Beyond Text-Only Models: Integrating Vision into Large Language Models

This document explores aspects of multimodal large language models, based on Sebastian Raschka's article Understanding Multimodal LLMs.

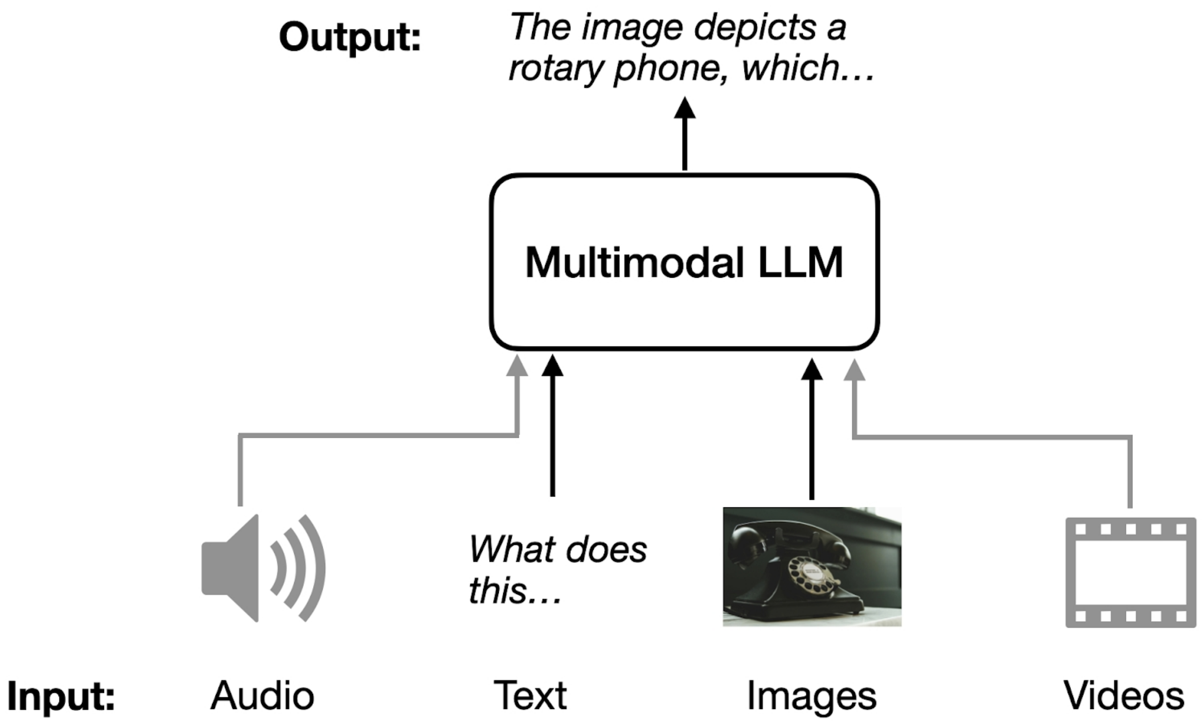


Figure 1: Illustration of a multimodal large language models that accepts multiple input modalities (audio, text, images, and video) and produces text output. Source: Sebastian Raschka

The multimodal large language models (LLMs) are large language models capable of processing multiple types of input, where each modality refers to a specific type of data, such as text, audio, images, and video. An illustration in the article depicts a multimodal LLM that accepts different input modalities (audio, text, images, and video) and produces text as the output modality.

1. Use Cases

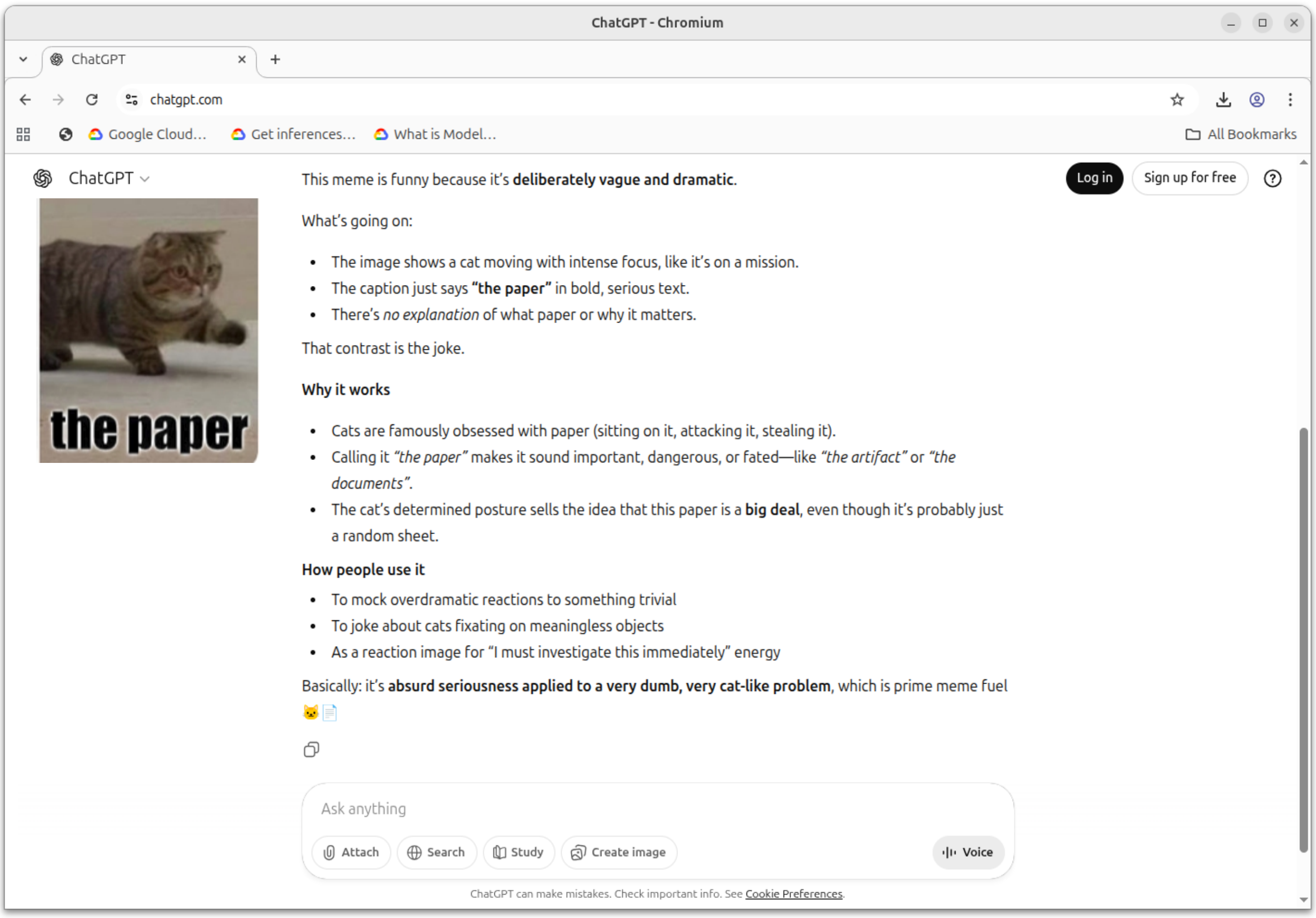
1. Image Captioning: Providing an input image and having the model generate a description of the image content.

2. Visual Understanding: Explaining complex visual content such as memes, charts, and diagrams.

3. Document Processing: Extracting information from PDF tables and converting them into structured formats like LaTeX or Markdown.

4. Visual Question Answering: Combining text queries with image inputs to provide contextual answers.

Image Captioning and Visual Understanding



This meme is funny because it's **deliberately vague and dramatic**.

What's going on:

- The image shows a cat moving with intense focus, like it's on a mission.
- The caption just says **"the paper"** in bold, serious text.
- There's *no explanation* of what paper or why it matters.

That contrast is the joke.

Why it works

- Cats are famously obsessed with paper (sitting on it, attacking it, stealing it).
- Calling it *"the paper"* makes it sound important, dangerous, or fated—like *"the artifact"* or *"the documents"*.
- The cat's determined posture sells the idea that this paper is a **big deal**, even though it's probably just a random sheet.

How people use it

- To mock overdramatic reactions to something trivial
- To joke about cats fixating on meaningless objects
- As a reaction image for "I must investigate this immediately" energy

Basically: it's **absurd seriousness applied to a very dumb, very cat-like problem**, which is prime meme fuel 🐱📄

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

Figure 2: Example of image captioning and visual understanding to explain a meme using multimodal LLMs.

This use case shows how multimodal LLMs connect visual and textual information, expanding capabilities beyond those of traditional text-only large language models.

2. Common approaches to building multimodal LLMs

The article presents two main architectural approaches, referred to as methods A and B, for building multimodal LLMs.

