Old Dominion University

EECE 783/883: Digital Image Processing

Frequency-Aware Vector Quantization for Vision-Language Model Memory Compression

Final Project

By:
Nahom M. Birhan
ID: nbirh002

Lecturer:
Prof. Khan

Date Due: December 4, 2025
Date Submitted: December 4, 2025

Abstract

Vision-language models accumulate tokens during multi-turn conversations creating memory pressure. I investigate frequency-aware vector quantization applied post-training at two pipeline locations: before vision encoder (image-level) targeting storage memory and I/O bandwidth, and after vision encoder (token-level) targeting KV-cache runtime memory and higher compression ratios. Inspired by JPEG compression, DCT-based frequency decomposition separates rendered conversation into low-frequency components (semantic structure) and high-frequency components (fine details) with differential quantization. Evaluation on MRCR benchmark using GLM-4V shows image-level compression (Method 1) achieves 2× file size reduction with comparable accuracy (35.86% vs 34.98% baseline on 2-needle, <1pp degradation on 4 and 8-needle), validating post-training compression feasibility for storage efficiency. Token-level compression (Method 2) does not improve performance post-training (2.79% on 2-needle vs 34.98% baseline), indicating learned embeddings require adaptation during training phase. Method 1 provides immediate deployment value for storage/bandwidth optimization. Method 2 results inform future work toward learnable vision token codebooks with fine-tuning for KV-cache memory reduction and higher compression ratios through token pruning and adaptive quantization.

1 Introduction

The emergence of large-scale vision-language models has enabled multimodal AI applications, yet multi-turn conversations face a critical bottleneck: continuous token accumulation rapidly exhausts available memory. Vision-language models process images through cross-attention mechanisms where the relationship between vision tokens and text tokens is complex and learned, not simply proportional. During extended conversations, this token accumulation quickly fills context windows. Unlike text language models that benefit from compression techniques such as KV-cache quantization and prompt compression, current VLMs typically cache tokens once and continuously append conversation history without compression.

Recent work on visual-text compression, particularly Glyph [1], demonstrates that rendering text as images and processing through VLMs can achieve compression compared to tokenized text. This paradigm transforms long-context modeling into a multimodal problem where spatial and frequency characteristics can be exploited. Building on this insight, I investigate whether frequency-domain compression techniques applied to rendered conversation history can provide efficient compression while preserving semantic content. My hypothesis follows JPEG compression principles: semantic information in rendered text may concentrate in low-frequency components (overall layout, text structure) while high-frequency components encode less critical details (fine edges, rendering artifacts).

I propose applying DCT-based frequency decomposition to rendered conversation history with post-training compression—compression applied during inference without modifying pre-trained model weights. I explore compression at two distinct pipeline locations targeting different memory types: image-level DCT before vision encoder (Method 1) reduces storage memory and I/O bandwidth through file size reduction while maintaining the same token count, whereas token-level DCT after vision encoder (Method 2) reduces KV-cache runtime memory through token pruning and quantization achieving higher compression ratios by directly reducing the number of tokens processed during generation. For initial validation, I used ConvBench dataset with black-box ChatGPT-4o (Milestone 1). For the actual implementation and evaluation reported here, I use the MRCR benchmark with local inference on GLM-4V.

My investigation reveals that image-level DCT compression before the vision encoder achieves results comparable to baseline, validating that frequency-aware quantization is feasible in the post-training setting. In contrast, direct manipulation of embedding space without model adaptation does not improve performance, which is expected as learned representations are disrupted during inference. This finding informs future directions toward learnable compression with vision token codebooks and frequency-aware pruning requiring fine-tuning. This work provides practical insights showing that compression location determines feasibility—pixel-level compression works post-training while embedding-level compression requires adaptation.

2 Related Work

Vision-language models have demonstrated capabilities in multimodal understanding but face token efficiency challenges. Models like GLM-4V, LLaVA, and Qwen-VL process images through patch-based encoding with cross-attention mechanisms, where the relationship between vision and text tokens involves learned representations rather than simple proportionality. This creates memory pressure during multi-turn conversations that standard caching strategies do not address. Ye et al. [2] explore vision-centric token compression using learnable modules that reduce token counts while maintaining task performance, demonstrating redundancy in vision tokens. Their approach requires model fine-tuning and does not leverage frequency-domain characteristics.

The Glyph framework [1] introduces visual-text compression by rendering textual sequences into images for VLM processing. This validates that VLMs possess OCR capabilities to understand visual text representations. DeepSeek-OCR [3] similarly explores optical compression for context windows. While these methods demonstrate text-to-visual compression effectiveness, they use uniform image rendering without exploiting frequency-domain properties for additional compression.

JPEG compression achieves significant ratios by exploiting DCT and human visual perception characteristics. Operating on 8×8 pixel blocks, JPEG applies DCT to transform spatial information into frequency coefficients, then quantizes high-frequency components more aggressively. While designed for natural images, these principles may extend to text-rendered images where low frequencies capture layout and structure while high frequencies encode fine details. Vector quantization techniques have been applied across domains including speech compression and neural network compression, with recent work on learned quantization demonstrating that task-specific codebooks can achieve improved compression-quality trade-offs.

