Carnegie Mellon University

Flamingo: a visual language model for few-shot learning

Haoyu Qi, Kelly Shi

Understanding Visual Language Models (VLMs)

Significance in AI

Visual Language Models are crucial for developing AI that can interpret and respond to complex multimodal contexts combining both visual and linguistic elements.

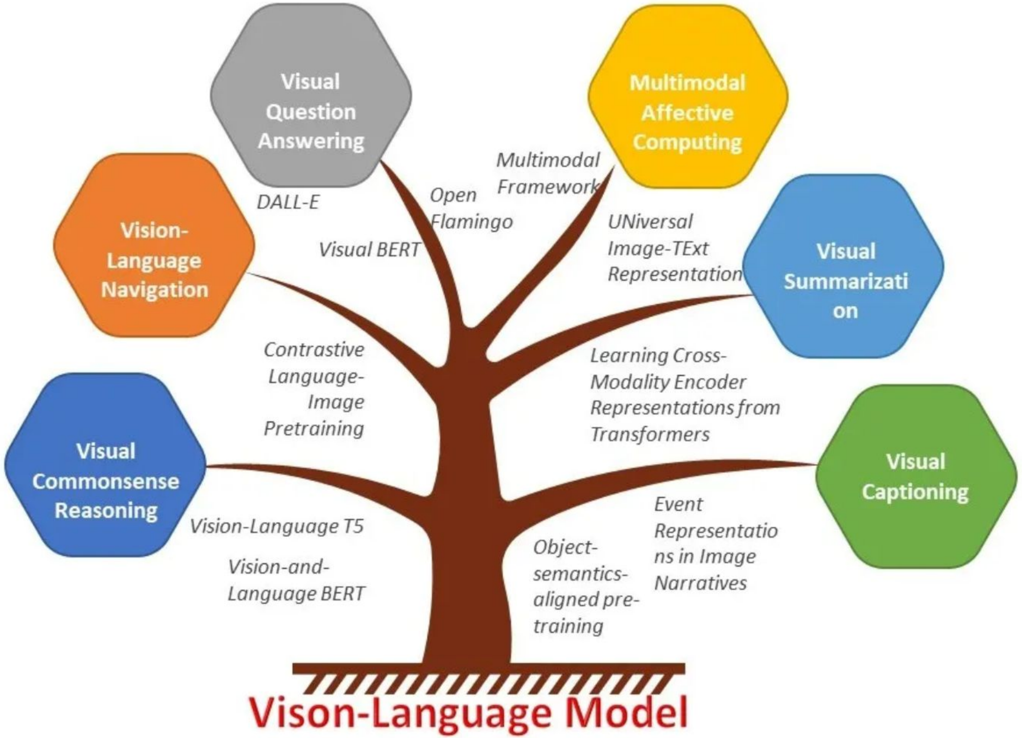
Integration of Sensory Data

These models are designed to process and integrate data from both visual inputs (like images and videos) and textual descriptions, mimicking human sensory and cognitive capabilities.

The Need for a Hybrid Solution

VLms aim to bridge the gaps between language and vision models to create a more versatile model.

Main Tasks Performed by VLMs



Overview of "Flamingo"

Key Features

- A family of visual language models(VLMs)
- Input: visual data interleaved with text
- Output: free-from text
- Training data: large-scale multi-modal web corpora
- In-context few-shot learning

Key Innovations

- Bridge powerful pretrained vision-only and language-only models
- Handle sequences of arbitrarily interleaved visual and textual data
- Seamlessly ingest images or videos as inputs.

Related Work

Language Modelling and Few-Shot Adaptation

- Emergence of Transformers as a substantial advancement in language modeling.
- Standard approach involves pretraining on a large dataset followed by adaptation to specific tasks.
- Flamingo builds on the Chinchilla language model, utilizing in-context few-shot learning, avoiding more complex methods like metric learning and meta-learning.

Related Work

Integration of Language and Vision Models

- The breakthroughs in language models have significantly influenced vision-language modeling.
- Inspiration from BERT has led to a large body of work integrating language with vision.
- Flamingo differs in that it does not require fine-tuning on new tasks, unlike many previous models.

Related Work

Contrastive Learning in Vision-Language Models

- A significant thread in vision-language models involves contrastive learning, which is foundational for models like Flamingo.
- However, Flamingo extends beyond merely using contrastive methods by enabling generative text capabilities.

Related Work

Pretrained Language Models and Their Adaptation

- Freezing the pretrained weights to prevent catastrophic forgetting has become a recent trend in model training.
- Flamingo innovates by freezing certain language model layers while adding learnable layers, allowing it to handle sequences of images, videos, and text seamlessly.

Challenges

- Unifying Strong Single-Modal Models
- Supporting Both Images and Videos
- Heterogeneous Training Data

Challenges

- **Unifying Strong Single-Modal Models**
  - Training large language models is extremely computationally expensive.
  - A text-only model has no built-in way to incorporate data from other modalities.

- Proposed approach
  - Interleave cross-attention layers.
  - A gating mechanism to minimize the effect of added layers, and to improve stability and final performance.

Challenges

- **Supporting Both Images and Videos**
  - Images and videos (of even modest resolution) are high dimensional.
  - Flattening them to 1D sequences is costly as the computation scales quadratically with the sequence length.