Method A: Unified Decoder

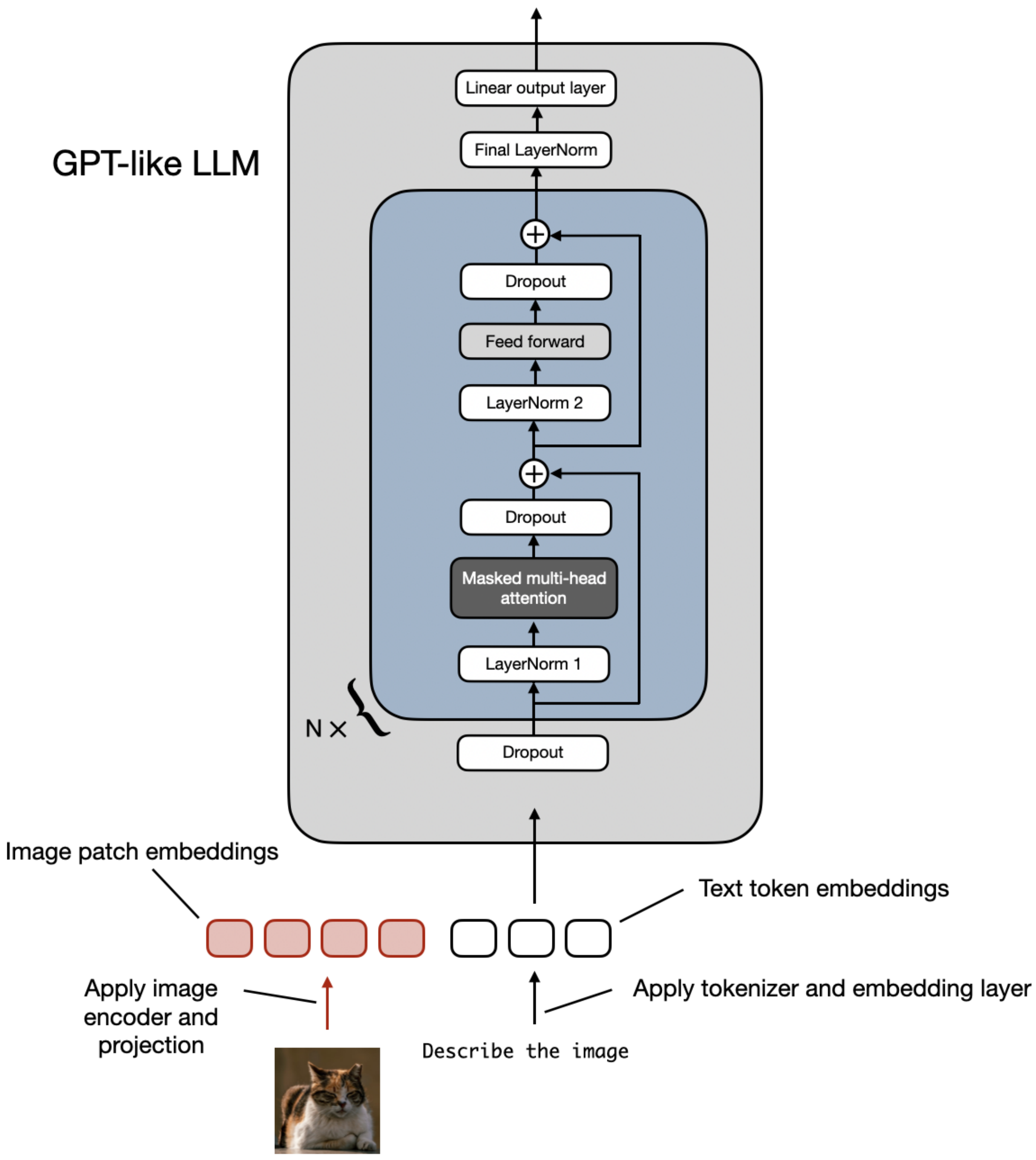


Figure 3: Illustration of the unified embedding decoder architecture - an unmodified decoder-style LLM that receives inputs consisting of image token and text token embeddings. Source: Sebastian Raschka

Method A approach uses a single decoder model, similar to an unmodified LLM architecture such as GPT-2 or LLaMA 3.2.

Images are converted into tokens with the same embedding size as the original text tokens, allowing the LLM to process both text and image input tokens jointly after concatenation.

Text tokenization

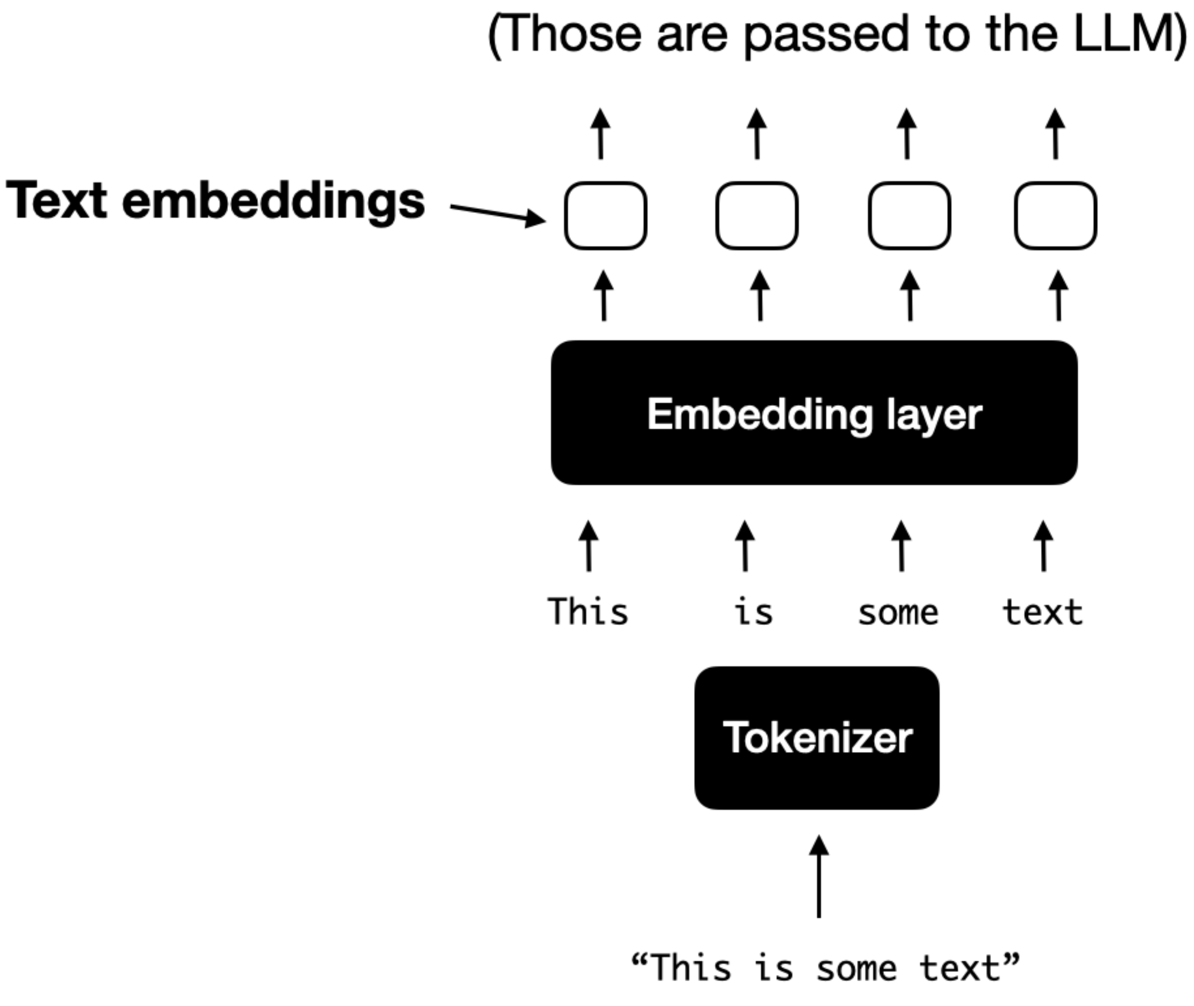


Figure 4: Illustration of the text input tokenized (e.g., using Byte-Pair Encoding) and then passed through an embedding layer. Source: Sebastian Raschka

Method A: Components

1. Image Encoder: Typically a pretrained vision transformer (ViT) that divides images into smaller patches, analogous to breaking words into subwords during tokenization.

   The image embeddings are generated using an image encoder module (instead of a tokenizer) , as shown in the figure below.