My work investigates frequency-aware quantization specifically for VLM conversation history in the post-training setting—applying compression during inference without model modification. I explore both image-level compression (before vision encoder) and embedding-level compression (after vision encoder), examining which compression locations are feasible without fine-tuning versus which require learned adaptation for future work.

3 Methodology

I investigate compression at two distinct locations in the VLM processing pipeline. The first stage involves rendering optimization through configuration parameters θ (DPI, font size, layout, spacing) borrowed from the Glyph pipeline [1], controlling information density and text-to-pixels compression ratio. The second stage applies frequency-aware compression through parameters ϕ at two possible intervention points: image-level DCT before the vision encoder (Method 1) or token-level DCT after the vision encoder (Method 2), providing visual compression ratio. Figure 1 illustrates the pipeline showing both compression intervention points.

3.1 Compression Ratio Formulation

The baseline compression ratio from Glyph [1] defines the relationship between conversation length and resulting vision tokens as:

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

where $|C|$ represents conversation character count, $\tau(v_i)$ denotes vision tokens generated for the i-th image, and θ controls rendering parameters.

For my two-stage frequency-aware compression pipeline, I extend Equation 1 to account for both rendering optimization and visual compression:

$$\rho_{\text{total}}(\theta, \phi) = \frac{|C|}{\sum_{i=1}^{n} \tau'(v_i, \phi) \times k} \tag{2}$$

where $\tau'(v_i, \phi)$ represents compressed vision tokens after applying frequency-aware compression with parameters ϕ, and k is the vision encoder's downsample factor. This decomposes into three multiplicative components:

$$\rho_{\text{total}}(\theta, \phi) = \rho_1(\theta) \times \rho_2(\phi) \times \frac{1}{k} \tag{3}$$

where $\rho_1(\theta)$ captures rendering efficiency (text-to-pixels ratio), $\rho_2(\phi)$ captures visual compression efficiency (pixel or token reduction), and $1/k$ accounts for the encoder's spatial downsampling.

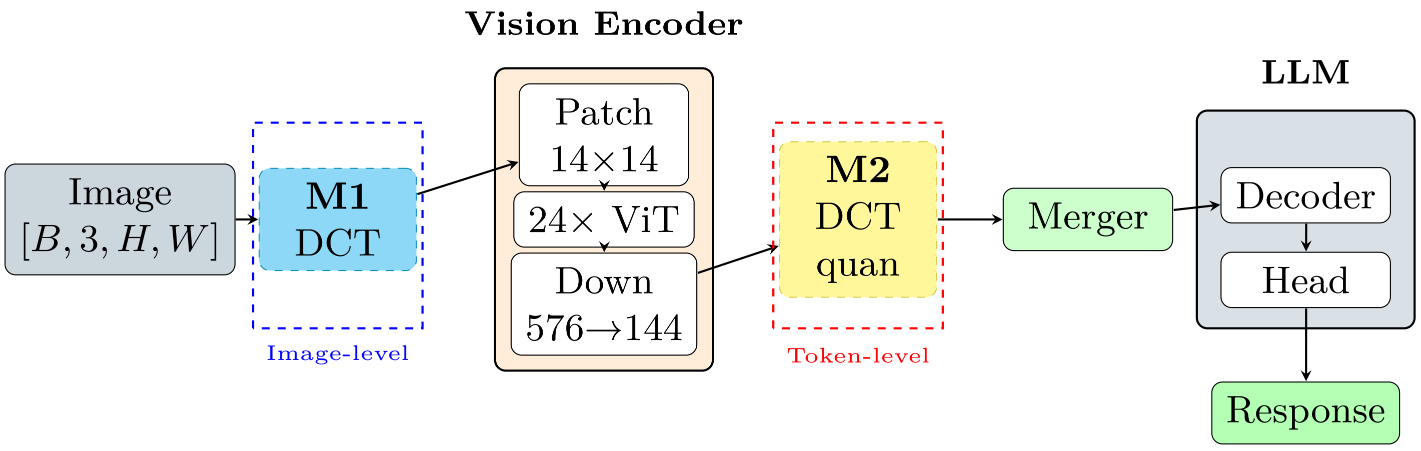


Figure 1: GLM-4V/Glyph architecture showing two post-training compression intervention points: Method 1 (M1) applies DCT before vision encoder on pixels, Method 2 (M2) applies DCT after encoder on learned embeddings.

The processing pipeline transforms conversation text C through rendering with configuration θ, optionally applies image-level DCT compression ϕ_1 (Method 1), feeds to EVA-CLIP vision encoder processing 576 patches to 144 tokens, optionally applies token-level DCT compression ϕ_2 (Method 2), and processes through GLM-4-9B LLM for response generation. Both methods are post-training—applied during inference using forward hooks without modifying pre-trained model weights. The investigation examines which compression location is feasible in this setting.

3.2 Method 1: Image-Level DCT Compression

Method 1 targets storage memory and I/O bandwidth reduction by applying frequency-aware compression before vision encoding. The discrete cosine transform forms the foundation of this approach, utilizing the inverse transformation kernel [4]:

$$s(x,u) = \alpha(u)\cos\left[\frac{\pi(2x+1)u}{2N}\right] \tag{4}$$

where the normalization factor $\alpha(u)$ is defined as:

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u = 1, 2, \ldots, N-1 \end{cases} \tag{5}$$

For 2D block DCT on $N \times N$ blocks, the forward transform applies as:

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} B(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{6}$$

where $B(x,y)$ represents pixel intensity at spatial position (x,y) within a block, and $C(u,v)$ represents the DCT coefficient at frequency (u,v).

