# CS 639: Foundation Models
# **Multimodal Models I**


Fred Sala

University of Wisconsin-Madison


**Feb. 26, 2026**

# Announcements

- Midterm: **March 11, 5:40 pm - 7:20 pm**
  - Sample problems out early next week
- Homework 1: due two days ago
  - Can submit late up to 1 week for 10% off.
  - HW 2: coming out on Tues
- Class roadmap:

| | |
|---|---|
| Thursday Feb. 26 | Multimodal Architectures I |
| Tuesday March 3 | Multimodal Architectures II |
| Thursday March 5 | Prompting, ICL, and Others |
| Tuesday March 10 | Specialization I |

Arch. and using FMs

# Outline

- **Finish up from last time**
  - Flash attention, non-attention based models
- **Multimodal Models Intro + One-Encoder Models**
  - Short history, adapting models to incorporate multiple modalities, BERT-like vision-language models, ViTs
- **VLM Variations and Types**
  - Multi-encoder setups, contrastive training, CLIP, joint training, few-shot models, visual instructions

# Outline

- **Finish up from last time**
  - Flash attention, non-attention based models
- **Multimodal Models Intro + One-Encoder Models**
  - Short history, adapting models to incorporate multiple modalities, BERT-like vision-language models, ViTs
- **VLM Variations and Types**
  - Multi-encoder setups, contrastive training, CLIP, joint training, few-shot models, visual instructions

# Attention Variations

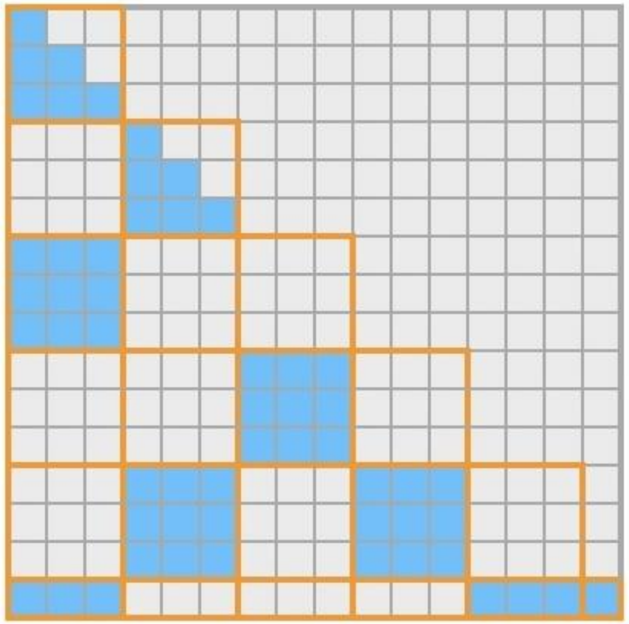
Instead of sparsity in terms of entries, sparsity in terms of blocks

- I.e., blocks generally contain information
- Attend to the whole thing

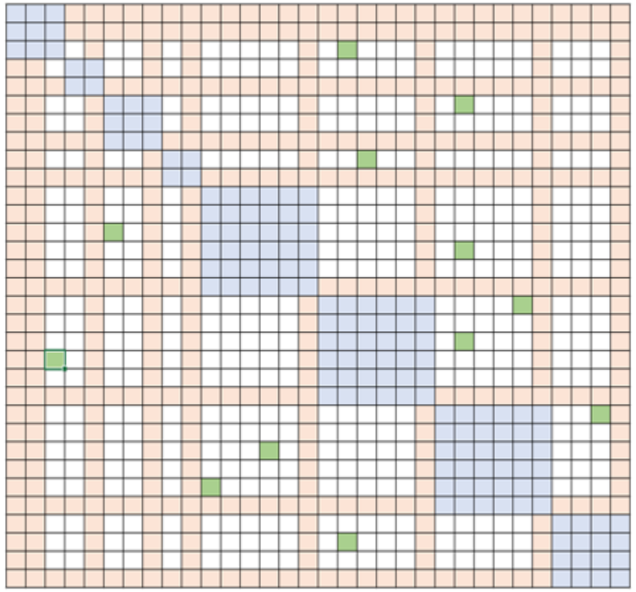
- Can also be combined with other approaches.

  **Ex**: block + global + random

  This is another design problem!



