Large Multi-modal Models (LMMs)

Fatemeh Seyyedsalehi

Sharif University of Technology

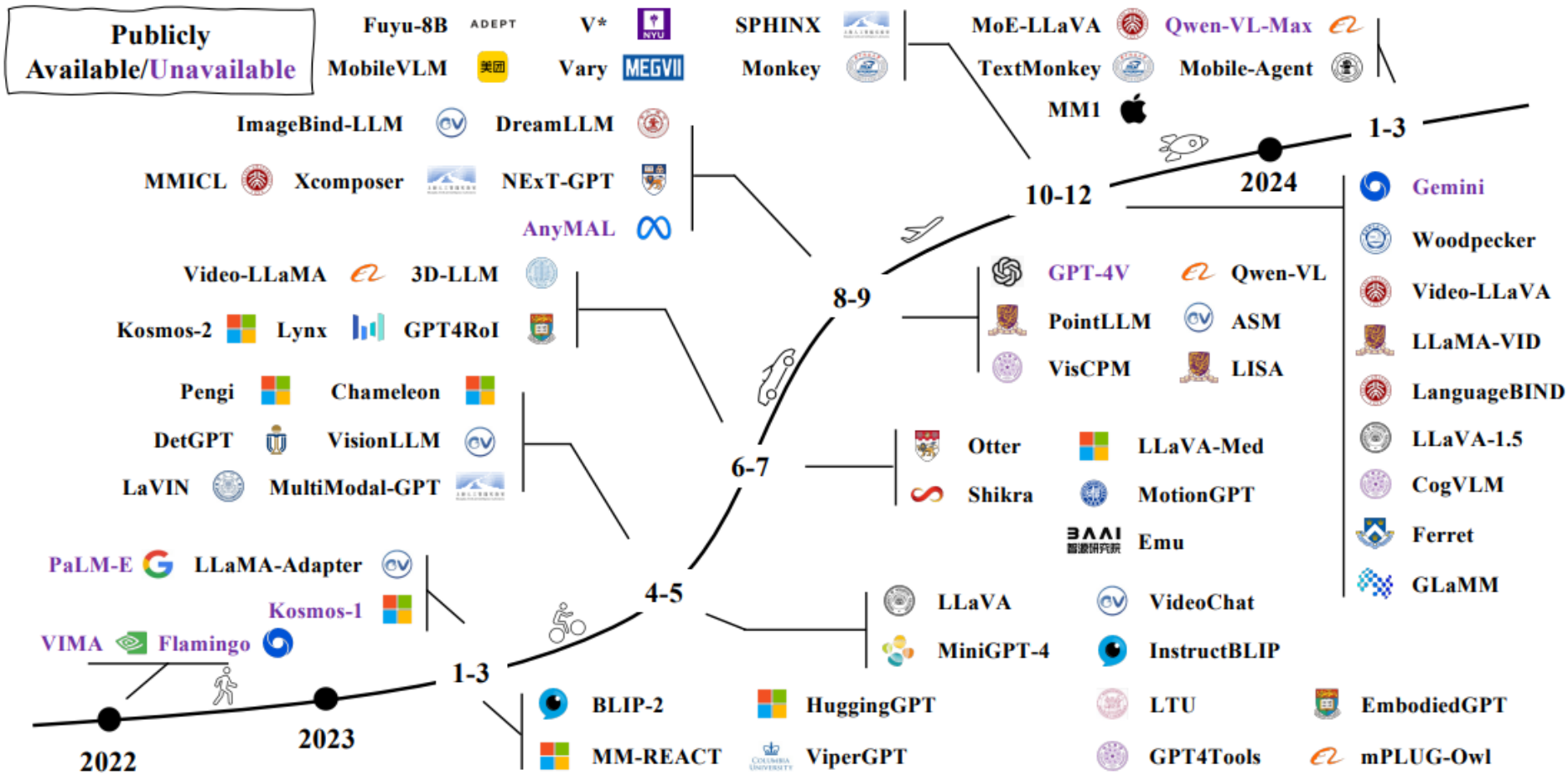
Multi-modal Models

- **Multi-modal Model (MM)**: AI system capable of processing or generating multiple data modalities (e.g., text, image, audio, video).
- **Multi-modal Foundation Models**: A multi-modal, large-scale, general-purpose AI model pre-trained on vast amounts of data (here multi-modal) that can be adapted (via fine-tuning or prompting) to a wide range of downstream tasks, ex. CLIP, DALL-E and **LMMs**.
- **Large Multi-modal Models (LMMs)**: They anchor on a Large Language Model (LLM) (e.g., GPT, LLaMA) as their reasoning core. Other modalities (vision, audio, etc.) are aligned to the LLM's text space for joint understanding/generation.

