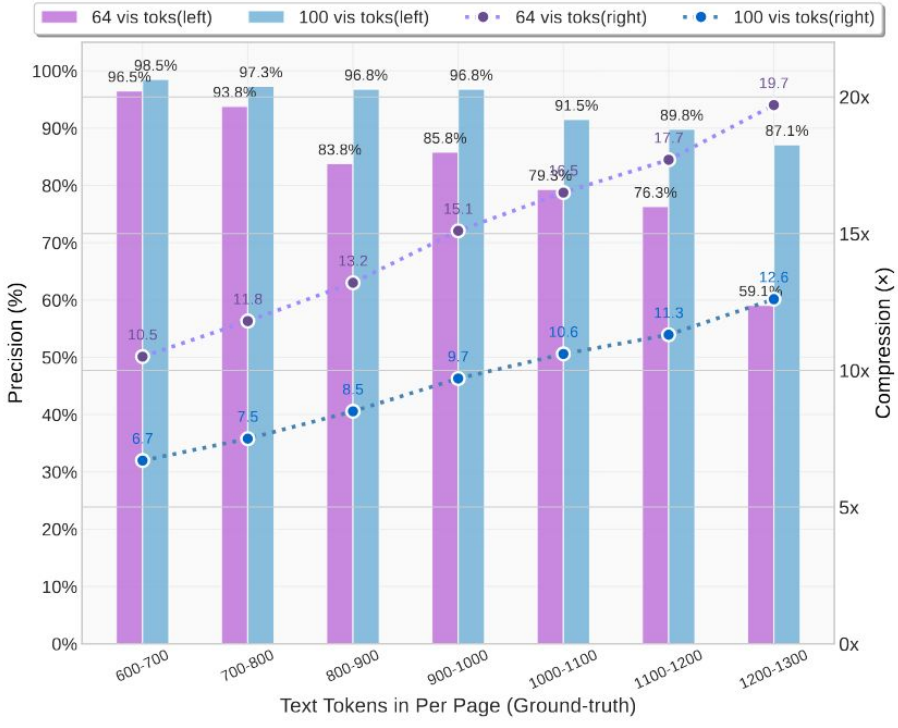


$$f_{\text{dec}} : \mathbb{R}^{n \times d_{\text{latent}}} \rightarrow \mathbb{R}^{N \times d_{\text{text}}}; \quad \hat{\mathbf{X}} = f_{\text{dec}}(\mathbf{Z}) \quad \text{where } n \leq N$$

DeepSeek-OCR: Data

- OCR 1.0 data: ~30M PDF pages + natural scene text (English, Chinese, ~100 languages)
- OCR 2.0 data: synthetic charts, geometry, and chemical formulas for structured parsing
- General vision data: small portion of image-caption, grounding, detection
- Text-only data: 10% of training to maintain language modeling ability