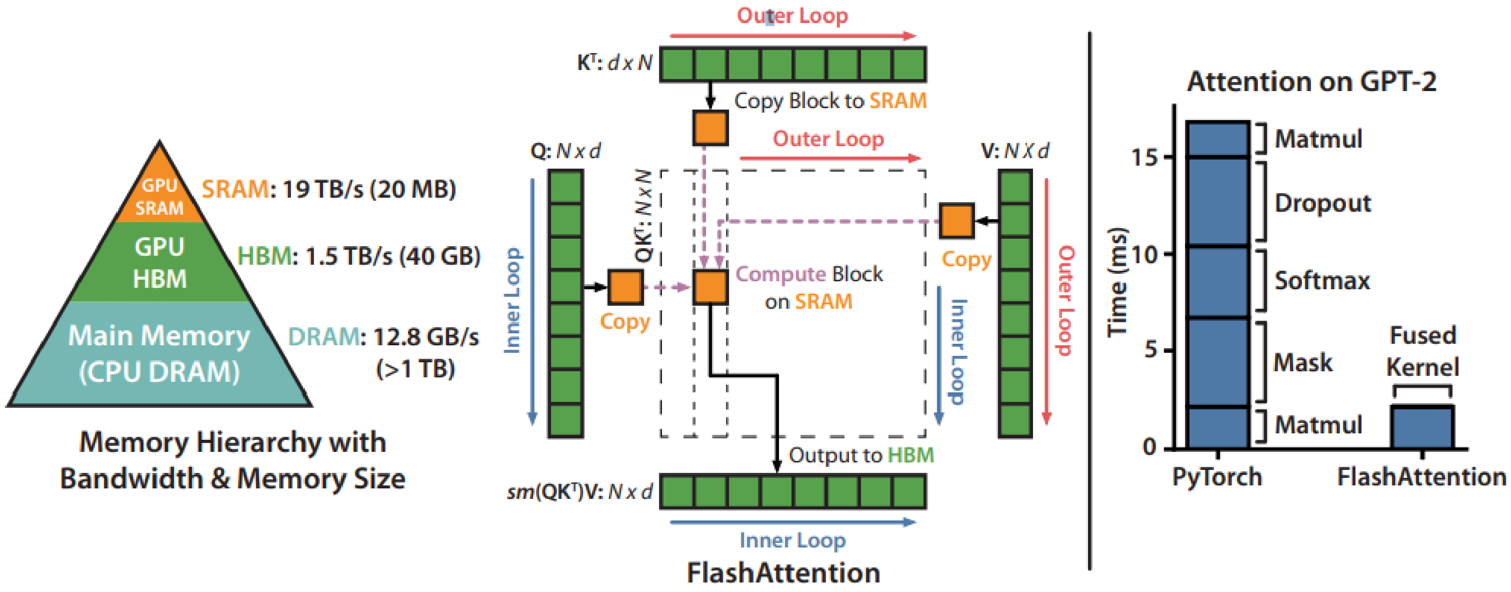
Variable Sparsity Structure



# Hardware Considerations?

So far we dealt with **computational complexity**... but we should also think about practical implementation issues

- GPUs: A little bit of fast memory, lots of slower memory
- Avoid using slow memory when possible?

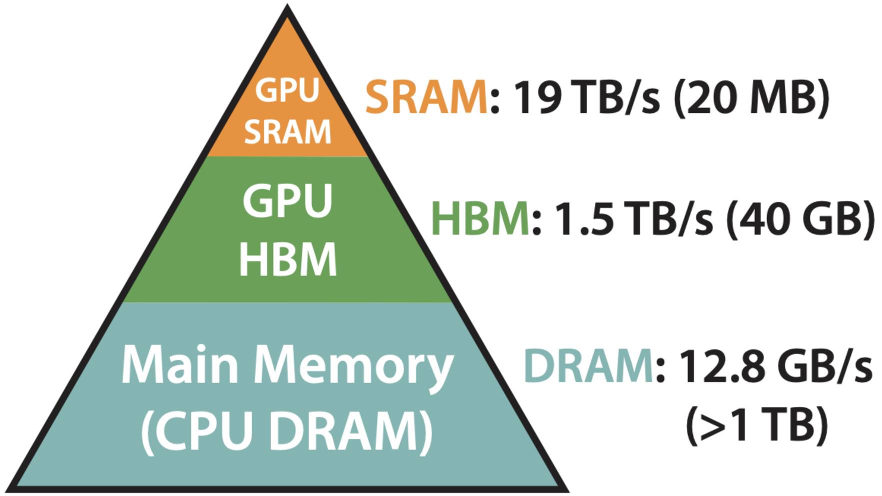



Dao et al '22

# Exact, Hardware-Aware Attention:

Idea for FlashAttention

- Different kinds of GPU memory




Memory Hierarchy with Bandwidth & Memory Size

  - Fast: on-chip SRAM
    - But very little of this: 192KB for each of ~100 processors for an A100 (20MB)
  - Slow(er): HBM
    - But lots: 40-80GB for an A100