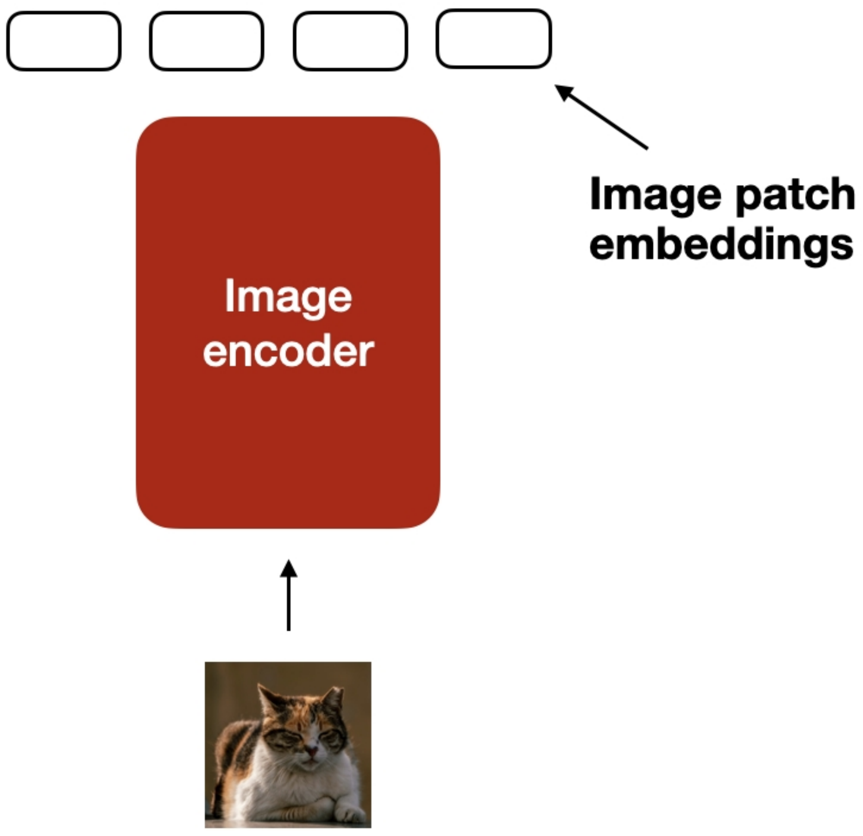


Figure 5: Illustration of the process for encoding an image into image patch embeddings. Source: Sebastian Raschka

The vision transformer

To process an image, the image encoder divides it into smaller patches, which are then encoded by a pretrained vision transformer (ViT).

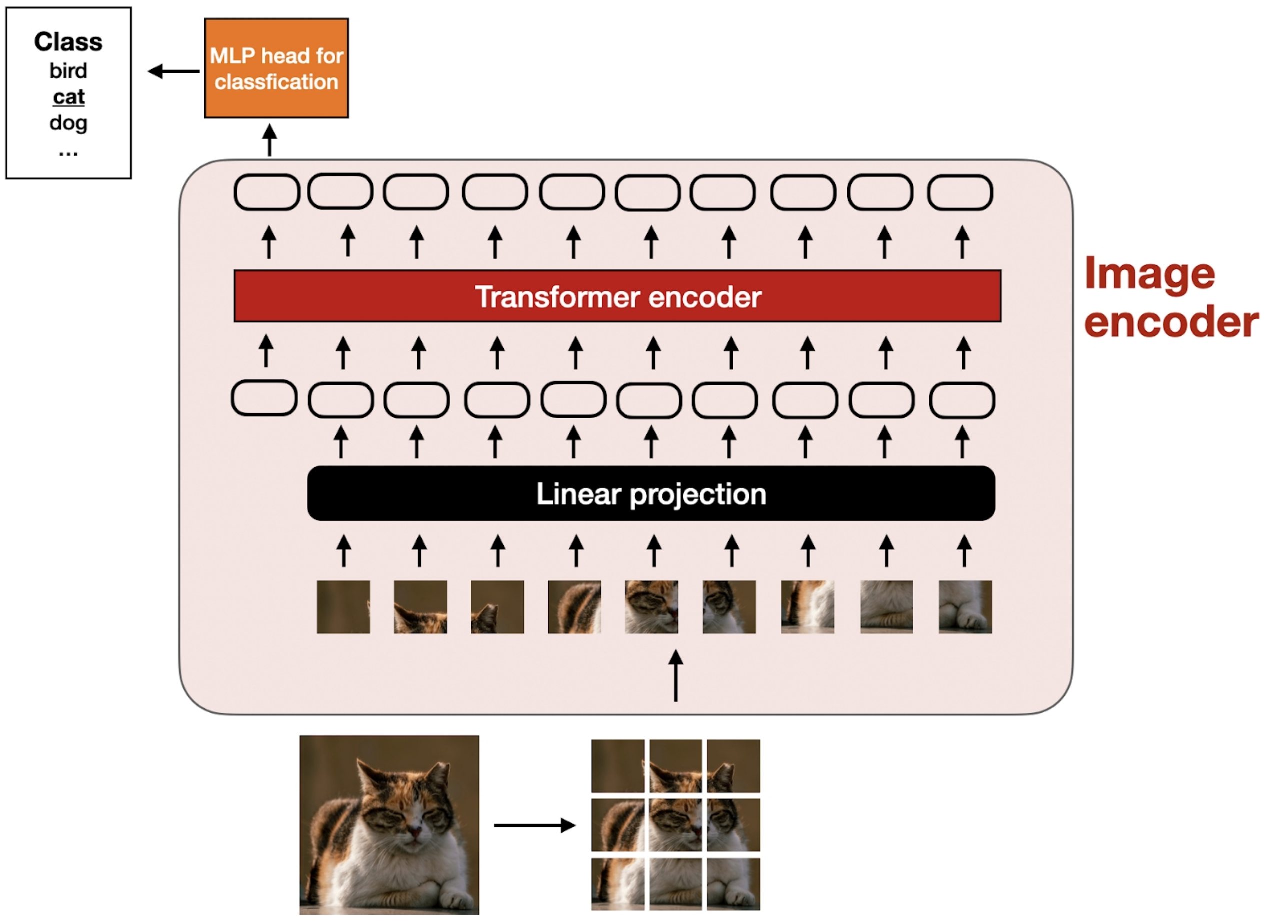
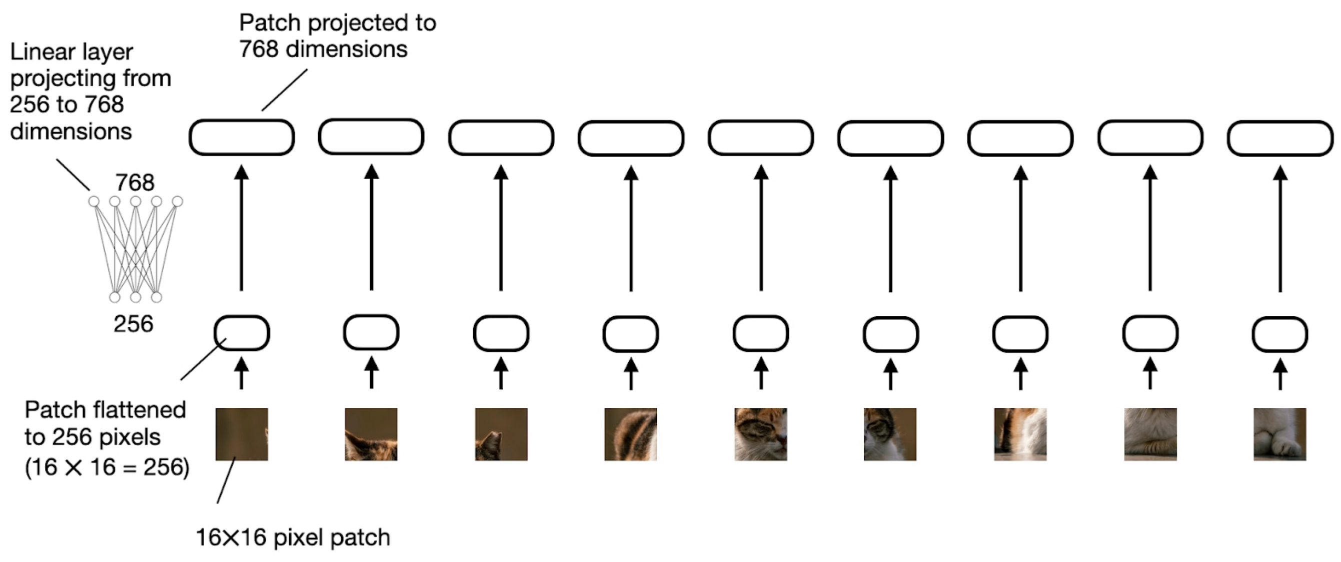


Figure 6: Illustration of a classic vision transformer (ViT) setup. Source: Sebastian Raschka

2. Linear Projection Module: The linear projection shown in the below figure consists of a single linear layer (i.e., a fully connected layer). The purpose of this layer is to project the image patches, which are flattened into a vector, into an embedding size compatible with the transformer encoder.



Patch projected to 768 dimensions
Linear layer projecting from 256 to 768 dimensions
768
256
Patch flattened to 256 pixels (16 × 16 = 256)
16×16 pixel patch

Figure 7: Illustration of a linear projection layer that projects flattened image patches from a 256-dimensional into a 768-dimensional embedding space. Source: Sebastian Raschka

3. Projector: An additional linear projection layer that projects the image encoder outputs into dimensions matching the embedded text tokens.

   The projector is sometimes also called adapter or connector.

4. Unmodified LLM Decoder: A standard decoder-style LLM that receives inputs consisting of image token and text token embeddings.

