Large Language Models

Large Multi-modal Models (LMMs)

M. Soleymani
Sharif University of Technology
Fall 2023

LLMs as a Universal Interface for Multi-modal Systems

- Multimodal systems can provide a more flexible way of interaction
  - typing, talking, or pointing your camera at something can be used throughout the question

- In the previous lecture, the interface had limited interactivity and adaptability to the user's instructions

- How can we utilize the potentials of LLMs for creating a general-purpose assistant?
  - various tasks can be explicitly represented in language

- They must be able to ingest a multimodal prompt containing images and/or videos interleaved with text.

Large Multi-modal Models (LMMs)

- Equip LLMs with eyes to see the world
  - by training them on vision-conditioned language generation tasks
  - and so use LLMs as a general interface for other modalities
  - and thus make use of facts that it has learned during language-only pre-training

- Multimodal In-Context Learning (M-ICL)
  - use demonstrations of some examples to conduct few-shot learning

- Multimodal Instruction-Tuning (M-IT)
  - Finetune the model on instruction-following & use task instructions to repurpose the model

GPT 4: Example

User What is funny about this image? Describe it panel by panel.

Source: https://www.reddit.com/r/hmmm/comments/ubab5v/hmmm/

GPT-4 The image shows a package for a "Lightning Cable" adapter with three panels.

Panel 1: A smartphone with a VGA connector (a large, blue, 15-pin connector typically used for computer monitors) plugged into its charging port.

Panel 2: The package for the "Lightning Cable" adapter with a picture of a VGA connector on it.

Panel 3: A close-up of the VGA connector with a small Lightning connector (used for charging iPhones and other Apple devices) at the end.

The humor in this image comes from the absurdity of plugging a large, outdated VGA connector into a small, modern smartphone charging port.

https://openai.com › gpt-4

(2) Prompt:
Describe the pointed region in the image.

Method	Validataion set								Test set							
	in.		near.		out.		overall		in.		near.		out.		overall	
	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S
OSCAR	85.4	11.9	84.0	11.7	80.3	10.0	83.4	11.4	84.8	12.1	82.1	11.5	73.8	9.7	80.9	11.3
Human	84.4	14.3	85.0	14.3	95.7	14.0	87.1	14.2	80.6	15.0	84.6	14.7	91.6	14.2	85.3	14.6
VIVO	92.2	12.9	87.8	12.6	87.5	11.5	88.3	12.4	89.0	12.9	87.8	12.6	80.1	11.1	86.6	12.4
VinVL	103.7	13.7	95.6	13.4	83.8	11.9	94.3	13.1	98.0	13.6	95.2	13.4	78.0	11.5	92.5	13.1
UFO	103.9	14.5	95.5	13.8	83.5	12.3	94.3	13.6	98.9	14.3	94.7	13.9	77.9	12.1	92.3	13.6
mPLUG	-	-	-	-	-	-	114.8	14.8	-	-	-	-	-	-	-	-
SimVLM	113.7	-	110.9	-	115.2	-	115.2	-	113.7	-	110.9	-	115.2	-	115.2	-
LEMON	118.0	15.4	116.3	15.1	120.2	14.5	117.3	15.0	112.8	15.2	115.5	15.1	110.1	13.7	114.3	14.9
UniversalCap	123.2	15.0	121.5	15.3	123.4	14.4	122.1	15.0	118.9	15.4	120.6	15.3	114.3	14.1	119.3	15.1
CoCa	-	-	-	-	-	-	122.4	15.5	-	-	-	-	-	-	120.6	15.5
GIT_B	100.7	13.8	97.7	13.5	89.6	12.5	96.6	13.4	-	-	-	-	-	-	-	-
GIT_L	107.7	14.9	107.8	14.5	102.5	13.7	106.9	14.4	-	-	-	-	-	-	-	-
GIT	**129.8**	**16.3**	124.1	16.0	127.1	15.7	125.5	16.0	122.4	16.2	123.9	16.0	122.0	**15.7**	123.4	15.9
GIT2	126.9	16.1	**125.8**	**16.2**	**130.6**	**15.8**	**126.9**	**16.1**	**124.2**	**16.4**	**125.5**	**16.1**	**122.3**	15.6	**124.8**	**16.1**

GPT-4V:
The highlighted red region in the image is a table cell that contains a numerical value of 122.3. It is located in the "Test set" section of the table, under the "out" column and the "GIT2" row. This value represents the overall performance metric for the GIT2 method on the test set, in the "out" category.

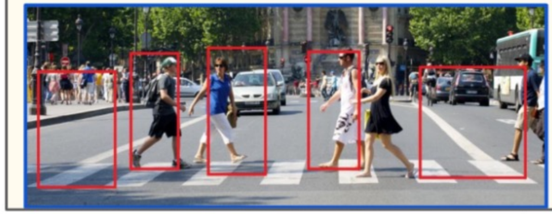
Prompt:
Localize each person in the image using bounding box. What is the image size of the input image?

GPT-4V:
The image size of the input image is 1000 x 333 pixels.

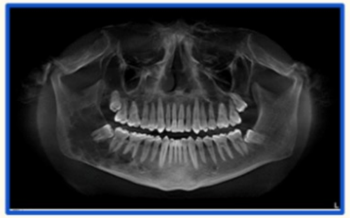
Bounding boxes for each person:
- Person 1 (leftmost): (x1, y1) = (22, 108), (x2, y2) = (170, 300)
- Person 2: (x1, y1) = (200, 80), (x2, y2) = (290, 270)
- Person 3: (x1, y1) = (350, 70), (x2, y2) = (460, 280)
- Person 4: (x1, y1) = (540, 80), (x2, y2) = (640, 270)
- Person 5 (rightmost): (x1, y1) = (760, 100), (x2, y2) = (920, 290)

Note: These bounding box coordinates are approximate and may not be exact.

Visualization of Detected Bbox:



Prompt:
Which wisdom tooth is hardest to remove?



Note: Same query image as shown above, resized to save space

GPT-4V:
The lower wisdom teeth, especially those that are impacted, are typically considered the hardest to remove. This is because the lower jawbone is denser than the upper jawbone, and the impacted wisdom teeth may be more difficult to access. Additionally, the roots of the lower wisdom teeth are often longer and more curved than those of the upper wisdom teeth, making them more challenging to remove.

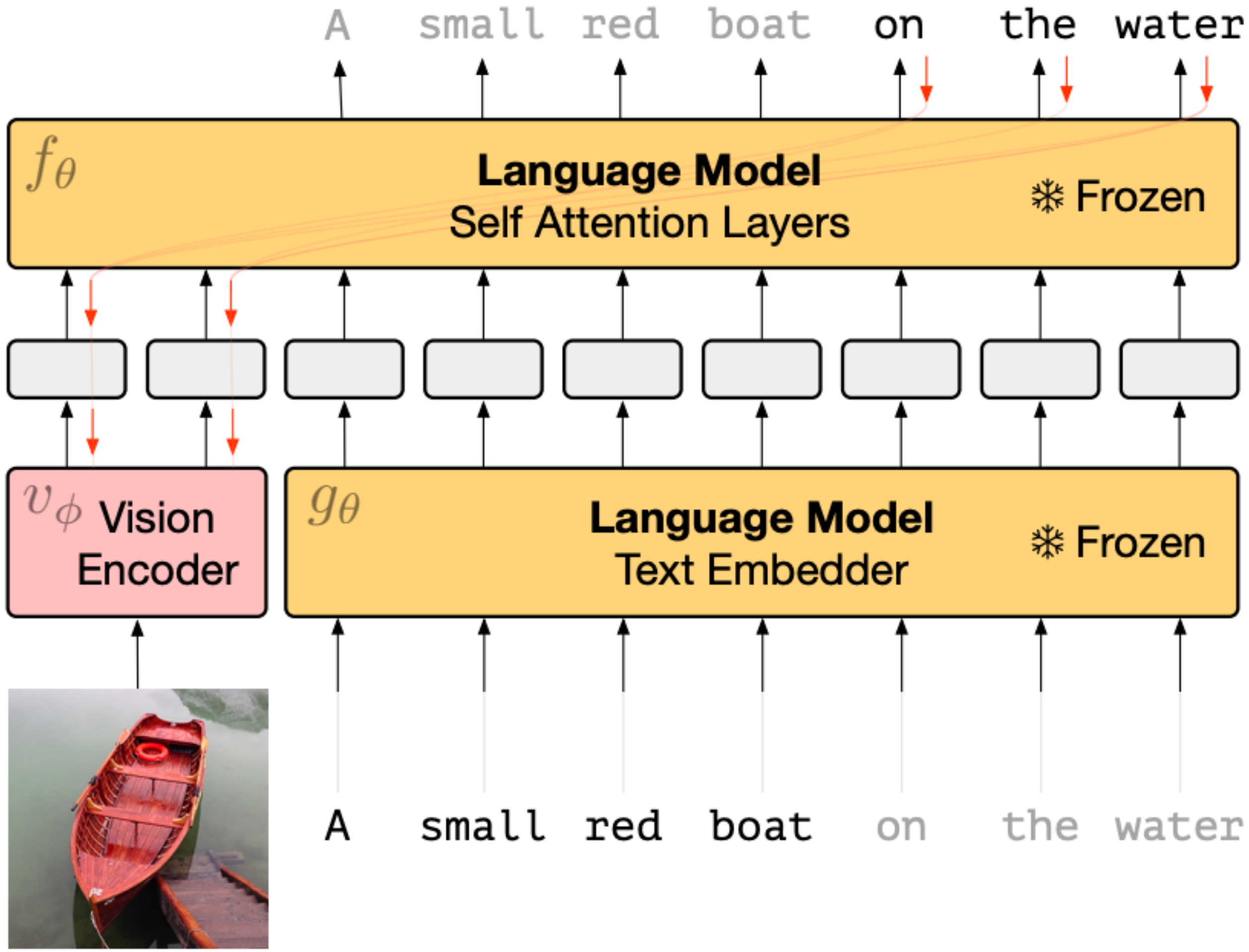
Prompt:
Count the number of apples in the image.

GPT-4V:
There are 7 apples in the image.

https://huyenchip.com/2023/10/10/multimodal.html

Frozen LM Prefix

NF-ResNet-50



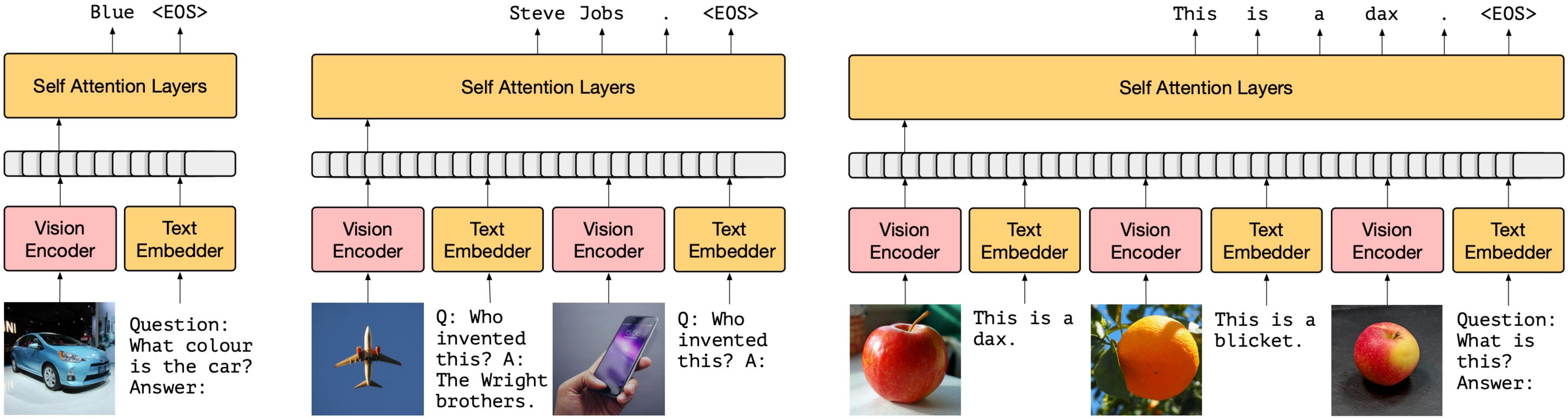
Tsimpoukelli et al., "Multimodal Few-Shot Learning with Frozen Language Models", ICLR 2022

Frozen LM Prefix

- Frozen: Visual modalities are incorporated as input of LLMs without needing to update their weights
- Concatenated textual and visual embeddings are fed to the decoder of the LLM, which generates a textual output autoregressively
- Finetunes an image encoder whose outputs are directly used as soft prompts for the LLM.

Tsimpoukelli et al., “Multimodal Few-Shot Learning with Frozen Language Models”, ICLR 2022

Frozen: Multimodal few shot learning



(a) **0-shot VQA** (b) **1-shot outside-knowledge VQA** (c) **Few-shot image classification**

Tsimpoukelli et al., "Multimodal Few-Shot Learning with Frozen Language Models", ICLR 2022

Flamingo

- Flamingo as the GPT-3 moment of multimodal models domain
  - due to its strong performance on zero-shot task transfer and in-context-learning.

- It bridges powerful pretrained vision-only and language-only models by novel architecture components

- The same can be done for image and video understanding tasks such as classification, captioning, or question-answering

Alayrac et al., “Flamingo: a Visual Language Model for Few-Shot Learning”, NeurIPS 2022

Input Prompt

Completion

This is a chinchilla. They are mainly found in Chile.

This is a shiba. They are very popular in Japan.

This is

→ **a flamingo. They are found in the Caribbean and South America.**

What is the title of this painting? Answer: The Hallucinogenic Toreador.

Where is this painting displayed? Answer: Louvres Museum, Paris.