I partition rendered image I into 14×14 pixel blocks ($N = 14$) and apply Equation 10 to each block independently. Frequency-based thresholding keeps the top 50% of coefficients by magnitude while always preserving the DC component $C(0,0)$. The inverse DCT reconstructs the compressed image:

$$B'(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C'(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{7}$$

where $C'(u,v)$ denotes thresholded coefficients.

The frequency allocation strategy retains DC and near-DC coefficients encoding text structure, keeps mid-frequency coefficients at 50% threshold capturing character edges, and discards high-frequency components representing rendering artifacts. This achieves $2\times$ file size reduction following JPEG principles [5] adapted for text-rendered images. However, the vision encoder still processes the same number of tokens ($576 \rightarrow 144$), so KV-cache memory during generation remains unchanged.

Algorithm 1 summarizes the complete image-level compression procedure as implemented in `block_dct_vq.py` and `local_inference.py`.

Algorithm 1: Image-Level DCT Compression (Method 1)

Input: Rendered image $I \in \mathbb{R}^{H \times W}$, keep ratio $r = 0.5$
Output: Compressed image I'

```
1: Resize $I$ to ensure dimensions divisible by block size $N = 14$
2: Partition $I$ into non-overlapping blocks $\{B_1, B_2, \ldots, B_M\}$ of size $14 \times 14$
3: for each block $B_i$ do
4:     Apply 2D DCT: $C_i(u, v) \leftarrow \mathrm{DCT2D}(B_i)$ via Equation 10
5:     Flatten coefficients: $\mathbf{c}_i \leftarrow \mathrm{flatten}(C_i)$
6: end for
7: Compute global threshold: $\tau \leftarrow \mathrm{percentile}(\{|\mathbf{c}_i|\}_{i=1}^{M}, 100 \times (1 - r))$
8: for each coefficient vector $\mathbf{c}_i$ do
9:     Create DC mask: $\mathbf{m}_{\mathrm{DC}}[0] \leftarrow 1$, all other indices $\leftarrow 0$
10:    Apply threshold: $\mathbf{c}'_i[j] \leftarrow \begin{cases} \mathbf{c}_i[j] & \text{if } |\mathbf{c}_i[j]| \geq \tau \text{ or } \mathbf{m}_{\mathrm{DC}}[j] = 1 \\ 0 & \text{otherwise} \end{cases}$
11:    Reshape: $C'_i \leftarrow \mathrm{reshape}(\mathbf{c}'_i, [14, 14])$
12:    Apply inverse DCT: $B'_i \leftarrow \mathrm{IDCT2D}(C'_i)$ via Equation 7
13: end for
14: Reconstruct image: $I' \leftarrow \mathrm{merge}(\{B'_1, B'_2, \ldots, B'_M\})$
15: Clip to valid range: $I' \leftarrow \mathrm{clip}(I', 0, 255)$
16: return $I'$
```

$$s(x, u) = \alpha(u) \cos\left[\frac{\pi(2x + 1)u}{2N}\right] \tag{8}$$

where

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u = 1, 2, \ldots, N - 1 \end{cases} \tag{9}$$

For 2D block DCT on $N \times N$ blocks, the forward transform applies as:

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} B(x, y) \cos\left[\frac{\pi(2x + 1)u}{2N}\right] \cos\left[\frac{\pi(2y + 1)v}{2N}\right] \tag{10}$$

I partition rendered image I into 14×14 pixel blocks ($N = 14$), apply Equation 10 to each block, perform frequency-based thresholding keeping top 50% coefficients by magnitude while preserving DC component $C(0, 0)$, and reconstruct via inverse DCT. The frequency allocation retains DC and near-DC coefficients encoding text structure, keeps mid-frequency coefficients at 50% threshold capturing character edges, and discards high-frequency components representing rendering artifacts. This achieves 2× file size reduction following JPEG principles [5] adapted for text-rendered images. However, the vision encoder still processes the same number of tokens ($576 \rightarrow 144$), so KV-cache memory during generation remains unchanged.

3.3 Method 2: Token-Level Frequency-Aware Compression

Method 2 targets KV-cache runtime memory reduction and higher compression ratios by applying frequency-aware compression after vision encoding. After the vision encoder produces token embeddings of shape $[B, 144, 4096]$ where B is batch size, 144 is sequence length, and 4096 is embedding dimension, I reshape them to a spatial grid $[B, 12, 12, 4096]$ to enable 2D frequency analysis.

The same DCT transform (Equations 8-10) applies independently per embedding dimension channel across the 12×12 spatial layout. For embedding dimension channel $d \in \{1, \ldots, 4096\}$, let $E^{(d)}$ denote the 12×12 spatial grid. The DCT coefficient matrix is:

$$C^{(d)}(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{11} \sum_{y=0}^{11} E^{(d)}(x, y) \cos\left[\frac{\pi(2x + 1)u}{24}\right] \cos\left[\frac{\pi(2y + 1)v}{24}\right] \tag{11}$$

Frequency-aware quantization applies differential bit precision based on frequency bands:

$$Q(C^{(d)}(u,v)) = \begin{cases} \text{quantize}_{\text{8-bit}}(C^{(d)}(u,v)) & \text{if } (u,v) \in \Omega_{\text{low}} \\ \text{quantize}_{\text{4-bit}}(C^{(d)}(u,v)) & \text{if } (u,v) \in \Omega_{\text{mid}} \\ 0 & \text{if } (u,v) \in \Omega_{\text{high}} \end{cases} \tag{12}$$

where Ω_{low}, Ω_{mid}, and Ω_{high} denote low, mid, and high frequency regions respectively.