Image vs Text Tokenization

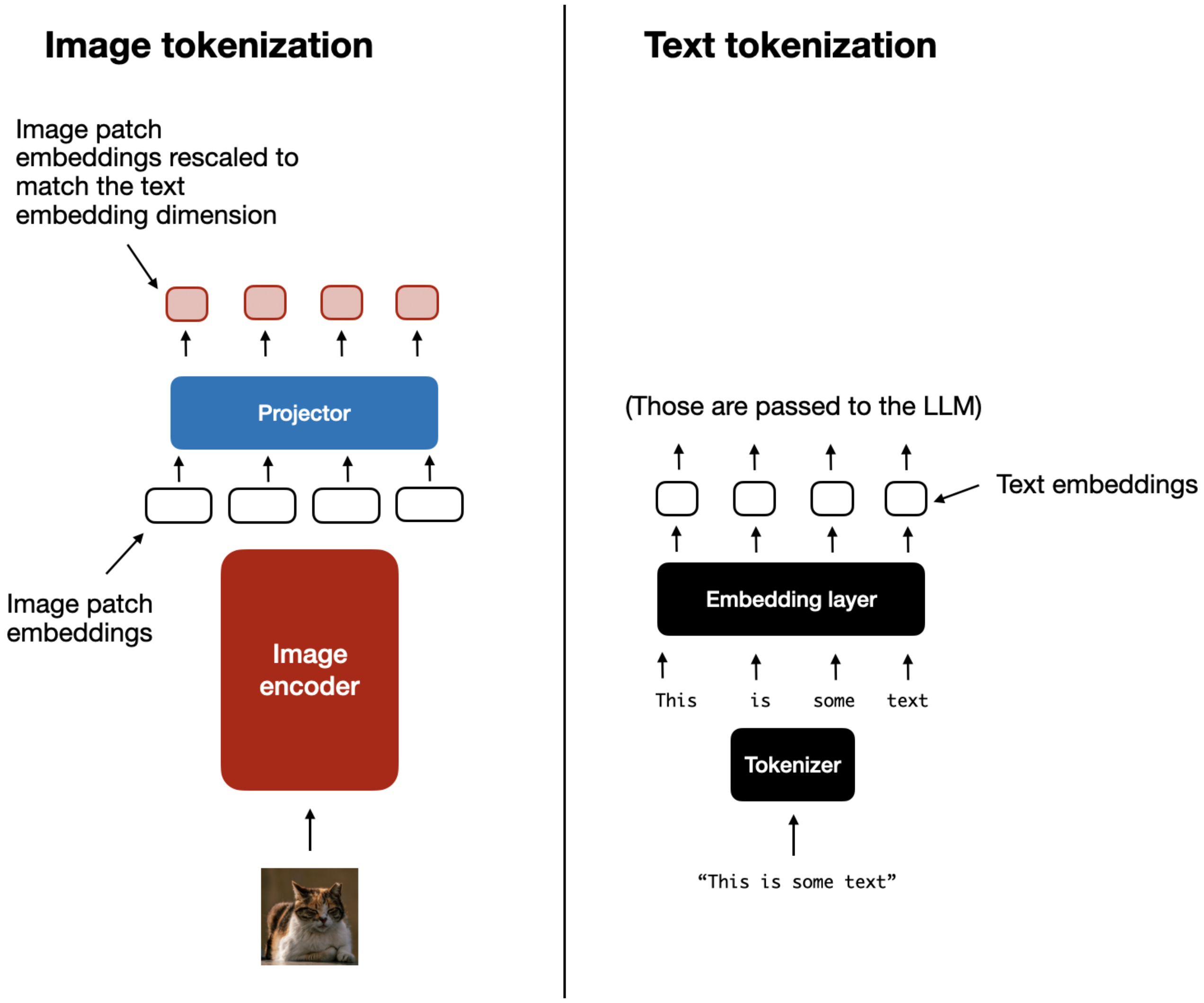


Figure 8: Image tokenization and embedding (left) and text tokenization and embedding (right) side by side. Source: Sebastian Raschka

The figure (left) shows an additional projector module that follows the image encoder.

This projector is typically another linear projection layer, similar to the one explained earlier.

Its function is to map the image encoder outputs into a dimensionality that matches that of the embedded text tokens.

Image Tokenization

1. Images are divided into smaller patches.

2. Patches are encoded by a pretrained vision transformer.

3. The linear projection layer projects flattened patches from their original dimension into a higher-dimensional embedding space (e.g. 256-dimensional to 768-dimensional).

4. The projector aligns image embeddings with text token dimensions.

5. Image and text embeddings are concatenated as input to the LLM.

Now that the image patch embeddings have the same embedding dimension as the text token embeddings, the projector can concatenate them as input to the LLM.

Method A: Advantages

1. Easier to implement as it doesn't require modifications to the LLM architecture.
2. Straightforward training process.
3. Can leverage existing pretrained LLMs directly.

Some versions of Method A, such as Fuyu, operate directly on image patches without a separate pretrained image encoder, instead using a linear projection to learn image patch embeddings, which simplifies the architecture.

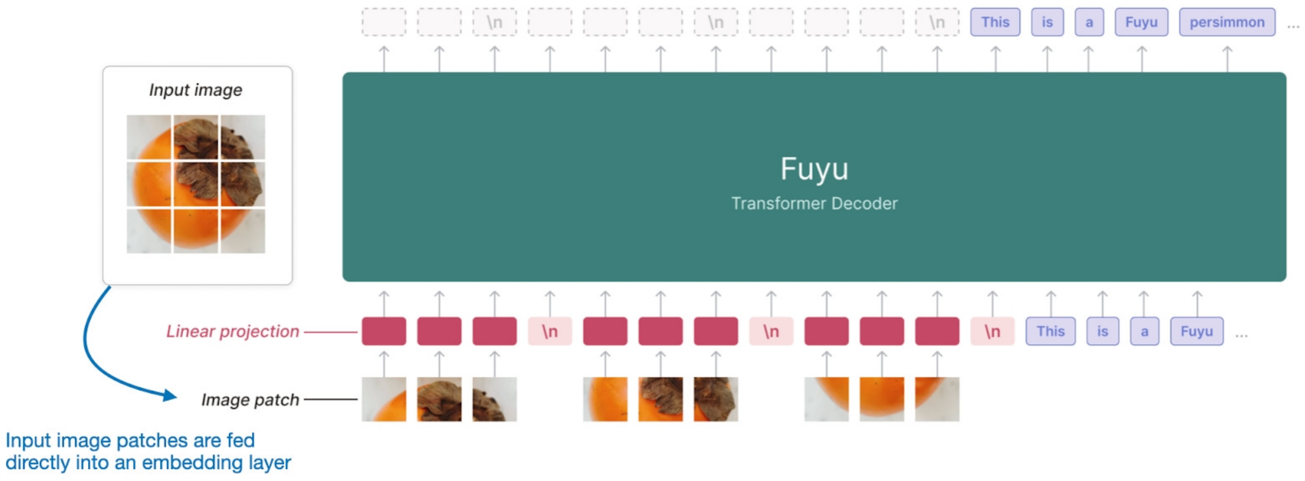


Figure 9: Annotated figure of the Fuyu multimodal LLM that operates directly on image patches without image encoder. Source: Sebastian Raschka