Large Multi-modal Models (LMMs)

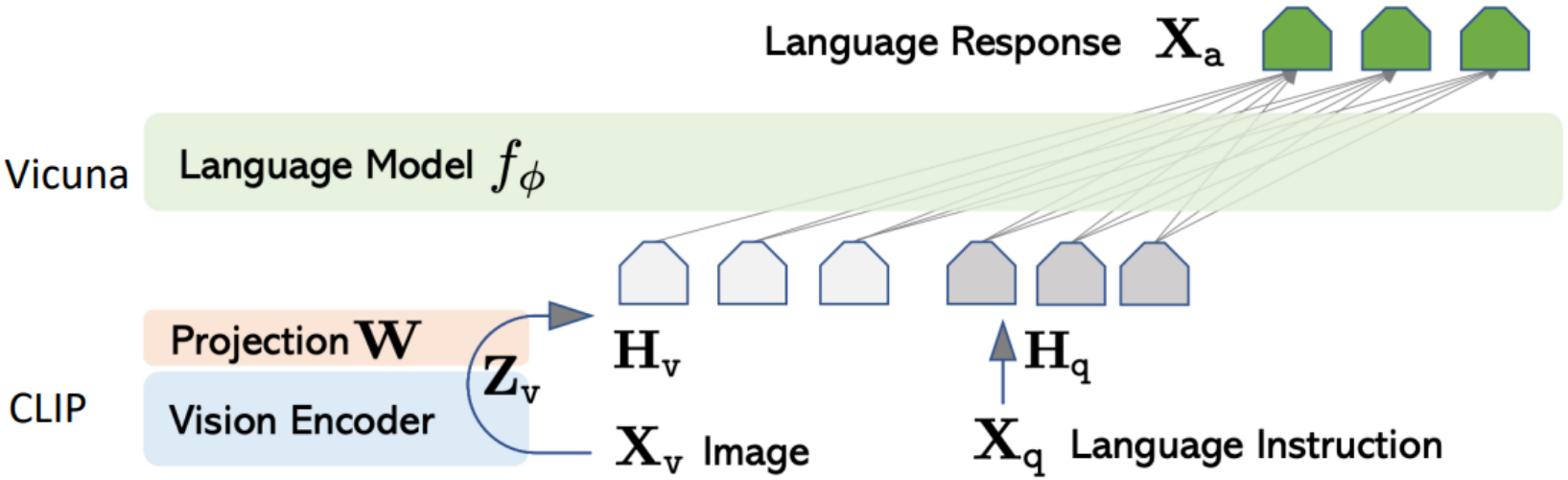
- LMMs are interactive AI systems that are expected to have emergent properties:
  - Chain of thought reasoning
  - In context learning
  - Instruction following

Large Multi-modal Models (LMMs)



LLaVA: Large Language-and-Vision Assistant

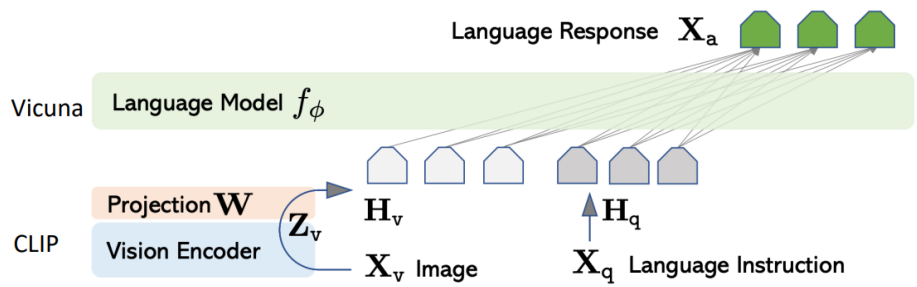
- The vision encoder converts input images into features Linear projection layer convert these features into a space compatible with the LLM.