- **Goal**: use fast as much as possible, avoid moving to HBM

# Flash Attention: **Basic Idea**

Will use two tricks for higher efficiency

- Tiling and re-computing.

First, recall standard attention

- Will use HBM memory repeatedly
  - Lots of reads and writes:

**Algorithm 0** Standard Attention Implementation

**Require:** Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.

2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.

3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.

4: Return $\mathbf{O}$.

# Flash Attention: **Tiling**

Will use two tricks for higher efficiency

- Tiling and re-computing.

How do we avoid writing and reading from HBM?

- A: don't load the whole thing, use custom **tiling** and save the pieces (small). Standard version

$$m(x) := \max_i \quad x_i, \quad f(x) := \begin{bmatrix} e^{x_1 - m(x)} & \dots & e^{x_B - m(x)} \end{bmatrix}, \quad \ell(x) := \sum_i f(x)_i, \quad \mathrm{softmax}(x) := \frac{f(x)}{\ell(x)}.$$

- Tiling version: two components (can extend)

$$m(x) = m(\begin{bmatrix} x^{(1)} \ x^{(2)} \end{bmatrix}) = \max(m(x^{(1)}), m(x^{(2)})), \quad f(x) = \begin{bmatrix} e^{m(x^{(1)}) - m(x)} f(x^{(1)}) & e^{m(x^{(2)}) - m(x)} f(x^{(2)}) \end{bmatrix},$$

$$\ell(x) = \ell(\begin{bmatrix} x^{(1)} \ x^{(2)} \end{bmatrix}) = e^{m(x^{(1)}) - m(x)} \ell(x^{(1)}) + e^{m(x^{(2)}) - m(x)} \ell(x^{(2)}), \quad \mathrm{softmax}(x) = \frac{f(x)}{\ell(x)}.$$

# Flash Attention: **Recomputing**

Will use two tricks for higher efficiency

- Tiling and re-computing.

How do we avoid writing and reading from HBM?

- A: don't load the whole thing, use custom **tiling** and save the pieces

*"Tiling enables us to implement our algorithm in one CUDA kernel, loading input from HBM, performing all the computation steps (matrix multiply, softmax, optionally masking and dropout, matrix multiply), then write the result back to HBM (masking and dropout in Appendix B). This avoids repeatedly reading and writing of inputs and outputs from and to HBM."*

Don't we need to store full S, P for backwards pass, anyway?

- A: **No!** Can recompute on the fly S, P on the fly

# Flash Attention: **Tradeoffs?**

Will use two tricks for higher efficiency

- Tiling and re-computing.

What's the tradeoff?

- Using tiling and computing/re-computing things normally trades off **memory consumption** for **speed**

- **But…** by reducing memory consumption, we can stick to fast memory only
  - And this makes us **much faster**
  - So **no tradeoff** at all (except for needing custom CUDA kernels ☺)

# Flash Attention: **Tradeoffs?**

Will use two tricks for higher efficiency

- Tiling and re-computing.

Results:

| Model implementations | OpenWebText (ppl) | Training time (speedup) |
|---|---|---|
| GPT-2 small - Huggingface [87] | 18.2 | 9.5 days (1.0×) |
| GPT-2 small - Megatron-LM [77] | 18.2 | 4.7 days (2.0×) |
| GPT-2 small - FlashAttention | 18.2 | **2.7 days (3.5×)** |
| GPT-2 medium - Huggingface [87] | 14.2 | 21.0 days (1.0×) |
| GPT-2 medium - Megatron-LM [77] | 14.3 | 11.5 days (1.8×) |
| GPT-2 medium - FlashAttention | 14.3 | **6.9 days (3.0×)** |

# Attention Alternatives?

Another approach is to get rid of attention and its quadratic cost altogether. Many new alternatives!