Method B: Cross-Modality Attention Architecture

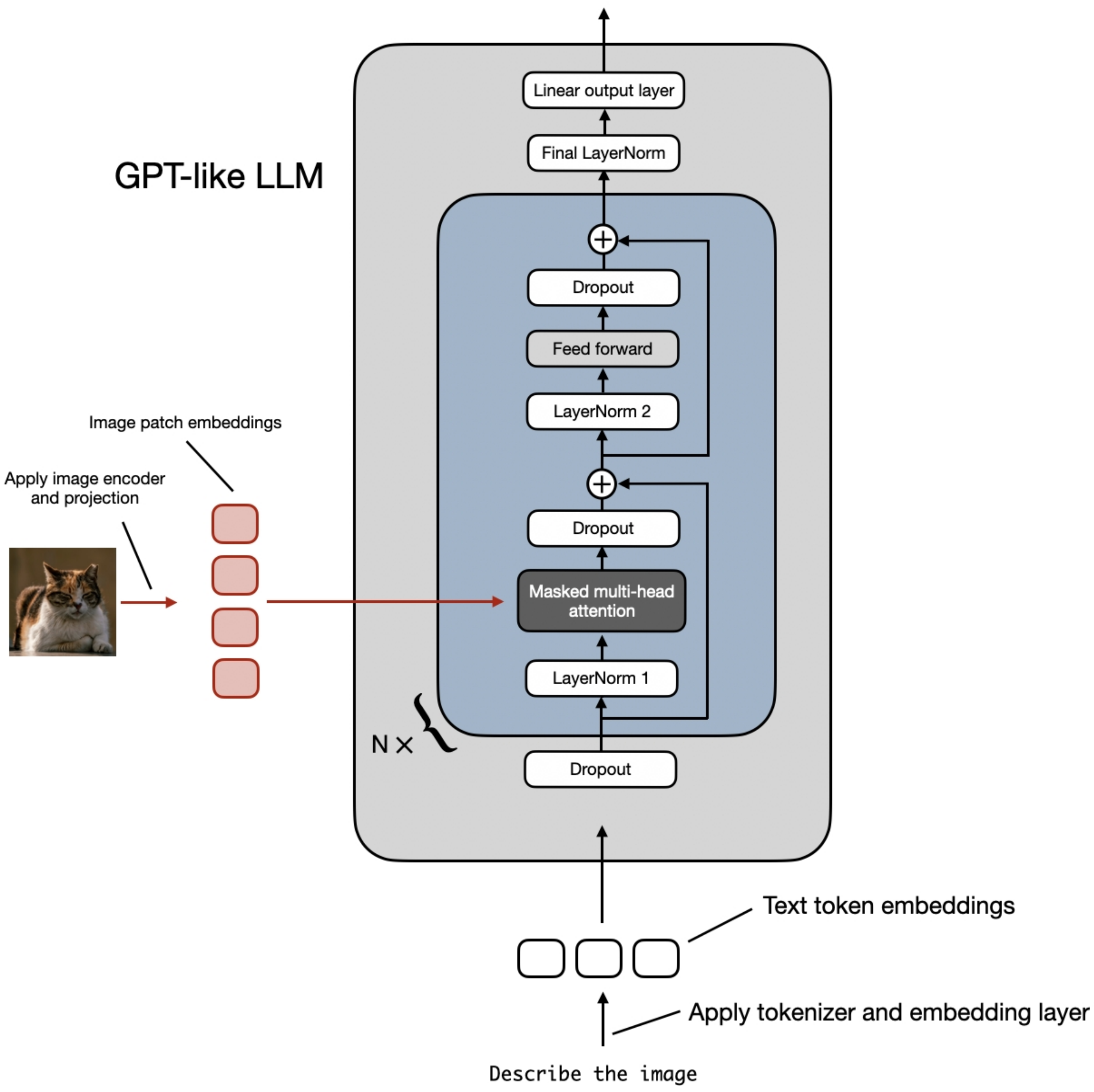


Figure 10: Illustration of the Cross-Modality Attention Architecture approach to building multimodal LLMs. Source: Sebastian Raschka

An alternative approach, **Method B**, connects input patches within the multi-head attention layer via a cross-attention mechanism derived from the original Transformer architecture introduced in the “**Attention Is All You Need**” paper.

Method B: Components

1. Image Encoder: Same as Method A, typically a pretrained vision transformer.

2. Cross-Attention Layers: Integrate image and text embeddings directly within the attention mechanism.

3. Modified LLM: The LLM architecture is modified to include cross-attention layers.

In the context of multimodal LLMs, the encoder is an **image encoder** rather than a text encoder, but the same underlying principle from the original Transformer architecture applies.

Cross-Attention

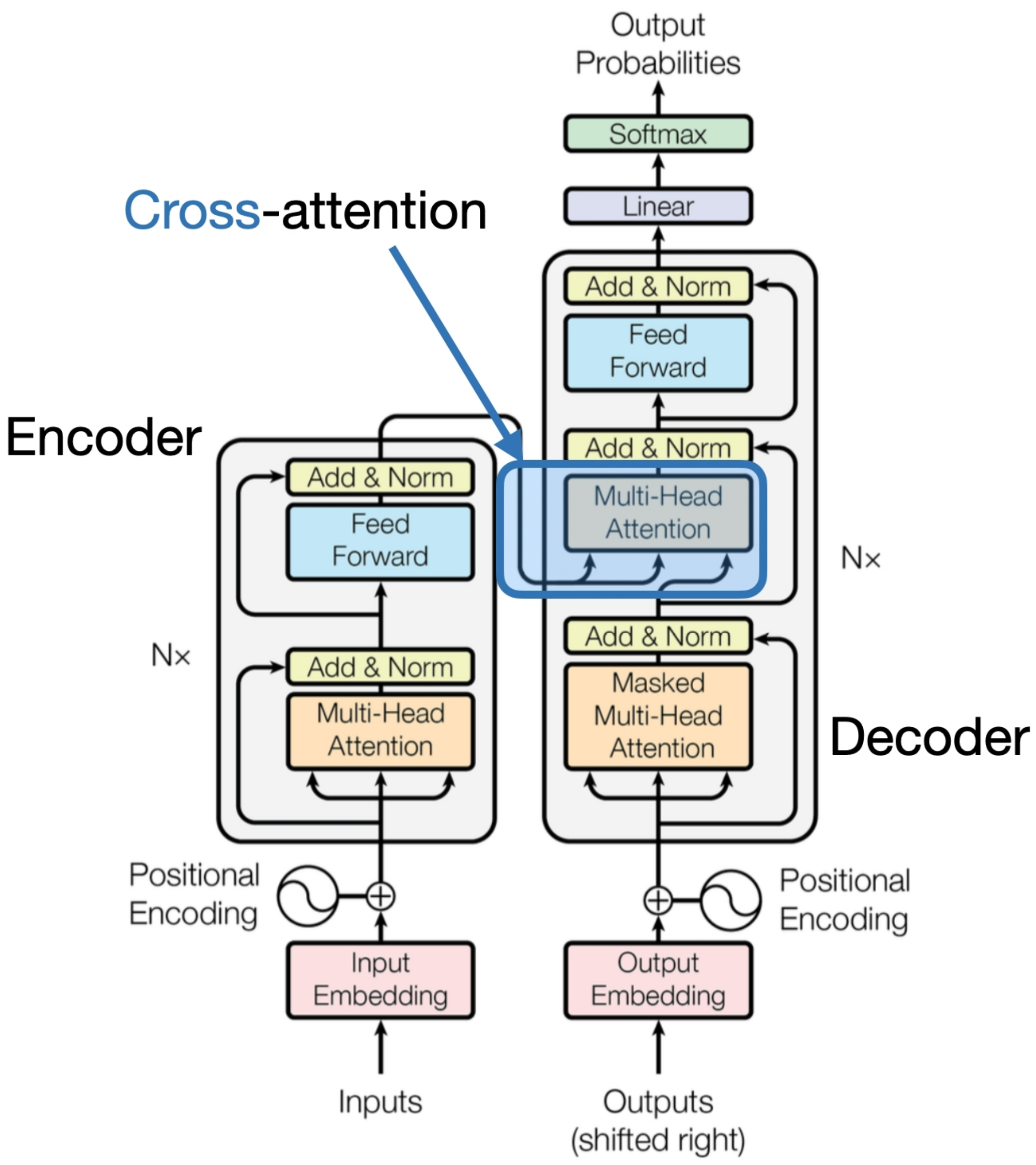


Figure 11: High-level illustration of the cross-attention mechanism used in the original transformer architecture. Source: Sebastian Raschka

Self-Attention Module

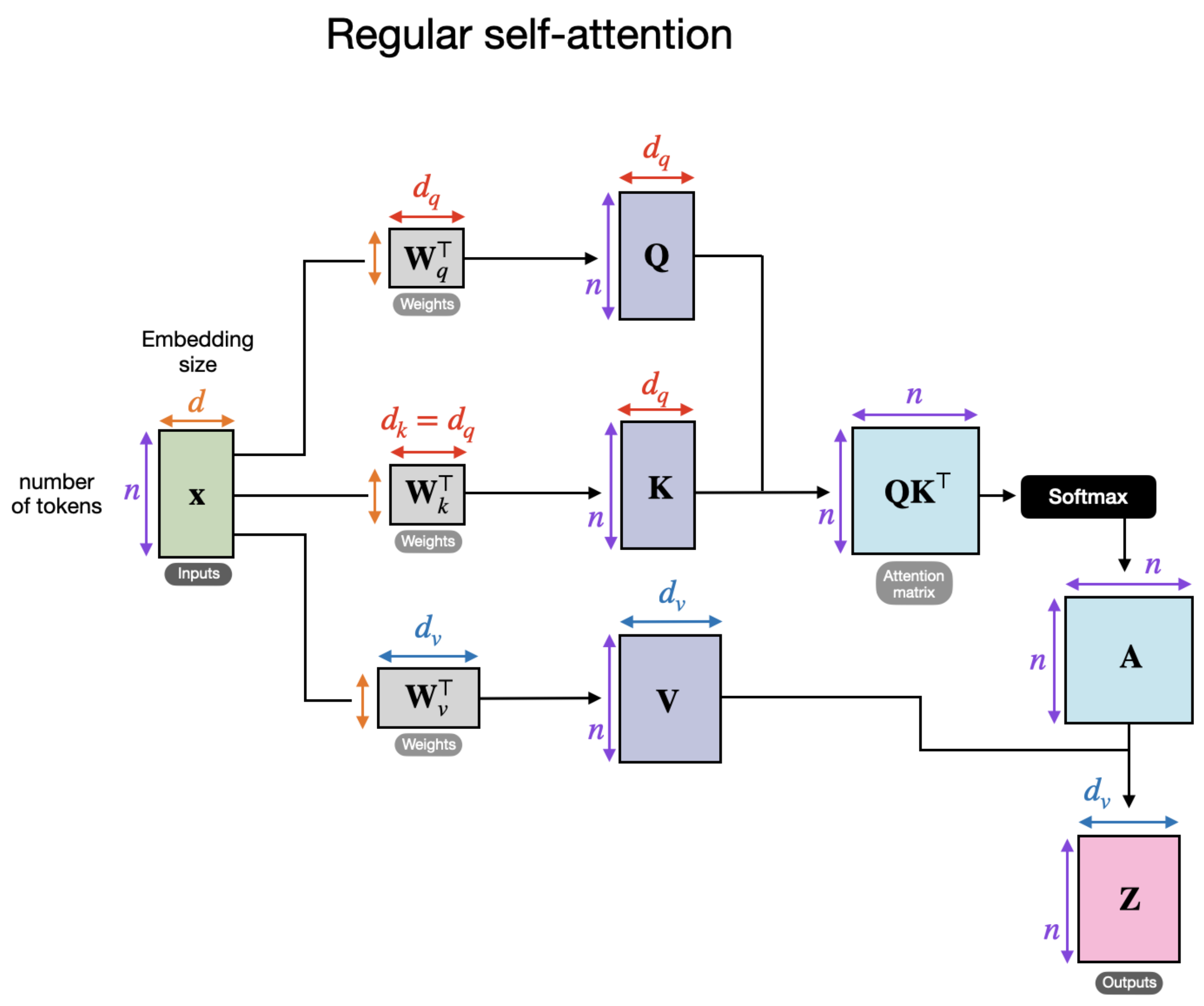


Figure 12: Outline of the regular self-attention mechanism. Source: Sebastian Raschka

In self-attention, queries (Q), keys (K), and values (V) all come from the same input sequence.