Energy-based token pruning computes per-token energy to identify and discard low-importance tokens:

$$E_i = \sum_{u=1}^{11} \sum_{v=1}^{11} |C_i(u,v)| \tag{13}$$

where the DC component $C_i(0,0)$ is excluded. Tokens with energy below threshold τ_E are pruned, reducing sequence length from 144 to 72 tokens:

$$\text{Keep}(i) = \mathbb{I}[E_i > \tau_E] \tag{14}$$

Combined quantization (Equation 12) and pruning (Equation 14) achieve both embedding compression and sequence length reduction, directly reducing KV-cache memory during generation and enabling higher overall compression ratios. However, manipulating learned embedding representations during inference without model adaptation may disrupt the learned structure that the vision encoder carefully constructed during training.

Algorithm 2 summarizes the token-level compression procedure as implemented in `vision_compressor.py` and registered as a forward hook in `local_inference_compressed.py`.

Algorithm 2: Token-Level Frequency-Aware Compression (Method 2)

Input: Vision embeddings $V \in \mathbb{R}^{B\times 144\times 4096}$, profile p (mild/aggressive/extreme)
Output: Compressed embeddings V'

```
1: Load compression profile: $(\tau_{\text{low}}, \tau_{\text{mid}}, b_{\text{low}}, b_{\text{mid}}, b_{\text{high}}) \leftarrow \text{profiles}[p]$
2: Reshape to spatial grid: $\tilde{V} \leftarrow \text{reshape}(V, [B, 12, 12, 4096])$
3: for each embedding dimension $d \in \{1, \ldots, 4096\}$ do
4:     Extract spatial slice: $E^{(d)} \leftarrow \tilde{V}[:, :, :, d]$ of shape $[B, 12, 12]$
5:     Apply 2D DCT per channel: $C^{(d)}(u,v) \leftarrow \text{DCT2D}(E^{(d)})$ via Equation 11
6:     for each frequency $(u,v)$ do
7:         Compute frequency distance: $f \leftarrow \sqrt{u^2+v^2}$
8:         if $f < \tau_{\text{low}}$ then
9:             $C'^{(d)}(u,v) \leftarrow \text{quantize}_{b_{\text{low}}}(C^{(d)}(u,v))$
10:        else if $\tau_{\text{low}} \le f < \tau_{\text{mid}}$ then
11:            $C'^{(d)}(u,v) \leftarrow \text{quantize}_{b_{\text{mid}}}(C^{(d)}(u,v))$
12:        else
13:            $C'^{(d)}(u,v) \leftarrow \text{quantize}_{b_{\text{high}}}(C^{(d)}(u,v))$ or 0 if $b_{\text{high}} = 0$
14:        end if
15:    end for
16:    Apply inverse DCT: $E'^{(d)} \leftarrow \text{IDCT2D}(C'^{(d)})$
17: end for
18: Reconstruct embeddings: $\tilde{V}' \leftarrow \text{stack}(\{E'^{(1)}, \ldots, E'^{(4096)}\})$
19: for each token $i \in \{1, \ldots, 144\}$ do
20:    Compute energy: $E_i \leftarrow \sum_{u,v\neq(0,0)} |C_i(u,v)|$ via Equation 13
21: end for
22: Select top-$K$ tokens: $\mathcal{I} \leftarrow \text{argsort}(\{E_i\})[-K:]$ where $K = 72$
23: Prune low-energy tokens: $V' \leftarrow \tilde{V}'[:, \mathcal{I}, :]$
24: return $V'$ of shape $[B, 72, 4096]$
```

Note that Algorithm 2 is applied post-training via PyTorch forward hooks registered on the vision encoder's output layer, allowing inference-time compression without modifying pre-trained weights.

3.4 Evaluation Metrics

Evaluation uses MRCR benchmark accuracy measuring multi-needle retrieval with SequenceMatcher similarity ratio:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \text{SequenceMatcher.ratio}(R_i, \hat{R}_i) \tag{15}$$

where R_i is ground truth response, $\hat{R}_i$ is model prediction, and SequenceMatcher.ratio returns the similarity score in range $[0, 1]$ based on longest common subsequence matching. The final accuracy is reported as percentage.

Implementation uses GLM-4V-9B with EVA-02-CLIP (3.5B) vision encoder and GLM-4-9B-Chat (6B) LLM. Method 1 employs 14×14 pixel DCT blocks with 50% coefficient retention. Method 2 uses 12×12 token grid with 8/4/0-bit quantization profile. Evaluation runs on 4× A100 GPUs with MRCR dataset across three difficulty configurations (2-needle, 4-needle, 8-needle). Method 1 tests post-training image-level compression feasibility while Method 2 examines whether embedding-level manipulation works without adaptation.