- Sometimes called **sub-quadratic** models.
- We'll briefly study a few.
- Step 1: let's get inspired by something RNN-like (well, fully linear for now). Borrow from continuous models:

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$
$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)$$

# State-Space Model

Step 1: let's get inspired by something RNN-like (well, fully linear for now). Borrow from continuous models:

$$\underbrace{}_{}\quad \text{State} \qquad \text{Input}$$

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$

$$\text{Output} \longrightarrow \quad y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)$$

- Can ignore the "D" (think of this as a skip connection).
- Inputs, outputs are 1-D, state is higher dimensional.

# State-Space Model: Discrete Form

Step 2: let's make this a discrete function

State

Input

Output

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k$$
$$y_k = \overline{\boldsymbol{C}} x_k$$

- Ignored D
- Can create approximations of A,B,C through discretizing.
- Looks a lot like an RNN! (or, a linear version of one)

# State-Space Model: Convolutional Form

Step 3: let's unroll the recursion

$$x_0 = \overline{\boldsymbol{B}} u_0 \qquad x_1 = \overline{\boldsymbol{A}\boldsymbol{B}} u_0 + \overline{\boldsymbol{B}} u_1 \qquad x_2 = \overline{\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{A}\boldsymbol{B}} u_1 + \overline{\boldsymbol{B}} u_2$$

$$y_0 = \overline{\boldsymbol{C}\boldsymbol{B}} u_0 \qquad y_1 = \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{B}} u_1 \qquad y_2 = \overline{\boldsymbol{C}\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_1 + \overline{\boldsymbol{C}\boldsymbol{B}} u_2$$

$$y_k = \overline{\boldsymbol{C}\boldsymbol{A}}^k \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{A}}^{k-1} \overline{\boldsymbol{B}} u_1 + \cdots + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_{k-1} + \overline{\boldsymbol{C}\boldsymbol{B}} u_k$$

- In general, $y = \overline{\boldsymbol{K}} * u.$

- This is a **convolution**!

# State-Space Model: Convolutional Form

Step 3: let's unroll the recursion

- Convolution

$$y_k = \overline{\boldsymbol{C}\boldsymbol{A}}^k \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}\boldsymbol{A}}^{k-1} \overline{\boldsymbol{B}} u_1 + \cdots + \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}} u_{k-1} + \overline{\boldsymbol{C}\boldsymbol{B}} u_k$$
$$y = \overline{\boldsymbol{K}} * u.$$

- But a weird one. It's a very **long** convolution.
  - Kernel as long as the input sequence (say, L).
  - Naively, is this better than attention?
  - Let's do **something else** instead.

# Interlude: Time & Frequency Domains

Back to Signals and Systems class,

- Convolution in the time-domain is element-wise multiplication in the frequency domain
- So low-complexity.
- But, need to convert to frequency domain
- Solution: **FFT.** O(L log L) (and also for iFFT, to invert back).
- So, can compute fast and use during training!

$$y_k = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{k-1}\overline{\boldsymbol{B}}u_1 + \cdots + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}}u_k$$

$$y = \overline{\boldsymbol{K}} * u.$$

# Back to SSM Picture

Back to the formula

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k$$
$$y_k = \overline{\boldsymbol{C}} x_k$$

- Just directly making all of these trainable parameters doesn't work so well.
  - Similar issues as in RNNs: stuff blowing up
  - Instead, various models propose approaches

S4 (Structured State Space Models) Gu et al' 22

  - Build A with a special fixed transition matrix that is good at memorization
  - Couple with a particular parametrization to get the discretization.

# Using SSMs as Layers

Back to the formula

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k$$
$$y_k = \overline{\boldsymbol{C}} x_k$$

S4 (Structured State Space Models) Gu et al’ 22

- Special A state transition matrix
- Special parametrization/choice of trainable parameters