Cross-Attention Module

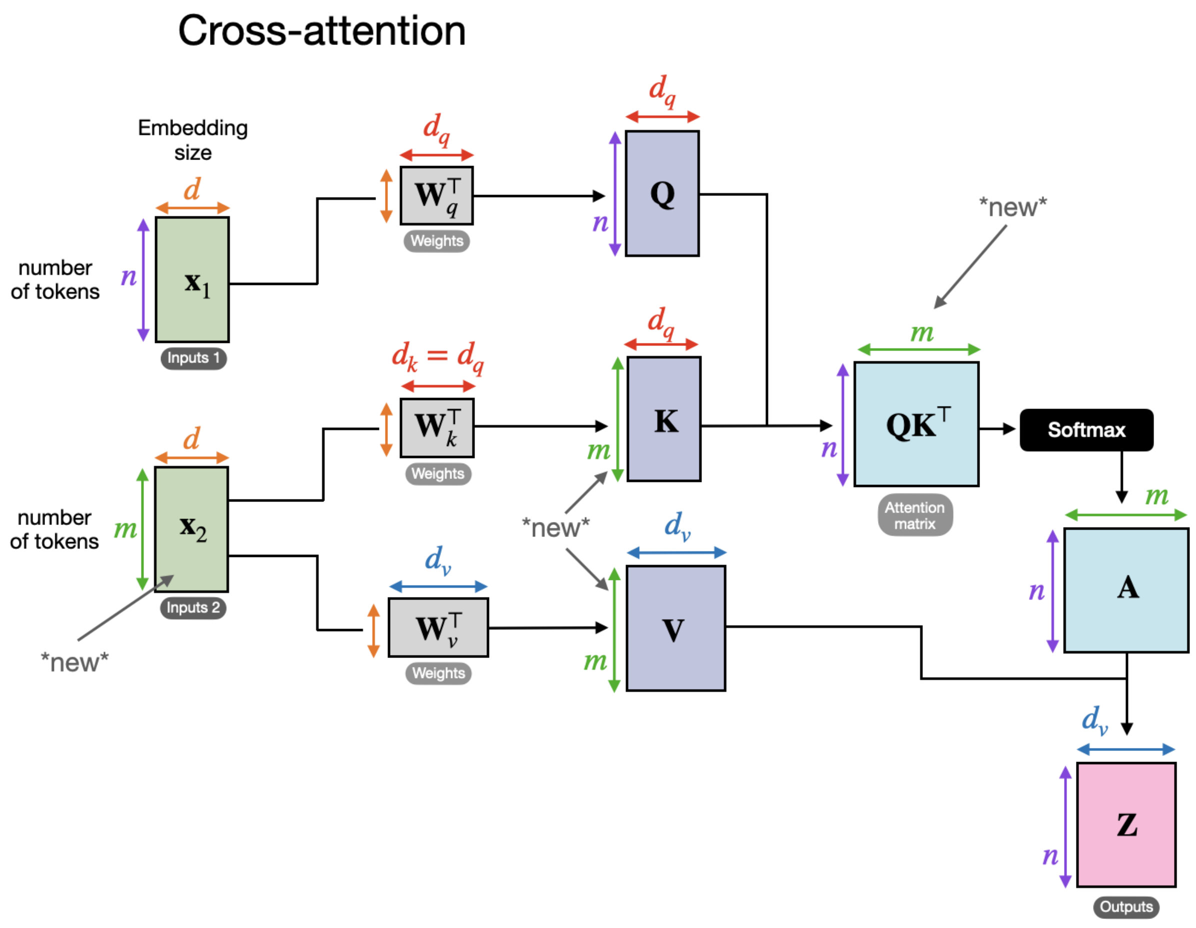


Figure 13: Illustration of cross-attention, where there can be two different inputs x1 and x2. Source: Sebastian Raschka

In cross-attention, there are two different input sources: one for queries (typically from the decoder text sequence) and another for keys and values (from the encoder image sequence).

The two input sequences can have different numbers of elements, but their embedding dimensions must match.

The cross-attention allows the model to "attend" to image features while processing text.

Method B: Advantages

1. More computationally efficient as it doesn't overload the input context with additional image tokens.

2. Maintains text-only performance of the original LLM if LLM parameters are kept frozen during training.

3. Introduces image information later in the processing pipeline through cross-attention layers.

3. Methods A and B model training

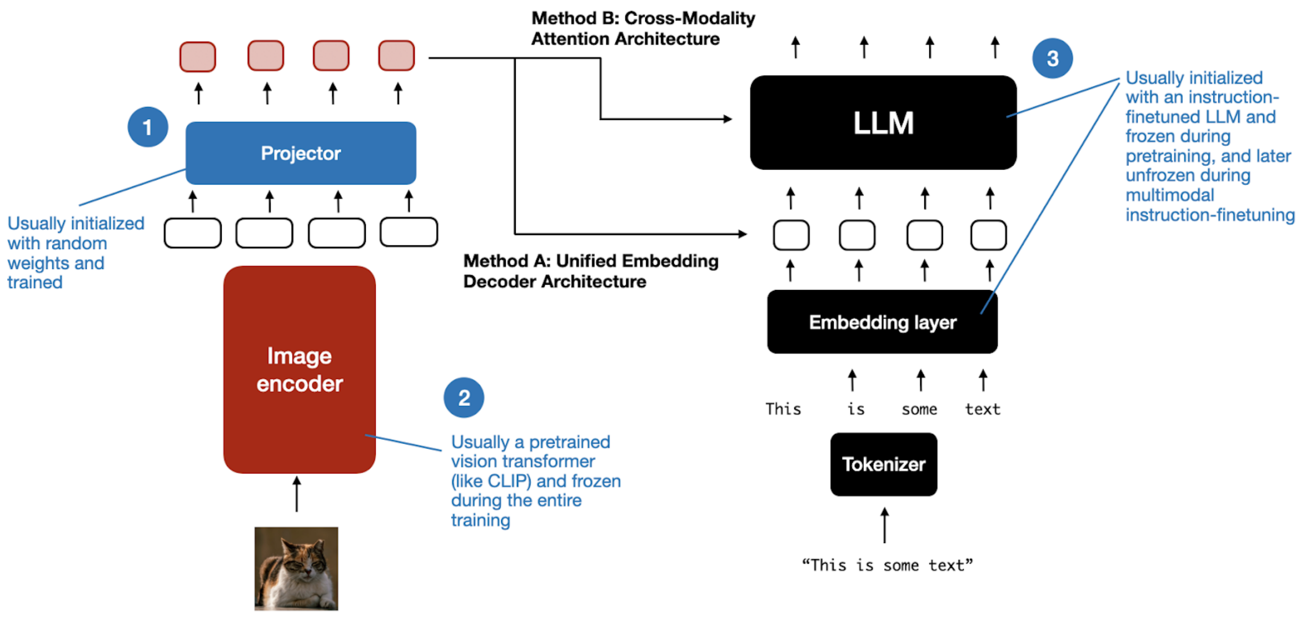

Method B: Cross-Modality Attention Architecture
1
Projector
Usually initialized with random weights and trained
Image encoder
2
Usually a pretrained vision transformer (like CLIP) and frozen during the entire training
Method A: Unified Embedding Decoder Architecture
3
LLM
Usually initialized with an instruction-finetuned LLM and frozen during pretraining, and later unfrozen during multimodal instruction-finetuning
Embedding layer
This
is
some
text
Tokenizer
"This is some text"

*Figure 14: An overview of **Unified Decoder and Cross-Attention t**raining for Multimodal LLMs. The components numbered 1-3 can be frozen or unfrozen during the multimodal training process. Source: Sebastian Raschka*

Training Process

Similar to traditional text-only LLMs, multimodal LLMs training involves two phases:

1. Pretraining

2. Instruction finetuning

However, unlike training from scratch, multimodal LLM training typically begins with a pretrained, instruction-finetuned text-only LLM as the base model.

Keeping the LLM part frozen during the pretraining phase is also common, with training focused on the projector, which can be linear layer or a small multi-layer perceptron.

Component-Specific Training Strategies

Image Encoder

- Commonly uses CLIP or OpenCLIP (pretrained vision transformers).
- Often remains frozen during the entire training process.

- Some approaches (like Llama 3.2) train the image encoder from scratch or update it during training.

Projector or Adapter
- Typically a linear layer or small multi-layer perceptron (1-2 layers).
- Usually trained during the pretraining phase while keeping the LLM frozen.
- Has limited learning capacity due to its small size.

