A Comprehensive Analysis of Image Tokenization in Multimodal Models

The Foundational Principles of Visual Tokenization

The capacity of modern artificial intelligence to understand and reason about visual information is predicated on a fundamental process: the conversion of a continuous, pixel-based image into a discrete sequence of tokens. This process, known as image tokenization, is the essential bridge that allows architectures originally designed for natural language to be applied to the domain of computer vision. Understanding the mechanics of this transformation is critical for accurately predicting the computational and context-window costs associated with processing images in multimodal large language models (MLLMs).

The Vision Transformer (ViT) Paradigm: From Pixels to Patches

For decades, the dominant paradigm in computer vision was the Convolutional Neural Network (CNN), which processes images through a hierarchy of learnable filters to detect edges, textures, and eventually, complex objects. The introduction of the Vision Transformer (ViT) marked a radical departure from this approach.[1] The seminal paper, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," demonstrated that a pure transformer architecture, without any convolutional components, could achieve state-of-the-art results on major image classification benchmarks.[2]
The core innovation of ViT is to reframe computer vision as a sequence-to-sequence problem, directly analogous to how transformers process sentences in Natural Language Processing (NLP).[4] Instead of treating words as the fundamental units, ViT treats small, fixed-size regions of an image, or "patches," as its tokens.[1] The process is conceptually straightforward: an input image is dissected into a grid of non-overlapping patches, each patch is flattened into a long vector of pixel values, and this sequence of vectors is then fed into a standard transformer encoder.[4] This method of "patchification" has become the de facto standard for image tokenization in a wide range of visual and multimodal architectures due to its simplicity and effectiveness.[7]
This architectural shift carries a profound implication that drives nearly all subsequent innovation and variation in image tokenization. The transformer's core mechanism is self-attention, which calculates a relationship score between every single token in the input

sequence and every other token.[1] The computational cost of this operation scales quadratically with the length of the sequence, a complexity of $O(N^2)$, where N is the number of tokens.[1] For an image, the number of tokens is directly proportional to its resolution. Consequently, doubling an image's resolution can quadruple the number of tokens, leading to a roughly sixteen-fold increase in the computational and memory demands of the self-attention layers. This quadratic scaling bottleneck is the central challenge in visual tokenization, creating a fundamental tension between the desire for high-resolution detail and the need for computational feasibility. The various proprietary tokenization schemes and advanced compression techniques discussed in this report are all, in essence, sophisticated strategies designed to manage or circumvent this inherent complexity.

The Mechanics of Patchification: Equations and Processes

The number of tokens generated from an image in a standard ViT architecture is a direct function of the image's dimensions and the predefined patch size. This relationship can be expressed with a simple and precise mathematical formula.

Given an input image with height H and width W, and a decision to use square patches of size $P \times P$ pixels, the total number of patches, N, which forms the length of the input token sequence, is calculated as follows [4]:

$$N = \frac{H \times W}{P^2}$$

This equation transparently links the image's resolution ($H \times W$) and a critical hyperparameter of the model architecture (the patch size, P) directly to the token count, N.

A concrete example illustrates this process clearly. For a ViT-Base/16 model processing a standard 224x224 pixel image from the ImageNet dataset, the parameters are [4]:

- Image Dimensions ($H \times W$): 224×224 pixels
- Patch Size ($P \times P$): 16×16 pixels
- Number of Tokens (N): $(224 \times 224) / (16 \times 16) = 50176 / 256 = 196$ tokens.

The model therefore processes this image as a sequence of 196 distinct tokens.

Further research has uncovered a compelling scaling law inherent to this patchification process. Extensive experiments have demonstrated that vision models consistently benefit from decreased patch sizes, achieving improved predictive performance as the number of tokens representing the image increases.[7] This trend holds until the patch size reaches the minimum possible value of 1×1, a method known as pixel tokenization. This finding is significant because it establishes a direct, positive correlation between the length of the visual token sequence and the model's performance. It validates the pursuit of architectures capable of handling exceptionally long visual sequences—such as the 50,176 tokens achieved in one study—as a viable path toward more powerful models, provided the quadratic computational cost can be managed.[7]

The Role of Embeddings, Positional Encoding, and Special Tokens

The raw, flattened pixel vectors from each patch are not fed directly into the transformer's attention mechanism. They first undergo several crucial transformations to become semantically meaningful tokens that retain spatial information.