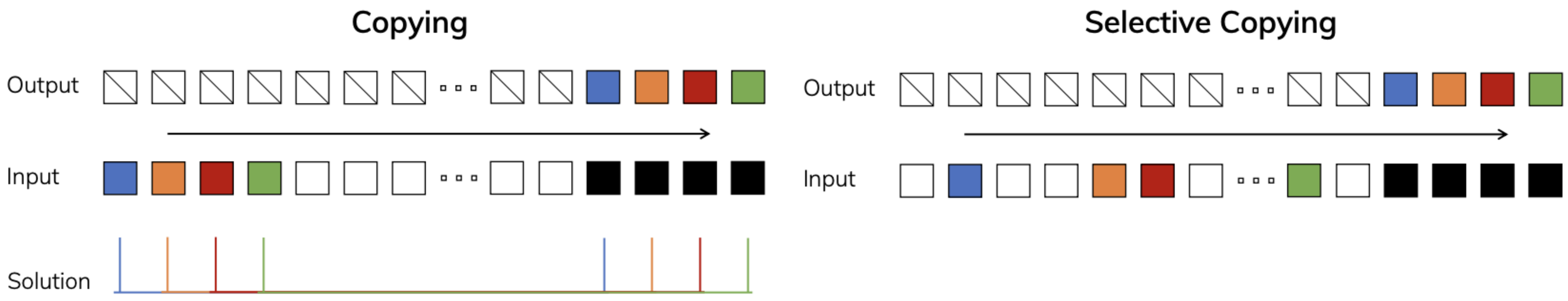
- How to actually use these? Need to define a layer,
  - Stack H of them together (similar to conv layers, multihead attn)
  - Mix with linear layer, place activation function at the end

# S4 Results: The Good and the Bad

Models like S4 can address **very long sequences**

- "S4 solves the **Path-X task**, an extremely challenging task that involves reasoning about LRDs over sequences of length … 16384. All previous models have failed…"

- But, can struggle with "selective" tasks.



# S4 Results: The Good and the Bad

Solution: need some type of context-aware approach

- **Mamba Model**
  - Gu and Dao ‘23, “Mamba: Linear-Time Sequence Modeling with Selective State Spaces”

**Algorithm 1** SSM (S4)

**Input:** $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
**Output:** $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$

$\boldsymbol{A} : (\mathsf{D}, \mathsf{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathsf{D}, \mathsf{N}) \leftarrow \mathsf{Parameter}$
$\boldsymbol{C} : (\mathsf{D}, \mathsf{N}) \leftarrow \mathsf{Parameter}$
$\Delta : (\mathsf{D}) \leftarrow \tau_\Delta(\mathsf{Parameter})$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathsf{D}, \mathsf{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
**return** $y$

**Algorithm 2** SSM + Selection (S6)

**Input:** $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
**Output:** $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$

$\boldsymbol{A} : (\mathsf{D}, \mathsf{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_B(x)$
$\boldsymbol{C} : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_C(x)$
$\Delta : (\mathsf{B}, \mathsf{L}, \mathsf{D}) \leftarrow \tau_\Delta(\mathsf{Parameter} + s_\Delta(x))$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathsf{B}, \mathsf{L}, \mathsf{D}, \mathsf{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-varying: recurrence (*scan*) only
**return** $y$

# Break & Questions

# Outline

- **Finish up from last time**
  - Flash attention, non-attention based models
- **Multimodal Models Intro + One-Encoder Models**
  - Short history, adapting models to incorporate multiple modalities, BERT-like vision-language models, ViTs
- **VLM Variations and Types**
  - Multi-encoder setups, contrastive training, CLIP, joint training, few-shot models, visual instructions

# Short History of Multimodal Models

**Multimodal models** pre-date foundation models

- Image-captioning models, VQA models, etc…
    - But it has become more popular

- Ex: **joint embedding spaces**

(Weston, Bengio, Usunier ’11)

| Image | One-vs-Rest | WSABIE |
|---|---|---|
|  | surf, bora, belize, sea world, balena, wale, tahiti, delfini, surfing, mahi mahi | delfini, orca, **dolphin**, mar, delfin, dauphin, whale, cancun, killer whale, sea world |
|  | **eiffel tower**, tour eiffel, snowboard, blue sky, empire state building, luxor, eiffel, lighthouse, jump, adventure | **eiffel tower**, statue, eiffel, mole antoneliana, la tour eiffel, londra, cctv tower, big ben, calatrava, tokyo tower |
|  | falco, barack, daniel craig, **obama**, barack obama, kanye west, pharrell williams, 50 cent, barrack obama, bono | barrack obama, barack obama, barack hussein obama, barack obama, james marsden, jay z, **obama**, nelly, falco, barack |