What is the name of the city where this was painted? Answer:

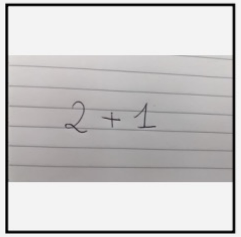
→ **Arles.**

Output: "Underground"

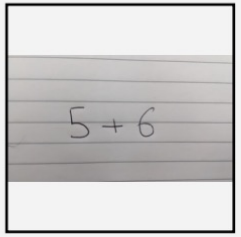
Output: "Congress"

Output:

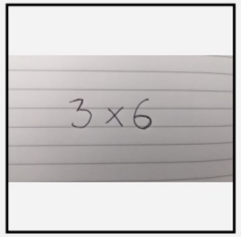
→ **"Soulomes"**



2+1=3



5+6=11



→ **3x6=18**

Output: A propaganda poster depicting a cat dressed as French emperor Napoleon holding a piece of cheese.

Output: A pink room with a flamingo pool float.

Output:

→ **A portrait of Salvador Dali with a robot head.**

This is a picture of two teddy bears on the moon.

What are they doing?

They are having a conversation.

What object are they using?

It looks like a computer.

Is this surprising?

Yes, it is surprising.

Why is this picture surprising to you?

I think it is surprising because teddy bears are not usually found on the moon.

What is the common thing about these three images?

They are all flamingos.

What is the difference between these three images?

The first one is a cartoon, the second one is a real flamingo, and the third one is a 3D model of a flamingo.

This is an apple with a sticker on it.

What does the sticker say?

The sticker says "iPod".

Where is the photo taken?

It looks like it's taken in a backyard.

Do you think it is printed or handwritten?

It looks like it's handwritten.

What color is the sticker?

It's white.

This is a cityscape. It looks like Chicago.

What makes you think this is Chicago?

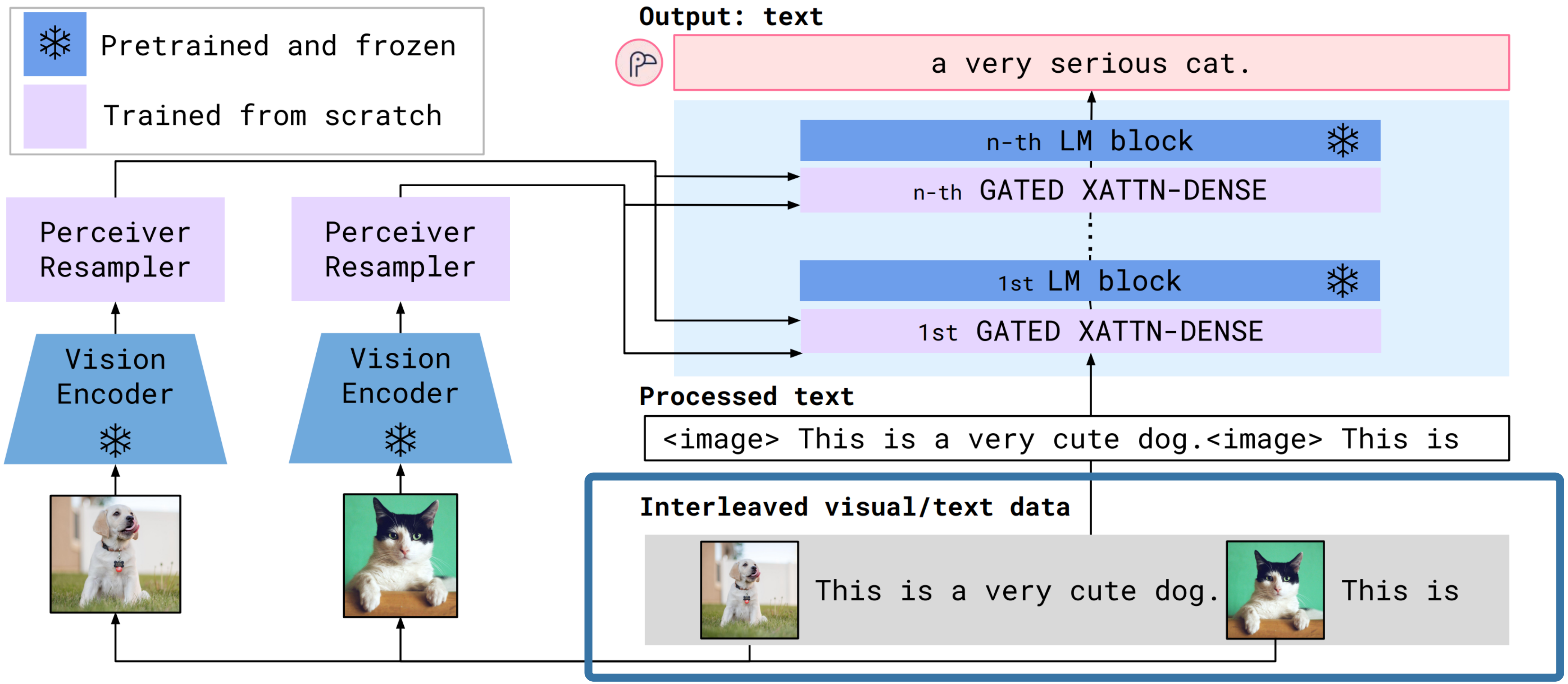
I think it's Chicago because of the Shedd Aquarium in the background.

What about this one? Which city is this and what famous landmark helped you recognise the city?

This is Tokyo. I think it's Tokyo because of the Tokyo Tower.

Flamingo Overview

Separately trained image + language models, with novel layers in between



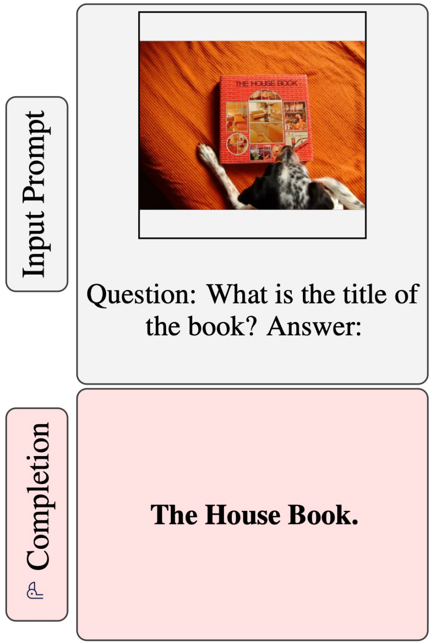
A vision model which can "perceive" visual scenes and an LLM which performs a basic form of reasoning.

Input/Output

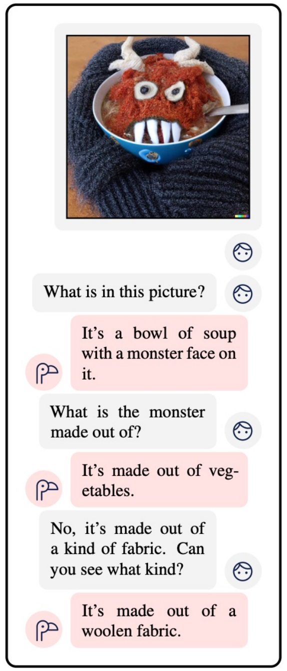
Interleaved inputs: text/images/video

Outputs: free-form text

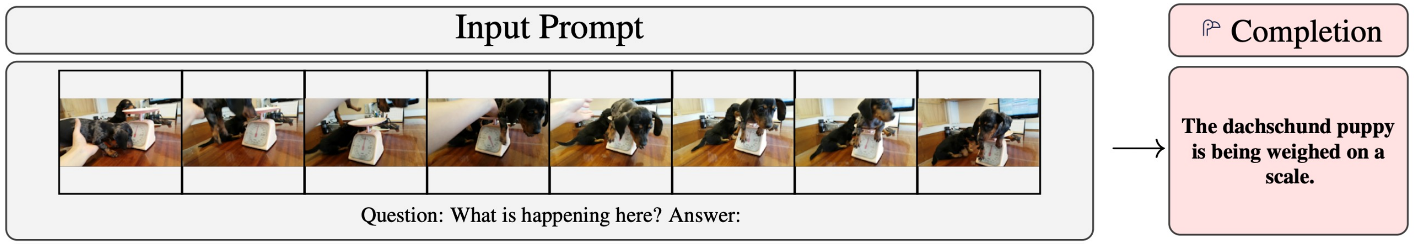
Selected single image samples



Selected dialogue samples



Selected video samples

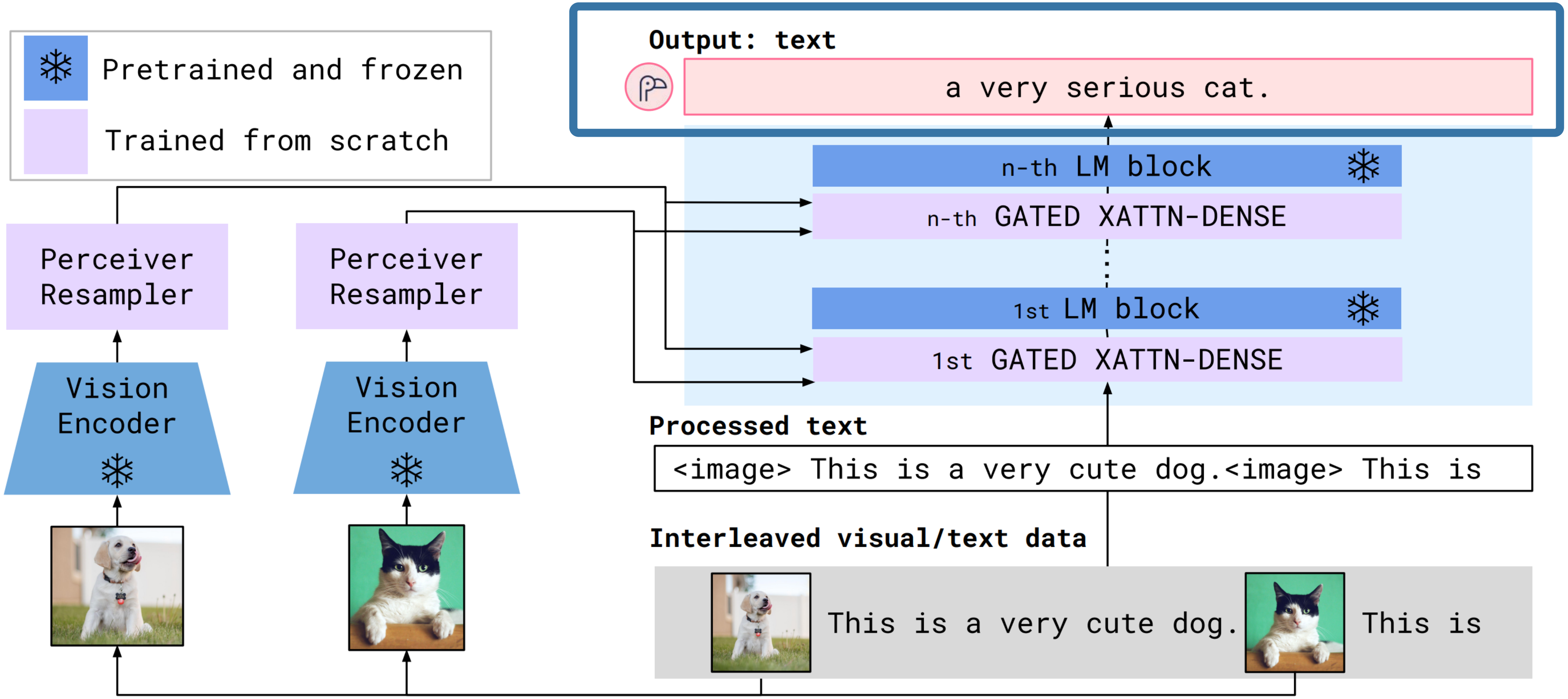


Handles sequences of arbitrarily interleaved visual and textual data.