- Proposed Approach
  - Use a Perceiver-based architecture that can produce a small fixed number of visual tokens (around a hundred) per image/video, given a large varying number of visual input features (up to several thousand).

Challenges

- **Heterogeneous Training Data**
  - Large models require huge datasets.
  - Paired image/caption datasets used in CLIP and ALIGN may not be general enough to reach GPT-3 style few-shot learning.

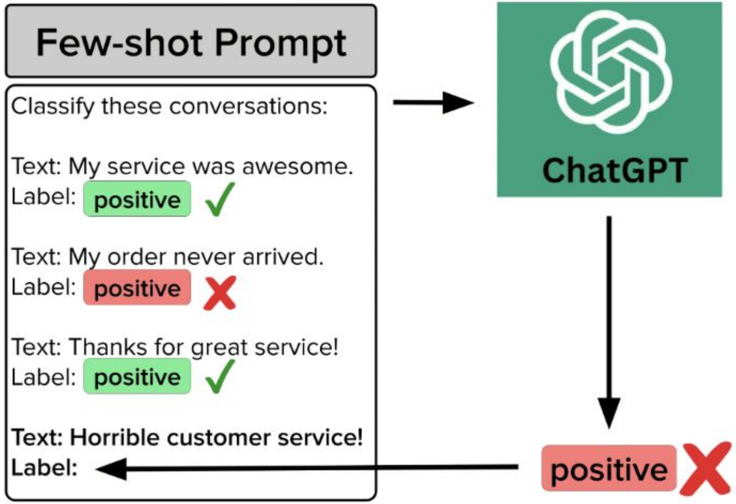
- Proposed Approach
  - Scrape webpages with interleaved images and text. Despite the generality of the data, the images and text are often weakly related.
  - Combine the interleaved dataset with standard paired image/text and video/text datasets where the visual and language are typically more strongly related.

Motivation

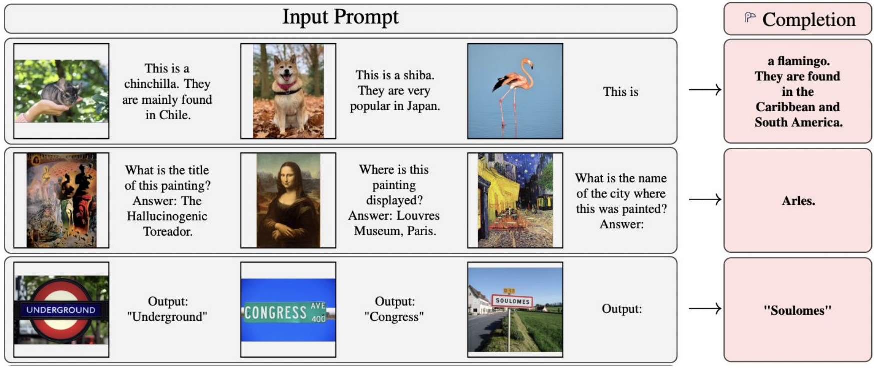
1. traditional solution to learn a new task given a shot instruction in computer vision is to **finetune a pretrained model**.
   - resource intensive
   - requires large amount of annotated data
   - requires careful hyperparameter tuning
2. multimodal vision-language models trained with **a contrastive objective** enables zero-shot adaptation to new tasks
   - **limited use case** as they need finite sets of outcomes to compute similarity scores
   - **Cannot generate language** (not suitable for open-ended Q&A)
   - or **generate visually-conditioned language**, performing bad in low-data regime