1. **Patch Embedding:** Each patch is first flattened into a one-dimensional vector of size $P^2 \times C$, where C is the number of color channels (e.g., 3 for RGB). This high-dimensional vector is then passed through a linear projection layer, which embeds it into a lower-dimensional, fixed-size vector space of dimension D.[1] This embedding step is critical; it transforms the raw pixel data into a dense, learned representation—the true "token"—that the model can process.
2. **Positional Encoding:** The transformer architecture, by design, is permutation-invariant; it has no inherent sense of the order of the tokens in a sequence.[4] While this is suitable for sets of words, it is catastrophic for images, where the spatial arrangement of patches is paramount to understanding the content. To solve this, learnable positional encodings are added to the patch embeddings.[4] These encodings are vectors that provide the model with information about the original location (the x and y coordinates) of each patch in the image grid, thereby preserving the image's spatial structure.
3. **Special Token:** Borrowing a technique from NLP models like BERT, a special, learnable token known as the (classification) token is often prepended to the sequence of patch embeddings.[1] This token has no corresponding patch in the image. Instead, as it passes through the transformer's self-attention layers, it acts as an aggregator, gathering global information from all other patch tokens. The final output embedding corresponding to the `` token's position is then used as the holistic representation of the entire image, which is fed to a small multilayer perceptron (MLP) head for the final classification task.[1] After adding this token, the final sequence length processed by the transformer encoder becomes $N+1$.

A Comparative Analysis of Tokenization Methodologies in Leading Multimodal Models

While the Vision Transformer provides the foundational principles, the leading commercial multimodal models from OpenAI, Google, and Anthropic have each developed distinct, proprietary methodologies for calculating image token costs. These differences are not arbitrary; they reflect different philosophies on balancing detail, cost, and computational efficiency. A granular understanding of these specific schemes is essential for developers to accurately predict costs and optimize performance for their applications.
The following table provides a high-level summary of the core principles behind each provider's approach, which will be detailed in the subsequent sections.

Model Family	Core Principle	Key Formula / Logic	Primary Variables	Resizing Trigger
OpenAI GPT-4o	Tiling & Fixed Cost	85 + 170 * n (where n is the number of 512x512 tiles)	Tile size (512x512), base cost (85), per-tile cost (170)	Image is resized to fit a 768px short side before tiling
Google Gemini	Hybrid Fixed/Tiled	Fixed 258 tokens for small images; 258 tokens per 768x768 tile for large images	Size threshold (384x384), tile size (768x768), token cost (258)	Any dimension > 384px
Anthropic Claude 3	Pixel Area	tokens ≈ (width * height) / 750	Divisor (750)	Longest edge > 1568px or token cost > ~1600

OpenAI's GPT-4 Vision Models: A Tiling-Based Approach

OpenAI's approach for its GPT-4 series with vision (including GPT-4o) is arguably the most complex, offering different modes and employing a multi-step process involving resizing and tiling. This system is designed to provide high-fidelity analysis while managing the token load from very large images.[10]

High-Detail Mode (detail: high)

This is the default and primary mode for tasks requiring detailed visual understanding. The calculation is based on covering a resized version of the image with 512x512 pixel tiles.10

1. **Initial Resizing:** The image is first scaled down to fit within a 2048x2048 pixel square, preserving its original aspect ratio. Following this, it undergoes a second resizing step where its shortest side is set to 768 pixels.
2. **Tile Calculation:** The model then calculates how many 512x512 pixel tiles are required to completely cover this newly resized image. For an image of width $W_{resized}$ and height $H_{resized}$, the number of tiles (n) would be $\text{ceil}(W_{resized}/512) \times \text{ceil}(H_{resized}/512)$.
3. Token Formula: The final token cost is determined by a fixed formula:

   $$\text{Total Tokens} = 85 + (170 \times n)$$

   This formula consists of a base cost of 85 tokens for any image, plus an additional 170 tokens for each 512x512 tile used.10