LLaVA: Large Language-and-Vision Assistant

Two-stage Training:

- Stage 1: Pre-training for Vision-language Alignment. Only the projection matrix is updated, based on a subset of CC3M.
- Stage 2: Fine-tuning End-to-End. Both the projection matrix and LLM are finetuned on curated dataset
  - Visual Chat: Generated multimodal instruction data for daily user-oriented applications.
  - Science QA: Multimodal reasoning dataset for the science domain.

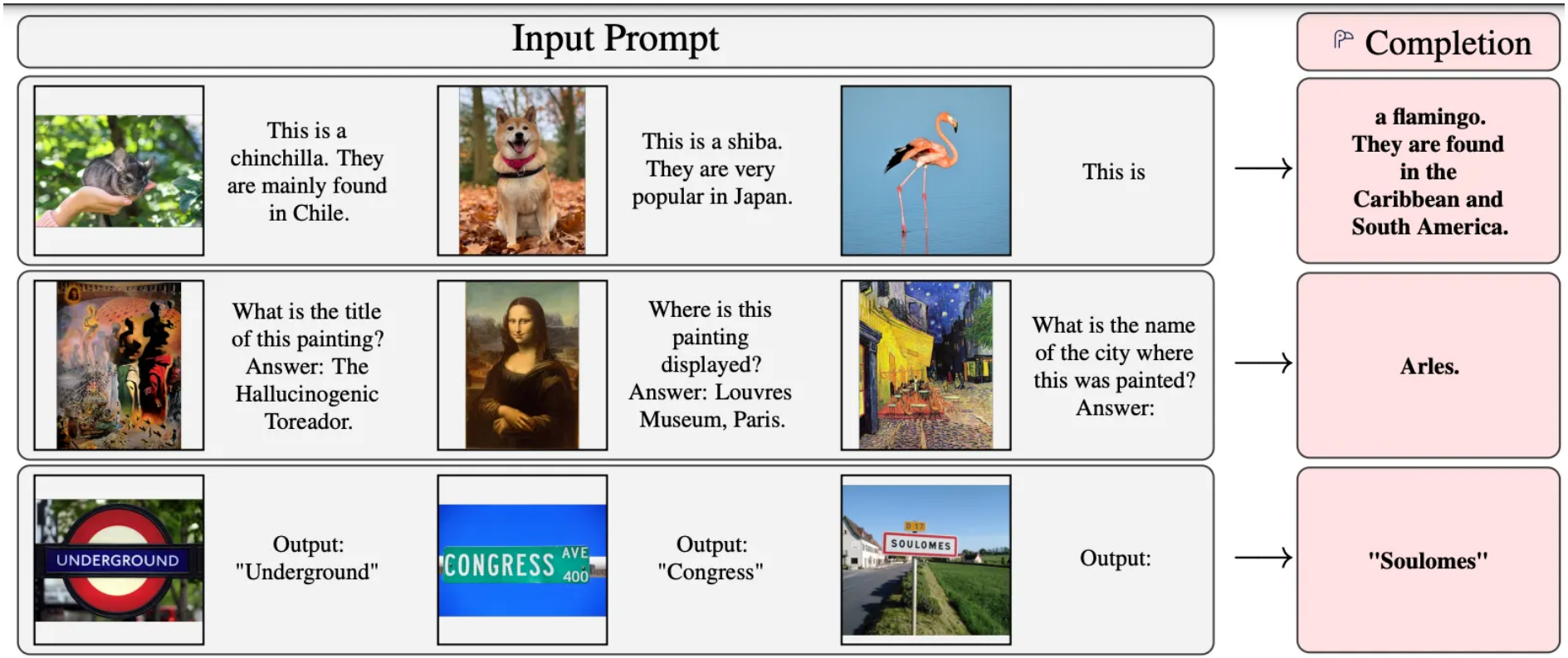


Flamingo

- Takes inputs a multimodal prompt containing images and/or videos interleaved with text and generates text like a standard Language Model.
- Introduced in paper Flamingo: a Visual Language Model for Few-Shot Learning in 2022.