Due to its flexibility, can be trained on large-scale multimodal web corpora containing arbitrarily interleaved text and images (key to endow them with in-context few-shot learning capabilities)

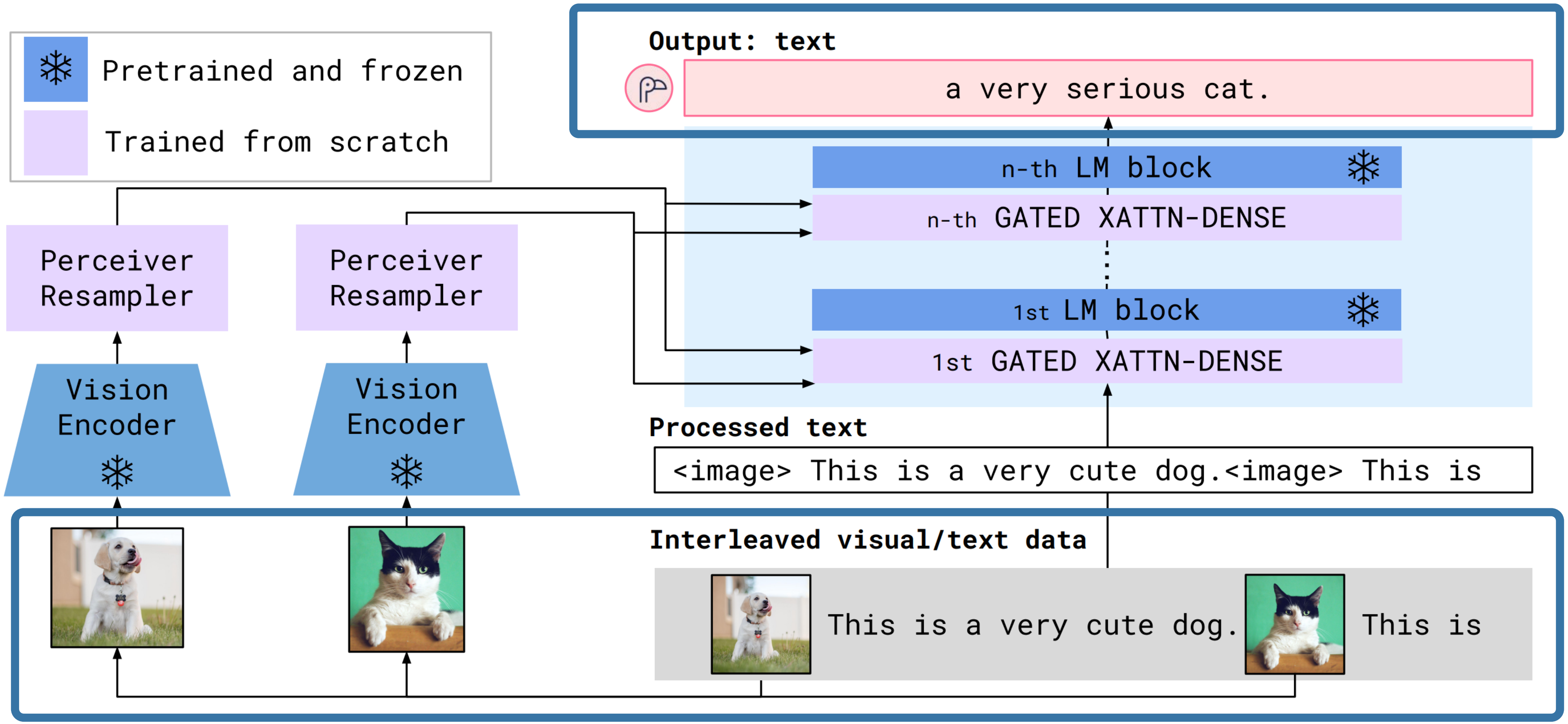
Flamingo Overview

Separately trained image + language models, with novel layers in between



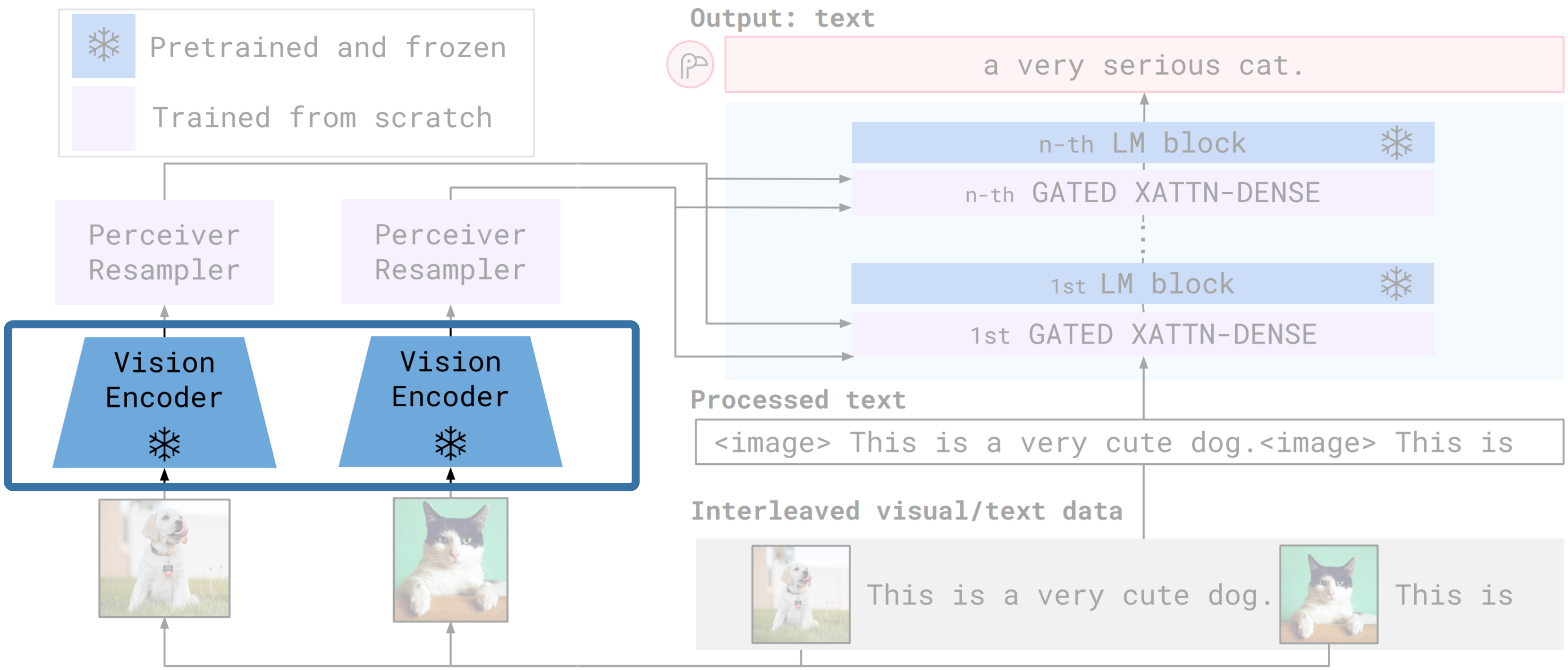
Flamingo Overview

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



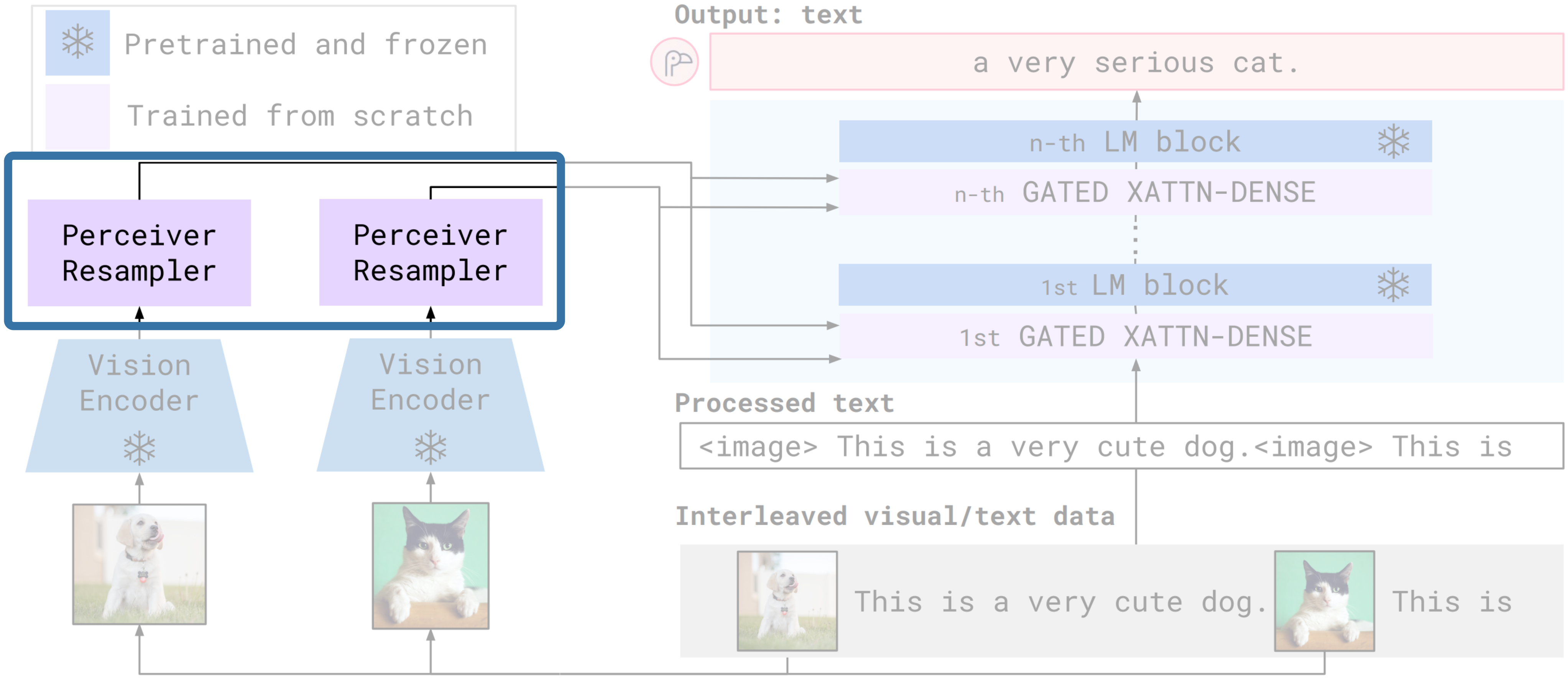
Vision Encoder

Pretrained and frozen Normalizer Free ResNet (NFNet)

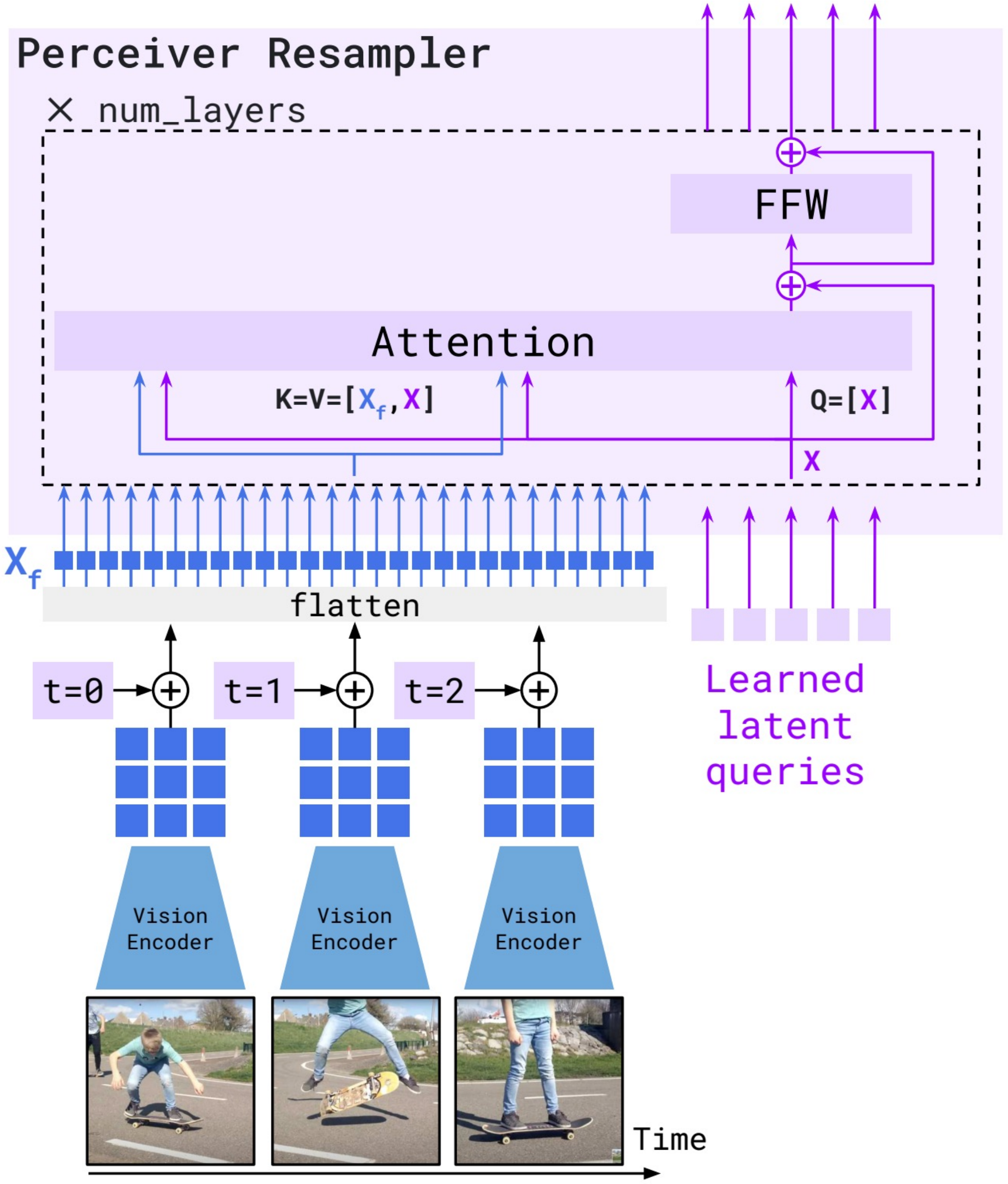


Brock et al. "High-performance large-scale image recognition without normalization", ICML 2021