LLM Decoder
- Often frozen during the pretraining phase.
- Commonly unfrozen during instruction finetuning to allow more comprehensive updates.
- In Method A (Unified Embedding Decoder), may be unfrozen in later stages.
- In Method B (Cross-Attention), the original LLM parameters may remain frozen while cross-attention layers are trained.

Cross-Attention Layers (Method B only)
- Unfrozen throughout the entire training process.
- Add substantial parameters (e.g., 3B parameters for 8B model, 20B for 70B model).
- Often added to every fourth transformer block to balance performance and computational cost.

Training Phase Strategy

Stage 1 (Pretraining)
- Focus on training the projector/adapter.
- LLM often frozen, image encoder typically frozen.
- Uses large-scale image-text pair datasets.

Stage 2 (Instruction Finetuning)
- Unfreeze LLM parameters for comprehensive updates.
- Adapter or projector continues training.
- Cross-attention layers (if applicable) remain unfrozen.
- Uses instruction-following datasets.

Comparisons and Considerations of Methods

- Method A is easier to implement but may require more input context.
- Method B is more computationally efficient but requires architectural modifications.
- The choice depends on trade-offs between implementation complexity, computational efficiency, and the preservation of text-only performance.

4. Multimodal Methods in Large Language Models

Recent methods demonstrate the evolution of the multimodal large language models, with researchers exploring various combinations of architectural choices, training strategies, and component designs to optimize trade-offs among performance, efficiency, and capabilities.

Image Encoder Choices

- Pretrained CLIP/OpenCLIP: Most common approach, leveraging existing vision-language models.

- Custom Vision Transformers: Some models (like Llama 3.2 and Pixtral) train image encoders from scratch for better control.

- SigLIP: An alternative vision encoder used in several recent models.

- InternViT-6B: A larger 6-billion-parameter image encoder used in NVLM.

Resolution Handling

- Fixed Resolution: Traditional approach using standard image sizes.

- Naive Dynamic Resolution: Qwen2-VL's mechanism allows images of varying resolutions without downsampling by removing absolute position embeddings and introducing 2D-RoPE.

- Native Variable Sizes: Pixtral and Qwen2-VL support variable image sizes natively, processing images at their original resolution.

Projector or Adapter

- Single Linear Layer: Simplest approach, projects image embeddings to match text token dimensions.

- Multi-layer Perceptron: More complex projector with 2-3 layers for better feature transformation.

- Convolutional Operations: Can mathematically replace linear layers while combining patch creation and projection.

Training Strategy

- All Parameters Unfrozen: Molmo's approach updates image encoder, connector, and LLM simultaneously in a unified training.

- Gradual Unfreezing: Baichuan-Omni's three-stage approach: (1) train projector only, (2) train vision encoder, (3) train entire model end-to-end.

- Selective Freezing: Llama 3.2 updates image encoder but freezes LLM parameters to preserve text-only capabilities.

- Phase-Specific Strategies: Qwen2-VL trains in three stages with different component freezing patterns.

Hybrid Approaches

- NVLM-H: Combines decoder-only (Method A) and cross-attention (Method B) approaches by providing image thumbnails as input followed by dynamic patches through cross-attention.

- Unified Understanding and Generation: Janus decouples visual encoding pathways for different tasks, using separate encoders for understanding (high-dimensional semantic) and generation (detailed local information).

Llama 3.2

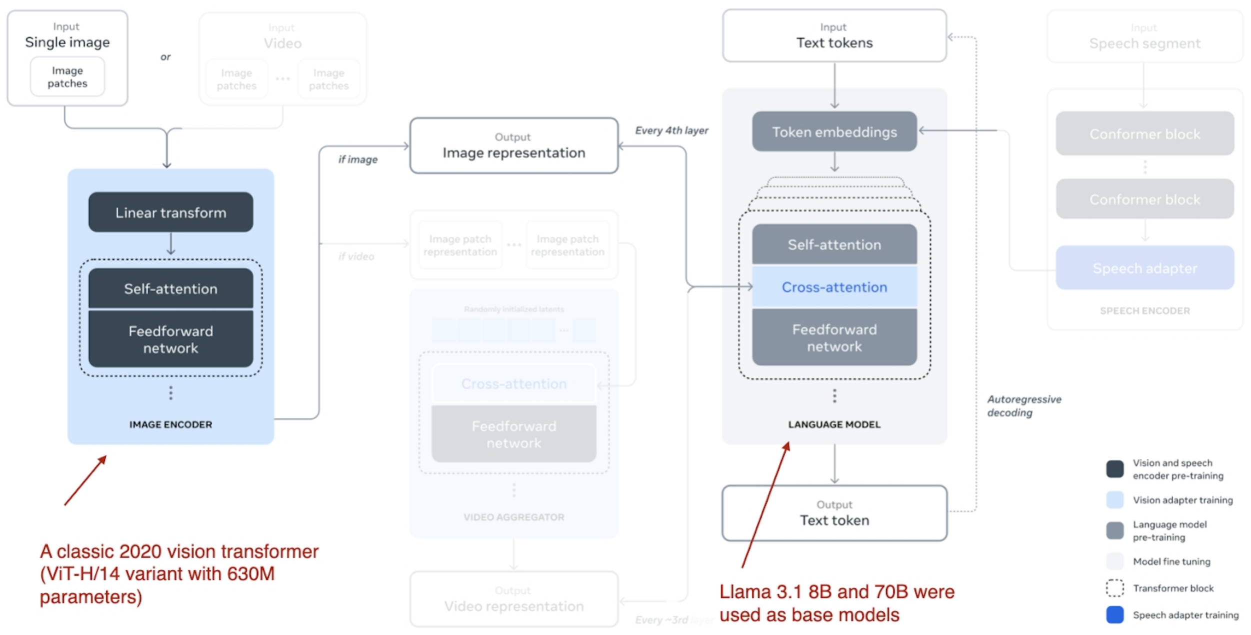


Figure 15: Illustration of the multimodal LLM approach used by Llama 3.2. Source: Sebastian Raschka

Molmo

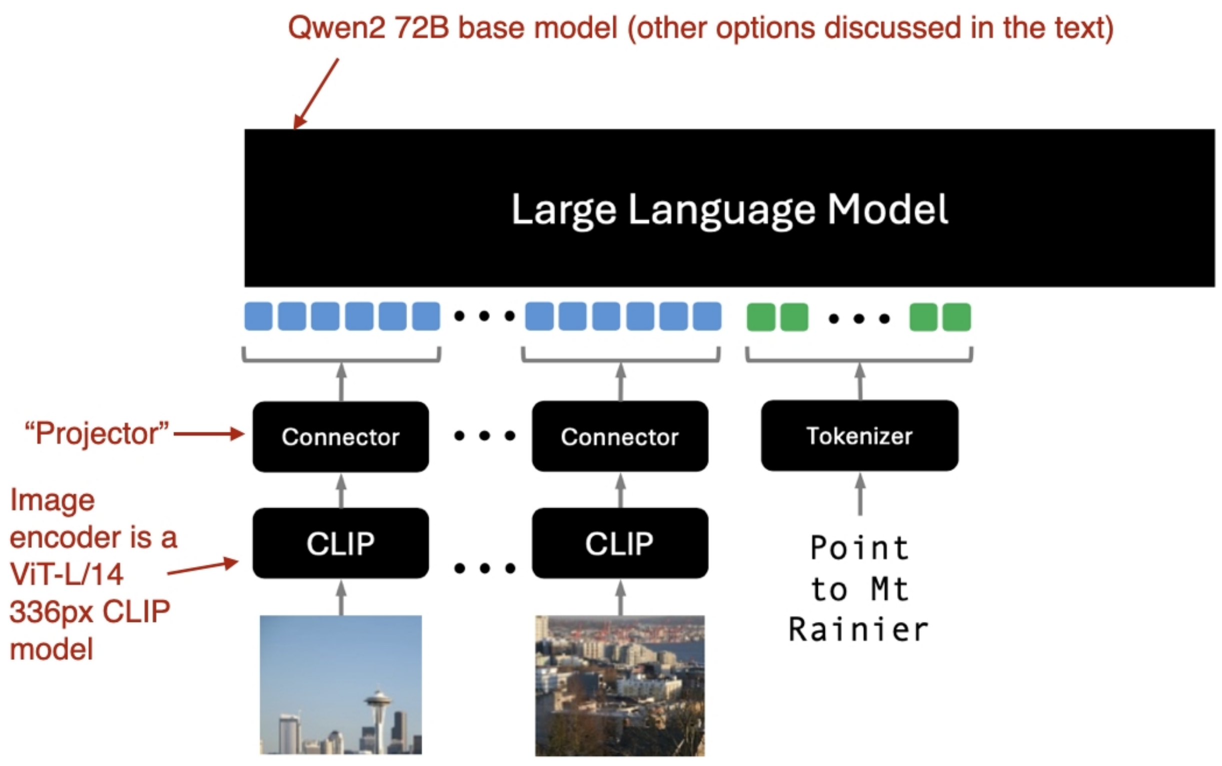
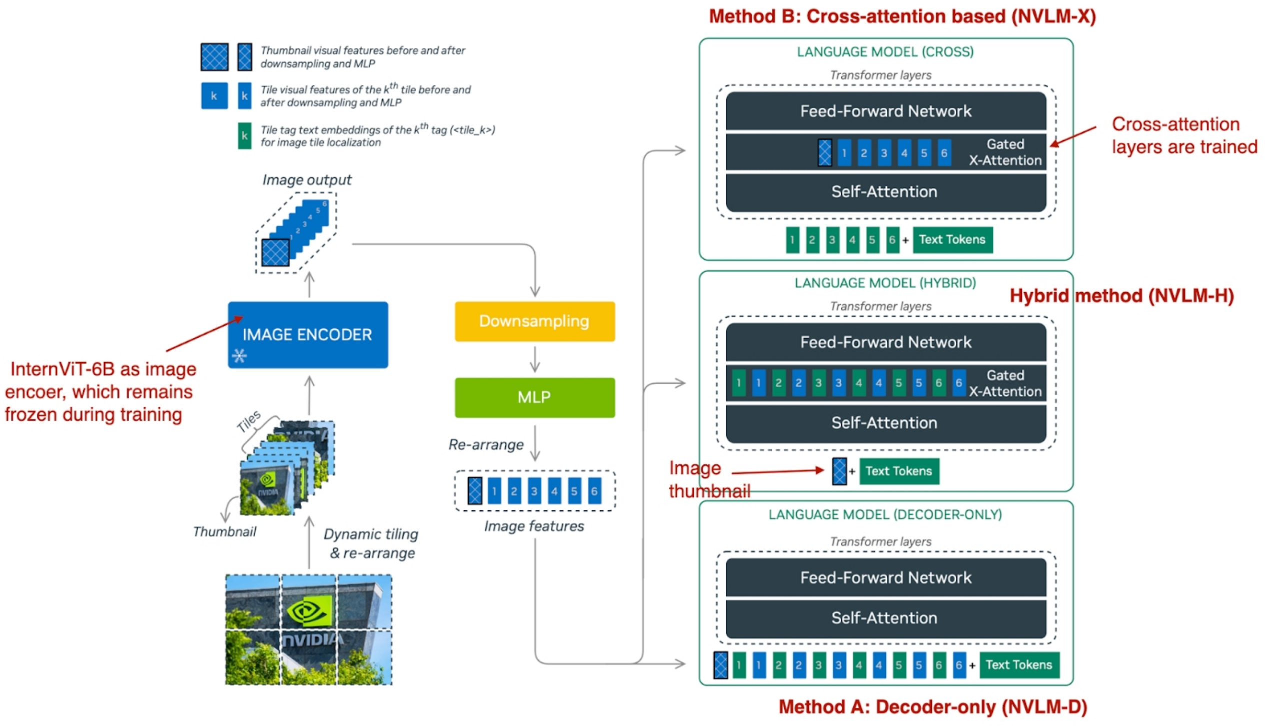