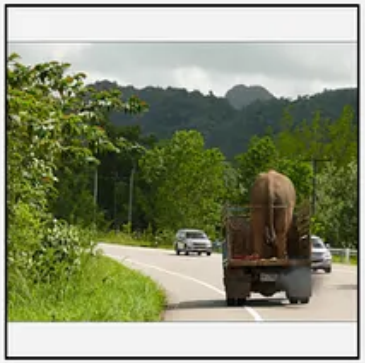
Flamingo

- Flamingo as a few-shot learner



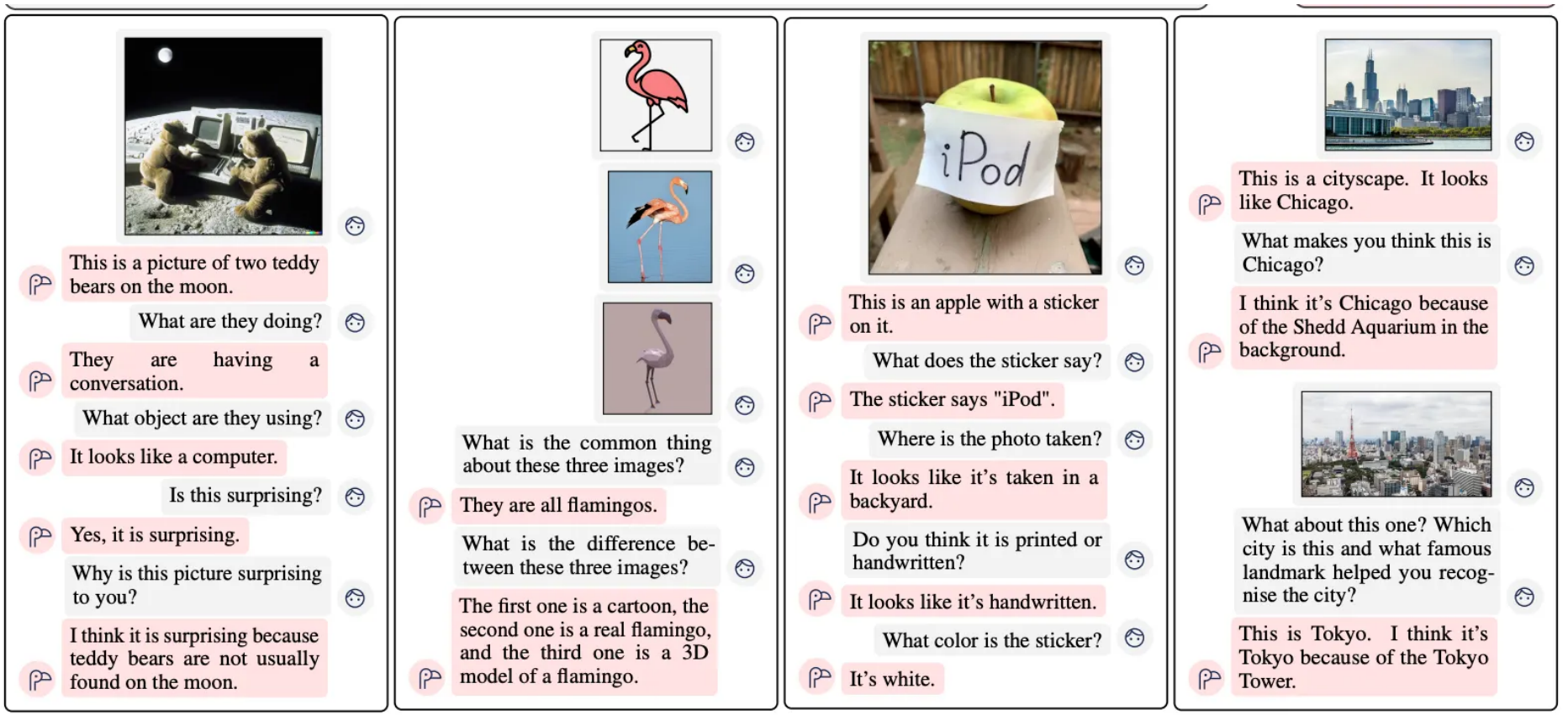
Flamingo

- One could also use Flamingo for visual question answering from an image.