Perceiver Resampler

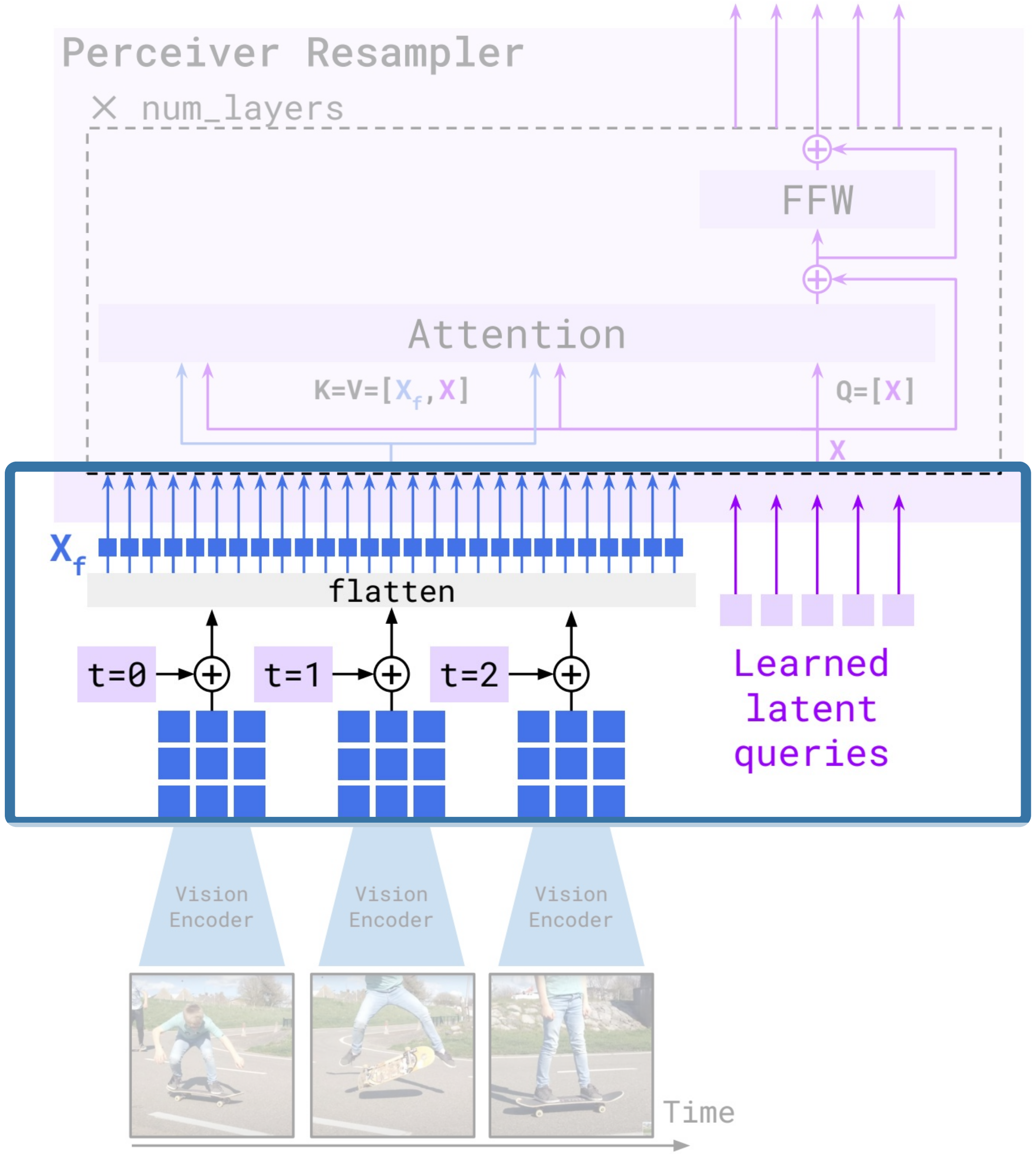


Perceiver Resampler



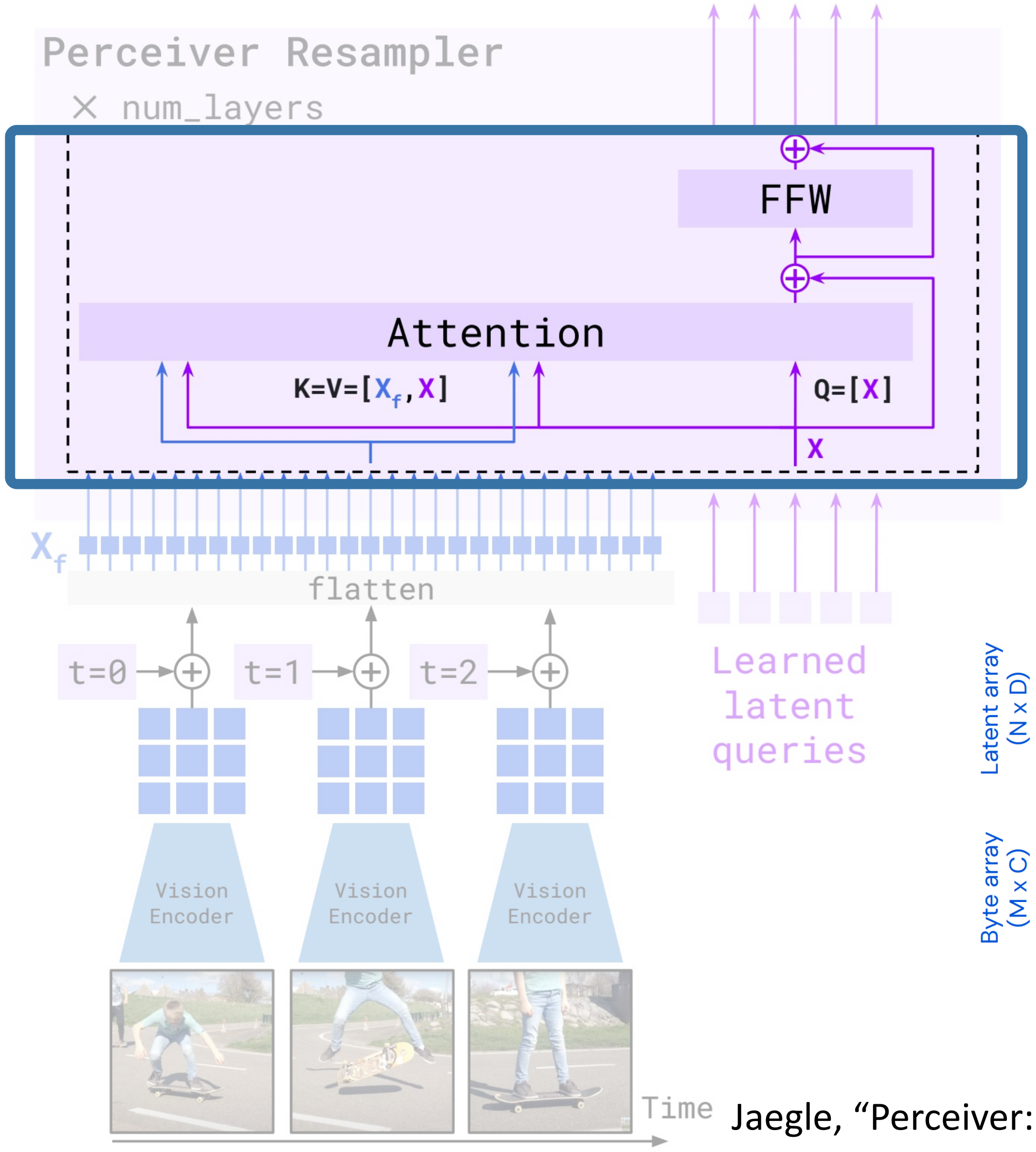

```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # 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
```

Perceiver Resampler

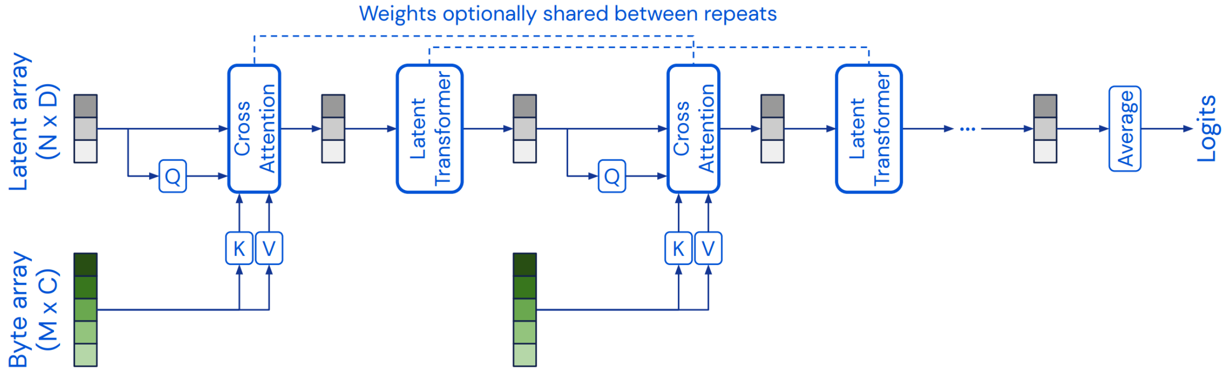



```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # 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
```

Perceiver Resampler

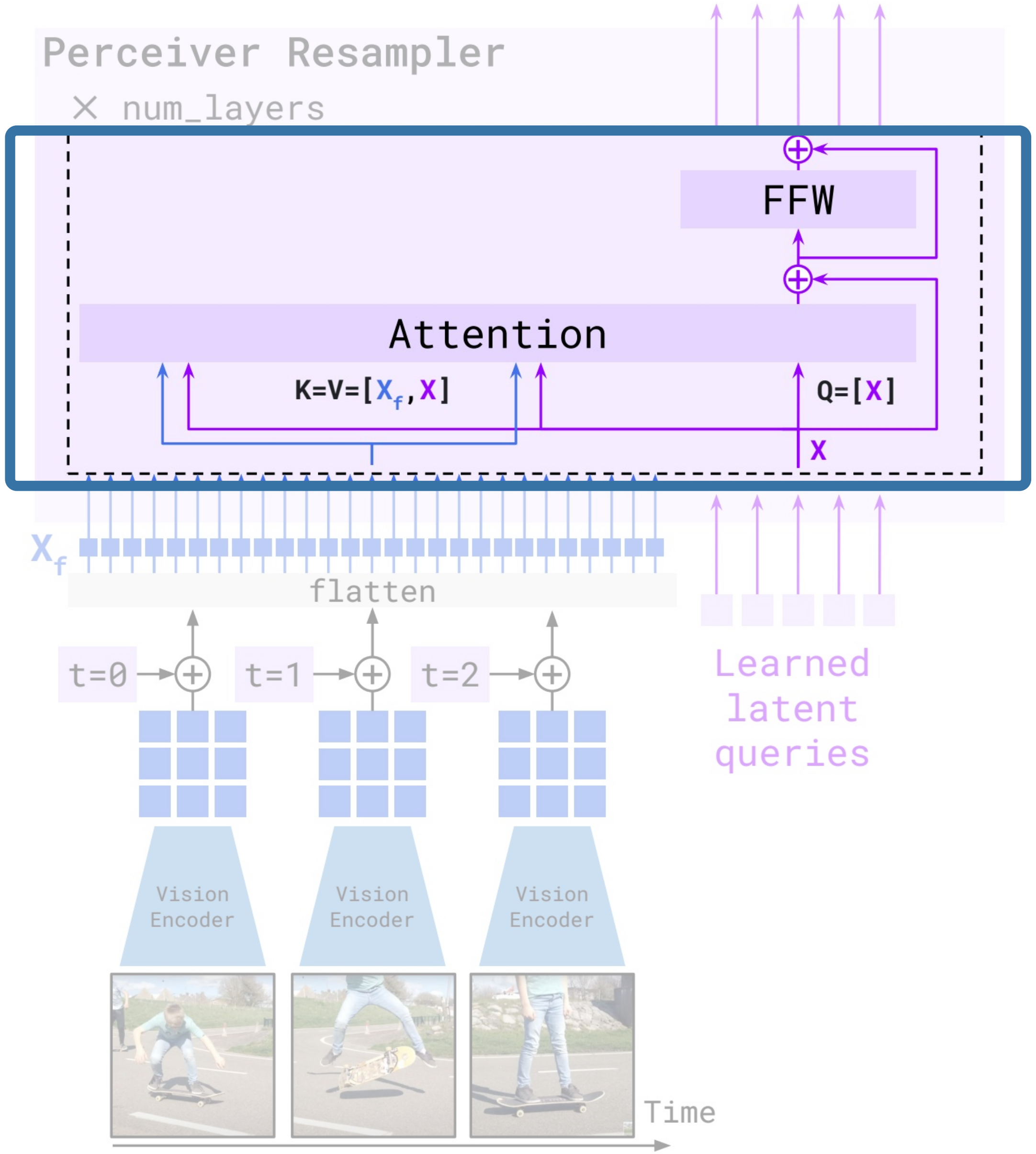



```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # Add the time position embeddings and flatten.
  x_f = x_f + time_embeddings
  x_f = flatten(x_f)  # [T, S, d] -> [T * S, d]
```



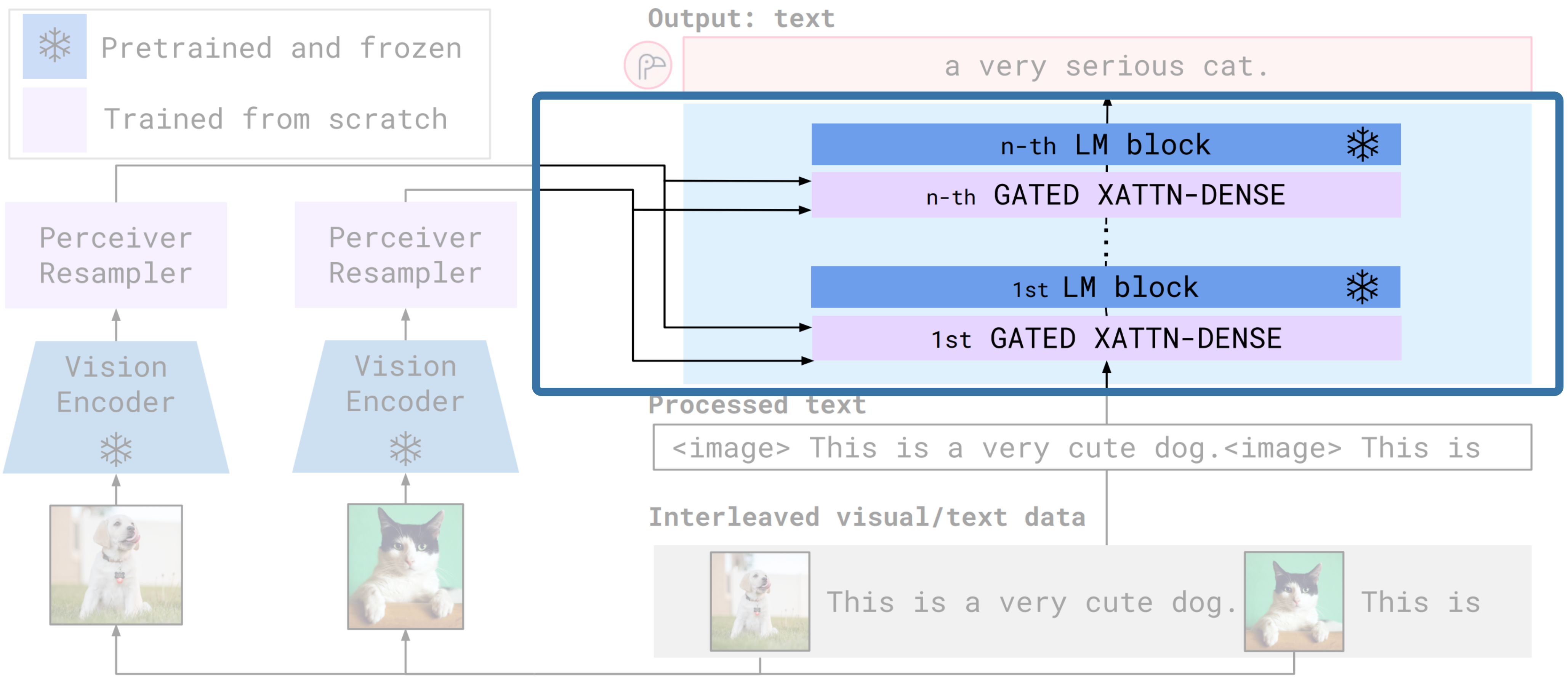
Jaegle, “Perceiver: General Perception with Iterative Attention”, ICML 2021