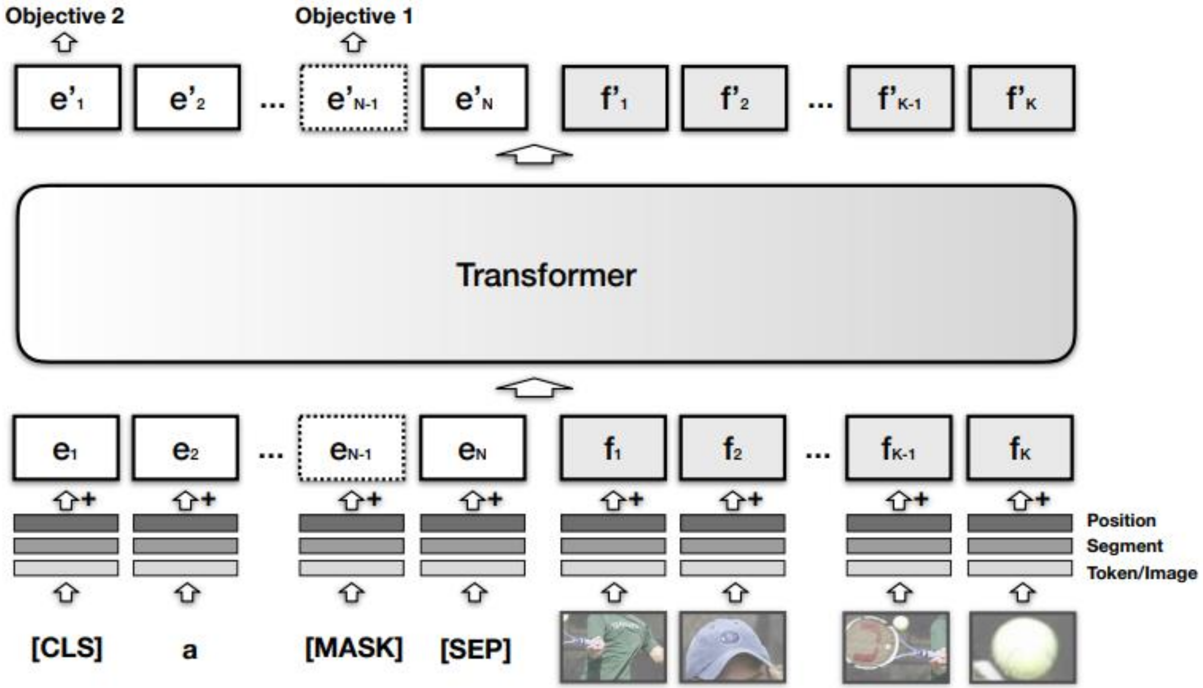
# Making LLMs Multimodal

How do we use a language architecture for multiple modalities?

