Un-LOCC: Universal Lossy Optical Context Compression for Vision-Based Language Models

MaxDevv

October 23, 2025

Abstract

The quadratic computational complexity of the Transformer attention mechanism makes large context LLM inferences prohibitively costly. This project, inspired by the research of DeepSeek-OCR, investigates a practical method for LLM context compression by rendering text as an image. The core hypothesis is that an image, which has a fixed token cost for a Vision-Language Model (VLM), can serve as a highly compressed representation of a much larger body of text.

I introduce the "Optical Needle in a Haystack" (O-NIH) evaluation framework to systematically optimize and quantify the performance of this technique. Through extensive experimentation with variables including font family, color contrast, image resolution, and font size, I've identified optimal configurations. These results show that it is possible to achieve compression ratios approaching 3:1 (e.g., **2.8:1**) while maintaining over **93% retrieval accuracy**. This demonstrates that Optical Compression is a viable, plug-and-play strategy for drastically extending the effective context length and reducing the operational cost of existing VLMs.

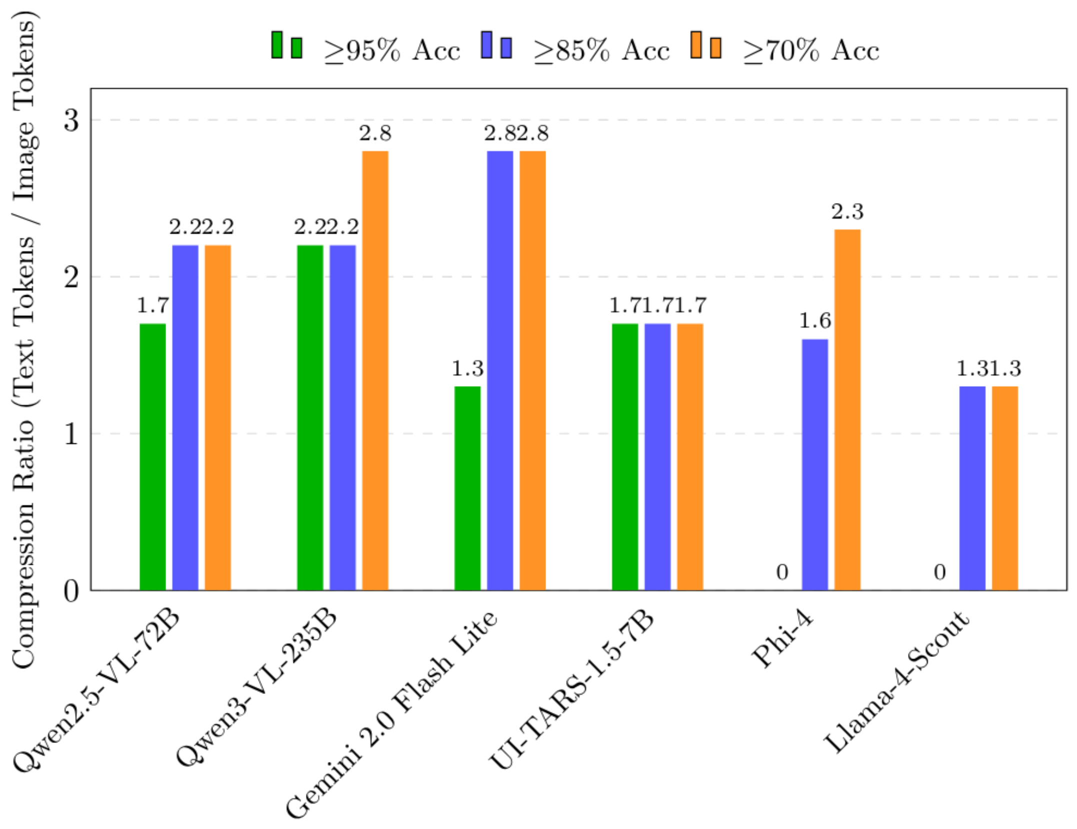

Figure 1: Maximum compression ratios achieved by top vision-language models at different accuracy thresholds.

1 Key Results

Analysis of over 90 experiments reveals that different models have unique "sweet spots" for optical compression[1]. However, a general trend emerges: high-fidelity compression is achievable across multiple state-of-the-art VLMs.

Table 1: Model Performance Overview

Model	Peak Performance (Max Accuracy)	High-Fidelity Compression (Max Ratio @ ≥85%)	High-Density Compression (Max Ratio @ ≥70%)
qwen/qwen2.5-vl-72b-instruct	**98.33%** @ 1.7:1 (Exp 44)	**2.2:1** @ 94.44% (Exp 81)	**2.2:1** @ 94.44% (Exp 81)
qwen/qwen3-vl-235b-a22b	**95.24%** @ 2.2:1 (Exp 50)	**2.2:1** @ 95.24% (Exp 50)	**2.8:1** @ 82.22% (Exp 90)
google/gemini-2.0-flash-lite	**100.0%** @ 1.3:1 (Exp 46)	**2.8:1** @ 93.65% (Exp 56)	**2.8:1** @ 93.65% (Exp 56)
bytedance/ui-tars-1.5-7b	**95.24%** @ 1.7:1 (Exp 72)	**1.7:1** @ 95.24% (Exp 72)	**1.7:1** @ 79.71% (Exp 88)
microsoft/phi-4-multimodal	**94.44%** @ 1.1:1 (Exp 59, 85)	**1.6:1** @ 91.11% (Exp 63)	**2.3:1** @ 73.55% (Exp 61)
meta-llama/llama-4-scout	**86.57%** @ 1.3:1 (Exp 53)	**1.3:1** @ 86.57% (Exp 53)	**1.3:1** @ 86.57% (Exp 53)