Perceiver Resampler



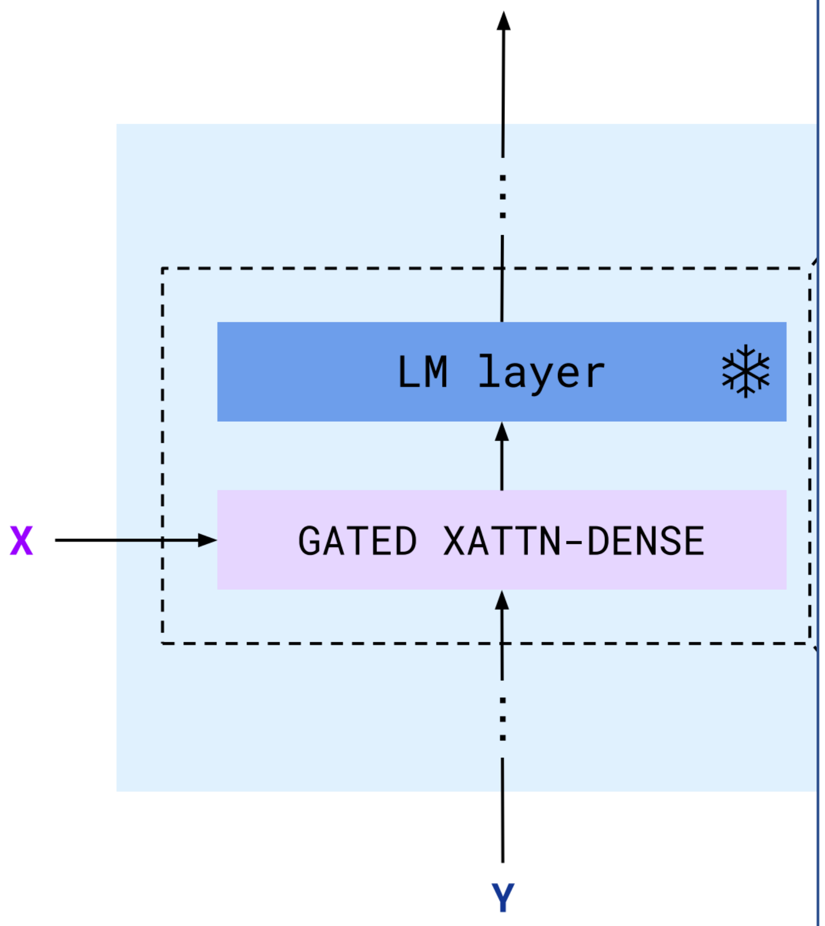

```
def perceiver_resampler(
    x_f,  # The [T, S, d] visual features (T=time, S=space)
    time_embeddings,  # The [T, 1, d] time pos embeddings.
    x,  # R learned latents of shape [R, d]
    num_layers,  # Number of layers
):
  """The Perceiver Resampler model."""

  # 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
```

Conditioning the Language Model

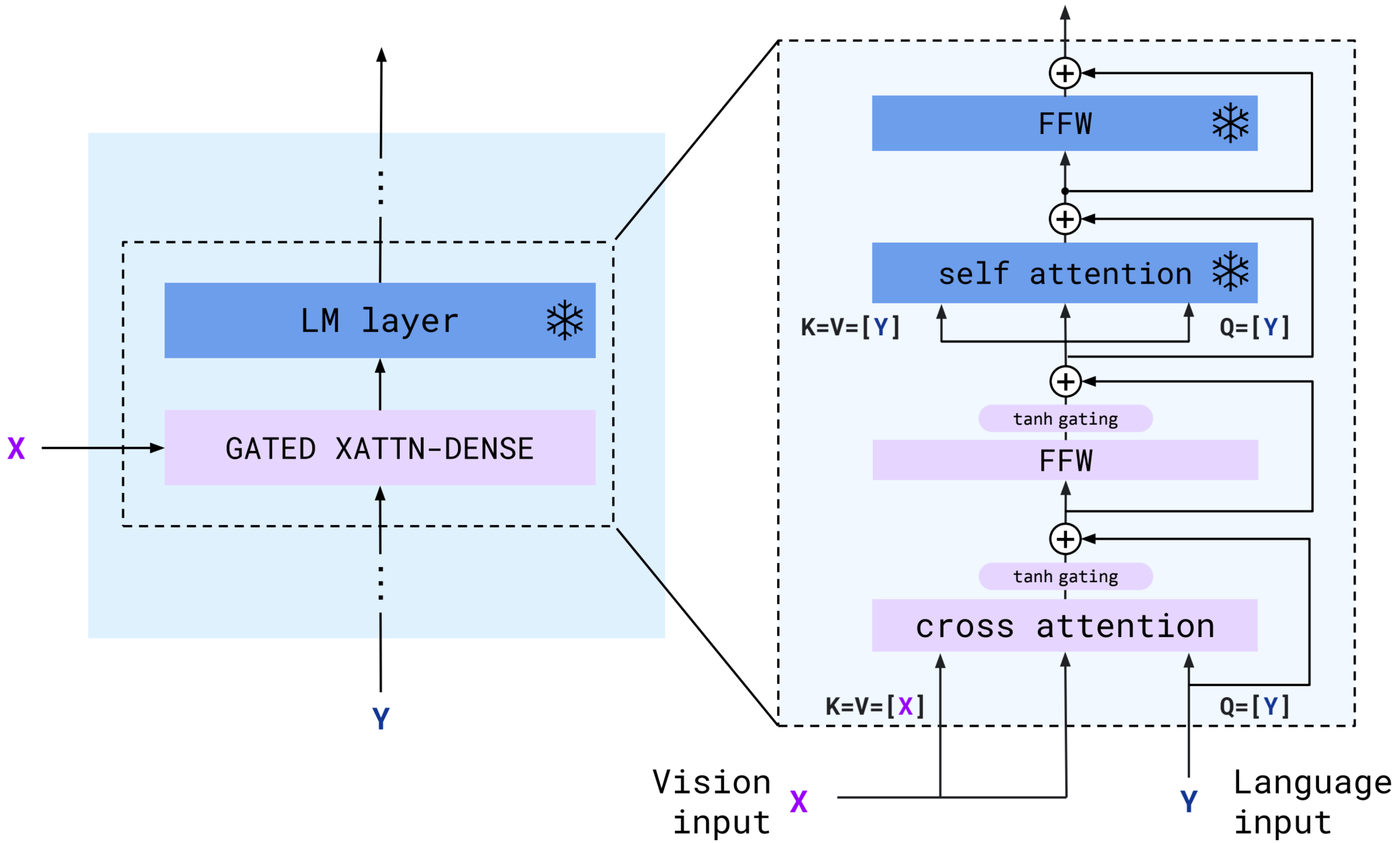


Brock et al. "High-performance large-scale image recognition without normalization", ICML 2021