Methodology

Few shot prompting in LM

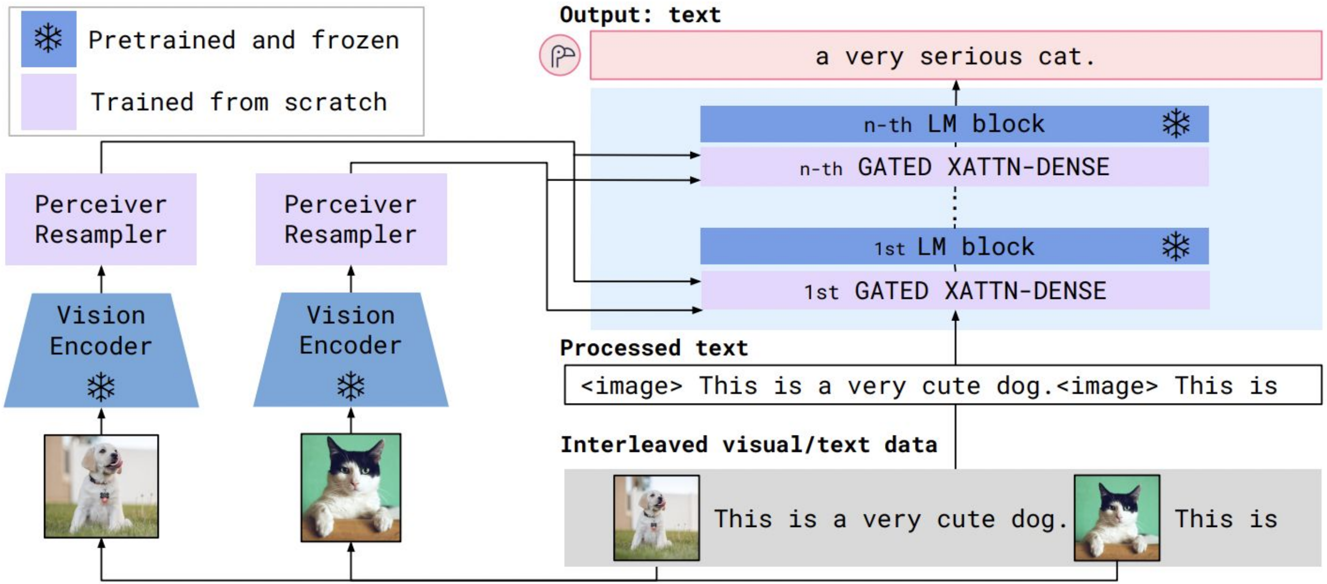


Few shot prompting in Flamingo



Methodology

Few shot prompting in Flamingo

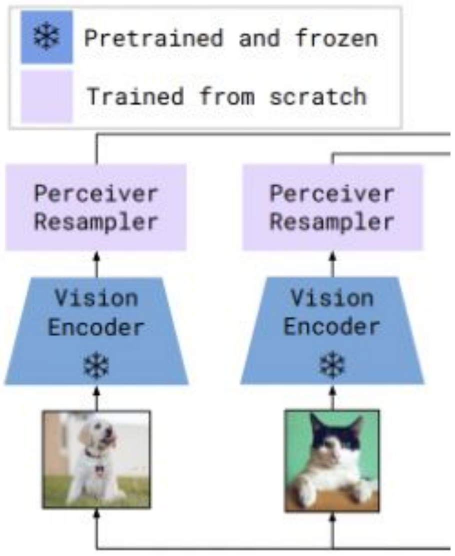


Methodology

Visual Encoder & Perceiver Resampler

Visual Encoder

- Normalizer Free ResNet (NFNet) F6 model
- Pretrained using **two-term contrastive loss of image and text pairs**
  - train Visual Encoder and Language Encoder from scratch
  - encode image and text pairs separately to shared embedding space
  - matched pairs as positive, others as negative
  - minimize sum of text-to-image loss and image-to-text loss
- images' 2D spatial features/videos' 3D spatio-temporal features -> flatten to 1D



Methodology

Visual Encoder & Perceiver Resampler

Visual Encoder

- minimize sum of text-to-image loss and image-to-text loss

normalized embedding of i-th element from language encoder

$$L_{contrastive:txt2im} = -\frac{1}{N}\sum_{i}^{N}\log\left(\frac{\exp(L_i^\intercal V_i\beta)}{\sum_j^N \exp(L_i^\intercal V_j\beta)}\right)$$

normalized embedding of j-th element from visual encoder

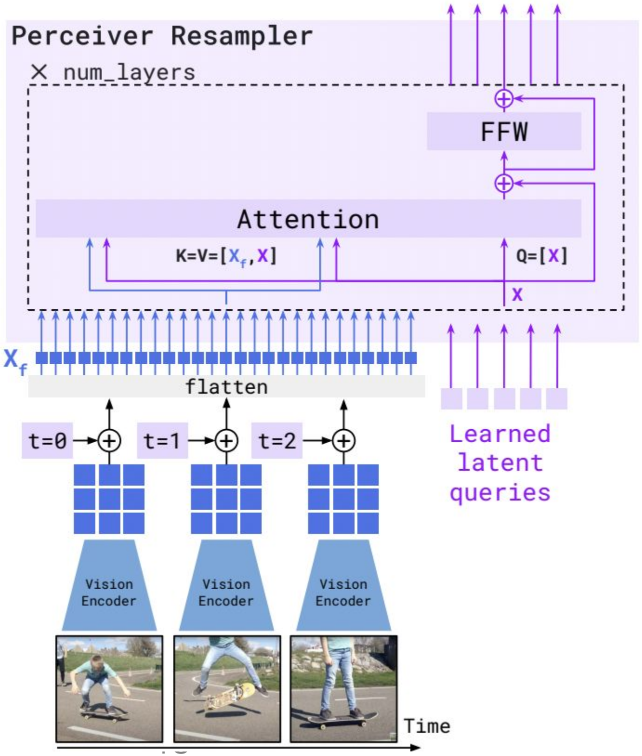
$$L_{contrastive:im2txt} = -\frac{1}{N}\sum_{i}^{N}\log\left(\frac{\exp(V_i^\intercal L_i\beta)}{\sum_j^N \exp(V_i^\intercal L_j\beta)}\right)$$