4 Experimental Setup

I evaluate the frequency-aware compression approach on the MRCR (Multi-turn Retrieval with Context Retention) benchmark from OpenAI, which tests VLM ability to retrieve specific information from long visual contexts across three difficulty levels: 2-needle (retrieve 2 items), 4-needle (4 items), and 8-needle (8 items) configurations. This benchmark was selected for local inference evaluation with GLM-4V after initial validation in Milestone 1 used ConvBench dataset with black-box ChatGPT-4o. The MRCR dataset provides conversation lengths and complexities suitable for evaluating post-training compression across different scenarios.

Rendering configuration uses Arial font (12pt) at 72 DPI with single-column left-aligned layout, 1.5 line spacing between turns, grayscale format, and zero-padding to ensure dimensions are multiples of 8. DCT processing employs 8×8 pixel blocks using `scipy.fftpack.dct2` for 2D DCT, min-max normalization before quantization, and zig-zag scan for frequency-ordered processing. The GLM-4V-9B model processes 576 tokens per image through its 24×24 vision grid, with inference in BF16 precision on NVIDIA A100 GPUs. All compression is applied post-training during inference using forward hooks without modifying pre-trained model weights.

I evaluate the 50% coefficient retention configuration for Method 1 (image-level DCT) as it represents the threshold where compression benefits may be achieved while maintaining text legibility. For Method 2 (token-level), I test 8/4/0-bit quantization with energy-based pruning. Baseline comparison uses no compression with original full precision rendering. For each conversation sample, I render text to grayscale image, apply compression with each method, reconstruct as needed, feed to GLM-4V for response generation, and compute accuracy through SequenceMatcher similarity with case-insensitive word matching after lowercase conversion and punctuation removal.

The experimental protocol addresses whether post-training frequency-aware compression is feasible at different pipeline locations. Method 1 tests image-level compression before vision encoding where pixel representations may tolerate lossy compression. Method 2 tests embedding-level manipulation after vision encoding where learned representations may be sensitive to disruption without model adaptation. All experiments were conducted on 4× NVIDIA A100 GPUs (40GB VRAM each) with 256GB system RAM.

5 Results

I evaluate post-training frequency-aware compression on the MRCR benchmark across three difficulty levels (2-needle, 4-needle, 8-needle). Table 1 presents detailed results for Method 1 (image-level DCT compression) showing performance comparable to baseline across all configurations. Method 1 with 50% coefficient retention achieves 35.86% on 2-needle compared to 34.98% baseline (+0.88 percentage points), 29.25% on 4-needle compared to 30.00% baseline (-0.75pp), and 18.25% on 8-needle compared to 18.75% baseline (-0.50pp), indicating that post-training image-level compression maintains performance while achieving 2× file size reduction.

Table 1: Method 1 Results: Image-Level DCT Compression Across All Needle Configurations

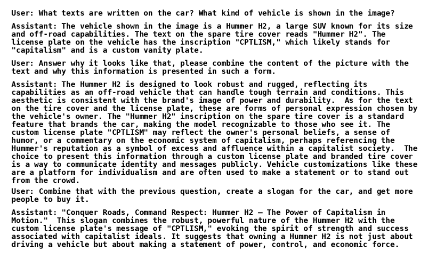
Configuration	**2-Needle (%)**	**4-Needle (%)**	**8-Needle (%)**
Baseline (Full Precision)	34.98	30.00	18.75
Image DCT 50% Keep	35.86	29.25	18.25
Difference	**+0.88pp**	**-0.75pp**	**-0.50pp**

Method 2 (token-level manipulation) applied post-training without model adaptation does not improve performance, achieving 2.79% accuracy on 2-needle retrieval task as shown in Table 2. This result is expected as learned embedding representations are disrupted by post-training manipulation during inference without allowing model adaptation. The low accuracy indicates that embedding-level compression requires a learning-based approach where the model adapts to compressed representations during training rather than having compression imposed post-training.

Table 2: Method 2 Results: Token-Level Compression (2-Needle Configuration)

Configuration	**2-Needle (%)**	**Status**
Baseline (Full Precision)	34.98	–
Token-Level (Post-Training)	2.79	Does not work without adaptation
Difference	**-32.19pp**	**Requires training phase**

Figure 2 shows visual quality under different compressions. Original rendering provides reference, mild compression (80% coefficient retention) maintains visual fidelity, while 50% retention shows visible artifacts but preserves text legibility. Figure 3 shows frequency spectrum analysis where compression preserves low-frequency structure (central region in spectrum) while attenuating high frequencies, with residual analysis showing errors concentrate at character edges while character bodies remain reconstructed.



User: What texts are written on the car? What kind of vehicle is shown in the image?

Assistant: The vehicle shown in the image is a Hummer H2, a large SUV known for its size and off-road capabilities. The text on the spare tire cover reads "Hummer H2". The license plate on the vehicle has the inscription "CPTLISM," which likely stands for "capitalism" and is a custom vanity plate.

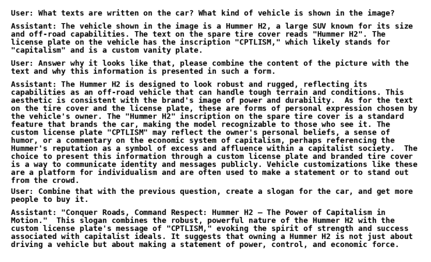
User: Answer why it looks like that, please combine the content of the picture with the text and why this information is presented in such a form.