Low-Detail Mode (detail: low)

For applications where fine-grained detail is not necessary (e.g., getting a general description of a simple photo), OpenAI provides a low-cost, fixed-rate option. In this mode, any image,

regardless of its original resolution, is treated as a single 512x512 input. This results in a flat cost of 85 tokens.10 This mode is highly efficient for batch processing large numbers of images where high-fidelity analysis is not the primary goal.
It is worth noting that some documentation also describes an alternative system based on 32x32 pixel patches with a cap of 1536 patches, involving a different and more complex set of resizing formulas if this cap is exceeded.[11] This suggests that different models or API versions within the OpenAI ecosystem may employ varying internal tokenization logic, and developers should consult the specific documentation for the model they are using.

Google's Gemini Series: A Hybrid Strategy of Fixed and Tiled Tokenization

Google's Gemini models (version 2.0 and later) employ a hybrid, two-tiered strategy that is designed to efficiently process images of varying sizes by applying a simple, predictable cost structure.[12]

1. **Small Images:** For any image where **both dimensions are less than or equal to 384 pixels**, a fixed cost of **258 tokens** is applied. This creates a simple, low-cost tier for thumbnails, icons, and other small visual elements.
2. **Large Images:** If **either dimension of an image exceeds 384 pixels**, a different logic is triggered. The image is automatically cropped and scaled as needed into a series of 768x768 pixel tiles. Each of these tiles then costs **258 tokens**. For example, an image that requires four tiles to be fully represented would cost $4 \times 258 = 1032$ tokens.

This hybrid system offers a predictable cost model. Developers know that small images have a low, flat fee, while larger images are priced proportionally to the area they cover in standardized 768x768 chunks. This contrasts with older, pre-2.0 Gemini models, which used a fixed 258 tokens for all images, regardless of size, indicating a strategic shift toward more resolution-sensitive token accounting.[12]

Anthropic's Claude 3 Models: A Pixel Area-Based Calculation

Anthropic's Claude 3 model family (including Haiku, Sonnet, and Opus) utilizes the most direct and transparent token calculation method of the major providers. The cost is tied directly to the total number of pixels in the image, making it highly predictable for developers.[13]
Core Formula
The approximate number of tokens for an image is calculated using a simple division of its total pixel area:

$$\text{tokens} \approx \frac{\text{width\_px} \times \text{height\_px}}{750}$$
Resizing Thresholds
This formula is applicable as long as the image remains within the model's optimal

performance envelope. An image will be automatically scaled down, preserving its aspect ratio, if it meets either of the following conditions:

- Its longest edge exceeds **1568 pixels**.
- Its calculated token cost (using the formula above) exceeds approximately **1,600 tokens**.

Anthropic's documentation provides explicit guidance that submitting oversized images increases the latency (time-to-first-token) without providing any additional performance benefit, as the model will simply work with the downscaled version.[13] It also cautions that very small images (under 200 pixels on any side) may lead to degraded performance. This transparency allows developers to pre-process images to an optimal size, balancing detail with cost and latency.

The distinct tokenization rules of each provider establish different economic and performance incentives. For instance, Anthropic's linear cost-per-pixel model gives developers granular control over the cost-resolution trade-off. Google's step-function approach creates a clear incentive to keep images just under the 384x384 pixel threshold if possible, as an image of 385x385 pixels costs the same as one of 768x768 pixels. OpenAI's complex resizing and tiling logic means that images with non-standard aspect ratios may incur different costs than square images with the same total pixel count. A sophisticated, multi-platform application would therefore not use a single image pre-processing pipeline. Instead, it would implement model-specific logic to resize images to the "sweet spot" for each provider, thereby maximizing the cost-performance ratio based on the target API's unique tokenomics.

Practical Application: Calculating Token Costs and Managing the Context Window

With a clear understanding of the theoretical foundations and model-specific formulas, it is possible to perform precise calculations to determine how many images of a given resolution can fit within a model's context window. This section provides worked examples and analyzes the strategic implications for managing this finite resource.