Input Prompt	 Question: What do you think the capacities of these are? Answer:	 Question: What is odd about this image? Explain why it is unusual. Answer:	 Question: What country is this? Why do you think so? Answer:
Completion	**The floppy disk is 1.44MB and the CD is 700MB.**	**The image is odd because the elephant is in the back of the truck. It is unusual because elephants are not usually transported in the back of a truck.**	**It is Canada. I think so because the flag is the Canadian flag.**

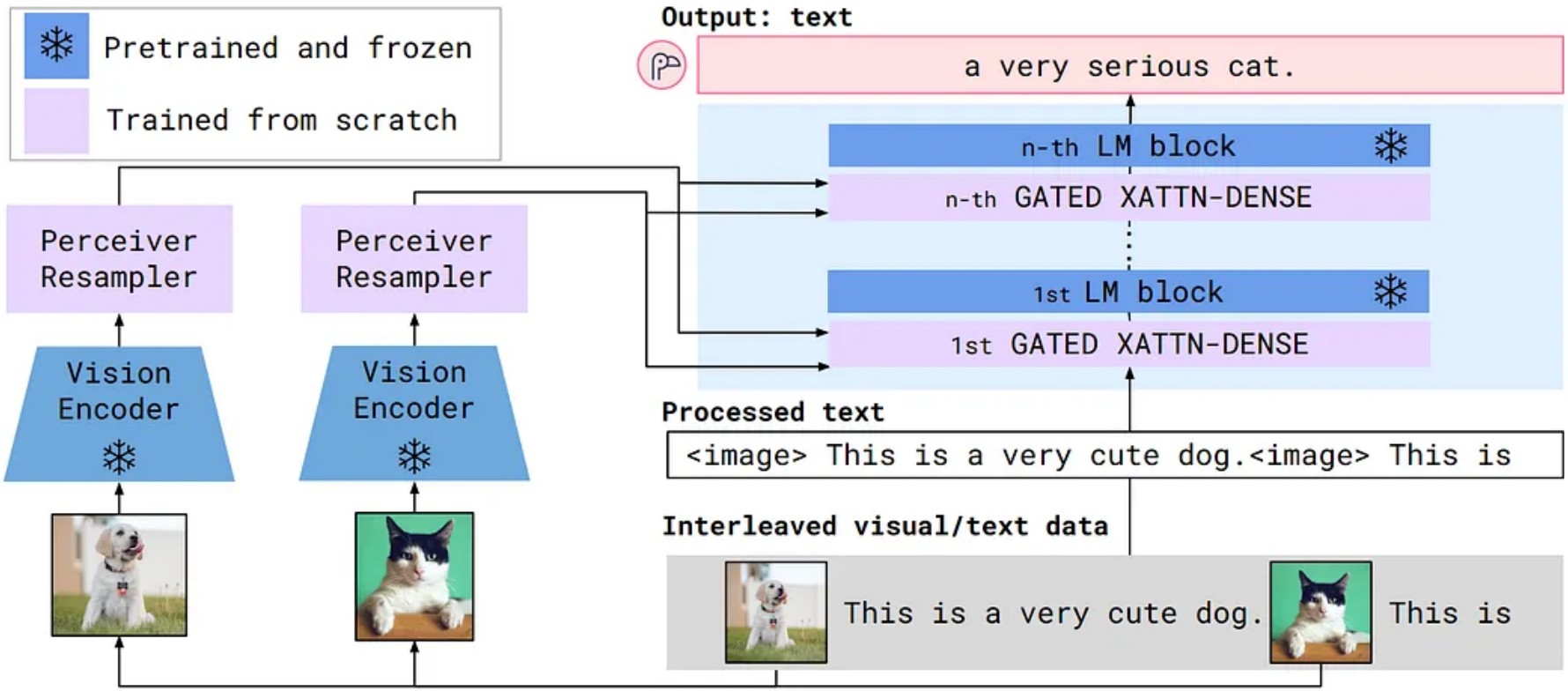
Flamingo

- Flamingo is also capable of multi-image visual dialogue out of the box.