DeepSeek-OCR: Results

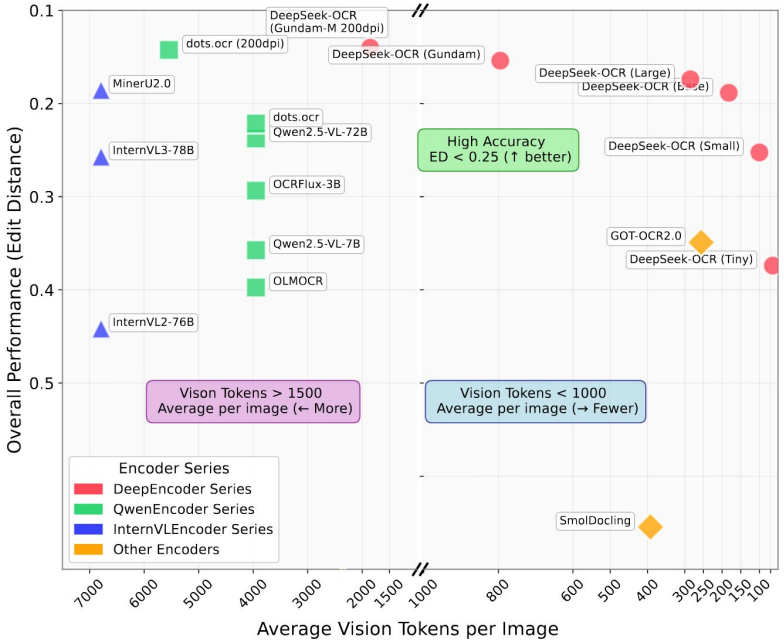


(a) Compression on Fox benchmark

- 97% OCR precision with text compressed by up to 10 times.
- 60% OCR precision even at a 20x text compression ratio
- Promising for handling long context

Text Tokens	Vision Tokens =64		Vision Tokens=100		
	Precision	Compression	Precision	Compression	Pages
600-700	96.5%	10.5×	98.5%	6.7×	7
700-800	93.8%	11.8×	97.3%	7.5×	28
800-900	83.8%	13.2×	96.8%	8.5×	28
900-1000	85.9%	15.1×	96.8%	9.7×	14
1000-1100	79.3%	16.5×	91.5%	10.6×	11
1100-1200	76.4%	17.7×	89.8%	11.3×	8
1200-1300	59.1%	19.7×	87.1%	12.6×	4

DeepSeek-OCR: Results



(b) Performance on Omnidocbench

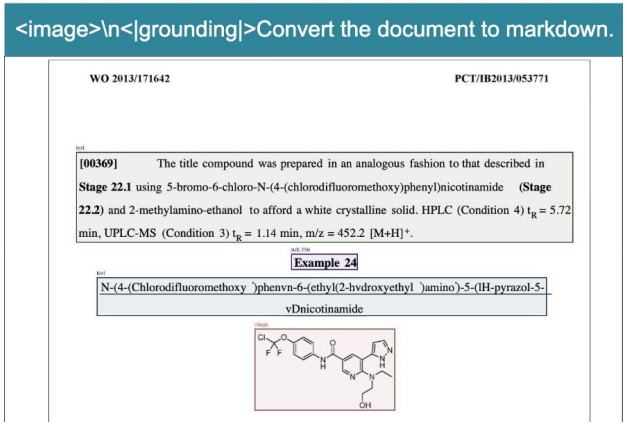
- Achieves **better** performance with **less** token needed compared to other model.
- Shows **adaptive** performance for diverse content.

Mode \ Type	Book	Slides	Financial Report	Textbook	Exam Paper	Magazine	Academic Papers	Notes	Newspaper	Overall
Tiny	0.147	0.116	0.207	0.173	0.294	0.201	0.395	0.297	0.94	0.32
Small	0.085	0.111	0.079	0.147	0.171	0.107	0.131	0.187	0.744	0.205
Base	0.037	0.08	0.027	0.1	0.13	0.073	0.052	0.176	0.645	0.156
Large	0.038	0.108	0.022	0.084	0.109	0.06	0.053	0.155	0.353	0.117
Gundam	0.035	0.085	0.289	0.095	0.094	0.059	0.039	0.153	0.122	0.083
Guandam-M	0.052	0.09	0.034	0.091	0.079	0.079	0.048	0.1	0.099	0.077

Edit distances for different categories of documents in OmniDocBench

DeepSeek-OCR: Results

- **Deep Parsing:** Extracts structured data from non-textual visual elements, such as converting text into Markdown, chemical formulas into SMILES strings, and geometric figures into structured data.
- **Multilingual Recognition:** Supports nearly 100 languages in PDF documents.
- **General Vision Understanding:** Describing image, object detection, grounding, etc.



DeepSeek-OCR: Discussion

- Simulates human memory forgetting by blurring and reducing tokens for older visual contexts, allowing distant memories to naturally fade.
- Enables scalable, theoretically unlimited context architectures.


JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING
Questions?
GILMAN HALL