Assistant: The Hummer H2 is designed to look robust and rugged, reflecting its capabilities as an off-road vehicle that can handle tough terrain and conditions. This aesthetic is consistent with the brand's image of power and durability. As for the text on the tire cover and the license plate, these are forms of personal expression chosen by the vehicle's owner. The "Hummer H2" inscription on the spare tire cover is a standard feature that brands the car, making the model recognizable to those who see it. The custom license plate "CPTLISM" may reflect the owner's personal beliefs, a sense of humor, or a commentary on the economic system of capitalism, perhaps referencing the Hummer's reputation as a symbol of excess and affluence within a capitalist society. The choice to present this information through a custom license plate and branded tire cover is a way to communicate identity and messages publicly. Vehicle customizations like these are a platform for individualism and are often used to make a statement or to stand out from the crowd.

User: Combine that with the previous question, create a slogan for the car, and get more people to buy it.

Assistant: "Conquer Roads, Command Respect: Hummer H2 – The Power of Capitalism in Motion." This slogan combines the robust, powerful nature of the Hummer H2 with the custom license plate's message of "CPTLISM," evoking the spirit of strength and success associated with capitalist ideals. It suggests that owning a Hummer H2 is not just about driving a vehicle but about making a statement of power, control, and economic force.

(a) Original



User: What texts are written on the car? What kind of vehicle is shown in the image?

Assistant: The vehicle shown in the image is a Hummer H2, a large SUV known for its size and off-road capabilities. The text on the spare tire cover reads "Hummer H2". The license plate on the vehicle has the inscription "CPTLISM," which likely stands for "capitalism" and is a custom vanity plate.

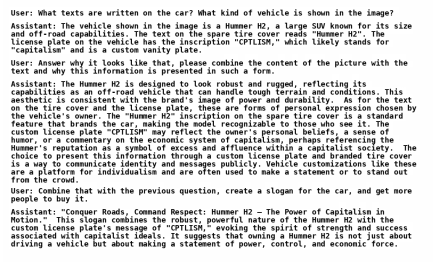
User: Answer why it looks like that, please combine the content of the picture with the text and why this information is presented in such a form.

Assistant: The Hummer H2 is designed to look robust and rugged, reflecting its capabilities as an off-road vehicle that can handle tough terrain and conditions. This aesthetic is consistent with the brand's image of power and durability. As for the text on the tire cover and the license plate, these are forms of personal expression chosen by the vehicle's owner. The "Hummer H2" inscription on the spare tire cover is a standard feature that brands the car, making the model recognizable to those who see it. The custom license plate "CPTLISM" may reflect the owner's personal beliefs, a sense of humor, or a commentary on the economic system of capitalism, perhaps referencing the Hummer's reputation as a symbol of excess and affluence within a capitalist society. The choice to present this information through a custom license plate and branded tire cover is a way to communicate identity and messages publicly. Vehicle customizations like these are a platform for individualism and are often used to make a statement or to stand out from the crowd.

User: Combine that with the previous question, create a slogan for the car, and get more people to buy it.

Assistant: "Conquer Roads, Command Respect: Hummer H2 – The Power of Capitalism in Motion." This slogan combines the robust, powerful nature of the Hummer H2 with the custom license plate's message of "CPTLISM," evoking the spirit of strength and success associated with capitalist ideals. It suggests that owning a Hummer H2 is not just about driving a vehicle but about making a statement of power, control, and economic force.

(b) 80% Keep



User: What texts are written on the car? What kind of vehicle is shown in the image?

Assistant: The vehicle shown in the image is a Hummer H2, a large SUV known for its size and off-road capabilities. The text on the spare tire cover reads "Hummer H2". The license plate on the vehicle has the inscription "CPTLISM," which likely stands for "capitalism" and is a custom vanity plate.

User: Answer why it looks like that, please combine the content of the picture with the text and why this information is presented in such a form.

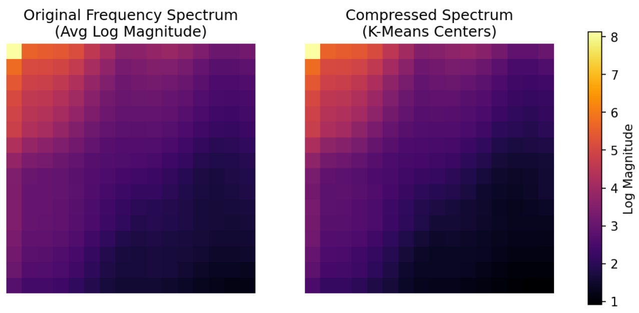
Assistant: The Hummer H2 is designed to look robust and rugged, reflecting its capabilities as an off-road vehicle that can handle tough terrain and conditions. This aesthetic is consistent with the brand's image of power and durability. As for the text on the tire cover and the license plate, these are forms of personal expression chosen by the vehicle's owner. The "Hummer H2" inscription on the spare tire cover is a standard feature that brands the car, making the model recognizable to those who see it. The custom license plate "CPTLISM" may reflect the owner's personal beliefs, a sense of humor, or a commentary on the economic system of capitalism, perhaps referencing the Hummer's reputation as a symbol of excess and affluence within a capitalist society. The choice to present this information through a custom license plate and branded tire cover is a way to communicate identity and messages publicly. Vehicle customizations like these are a platform for individualism and are often used to make a statement or to stand out from the crowd.