**VisualBERT**: take all the ideas from BERT, add images

- Use bounding boxes from image detector + image embedder

A person hits a ball with a tennis racket

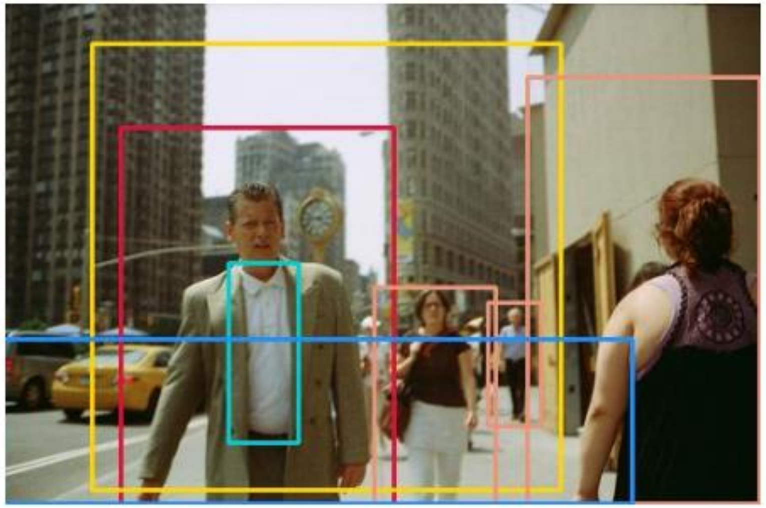


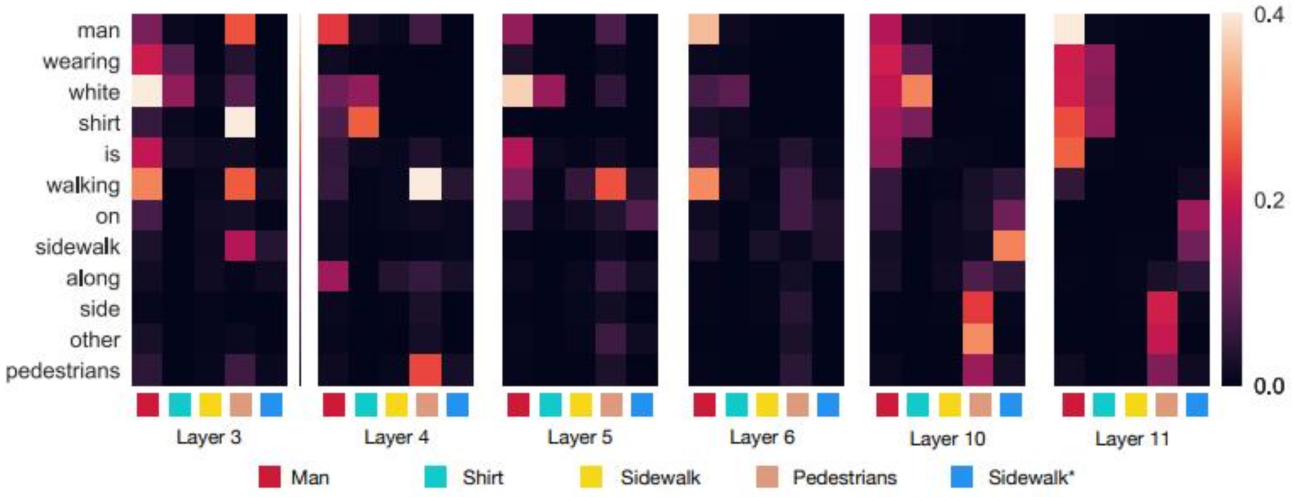

Li et al '19

# Making LLMs Multimodal: **VisualBERT**

**VisualBERT**: take all the ideas from BERT, add images

- What about training? Recall BERT training…
  - Masked language modeling + image (text is masked, image same)
  - Sentence-image prediction
- Results (Li et al, ‘19)





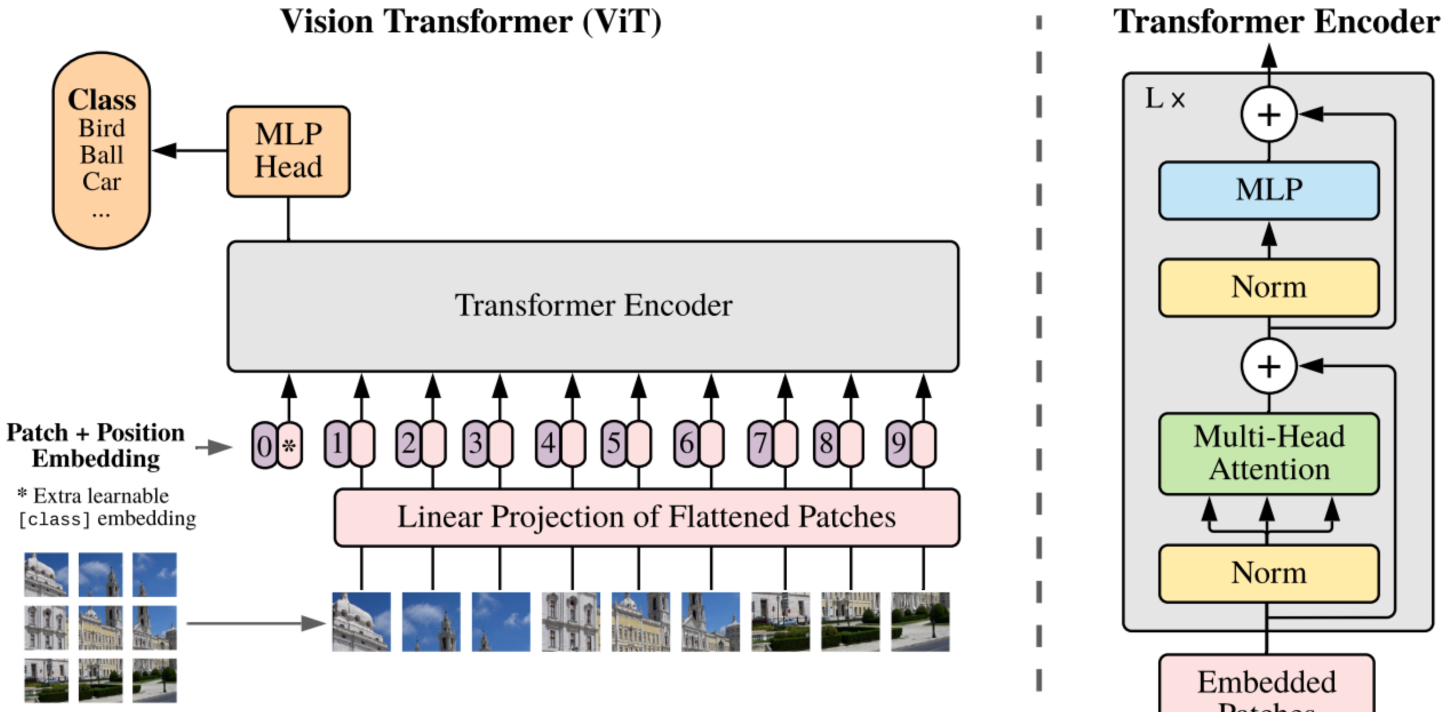
# How Do We Get Image Embeddings?