Flamingo

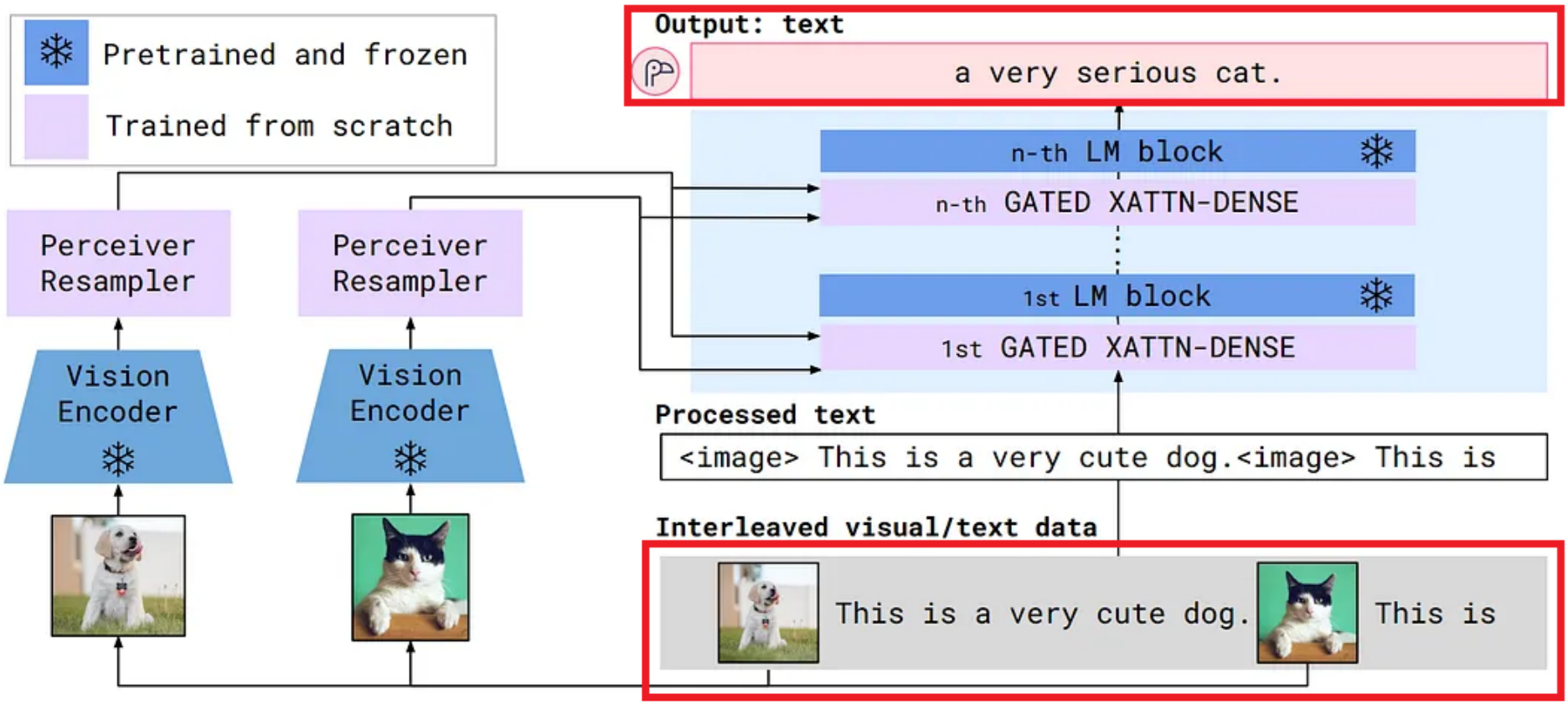
- Flamingo architecture



Flamingo

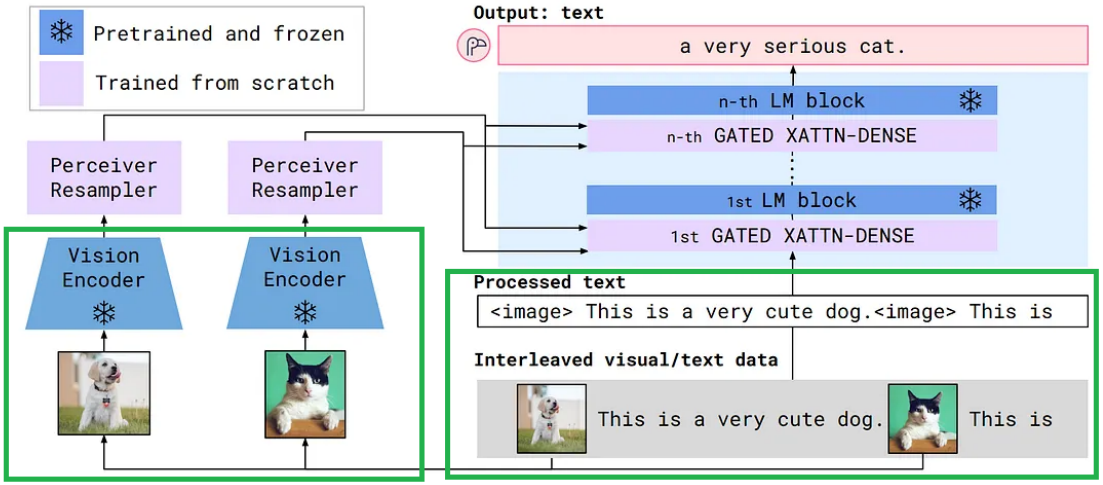
- Flamingo loss function

$$p(y|x) = \prod_{\ell=1}^{L} p(y_\ell | y_{<\ell}, x_{\leq \ell})$$



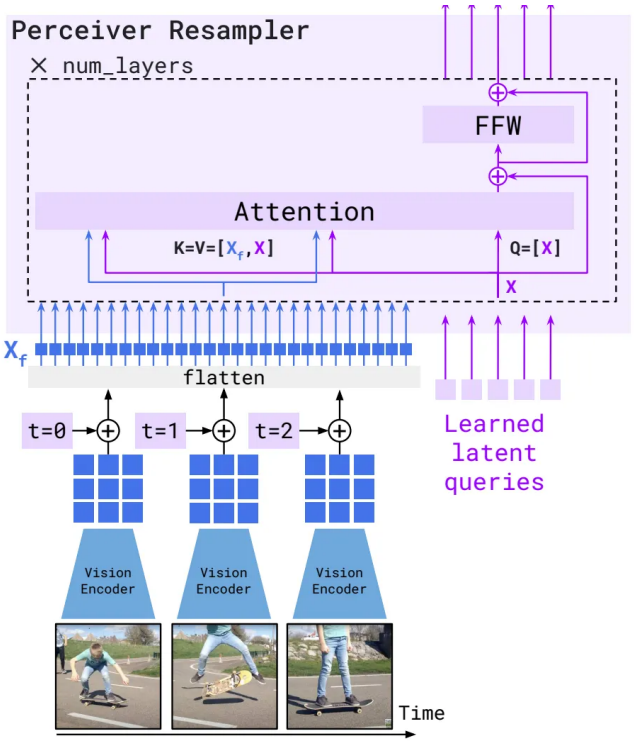
Flamingo

- Flamingo inputs
  - The model takes interleaved visual/text data as input. The images are extracted from the text and replaced with a common token e.g. <image>. This can be then passed into the plain Language Model component. The images are separately passed in through a vision encoder model to convert them into fixed size embeddings.



Flamingo

- **The Perceiver Resampler module**