trainable inverse temperature parameter

Methodology

Visual Encoder & Perceiver Resampler

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo




```python
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

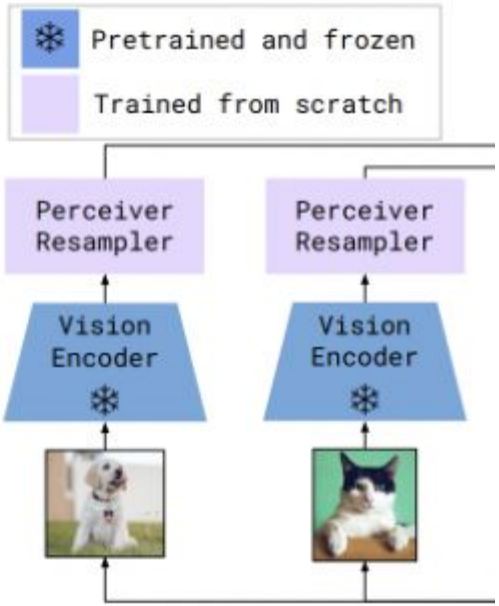
- input: variable number of image/video features
- output: fix number(64) of visual tokens
- Goal: reduce computation complexity of visual-text cross-attention

Methodology

Visual Encoder & Perceiver Resampler

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo

Perceiver Resampler
- learned parameters:
  - latent queries
  - temporal embedding
  - no spatial embedding (CNN include spatial information)
- # of output tokens = # of learned latent queries



Carnegie Mellon University

Methodology

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo

Visual Encoder & Perceiver Resampler

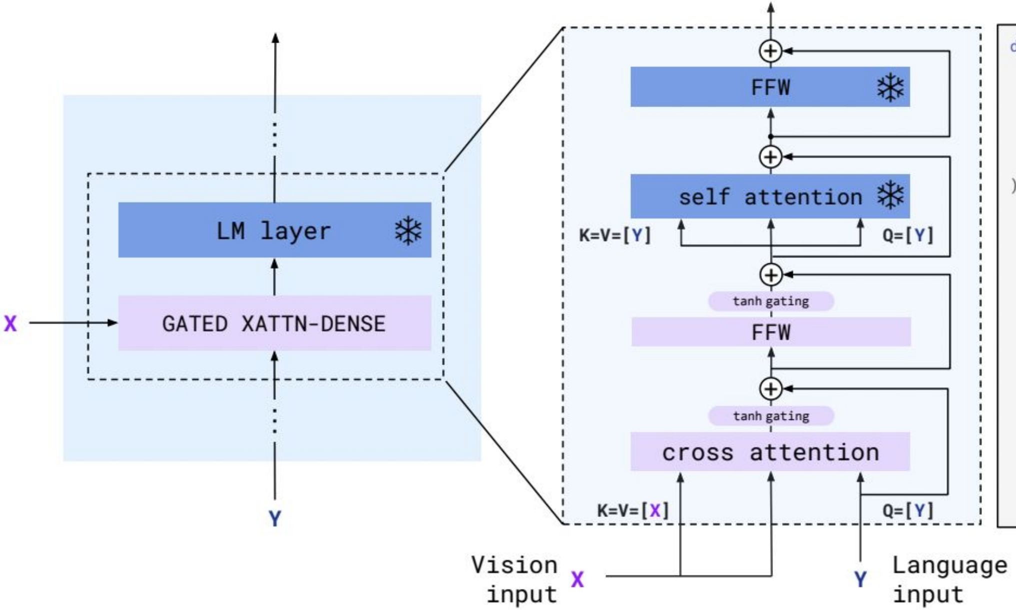
```
def forward(self, x):
    """
    Args:
        x (torch.Tensor): image features
            shape (b, T, F, v, D)
    Returns:
        shape (b, T, n, D) where n is self.num_latents
    """
    b, T, F, v = x.shape[:4]

    # frame and media time embeddings
    if exists(self.frame_embs):
        frame_embs = repeat(self.frame_embs[:F], "F d -> b T F v d", b=b, T=T, v=v)
        x = x + frame_embs
    x = rearrange(
        x, "b T F v d -> b T (F v) d"
    )  # flatten the frame and spatial dimensions
    if exists(self.media_time_embs):
        x = x + self.media_time_embs[:T]

    # blocks
    latents = repeat(self.latents, "n d -> b T n d", b=b, T=T)
    for attn, ff in self.layers:
        latents = attn(x, latents) + latents
        latents = ff(latents) + latents
    return self.norm(latents)
```

Methodology

The “bridge” between visual encoder & LM

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo

Gated Cross Attention Dense block



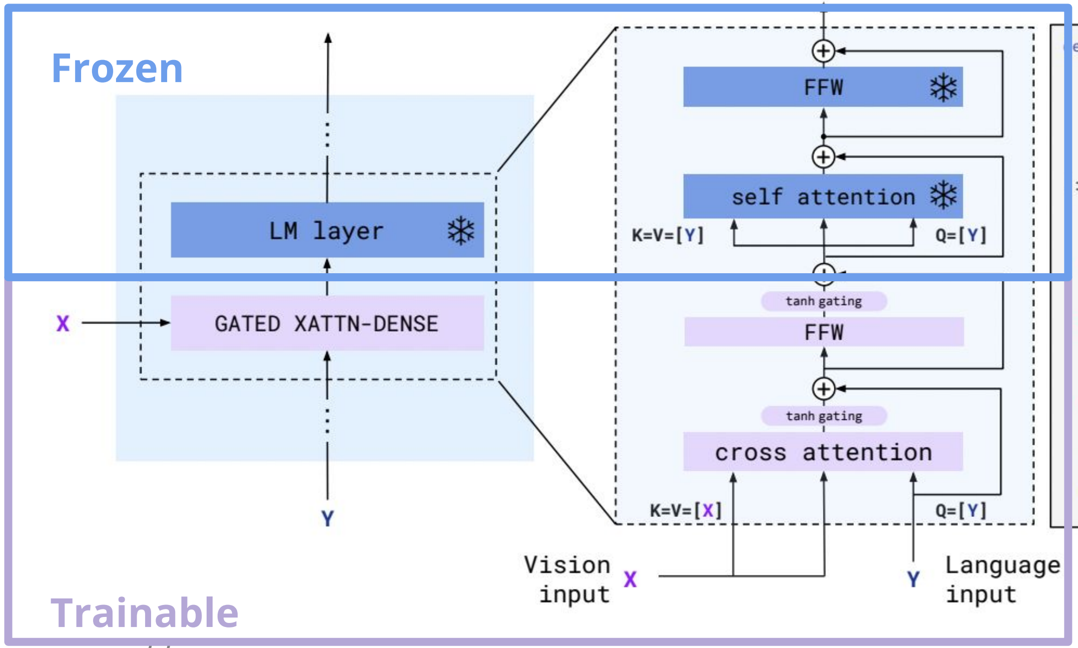