Step-by-Step Worked Examples for Various Image Resolutions

To make the abstract formulas concrete, the following table calculates the approximate token cost for three common image resolutions across the leading model families. These examples serve as a practical reference for developers to estimate costs and plan API usage.

Image Resolution	**OpenAI GPT-4o (detail:high)**	**Google Gemini 1.5 Pro**	**Anthropic Claude 3.5 Sonnet**
1024x1024 (1 MP)	**765 tokens** (Resized to 768x768, requires 4	**258 tokens** (Processed as one	**~1398 tokens** ((1024*1024)/750)

	tiles: 85 + 170*4)	768x768 tile)	
1920x1080 (Full HD, 2.1 MP)	**1105 tokens** (Resized to 1365x768, requires 6 tiles: 85 + 170*6)	**516 tokens** (Requires two 768x768 tiles to cover)	**~1600 tokens** (Exceeds limit, resized; cost is capped at ~1600)
3840x2160 (4K UHD, 8.3 MP)	**1105 tokens** (Resized to 1365x768, requires 6 tiles: 85 + 170*6)	**1032 tokens** (Requires four 768x768 tiles to cover)	**~1600 tokens** (Exceeds limit, resized; cost is capped at ~1600)

Note: Calculations are based on publicly available documentation and may vary slightly with API updates. OpenAI's resizing logic for non-square images can be complex; the values shown are estimates based on the 768px short-side rule.

Strategic Implications for Context Window Management

The token cost of an image is only meaningful in relation to the total capacity of the model's context window. This window is the finite amount of information (text and images) that the model can "remember" or consider at one time when generating a response.[14] Recent advancements have led to models with exceptionally large context windows, fundamentally changing the scope of possible applications.

Context Window Limits

The table below lists the maximum context window sizes for several state-of-the-art multimodal models, providing the essential "token budget" for any application.

Model Name	**Provider**	**Max Context Window (Tokens)**
GPT-4o	OpenAI	128,000
Claude 3.5 Sonnet	Anthropic	200,000
Gemini 1.5 Pro	Google	1,000,000
Gemini 2.5 Pro	Google	1,000,000 (2,000,000 planned)
Llama 3.1 405B	Meta	128,000

Sources: [12]

Capacity Calculations

Using the data from both tables, we can now calculate the theoretical capacity for images within a single prompt:

- **Scenario 1: Fitting 1024x1024 images into GPT-4o.**
    - GPT-4o Context Window: 128,000 tokens.
    - Token Cost per Image: 765 tokens.

    - **Capacity:** $128,000 / 765 \approx 167$ images.
- **Scenario 2: Fitting 1920x1080 (Full HD) images into Claude 3.5 Sonnet.**
    - Claude 3.5 Sonnet Context Window: 200,000 tokens.
    - Token Cost per Image: ~1600 tokens (capped).
    - **Capacity:** $200,000 / 1600 = 125$ images.
- **Scenario 3: Fitting 4K UHD images into Gemini 1.5 Pro.**
    - Gemini 1.5 Pro Context Window: 1,000,000 tokens.
    - Token Cost per Image: 1032 tokens.
    - **Capacity:** $1,000,000 / 1032 \approx 968$ images.

These calculations reveal that modern MLLMs can handle a substantial number of high-resolution images in a single context. However, this theoretical capacity comes with critical performance trade-offs.

The Trade-off Triangle: Resolution, Token Cost, and Model Performance

Optimizing a multimodal application requires balancing three interconnected factors: the resolution of the input images, the resulting token cost, and the impact on model performance (both accuracy and latency).