  - Maps a variable size grid of spatio-temporal visual features output by the Vision Encoder to a fixed number of output tokens (five in the figure), independently from the input image resolution or the number of input video frames.

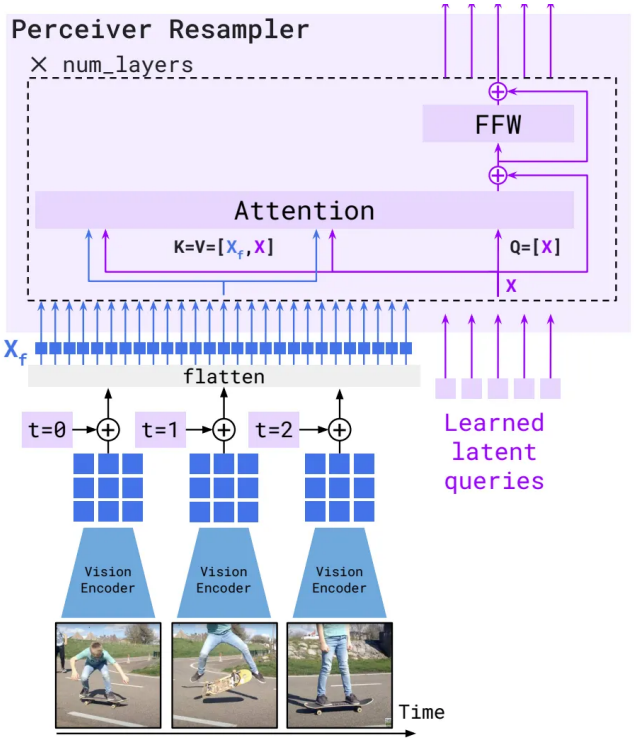



```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # Add the time position embeddings and flatten.
  x_f = x_f + time_embeddings
  x_f = flatten(x_f)  # [T, S, d] -> [T * S, d]
  # Apply the Perceiver Resampler layers.
  for i in range(num_layers):
    # Attention.
    x = x + attention_i(q=x, kv=concat([x_f, x]))
    # Feed forward.
    x = x + ffw_i(x)
  return x
```

Flamingo

- **The Perceiver Resampler module**
  - This transformer has a set of learned latent vectors as queries, and the keys and values are a concatenation of the spatio-temporal visual features with the learned latent vectors.