```
def gated_xattn_dense(
    y,  # input language features
    x,  # input visual features
    alpha_xattn, # xattn gating parameter - init at 0.
    alpha_dense, # ffw gating parameter - init at 0.
):
  """Applies a GATED XATTN-DENSE layer."""

  # 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
```

Methodology

The “bridge” between visual encoder & LM

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo

Gated Cross Attention Dense block




```
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
```

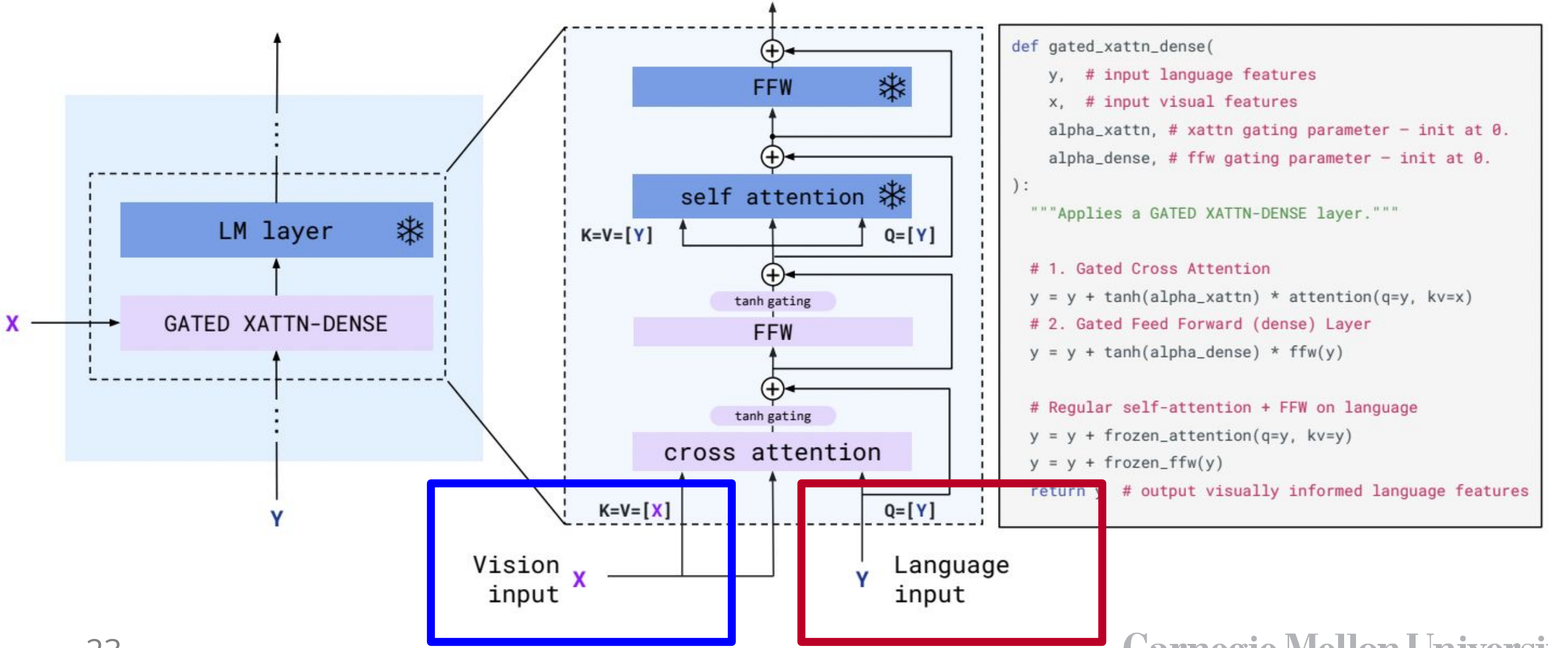

Carnegie Mellon University

Methodology

The “bridge” between visual encoder & LM

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo

Gated Cross Attention Dense block



Methodology

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo

The “bridge” between visual encoder & LM

Gated Cross Attention Dense block