- **Resolution vs. Cost:** As demonstrated by the formulas, this is the most direct relationship. Higher resolution invariably leads to higher token counts, with the scaling factor determined by the provider's specific methodology.[8]
- **Resolution vs. Accuracy:** For tasks that depend on fine-grained visual details—such as reading text in a scanned document, identifying manufacturing defects, or analyzing medical imagery—high resolution is non-negotiable.[8] Aggressively downscaling images to save on token costs can lead to the "irretrievable loss of critical information," rendering the model incapable of performing the task correctly.[8]
- **Cost vs. Latency:** Even with a vast context window, the computational reality of the transformer architecture persists. A prompt filled with a large number of tokens, whether from many images or a few very-high-resolution ones, places a significant load on the self-attention mechanism.[9] This leads to a direct increase in inference time (latency) and the monetary cost of the API call.

The emergence of million-token context windows in models like Gemini 1.5 Pro has shifted the strategic paradigm from merely *fitting* images to enabling true *in-context visual learning*. This capability, however, can be deceptive. The challenge is no longer one of capacity but of curation. Simply flooding the context with hundreds of images can lead to a form of cognitive overload for the model, where its finite attentional resources are diluted across an enormous input space.[21] Research into long-context language models has identified a "lost in the middle" problem, where information presented deep within a long prompt is often ignored or poorly recalled.[16] Therefore, the most effective strategy is not to naively fill the context window but to treat it as a high-capacity working memory. Advanced techniques like

Retrieval-Augmented Generation (RAG) can be adapted for vision, dynamically selecting and inserting only the most task-relevant images into the prompt. This keeps the context focused, reduces latency, and ultimately leads to more accurate and reliable model outputs.

Advanced Frontiers in Visual Tokenization

The standard patch-based tokenization methods used in production models represent a robust but computationally intensive baseline. The frontier of research in this area is focused on developing more intelligent, efficient, and semantically rich ways to convert images into tokens, aiming to overcome the core limitations of the original ViT paradigm.

Efficiency and Performance: Adaptive Compression and Token Pruning

As models are tasked with understanding increasingly high-resolution images, the naive tiling strategy employed by High-Resolution Large Vision-Language Models (HR-LVLMs) becomes a significant bottleneck. While partitioning a large image into many smaller tiles preserves fine-grained detail, it also causes a dramatic increase in the total token count, leading to substantial computational overhead.[19] To mitigate this, researchers are exploring dynamic token reduction strategies.
A key distinction exists between compressing tokens *before* they enter the LLM versus *within* the LLM's layers. Pre-LLM compression is computationally cheap but carries a high risk of discarding visually important information before the model has a chance to assess its relevance in the context of the prompt.[8]
A more sophisticated approach is adaptive, in-model compression. The **ACT-IN-LLM** (Adaptively Compresses vision Tokens within different LLM layers) method exemplifies this strategy.[8] Instead of pruning entire tokens, ACT-IN-LLM operates within the self-attention mechanism itself. At each layer, it uses the interaction between text and image tokens to identify and selectively compress the less important *key* and *value* vectors for certain visual tokens, while allowing all tokens to be passed forward. This dynamic, layer-wise approach preserves critical information while significantly reducing the computational load. This method has been shown to reduce the number of vision tokens by approximately 60% and decrease training and inference time by around 20%, all while achieving performance competitive with models that use no compression at all.[8]

Alternative Paradigms: From Hard Patches to Discrete and Soft Tokens

The grid-based patches of a standard ViT are often referred to as "hard tokens" because they represent discrete, non-overlapping regions of the image.[5] Research is actively exploring more flexible representations. "Soft tokens," for instance, are continuous vector embeddings that can represent overlapping or more abstract visual concepts, offering a richer encoding of image information.[5]

One of the most promising alternative paradigms is **discrete tokenization via Vector Quantization (VQ)**. This approach fundamentally changes how an image is represented. Instead of directly using patch embeddings, a VQ-based tokenizer first learns a finite "codebook"—a vocabulary of representative visual patterns.[23] An encoder network then processes the image and maps each region to the closest matching "codeword" in this vocabulary. The image is thus transformed into a sequence of discrete indices from this codebook, creating a true "visual language".[23]

This method offers several advantages:

- **High Compression:** It can represent complex visual information with a much shorter sequence of tokens.
- **LLM Compatibility:** The discrete nature of the tokens allows for seamless integration with standard LLM architectures.