1.1 Per-Model Optimization Summary

While optimal variables vary, this research provides a strong starting point for applying this technique:

- **Image Size:** The best results are often achieved when the input image resolution is slightly smaller than the model's maximum supported single-tile resolution. An image size of **864x864 pixels** proved to be an excellent default.
- **Font Size:** A font size between **12px and 16px** is generally optimal. For `qwen/qwen2.5-vl-72b`, 13px was the clear winner, while `gemini-2.0-flash-lite` performed well with 12px.
- **Font Family:** High-legibility, sans-serif fonts are critical. Top performers included:
    1. **Atkinson Hyperlegible** (especially the Italic variant)
    2. **Lato**
    3. **Lexica Ultralegible**
- **Color Contrast:** Standard black-on-white or yellow-on-blue perform best. Low-contrast pairs (e.g., red-on-blue) fail completely.

[1]Detailed raw logs of the 90+ experiments (the Appendix) have been excluded from this document for brevity. They can be found in the `README.md` of the actual GitHub repository: `https://github.com/MaxDevv/Un-LOCC`.

2 The O-NIH Methodology

The core challenge of this research was to create a reliable method for testing a model's text processing capabilities across different image configurations. To solve this, I developed the **Optical Needle in a Haystack (O-NIH)** test.

The O-NIH test works by injecting a unique, non-contextual code (the "needle") into a large, cohesive body of text (the "haystack"). The VLM is then tasked with retrieving only that code. The haystack and needle are prepared programmatically:

```python
import random

def generate_needle(length=9):
    """Generates a random, unique code (the 'needle')."""
    chars = 'ABCDEFGHJKLMNPQRSTUVWXYZ23456789'
    result = ''
    for i in range(length):
        result += random.choice(chars)
        if (i + 1) % 3 == 0 and i < length - 1:
            result += '-'
    return result

def prepare_text(source_text, word_count):
    """Extracts a random chunk of text and injects the needle."""
    words = source_text.split()
    if len(words) < word_count:
        raise ValueError(f"Source text is too short.")

    start_index = random.randint(0, len(words) - word_count)
    text_chunk = words[start_index : start_index + word_count]

    needle = generate_needle()
    # Inject the needle at a random position within the text chunk
    injection_index = random.randint(1, len(text_chunk) - 1)
    text_chunk.insert(injection_index, needle)

    haystack = " ".join(text_chunk)
    return haystack, needle
```

Listing 1: Needle Generation and Injection

The model's accuracy is then graded using a "fuzzy" comparison that accounts for common OCR errors (e.g., mistaking 'I' for '1') by calculating the Levenshtein distance.

```python
def normalize_for_ocr(text):
    """Normalizes a string to account for common OCR errors."""
    replacements = {
        'O': '0', 'o': '0', 'I': '1', 'l': '1', 'S': '5', 's': '5',
        'B': '8', 'A': '4', '-': '', ' ': ''
    }
    text = text.upper()
    for old, new in replacements.items():
        text = text.replace(old, new)
    return text

def fuzzy_similarity(s1, s2):
    """Calculates a normalized similarity score based on Levenshtein distance."""
    norm_s1 = normalize_for_ocr(s1)
    norm_s2 = normalize_for_ocr(s2)
    # Standard Levenshtein implementation omitted for brevity
    distance = levenshtein_distance(norm_s1, norm_s2)
    max_len = max(len(norm_s1), len(norm_s2))
    if max_len == 0: return 1.0
    return (max_len - distance) / max_len
```

Listing 2: Fuzzy Similarity Scoring

3 Limitations

- **Contextual Understanding:** This technique tests perceptual retrieval, not deep contextual understanding. While a model can *find* text, its ability to reason over the optically compressed context may be different than with standard text tokens.
- **Prompt Injection Risk:** This technique could potentially be used to circumvent safety filters by rendering harmful text within an image.

4 Vision & Future Work

The field of Lossy Optical Context Compression offers an immediate path to reducing costs and increasing context length for developers without fine-tuning.

The strong performance of smaller models like **Phi-4 Multimodal** opens a particularly exciting avenue for **on-device context expansion**. By offloading historical context to an optically compressed image, it may be possible to run agents with vast memory on edge devices with limited VRAM.

I foresee a series of libraries that wrap around existing API and SDK interactions to make large-context, low-cost VLM inference easily accessible.

5 Acknowledgements

This project would not have been possible without the foundational ideas presented in the **DeepSeek-OCR** paper ("Contexts Optical Compression"). Their work provided the initial spark that demonstrated the feasibility of representing large amounts of text in a compressed optical format.

6 Code and Data Availability

To facilitate reproducibility and further research, all resources are open-source:

- **Research Data & Logs:** The full dataset of 90+ experiments and original research notes can be found at `https://github.com/MaxDevv/Un-LOCC`.
- **Python Implementation:** A usable wrapper library for implementing O-NIH tests and optical compression is available at `https://github.com/MaxDevv/Un-LOCC-Wrapper`.