```
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
```

je

- alpha_xattn and alpha_dense are set to 0 initially
- tanh(alpha_*) = 0 initially
- LM is kept intact at initialization for improved stability and performance

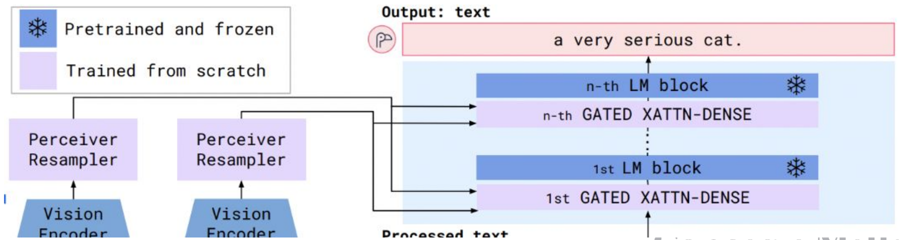
Methodology

The “bridge” between visual encoder & LM

All the codes referenced in methodology are from OpenFlamingo - the open source version of flamingo, which is unofficial.
https://github.com/mlfoundations/open_flamingo

Gated Cross Attention Dense block

- Bridge visual encoder & LM
- Without it, the overall score drops by 4.2% and training becomes unstable
- Trade-off between performance & Resources
    - Add them at every layer is better for overall score, but leads to increasing time complexity
    - inserting them every fourth block accelerates training by 66% while decreasing the overall score by 1.9%



Methodology

Training

Training in LM

Trained on a large amount of text data, providing the model general-purpose generation capabilities.

Training in Flamingo

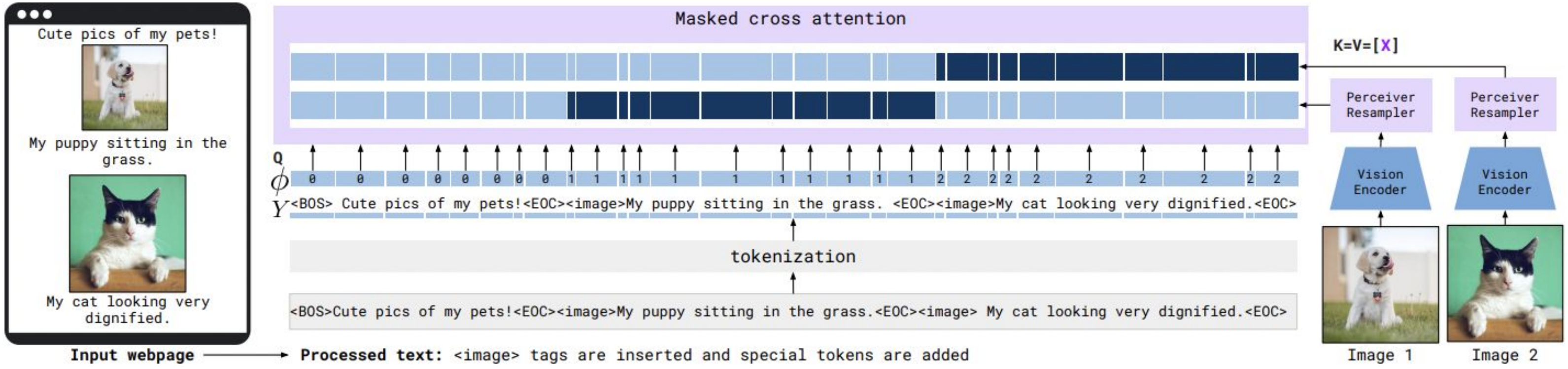
Trained on a carefully chosen mixture of complementary large-scale multimodal data coming only from the web, without using any data annotated for machine learning purposes.

- arbitrary images
- arbitrary position

Methodology

Training

- Input
  - a sequence of text y
  - a sequence of images/videos x
  - $\varphi : [1, L] \mapsto [0, N]$ assigns to each text position the index of the last image/video appearing before this position



Methodology

Training

- model

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

$$y_{<\ell} \triangleq (y_1, \ldots, y_{\ell-1}),\ x_{\leq \ell} \triangleq \{x_i | i \leq \phi(\ell)\}$$

- benefits
  - allows the model to generalise to any number of visual inputs
  - the dependency on all previous images remains via self-attention in the LM.

Methodology

Training

- loss
  - weighted sum of per-dataset expected negative log-likelihoods of text, given the visual inputs
  - accumulate gradients over all datasets > round-robin approach

$$\sum_{m=1}^{M} \lambda_m \cdot \mathbb{E}_{(x,y)\sim\mathcal{D}_m} \left[-\sum_{\ell=1}^{L} \log p(y_\ell | y_{<\ell}, x_{\leq \ell}) \right]$$

Code Walk through

- link to VScode

Training Data

Image-Text pairs data

- ALIGN dataset - 1.8 billion noisy image-text pairs, **12.4** text tokens per image on average.
- LTIP dataset - 312 million image-text pairs, **20.5** text tokens per image on average
- Resolution of **320 x 320** pixels is used for images.
- **32/64** tokens sequence length is used for text.

Video-Text pairs data

- VTP dataset with 27 million short videos.