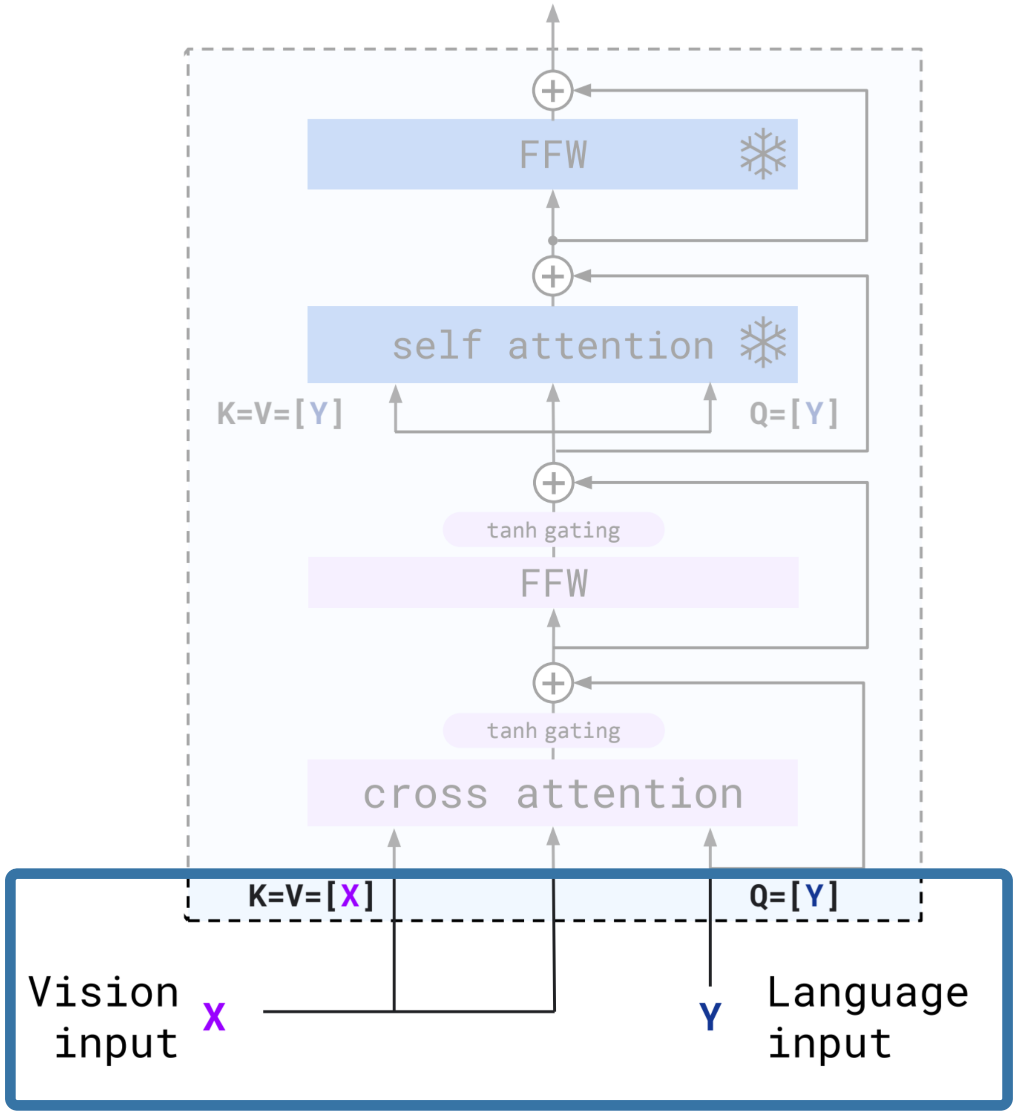
Gated XATTN-Dense layers



Gated XATTN-Dense layers

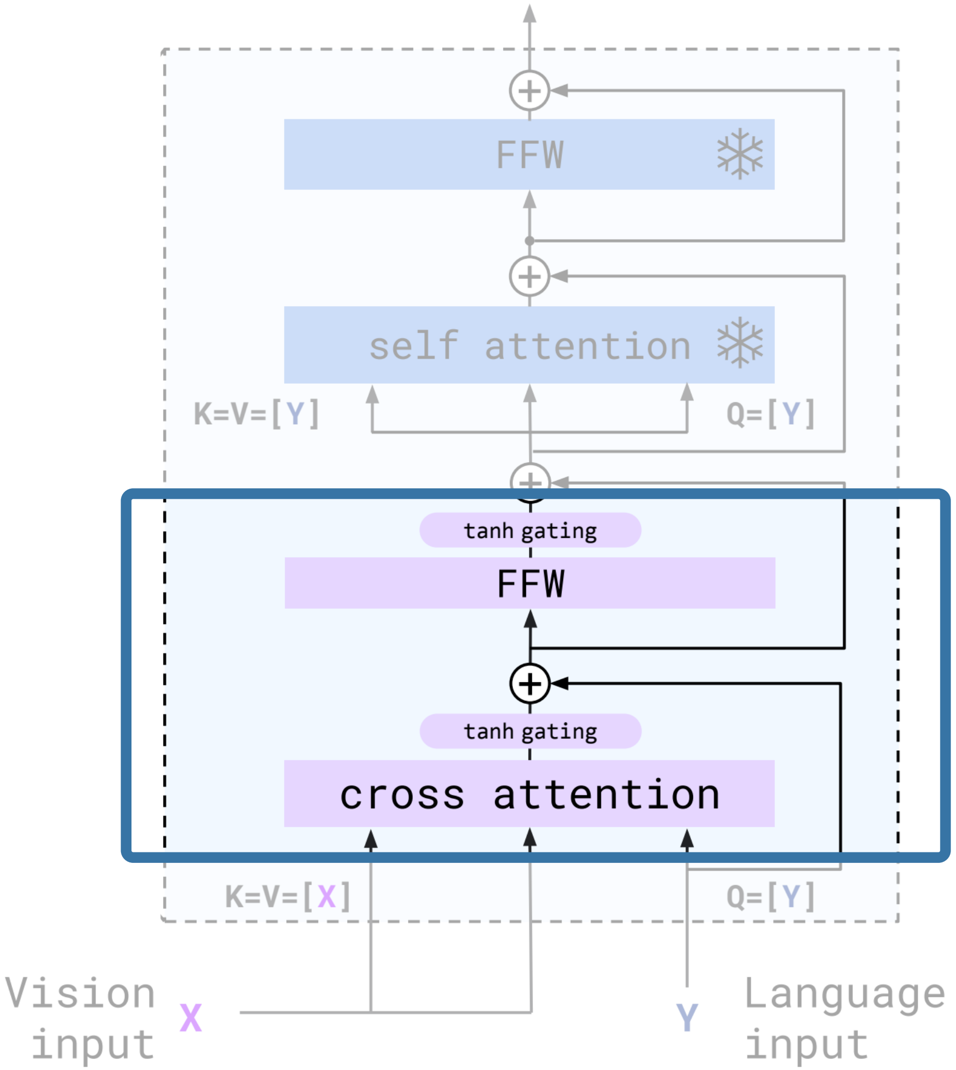


Gated XATTN-Dense layers



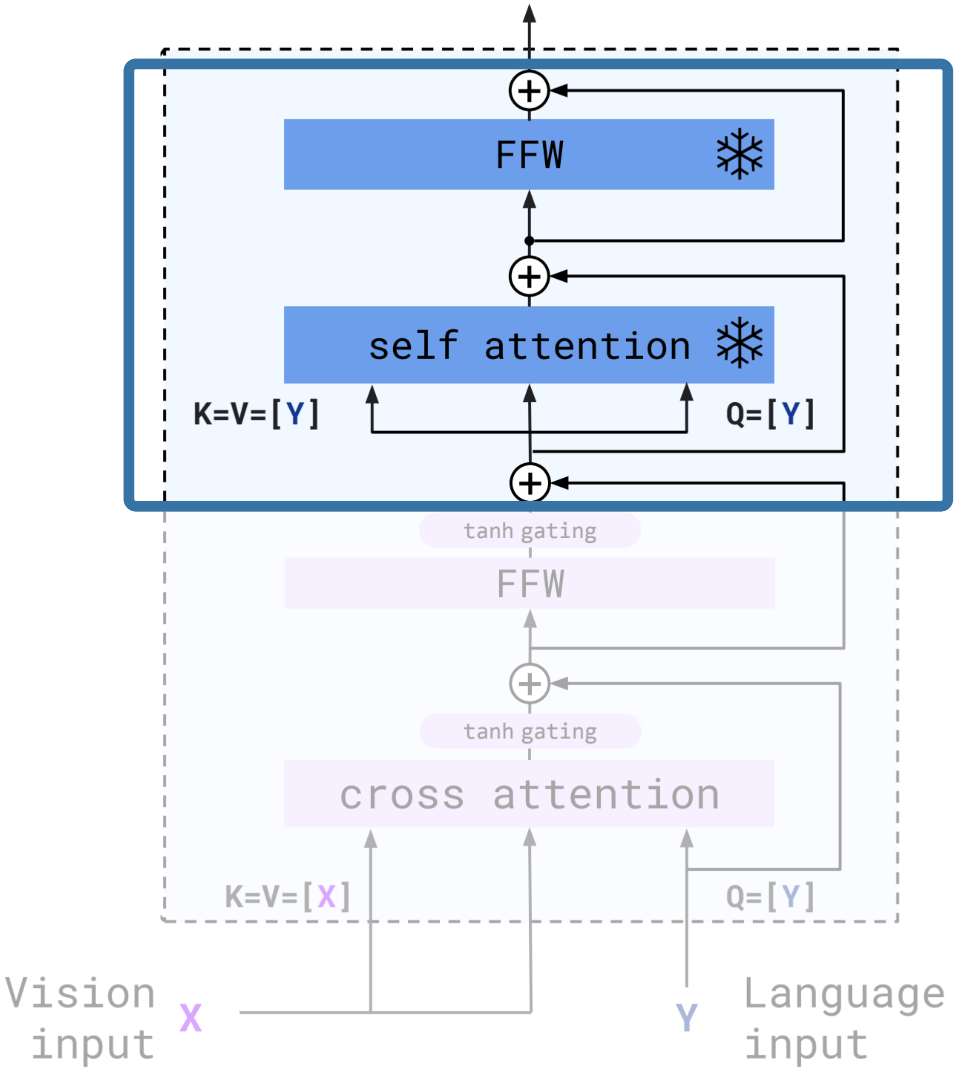

```
def gated_xattn_dense(
    y,  # input language features
    x,  # input visual features
    alpha_xattn, # xattn gating parameter – init at 0.
    alpha_dense, # ffw gating parameter – init at 0.
):
  """Applies a GATED XATTN-DENSE layer."""

  # 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
```

Gated XATTN-Dense layers



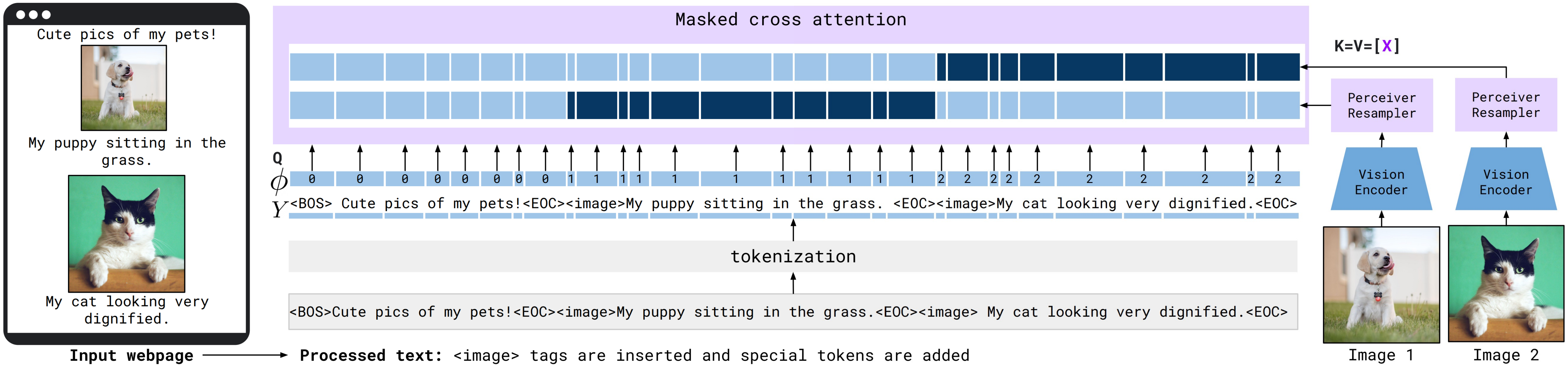

```
def gated_xattn_dense(
    y,  # input language features
    x,  # input visual features
    alpha_xattn, # xattn gating parameter – init at 0.
    alpha_dense, # ffw gating parameter – init at 0.
):
  """Applies a GATED XATTN-DENSE layer."""

  # 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
```

Gated XATTN-Dense layers

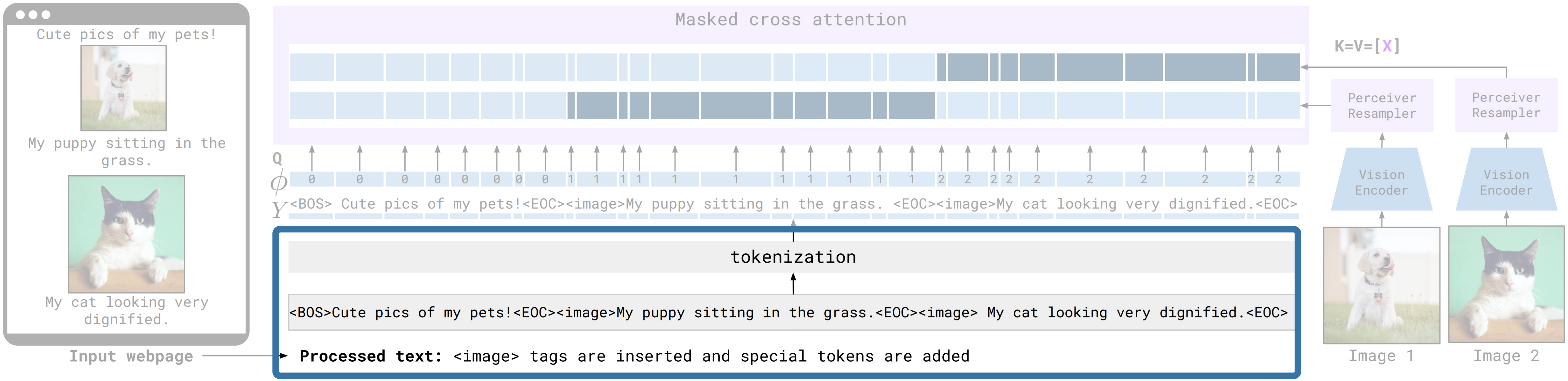



```
def gated_xattn_dense(
    y,  # input language features
    x,  # input visual features
    alpha_xattn, # xattn gating parameter – init at 0.
    alpha_dense, # ffw gating parameter – init at 0.
):
  """Applies a GATED XATTN-DENSE layer."""

  # 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
```

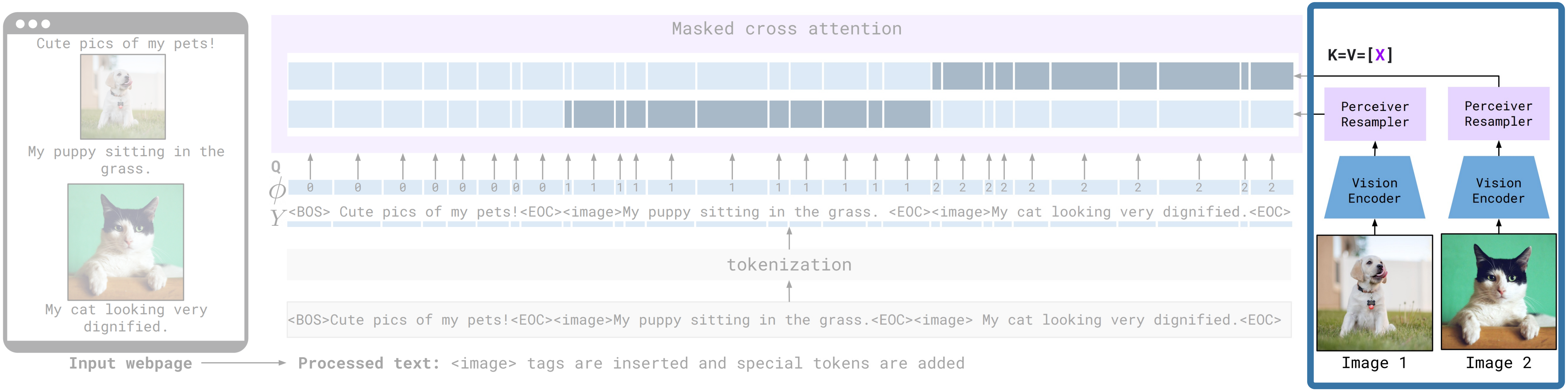
Multi-Visual Input Support



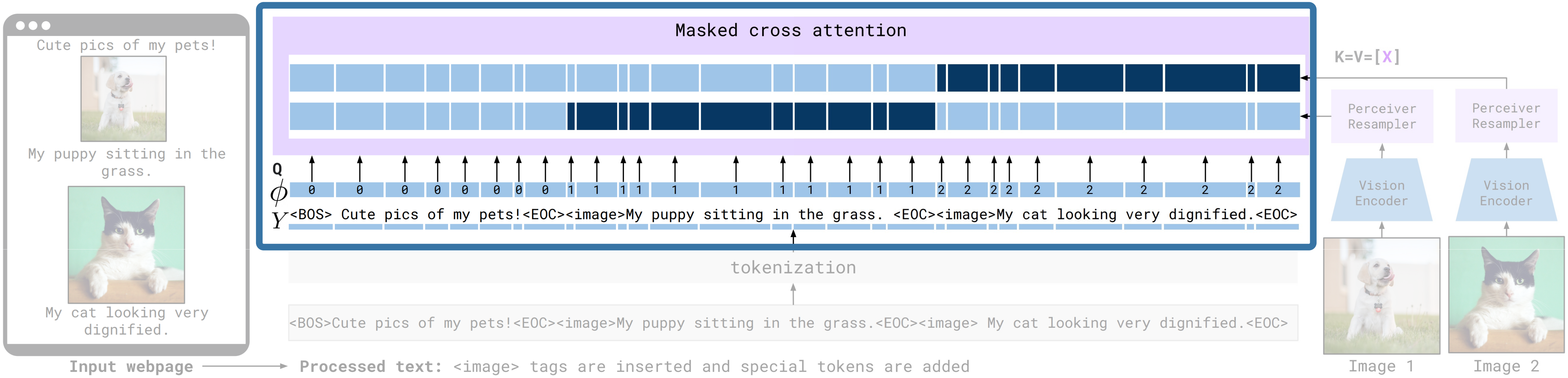
Multi-Visual Input Support



Multi-Visual Input Support



Multi-Visual Input Support



Training Data: Mixture of Datasets

Image-Text Pairs dataset
[N=1, T=1, H, W, C]