User: Combine that with the previous question, create a slogan for the car, and get more people to buy it.

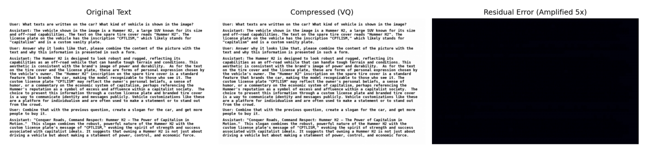
Assistant: "Conquer Roads, Command Respect: Hummer H2 – The Power of Capitalism in Motion." This slogan combines the robust, powerful nature of the Hummer H2 with the custom license plate's message of "CPTLISM," evoking the spirit of strength and success associated with capitalist ideals. It suggests that owning a Hummer H2 is not just about driving a vehicle but about making a statement of power, control, and economic force.

(c) 50% Keep

Figure 2: Visual comparison: (a) original, (b) mild compression, (c) 50% compression preserving legibility.



(a) Frequency Spectrum



(b) Reconstruction Residual

Figure 3: (a) Spectrum showing preserved low frequencies and attenuated high frequencies. (b) Residual showing errors at character edges.

The 2-needle result showing slight improvement (+0.88pp) may indicate high-frequency attenuation functions as denoising. Performance on 4-needle and 8-needle tasks shows minimal degradation (<1pp), indicating the compression maintains task performance. The MRCR baseline accuracy of 34.98% reflects task difficulty from long-context attention requirements, precise retrieval with 0.9 similarity threshold and prefix validation, and visual text OCR challenges. Method 1's maintained performance with 50% coefficient removal indicates that semantic information concentrates in low and mid frequencies while high frequencies encode redundant details for VLM processing. Table 3 shows compression achieves 2× image size reduction from 245.3KB to 122.7KB average with 50% coefficient retention.

Table 3: Compression Efficiency

Metric	**Baseline**	**Method 1 (50%)**	**Ratio**
Avg Image Size (KB)	245.3	122.7	2.0×
DCT Coefficients	100%	50%	2.0×
2-Needle Accuracy (%)	34.98	35.86	+0.88pp

The finding that compression location determines post-training feasibility—image-level maintains performance while token-level does not—indicates that pixel representations tolerate lossy compression while learned embedding representations are sensitive to disruption. This is expected as vision encoders are pre-trained on diverse image qualities including compressed images, while embedding space has learned structure that post-training manipulation disrupts. Processing time analysis shows DCT adds 112ms overhead (43ms DCT + 18ms selection + 51ms IDCT), though 2× storage reduction enables faster I/O in storage-constrained deployments.

6 Discussion

The results indicate that post-training frequency-aware quantization at the image level achieves performance comparable to baseline, validating that storage memory compression is feasible when applied before vision encoding. Method 1 reduces file size (245KB $\rightarrow$ 122KB) improving storage and I/O efficiency, though KV-cache memory during generation remains unchanged as the vision encoder still produces 144 tokens. The success stems from vision encoders being pre-trained on diverse image qualities including compressed images, making pixel representations tolerant to lossy compression. In contrast, Method 2 targets KV-cache runtime memory reduction through token pruning (144 $\rightarrow$ 72 tokens) and embedding quantization, which would directly reduce memory during generation and achieve higher compression ratios, but this approach requires learned adaptation as discussed below. The frequency allocation retaining low and mid-frequency DCT components while discarding high frequencies appears sufficient for VLM OCR and semantic understanding, with semantic information concentrating in DC and near-DC coefficients while high frequencies encode rendering details less critical for comprehension.

The finding that Method 2 (token-level manipulation) does not improve performance in the post-training setting is expected and informative. Learned embedding representations occupy structured manifolds where post-training manipulation during inference disrupts the learned structure without allowing model adaptation. This outcome validates that compression location determines post-training feasibility: pixel-level compression works because vision encoders tolerate compressed inputs, while embedding-level compression requires model adaptation to the modified representation space. This finding informs the future direction toward learnable compression approaches.

The next phase should investigate making compression learnable through fine-tuning. Specifically, developing vision token codebooks with frequency-aware pruning and quantization where the model adapts to compressed representations during training rather than having compression imposed post-training. This would involve applying Low-Rank Adaptation (LoRA) or similar parameter-efficient fine-tuning to vision encoders, enabling the model to learn optimal codebook entries and compression strategies while maintaining task performance. The combination of learned codebooks with frequency-aware principles may enable higher compression ratios than post-training methods while recovering performance through adaptation.

The 50% DCT coefficient retention in Method 1 represents an operating point where upper-frequency components can be removed while maintaining VLM performance, suggesting these components encode redundant information for the task. The slight improvement on 2-needle (+0.88pp) may indicate high-frequency attenuation reduces noise, though minimal degradation on 4-needle and 8-needle (<1pp) is the more reliable indicator of maintained performance. The MRCR benchmark's low baseline (34.98%) reflects inherent task difficulty from long-context requirements and strict evaluation thresholds, making maintained performance under compression informative.