- **22 seconds** duration on average.
- Resolution of **320 x 320** pixels is used for frames.
- Temporal dimension is **8** (T = 8).
- **32** tokens sequence length is used for text.

MultiModel Massive Web (M3W)

- Extracted text and images from 43 million webpages
- M3W contains **185 million images** and **182 GB of text**.
- Text filter and image filters are used to remove low quality data.
- Resolution of **320 x 320** pixels is used for images.
- Token sequence length of **256** is used for text.

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]

Experiment Setup

Data Augmentation and Pre-Processing

- During training, 50% of text samples are prepended with a space character.
- The authors attribute the effectiveness to subword tokenizer (tokens depend on preceding space).
- Visual inputs are processed at 320 pixels (rather than 288 pixels used in pretraining)
- Image indices ϕ are also perturbed (next/prev prob. 0.5) on interleaved dataset.
- For videos clips of 8 frames (1 fps) are sampled from each training video.
- However, while inference 30 video frames are processed at 3 fps.

Experiment Setup

Infrastructure/Implementation

- Model and associated infrastructure implemented using JAX and Haiku.
- All training and evaluation done on TPUv4 instances.
- Largest (80B) model trained for 15 days on 1536 chips over 16 devices.
- Megatron sharding used for Embedding/S-Attention/X-Attention/FFW.
- ZeRO stage 1 is used to shard optimizer state.
- Activations + gradients are kept in bfloat16 and params + optimizer accumulators are kept in float32.

Experiment Setup

Training and Model Details

- Model Sizes: three different sizes of the Flamingo model, scaling from 1.4 billion to 7 billion and up to 70 billion parameters.
- Vision Encoder: The pretrained vision encoder remains frozen throughout the experiments and utilizes a NFNet-F6 model trained contrastively.

Parameter counts for Flamingo models

	Requires model sharding	Frozen Language	Frozen Vision	Trainable GATED XATTN-DENSE	Trainable Resampler	Total 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**

Parameter counts for Flamingo models

Experiment Setup

Evaluation Benchmarks

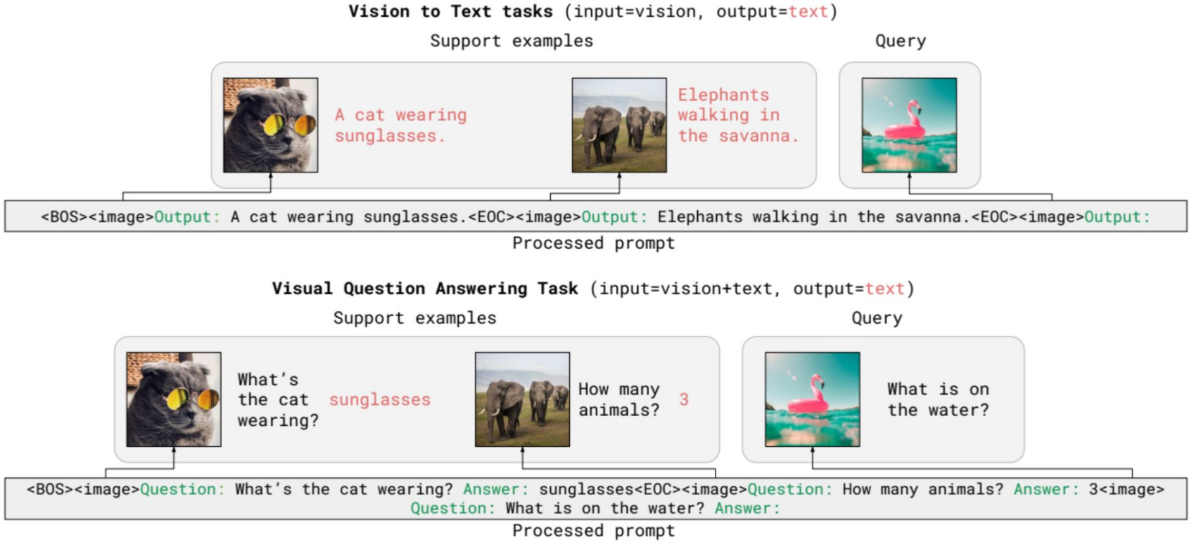
- Development (DEV) Benchmarks: A subset of multimodal image/video and language benchmarks were selected for detailed analysis, including tasks like captioning, visual question answering, and classification.

- Testing Protocols: Evaluation focused on few-shot learning performance, where the model adapts to new tasks using a small number of support samples and is then evaluated on a separate set of query samples.

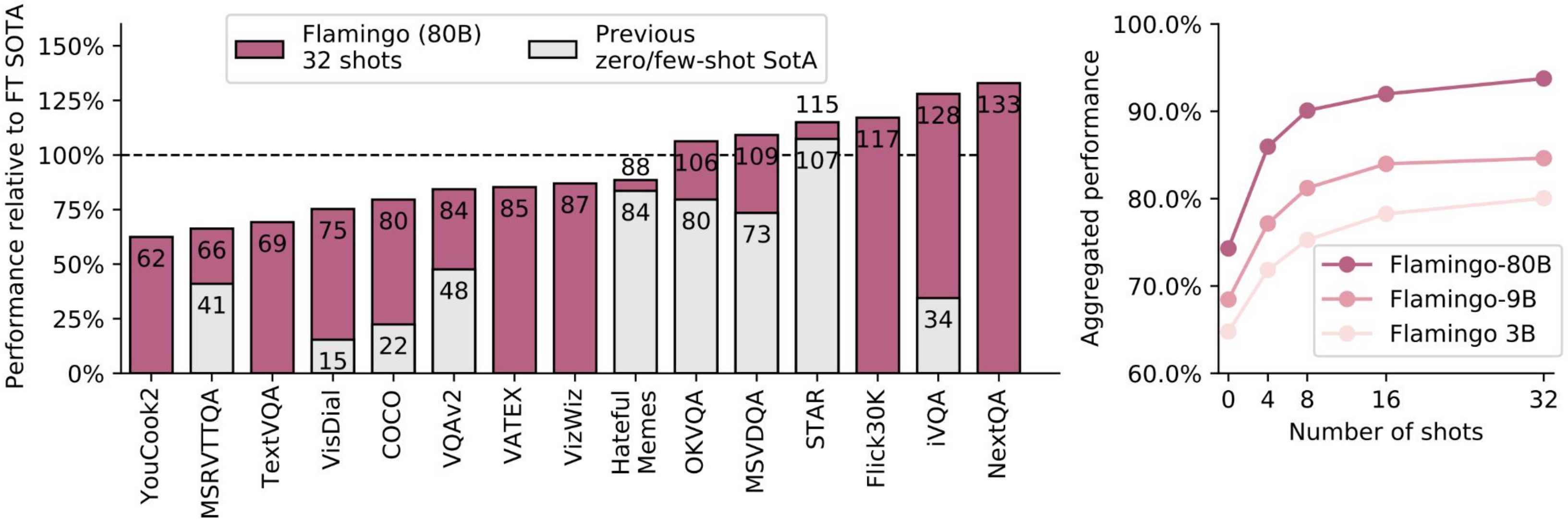