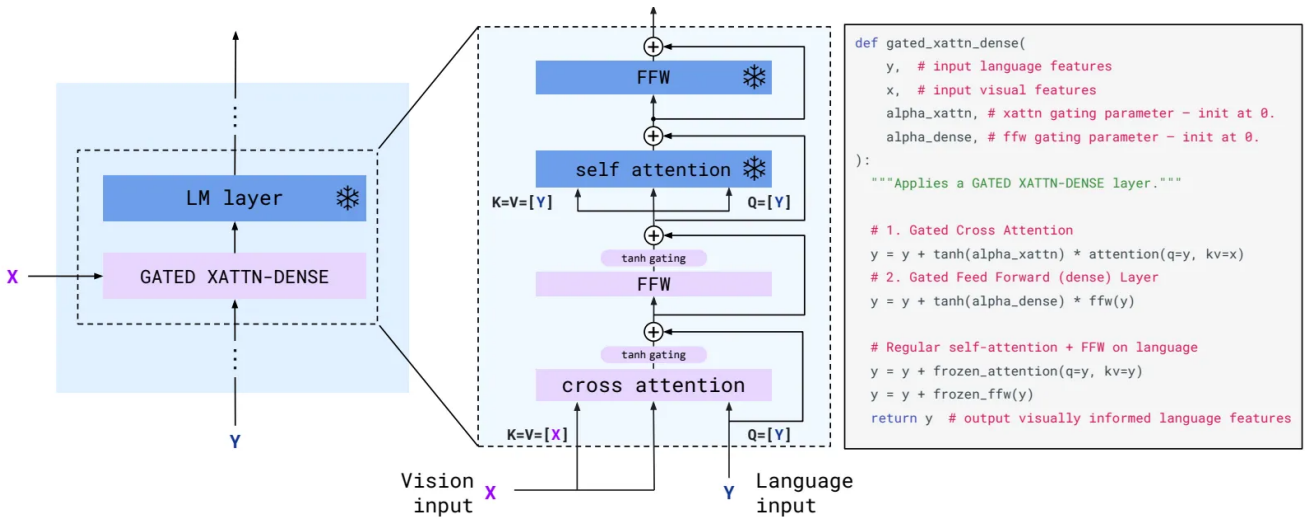



```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # Add the time position embeddings and flatten.
  x_f = x_f + time_embeddings
  x_f = flatten(x_f)  # [T, S, d] -> [T * S, d]
  # Apply the Perceiver Resampler layers.
  for i in range(num_layers):
    # Attention.
    x = x + attention_i(q=x, kv=concat([x_f, x]))
    # Feed forward.
    x = x + ffw_i(x)
  return x
```

Flamingo

- **GATED XATTN-DENSE layers**
  - To condition the LM on visual inputs, we insert new cross-attention layers between existing pretrained and frozen LM layers.




```
def gated_xattn_dense(
    y,  # input language features
    x,  # input visual features
    alpha_xattn, # xattn gating parameter – init at 0.
    alpha_dense, # ffw gating parameter – init at 0.
):
  """Applies a GATED XATTN-DENSE layer."""

  # 1. Gated Cross Attention
  y = y + tanh(alpha_xattn) * attention(q=y, kv=x)
  # 2. Gated Feed Forward (dense) Layer
  y = y + tanh(alpha_dense) * ffw(y)

  # Regular self-attention + FFW on language
  y = y + frozen_attention(q=y, kv=y)
  y = y + frozen_ffw(y)
  return y  # output visually informed language features
```

Flamingo

- **GATED XATTN-DENSE layers**
  - The keys and values in these layers are obtained from the vision features while the queries are derived from the language inputs.
  - They are followed by dense feed-forward layers. These layers are gated so that the LM is kept intact at initialization for improved stability and performance.