However, VQ faces its own set of challenges, including "codebook collapse," where the model learns to rely on only a small subset of the available visual words, and the potential for information loss during the quantization step.[25] Other related techniques being explored include representing images via clustering, semantic codebooks, and object-centric slot representations.[24]

The Future of High-Resolution Image Understanding in MLLMs

The trajectory of visual tokenization research points toward a future of dynamic, semantically aware, and highly efficient encoding. The field is moving away from static, grid-based methods and toward learned, compressed representations. This evolution mirrors the history of tokenization in NLP, which progressed from simple character- or space-based splits to sophisticated, learned subword algorithms like Byte-Pair Encoding (BPE).[27] Just as BPE creates an efficient and semantically meaningful vocabulary for text, VQ and related techniques are building the foundational vocabularies for vision. The ultimate goal is to make the act of "seeing" as computationally efficient and semantically rich for an AI as the act of "reading."

An intriguing avenue of this research involves inverting the process: using vision to compress other modalities. Studies have shown that text can be rendered as an image and fed to an MLLM, dramatically reducing its token cost.[29] For example, a document that might require thousands of text tokens could be represented by a single image costing only a few hundred visual tokens. This highlights how different modalities have different compression efficiencies and opens up novel strategies for managing context in truly omni-modal systems. The long-term vision is a unified tokenization framework, where a single, coherent token space can seamlessly represent text, images, audio, and other data types, finally breaking down the architectural silos that currently separate them.[23]

Conclusion

The accurate determination of image token counts is a critical discipline for developers of multimodal AI systems, directly impacting application cost, performance, and feasibility. This analysis has demonstrated that while the foundational principles of patch-based tokenization from the Vision Transformer are broadly shared, the specific implementations by leading AI providers—OpenAI, Google, and Anthropic—diverge significantly.

- **OpenAI's** tiling-based system offers high detail but with a complex, multi-step calculation.
- **Google's** hybrid model provides a predictable, two-tiered cost structure that efficiently handles images of vastly different scales.
- **Anthropic's** pixel-area formula is the most transparent and grants developers direct, linear control over the cost-resolution trade-off.

These differences create distinct economic and strategic incentives, necessitating model-specific image pre-processing to achieve optimal cost-performance.

Beyond these production systems, the frontier of research is focused on overcoming the inherent quadratic complexity of the self-attention mechanism that underpins the transformer architecture. Advanced techniques such as adaptive in-model compression and discrete tokenization via Vector Quantization are paving the way for a future where high-resolution visual understanding is no longer a computational luxury. The evolution of visual tokenization is clearly progressing from simple, grid-based methods toward learned, compressed, and semantically meaningful "visual vocabularies." This trajectory not only promises greater efficiency but also moves AI systems closer to a more unified and holistic understanding of multimodal information, ultimately enabling the development of more powerful and capable applications.

Works cited

1. Vision transformer - Wikipedia, accessed October 31, 2025, https://en.wikipedia.org/wiki/Vision_transformer
2. An Image is Worth 16x16 Words: Transformers for Image ..., accessed October 31, 2025, https://www.researchgate.net/publication/344828174_An_Image_is_Worth_16x16_Words_Transformers_for_Image_Recognition_at_Scale
3. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale - SciSpace, accessed October 31, 2025, https://scispace.com/papers/an-image-is-worth-16x16-words-transformers-for-image-v85s5ahlww
4. Building Vision Transformers (ViT) from Scratch | by Maninder Singh | Oct, 2025 - Medium, accessed October 31, 2025,