Task Adaptation With Few-Shot In-Context Learning

Few-shot interleaved prompt generation

- Evaluate the ability of the model to rapidly adapt to new tasks using in-context learning, popularised by GPT-3.



Results Overview



Comparison to State of the Art

Method	FT	Shot	OKVQA (I)	VQAv2 (I)	COCO (I)	MSVDQA (V)	VATEX (V)	VizWiz (I)	Flick30K (I)	MSRVTTQA (V)	iVQA (V)	YouCook2 (V)	STAR (V)	VisDial (I)	TextVQA (I)	NextQA (I)	HatefulMemes (I)	RareAct (V)
Zero/Few shot SOTA	✗	(X)	[34] 43.3 (16)	[114] 38.2 (4)	[124] 32.2 (0)	[58] 35.2 (0)	-	-	-	[58] 19.2 (0)	[135] 12.2 (0)	-	[143] 39.4 (0)	[79] 11.6 (0)	-	-	[85] 66.1 (0)	[85] 40.7 (0)
Flamingo-3B	✗	0	41.2	49.2	73.0	27.5	40.1	28.9	60.6	11.0	32.7	55.8	39.6	46.1	30.1	21.3	53.7	58.4
	✗	4	43.3	53.2	85.0	33.0	50.0	34.0	72.0	14.9	35.7	64.6	41.3	47.3	32.7	22.4	53.6	-
	✗	32	45.9	57.1	99.0	42.6	59.2	45.5	71.2	25.6	37.7	76.7	41.6	47.3	30.6	26.1	56.3	-
Flamingo-9B	✗	0	44.7	51.8	79.4	30.2	39.5	28.8	61.5	13.7	35.2	55.0	41.8	48.0	31.8	23.0	57.0	57.9
	✗	4	49.3	56.3	93.1	36.2	51.7	34.9	72.6	18.2	37.7	70.8	**42.8**	50.4	33.6	24.7	62.7	-
	✗	32	51.0	60.4	106.3	47.2	57.4	44.0	72.8	29.4	40.7	77.3	41.2	50.4	32.6	28.4	63.5	-
Flamingo	✗	0	50.6	56.3	84.3	35.6	46.7	31.6	67.2	17.4	40.7	60.1	39.7	52.0	35.0	26.7	46.4	**60.8**
	✗	4	57.4	63.1	103.2	41.7	56.0	39.6	75.1	23.9	44.1	74.5	42.4	**55.6**	36.5	30.8	68.6	-
	✗	32	**57.8**	**67.6**	**113.8**	**52.3**	**65.1**	**49.8**	**75.4**	**31.0**	**45.3**	**86.8**	42.2	**55.6**	**37.9**	**33.5**	**70.0**	-
Pretrained FT SOTA	✔	(X)	54.4 [34] (10K)	80.2 [140] (444K)	143.3 [124] (500K)	47.9 [28] (27K)	76.3 [153] (500K)	57.2 [65] (20K)	67.4 [150] (30K)	46.8 [51] (130K)	35.4 [135] (6K)	138.7 [132] (10K)	36.7 [128] (46K)	75.2 [79] (123K)	54.7 [137] (20K)	25.2 [129] (38K)	79.1 [62] (9K)	-

Carnegie Mellon University

Comparison to SotA when Fine-tuning

Method	VQAV2		COCO	VATEX	VizWiz		MSRVTTQA	VisDial		YouCook2	TextVQA		HatefulMemes
	test-dev	test-std	test	test	test-dev	test-std	test	valid	test-std	valid	valid	test-std	test seen
32 shots	67.6	-	113.8	65.1	49.8	-	31.0	56.8	-	86.8	36.0	-	70.0
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**
SotA	81.3†	81.3†	**149.6**†	81.4†	57.2†	60.6†	46.8	**75.2**	**75.4**†	**138.7**	54.7	**73.7**	84.6†
	[133]	[133]	[119]	[153]	[65]	[65]	[51]	[79]	[123]	[132]	[137]	[84]	[152]

Ablation Study

	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

Limitations

- Worse performance on classification tasks than contrastive models
- Direct inheritance of all the biases
- Toxicity and weaknesses of the Language Model
- Occasional hallucination and un-grounded guesses in open-ended visual question answering tasks

Future Work

Integration of other modalities such as audio