Video-Text Pairs dataset
[N=1, T>1, H, W, C]

Multi-Modal Massive Web (M3W) dataset
[N>1, T=1, H, W, C]

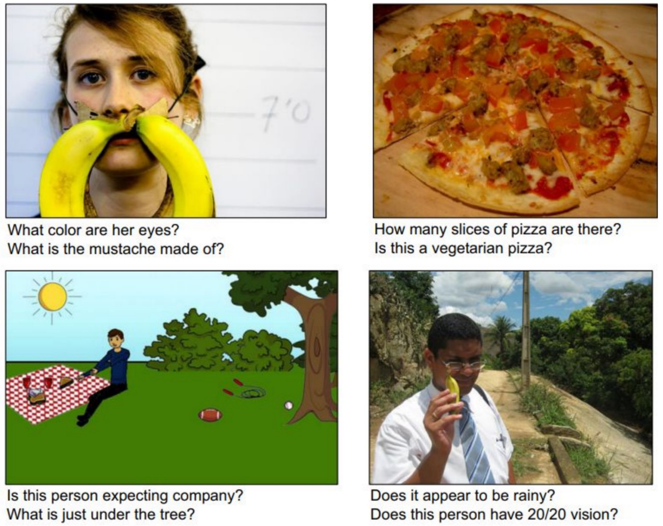
- N: Number of visual inputs for a single example
- T: Number of video frames
- H, W, C: height, width, color channels

Benchmark Tasks

Image Net-1k

Source: https://link.springer.com/article/10.1007/s11263-015-0816-y

VQA



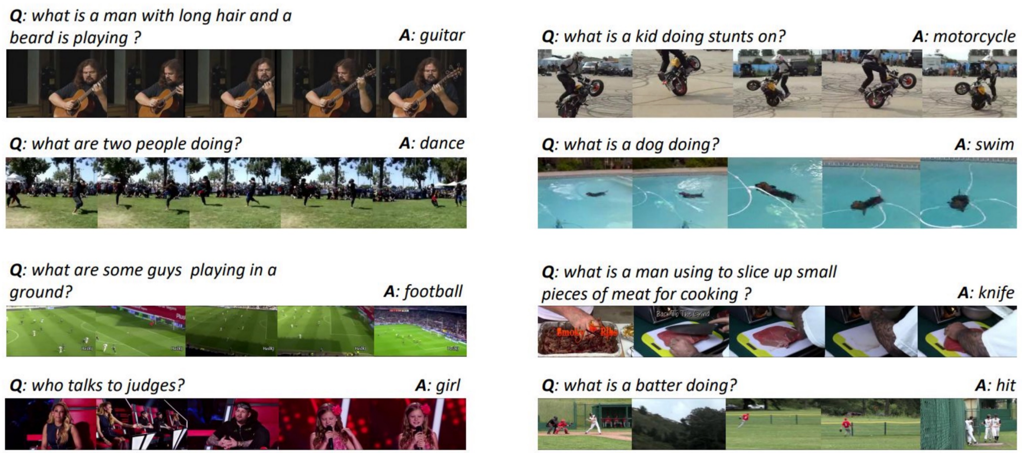
Source: https://link.springer.com/content/pdf/10.1007/s11263-016-0966-6.pdf

	Dataset	DEV	Gen.	Custom prompt	Task description
Image	ImageNet-1k [94]	✓			Object classification
	MS-COCO [15]	✓	✓		Scene description
	VQAv2 [3]	✓	✓		Scene understanding QA
	OKVQA [69]	✓	✓		External knowledge QA
	Flickr30k [139]		✓		Scene description
	VizWiz [35]		✓		Scene understanding QA
	TextVQA [100]		✓		Text reading QA
	VisDial [20]				Visual Dialogue
	HatefulMemes [54]			✓	Meme classification

Benchmark Tasks

Kinetics700 2020: Taken from YouTube videos

MSVDQA



Source: https://arxiv.org/pdf/2210.10864.pdf

	Dataset	DEV	Gen.	Custom prompt	Task description
Image	ImageNet-1k [94]	✓			Object classification
	MS-COCO [15]	✓	✓		Scene description
	VQAv2 [3]	✓	✓		Scene understanding QA
	OKVQA [69]	✓	✓		External knowledge QA
	Flickr30k [139]		✓		Scene description
	VizWiz [35]		✓		Scene understanding QA
	TextVQA [100]		✓		Text reading QA
	VisDial [20]				Visual Dialogue
	HatefulMemes [54]			✓	Meme classification
Video	Kinetics700 2020 [102]	✓			Action classification
	VATEX [122]	✓	✓		Event description
	MSVDQA [130]	✓	✓		Event understanding QA
	YouCook2 [149]		✓		Event description
	MSRVTTQA [130]		✓		Event understanding QA
	iVQA [135]		✓		Event understanding QA
	RareAct [73]			✓	Composite action retrieval
	NextQA [129]		✓		Temporal/Causal QA
	STAR [128]				Multiple-choice QA

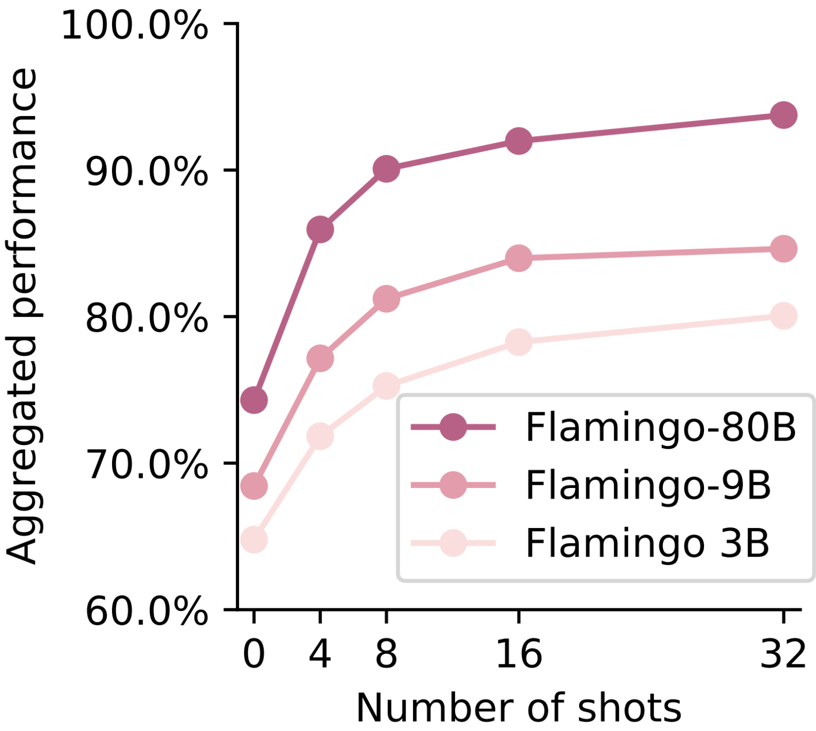
Classification Task Results

Model	Method	Prompt size	shots/class	ImageNet top 1	Kinetics700 avg top1/5
SotA	Fine-tuned	-	full	90.9 [127]	89.0 [134]
SotA	Contrastive	-	0	**85.7 [82]**	**69.6 [85]**
NFNetF6	Our contrastive	-	0	77.9	62.9
Flamingo-3B	RICES	8	1	70.9	55.9
		16	1	71.0	56.9
		16	5	72.7	58.3
Flamingo-9B	RICES	8	1	71.2	58.0
		16	1	71.7	59.4
		16	5	75.2	60.9
Flamingo-80B	Random	16	≤ 0.02	66.4	51.2
	RICES	8	1	71.9	60.4
		16	1	71.7	62.7
		16	5	76.0	63.5
	RICES+ensembling	16	5	77.3	64.2

Fine Tuning Results

Method	VQAV2 test-dev	VQAV2 test-std	COCO test	VATEX test	VizWiz test-dev	VizWiz test-std	MSRVTTQA test	VisDial valid	VisDial test-std	YouCook2 valid	TextVQA valid	TextVQA test-std	HatefulMemes test seen
Flamingo - 32 shots	67.6	-	113.8	65.1	49.8	-	31.0	56.8	-	86.8	36.0	-	70.0
SimVLM [124]	80.0	80.3	**143.3**	-	-	-	-	-	-	-	-	-	-
OFA [119]	79.9	80.0	149.6	-	-	-	-	-	-	-	-	-	-
Florence [140]	80.2	80.4	-	-	-	-	-	-	-	-	-	-	-
Flamingo Fine-tuned	**82.0**	**82.1**	138.1	**84.2**	**65.7**	**65.4**	**47.4**	61.8	59.7	118.6	**57.1**	54.1	**86.6**
Restricted SotA†	80.2 [140]	80.4 [140]	**143.3** [124]	76.3 [153]	-	-	46.8 [51]	**75.2** [79]	**74.5** [79]	**138.7** [132]	54.7 [137]	**73.7** [84]	79.1 [62]
Unrestricted SotA	81.3 [133]	81.3 [133]	149.6 [119]	81.4 [153]	57.2 [65]	60.6 [65]	-	-	75.4 [123]	-	-	-	84.6 [152]

Model Scaling & Number of Shots



	Requires model sharding	Frozen		Trainable		Total
		Language	Vision	GATED XATTN-DENSE	Resampler	count
Flamingo-3B	✗	1.4B	435M	1.2B (every)	194M	**3.2B**
Flamingo-9B	✗	7.1B	435M	1.6B (every 4th)	194M	**9.3B**
Flamingo	✓	70B	435M	10B (every 7th)	194M	**80B**

Ablation Studies

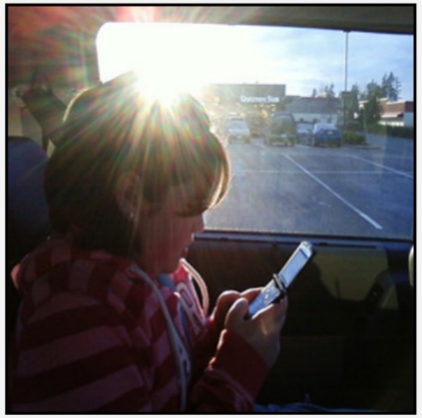
	Ablated setting	*Flamingo*-3B original value	Changed value	Param. count ↓	Step time ↓	COCO CIDEr↑	OKVQA top1↑	VQAv2 top1↑	MSVDQA top1↑	VATEX CIDEr↑	Overall score↑
		***Flamingo*-3B model**		3.2B	1.74s	86.5	42.1	55.8	36.3	53.4	**70.7**
(i)	Training data	All data	w/o Video-Text pairs	3.2B	1.42s	84.2	43.0	53.9	34.5	46.0	67.3
			w/o Image-Text pairs	3.2B	0.95s	66.3	39.2	51.6	32.0	41.6	60.9
			Image-Text pairs→ LAION	3.2B	1.74s	79.5	41.4	53.5	33.9	47.6	66.4
			w/o M3W	3.2B	1.02s	54.1	36.5	52.7	31.4	23.5	53.4
(ii)	Optimisation	Accumulation	Round Robin	3.2B	1.68s	76.1	39.8	52.1	33.2	40.8	62.9
(iii)	Tanh gating	✓	✗	3.2B	1.74s	78.4	40.5	52.9	35.9	47.5	66.5
(iv)	Cross-attention architecture	GATED XATTN-DENSE	VANILLA XATTN	2.4B	1.16s	80.6	41.5	53.4	32.9	50.7	66.9
			GRAFTING	3.3B	1.74s	79.2	36.1	50.8	32.2	47.8	63.1
(v)	Cross-attention frequency	Every	Single in middle	2.0B	0.87s	71.5	38.1	50.2	29.1	42.3	59.8
			Every 4th	2.3B	1.02s	82.3	42.7	55.1	34.6	50.8	68.8
			Every 2nd	2.6B	1.24s	83.7	41.0	55.8	34.5	49.7	68.2
(vi)	Resampler	Perceiver	MLP	3.2B	1.85s	78.6	42.2	54.7	35.2	44.7	66.6
			Transformer	3.2B	1.81s	83.2	41.7	55.6	31.5	48.3	66.7
(vii)	Vision encoder	NFNet-F6	CLIP ViT-L/14	3.1B	1.58s	76.5	41.6	53.4	33.2	44.5	64.9
			NFNet-F0	2.9B	1.45s	73.8	40.5	52.8	31.1	42.9	62.7
(viii)	Freezing LM	✓	✗ (random init)	3.2B	2.42s	74.8	31.5	45.6	26.9	50.1	57.8
			✗ (pretrained)	3.2B	2.42s	81.2	33.7	47.4	31.0	53.9	62.7

Failures: Hallucinations

Input Prompt

Question: What is on the phone screen? Answer:

Question: What can you see out the window? Answer:



Question: Whom is the person texting? Answer:

Output

A text message from a friend.