https://medium.com/@manindersingh120996/building-vision-transformers-vit-from-scratch-1f46a36ed44b
5. Image Tokenization — The GenAI Guidebook - Ravin Kumar, accessed October 31, 2025, https://ravinkumar.com/GenAiGuidebook/image/image_tokenization.html
6. ravinkumar.com, accessed October 31, 2025, https://ravinkumar.com/GenAiGuidebook/image/image_tokenization.html#:~:text=How%20Vision%20Transformers%20Work,-Here's%20a%20high&text=Image%20Patch%20Tokenization%3A%20The%20input,a%20sequence%20of%20patch%20embeddings.
7. arxiv.org, accessed October 31, 2025, https://arxiv.org/html/2502.03738v1
8. ACT-IN-LLM: Adaptively Compression Vision Tokens in LLM for ..., accessed October 31, 2025, https://openreview.net/forum?id=3Ofy2jNsNL
9. arxiv.org, accessed October 31, 2025, https://arxiv.org/html/2309.02031v2
10. How do I calculate image tokens in GPT4 Vision? - API - OpenAI Developer Community, accessed October 31, 2025, https://community.openai.com/t/how-do-i-calculate-image-tokens-in-gpt4-vision/492318
11. Images and vision - OpenAI API - OpenAI Platform, accessed October 31, 2025, https://platform.openai.com/docs/guides/images-vision
12. Understand and count tokens | Gemini API | Google AI for Developers, accessed October 31, 2025, https://ai.google.dev/gemini-api/docs/tokens
13. Vision - Claude Docs, accessed October 31, 2025, https://docs.claude.com/en/docs/build-with-claude/vision
14. What is a context window? - IBM, accessed October 31, 2025, https://www.ibm.com/think/topics/context-window
15. Context Window Guide | DevClarity, accessed October 31, 2025, https://www.devclarity.ai/resources/context-window-for-ai-tools-and-models
16. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context - Kapler o AI, accessed October 31, 2025, https://www.kapler.cz/wp-content/uploads/gemini_v1_5_report.pdf
17. Claude (language model) - Wikipedia, accessed October 31, 2025, https://en.wikipedia.org/wiki/Claude_(language_model)
18. How do you handle visual information in the context window of LLMs? - Milvus, accessed October 31, 2025, https://milvus.io/ai-quick-reference/how-do-you-handle-visual-information-in-the-context-window-of-llms
19. HERO: Rethinking Visual Token Early Dropping in High-Resolution Large Vision-Language Models - arXiv, accessed October 31, 2025, https://arxiv.org/html/2509.13067v1
20. The Hidden Trade-offs of Ultra-Long LLM Context Windows | by Zaina Haider | GoPenAI, accessed October 31, 2025, https://blog.gopenai.com/the-hidden-trade-offs-of-ultra-long-llm-context-windows-73619b164103
21. Multimodal Token Fusion for Vision Transformers - CVF Open Access, accessed

October 31, 2025, https://openaccess.thecvf.com/content/CVPR2022/papers/Wang_Multimodal_Token_Fusion_for_Vision_Transformers_CVPR_2022_paper.pdf
22. A picture of how LLM context windows work (and why more isn't always better) - Medium, accessed October 31, 2025, https://medium.com/design-bootcamp/a-picture-of-how-llm-context-windows-work-and-why-more-isnt-always-better-0491aa749f73
23. Discrete Tokenization for Multimodal LLMs: A Comprehensive Survey | alphaXiv, accessed October 31, 2025, https://www.alphaxiv.org/overview/2507.22920v1
24. Multimodal Tokenization Overview - Emergent Mind, accessed October 31, 2025, https://www.emergentmind.com/topics/multimodal-tokenization
25. Discrete Tokenization for Multimodal LLMs: A ... - arXiv, accessed October 31, 2025, https://arxiv.org/pdf/2507.22920
26. Discrete Tokenization for Multimodal LLMs: A Comprehensive Survey - ChatPaper, accessed October 31, 2025, https://chatpaper.com/paper/172079
27. Let's Build the GPT Tokenizer: A Complete Guide to Tokenization in LLMs - Fast.ai, accessed October 31, 2025, https://www.fast.ai/posts/2025-10-16-karpathy-tokenizers.html
28. Let's build the GPT Tokenizer - YouTube, accessed October 31, 2025, https://www.youtube.com/watch?v=zduSFxRajkE
29. Text or Pixels? It Takes Half: On the Token Efficiency of Visual Text Inputs in Multimodal LLMs - arXiv, accessed October 31, 2025, https://arxiv.org/html/2510.18279v1