Qwen2 72B base model (other options discussed in the text)
Large Language Model
"Projector"
Connector
Connector
Tokenizer
Image encoder is a ViT-L/14 336px CLIP model
CLIP
CLIP
Point to Mt Rainier

Figure 16: Illustration of the Molmo decoder-only approach (Method A). Source: Sebastian Raschka

NVLM



Method B: Cross-attention based (NVLM-X)
Thumbnail visual features before and after downsampling and MLP
Tile visual features of the kth tile before and after downsampling and MLP
Tile tag text embeddings of the kth tag (<tile_k>) for image tile localization
Image output
IMAGE ENCODER
InternViT-6B as image encoer, which remains frozen during training
Tiles
Thumbnail
Dynamic tiling & re-arrange
Downsampling
MLP
Re-arrange
Image features
LANGUAGE MODEL (CROSS)
Transformer layers
Feed-Forward Network
Gated X-Attention
Self-Attention
Text Tokens
Cross-attention layers are trained
LANGUAGE MODEL (HYBRID)
Hybrid method (NVLM-H)
Image thumbnail
LANGUAGE MODEL (DECODER-ONLY)
Method A: Decoder-only (NVLM-D)

Figure 17: Overview of the three multimodal approaches in NVLM. Source: Sebastian Raschka

Qwen2-VL

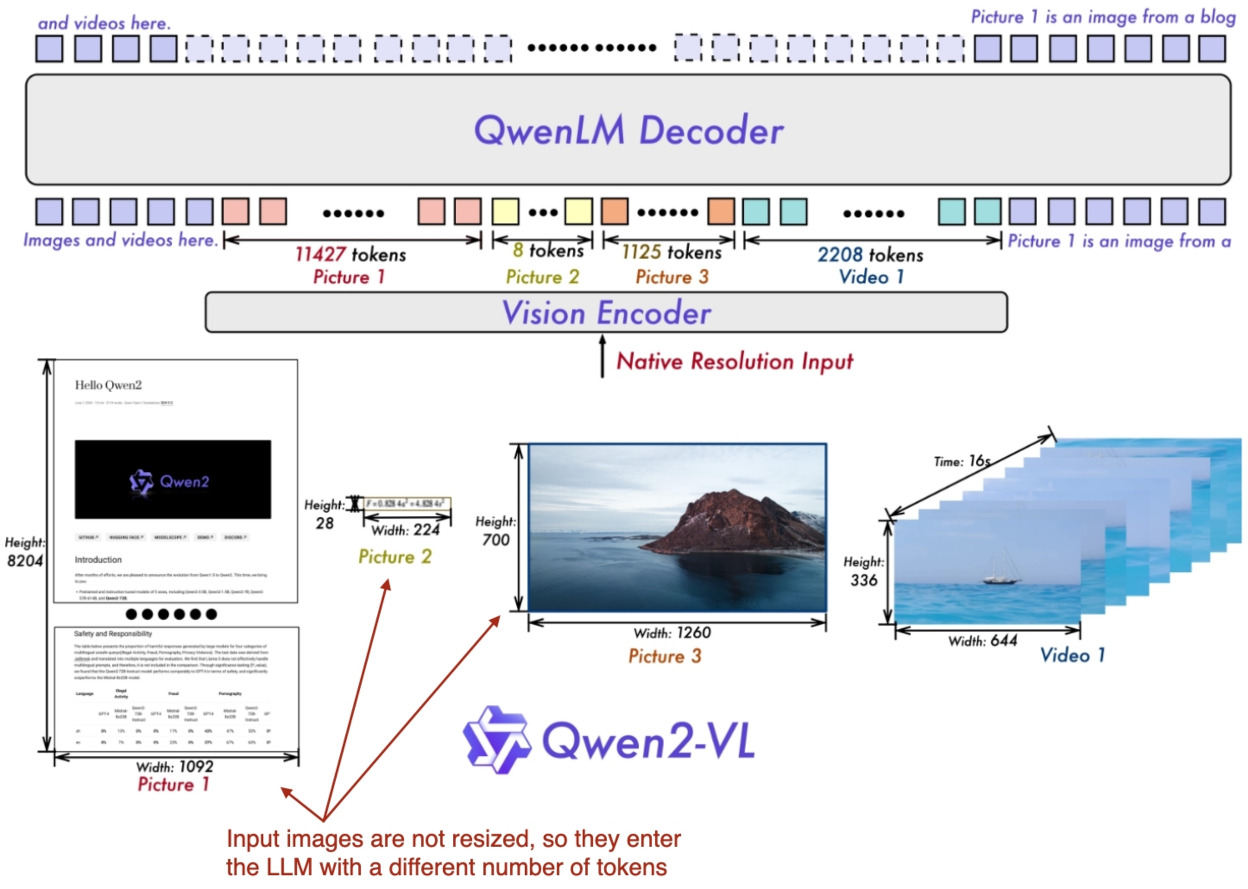


Figure 18: An overview of the multimodal Qwen model, which can process input images with various different resolutions natively. Source: Sebastian Raschka

Conclusion

The **multimodal LLMs** demonstrates remarkable flexibility in architectural choices, ranging from **unified decoder** approaches to **cross-attention** methods, and from using pretrained components to training custom encoders from scratch.

- Both Method A (**Unified Embedding Decoder**) and Method B (**Cross-Modality Attention**) are viable approaches with different trade-offs.

- There is no single best way to build multimodal LLMs because success depends on specific requirements and constraints.

- The choice of image encoder (CLIP, custom ViT, SigLIP) and training strategy (frozen vs. unfrozen components) varies widely across successful implementations.

- Recent innovations in resolution handling, hybrid architectures, and training strategies continue to push the boundaries of what's possible.

As the multimodal LLMs continues to evolve, further innovations in architectural design, training efficiency, and multimodal capabilities can be expected.

The diversity of successful approaches suggests that multimodal LLMs represent a resilient concept that can be implemented in multiple ways.

In any case, the key insight from this article is that multimodal LLMs can be successfully built in a variety of ways.

References

1. Raschka, Sebastian., "Understanding Multimodal LLMs", November 3, 2024. https://magazine.sebastianraschka.com/p/understanding-multimodal-llms

2. A. Vaswani et al., "Attention is all you need", arXiv, 12 June 2017, https://arxiv.org/pdf/1706.03762v1

3. Various technical papers on recent multimodal models (Llama 3.2, Molmo, NVLM, Qwen2-VL, Pixtral, MM1.5, Aria, Baichuan-Omni, Emu3, Janus) published in 2024 as referenced in the article.