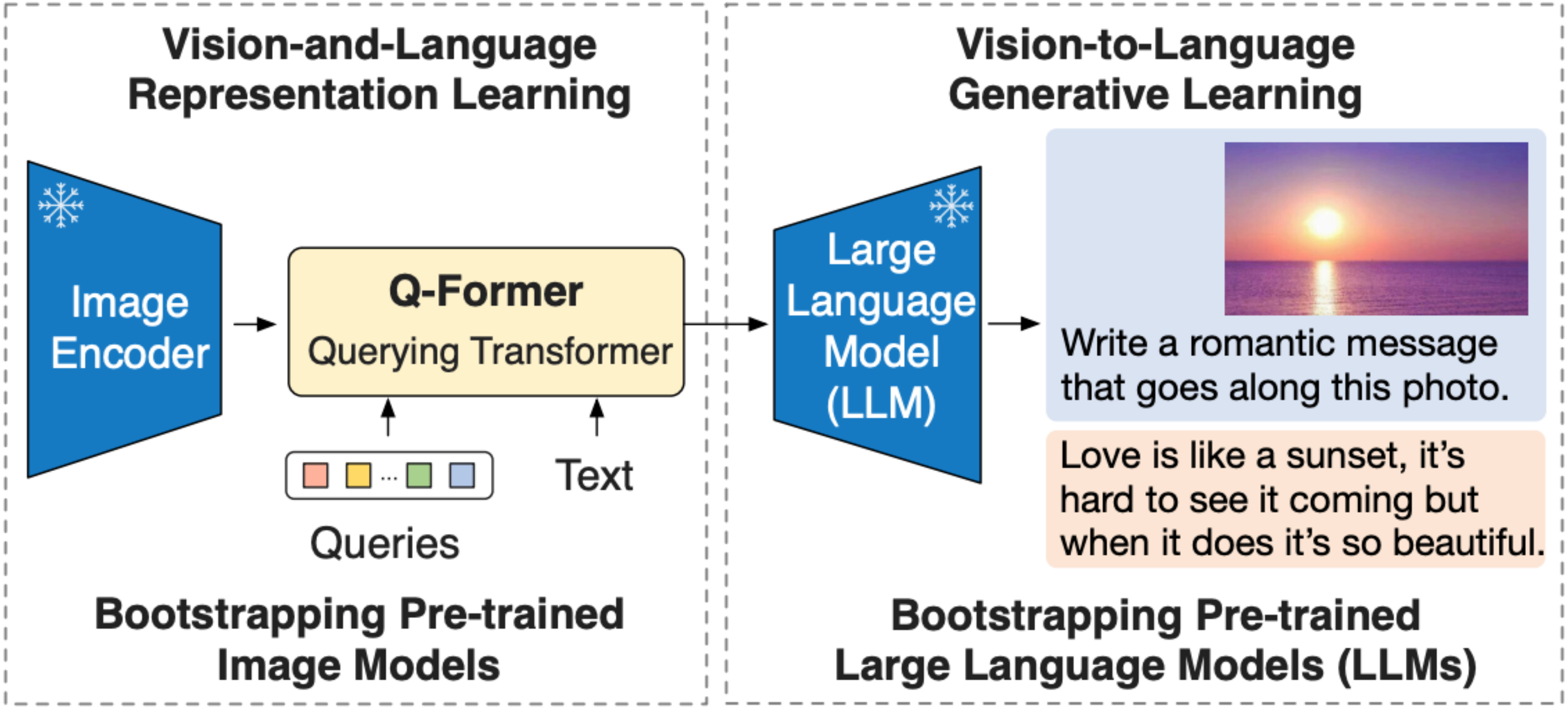
A parking lot.

The driver.

Flamingo: Summary

- Unifying strong single-modal models by connectors
  - Perceiver-based architecture with a fixed number of visual tokens to support images and videos
  - Interleave cross-attention layers with language only self-attention layers
- Heterogeneous training data
  - Combine web scraping with existing image-text or video-text datasets.

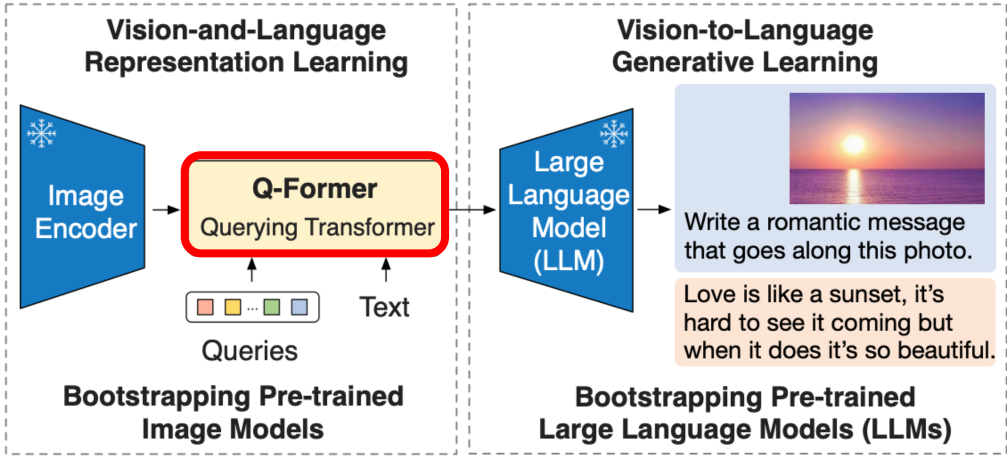
BLIP-2

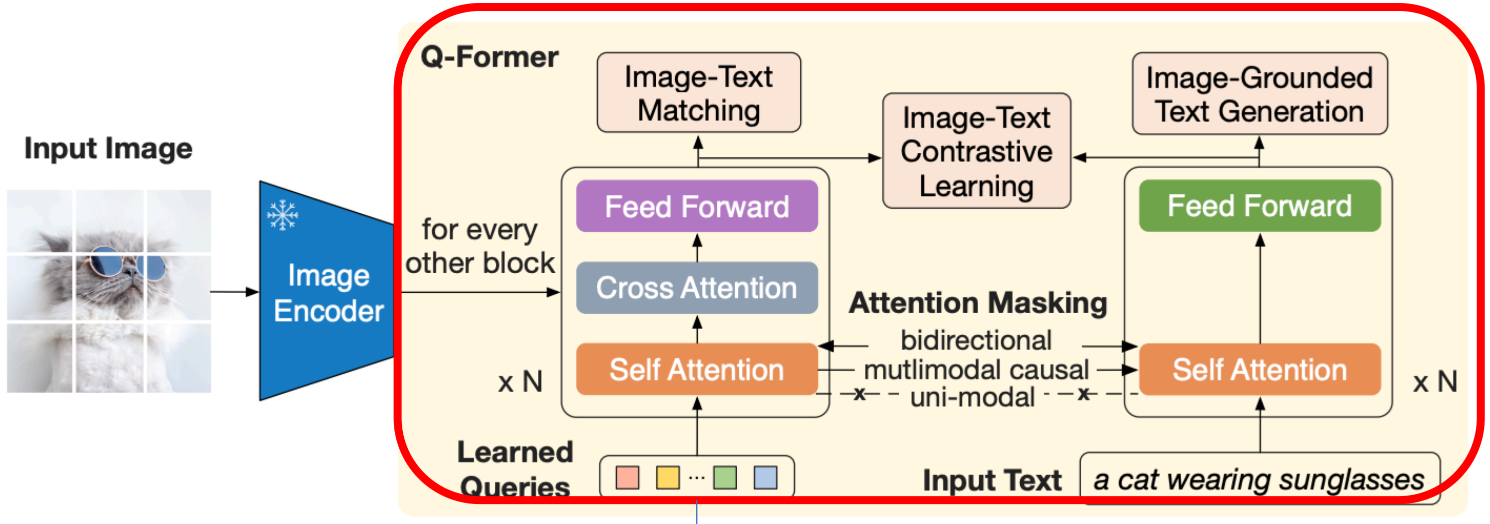


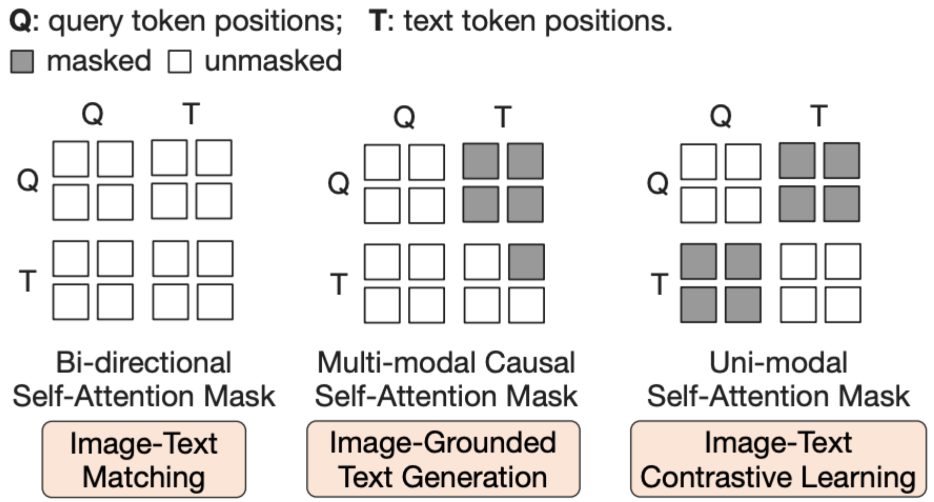
Aligns pretrained and frozen image encoders and language models by a lightweight Q-Former

Li et al., "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models", ICML 2023

BLIP-2: Q-former







a set of learnable query vectors to extract visual features from the frozen image encoder

Li et al., "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models", ICML 2023

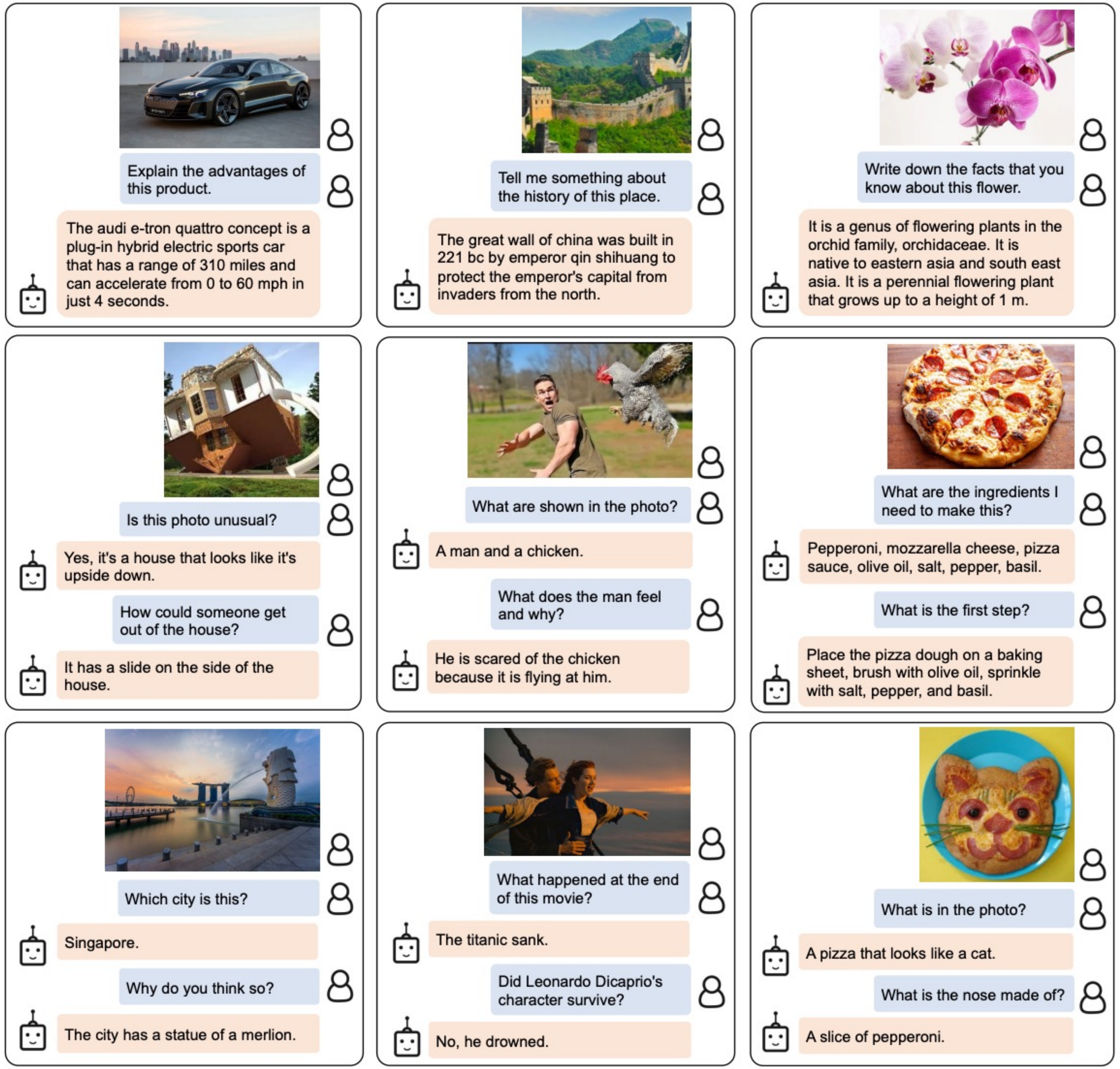
Architectures

- Dual-encoder
- Encoder-decoder
- Fusion-encoder
- **Unified transformer**

BLIP-2: Emerging capabilities

- Powered by LLMs (e.g. OPT and FlanT5), BLIP-2 can be prompted to perform zero-shot image-to-text generation
  - follows natural language instructions
  - enables emerging capabilities such as visual knowledge reasoning and visual conversation

Li et al., "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models", ICML 2023



Li et al., "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models", ICML 2023

Zero-shot Results on Various Tasks

Models	#Trainable Params	Open-sourced?	**Visual Question Answering** VQAv2 (test-dev) VQA acc.	**Image Captioning** NoCaps (val) CIDEr	SPICE	**Image-Text Retrieval** Flickr (test) TR@1	IR@1
BLIP (Li et al., 2022)	583M	✓	-	113.2	14.8	96.7	86.7
SimVLM (Wang et al., 2021b)	1.4B	✗	-	112.2	-	-	-
BEIT-3 (Wang et al., 2022b)	1.9B	✗	-	-	-	94.9	81.5
Flamingo (Alayrac et al., 2022)	10.2B	✗	56.3	-	-	-	-
BLIP-2	188M	✓	**65.0**	**121.6**	**15.8**	**97.6**	**89.7**

Li et al., "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models", ICML 2023