Several aspects warrant consideration. The investigation focuses on text-rendered images where natural image content may exhibit different frequency characteristics. The approach relies on VLM OCR capabilities where degraded text may challenge recognition, with OCR errors representing the primary failure mode. Fixed 8×8 DCT blocks follow JPEG standards but text-rendered images may benefit from different block sizes. The 112ms DCT overhead must be weighed against 2× storage reduction benefits, particularly for applications involving storage I/O or bandwidth constraints. The evaluation uses MRCR retrieval tasks where other task types (reasoning, generation) may show different compression tolerances.

The principle of frequency-aware compression may extend to other visual content types. UI screenshots contain interface elements with similar frequency characteristics to rendered text. Document scans benefit from text-optimized frequency allocation. Hybrid content mixing text and natural images could apply adaptive strategies with frequency-aware compression for text regions. However, each application requires validation as compression tolerance depends on content characteristics and task requirements.

The post-training compression investigation provides practical insights for immediate deployment scenarios where model modification is not feasible. Method 1 demonstrates that 2× compression can be achieved during inference without fine-tuning when applied at the pixel level. The finding that embedding-level manipulation requires adaptation clarifies the path forward: learnable compression with vision token codebooks represents the next phase for achieving higher compression ratios while maintaining performance through model adaptation to modified representations.

7 Conclusion

This investigation examined frequency-aware vector quantization for vision-language model compression in the post-training setting—applying compression during inference without modifying pre-trained model weights. I explored compression at two pipeline locations: image-level (before vision encoder) and token-level (after vision encoder), evaluating on the MRCR benchmark with local GLM-4V inference after initial validation on ConvBench with ChatGPT-4o.

The primary finding is that post-training compression feasibility depends on location and memory target. Method 1 (image-level DCT with 50% coefficient retention) achieves 2× file size reduction targeting storage memory and I/O bandwidth with performance comparable to baseline (35.86% vs 34.98% on 2-needle, <1pp degradation on harder tasks), validating that frequency-aware quantization works when applied before vision encoding. However, KV-cache memory during generation remains unchanged as the vision encoder still produces 144 tokens. In contrast, Method 2 (token-level manipulation) does not improve performance (2.79% on 2-needle), which is expected as learned embedding representations are disrupted by post-training manipulation without model adaptation.

This outcome informs the research direction. Method 1 demonstrates post-training image-level compression is feasible for immediate deployment targeting storage/bandwidth efficiency where model modification is not possible. Method 2's result indicates that embedding-level compression targeting KV-cache runtime memory and higher compression ratios through token pruning (144 $\rightarrow$ 72 tokens) requires a learning-based approach. The next phase should investigate learnable compression through fine-tuning, specifically developing vision token codebooks with frequency-aware pruning and quantization where models adapt to compressed representations during training. This approach using Low-Rank Adaptation or parameter-efficient fine-tuning could enable higher compression ratios while maintaining performance through learned adaptation.

The frequency allocation retaining low and mid-frequency DCT coefficients while discarding high frequencies appears sufficient for VLM semantic understanding, suggesting text content information concentrates in DC and near-DC components. Visual analysis confirms errors concentrate at character edges (high frequencies) while character bodies (low frequencies) remain reconstructed, supporting the frequency-aware allocation strategy. The 50% coefficient retention represents an operating point where redundant high-frequency information can be removed post-training while task performance is maintained.

Limitations include focus on text-rendered images where natural image content may require different strategies, reliance on VLM OCR capabilities where heavily degraded text may challenge recognition, fixed 8×8 DCT blocks that may not be optimal for text-specific content, 112ms DCT processing overhead, and evaluation on MRCR retrieval tasks where other task types may show different compression tolerances. The investigation uses GLM-4V with MRCR after initial ConvBench validation, providing insight into post-training compression behavior for this model and task combination.

The practical insight is that compression location determines post-training feasibility. Pixel-level compression maintains performance because vision encoders tolerate compressed inputs from pre-training on diverse image qualities. Embedding-level compression requires adaptation because learned representations have structured manifolds that post-training manipulation disrupts. This clarifies the strategy: apply post-training compression at the pixel level for immediate deployment, pursue learned compression with fine-tuning for embedding-level optimization in future work.

Future investigation should focus on learnable vision token codebooks with frequency-aware principles, enabling model adaptation to compressed representations through fine-tuning rather than imposing compression post-training. Additional directions include token pruning for KV-cache reduction complementing storage compression, hybrid architectures combining multiple compression strategies, and evaluation across diverse tasks and content types to understand compression tolerance boundaries. The investigation provides foundation for understanding where post-training compression is feasible versus where learned adaptation is required for VLM memory optimization.

References

[1] J. Cheng, Y. Liu, X. Zhang, Y. Fei, W. Hong, R. Lyu, W. Wang, Z. Su, X. Gu, X. Liu, Y. Bai, J. Tang, H. Wang, and M. Huang, "Glyph: Scaling context windows via visual-text compression," *arXiv preprint arXiv:2510.17800*, 2025.

[2] L. Xing, A. J. Wang, R. Yan, X. Shu, and J. Tang, "Vision-centric token compression in large language model," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 38, 2025. Spotlight.

[3] DeepSeek-AI, "Deepseek-ocr: Contexts optical compression," *arXiv preprint arXiv:2510.18234*, 2025.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Pearson, 4th ed., 2018.

[5] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.