An Image is Worth 16x16 Words: Transformers for ...

by A Dosovitskiy · 2020 · Cited by 46708 — A pure transformer applied directly to sequences of image patches can perform very well on image classification tasks.

Could always use Resnets, etc., but…

- Didn’t Transformers make a big difference for text?
- Can also use for vision: **ViT.** Just use patches!

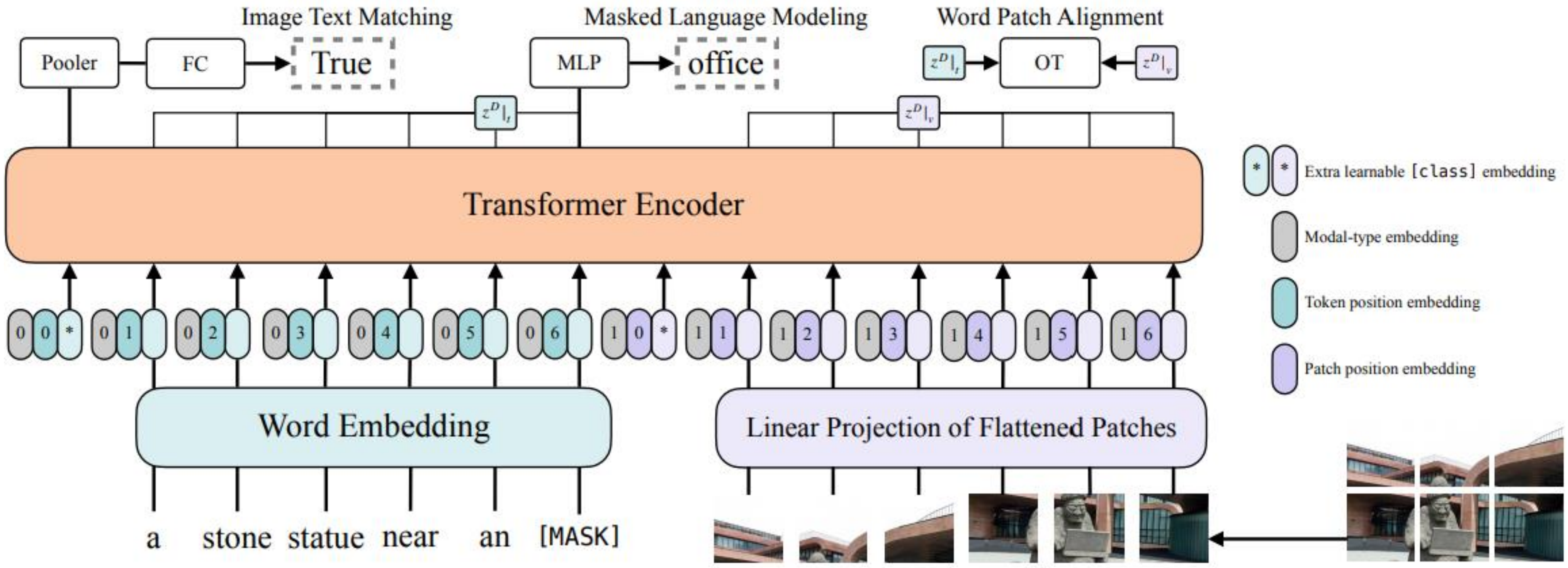



Dosovitskiy et al ‘21

# Put It Together

Multimodal with language and vision transformers: **ViLT**

- Kim et al '21



# Variations…

Lots of different approaches!

- Du et al ‘22, “A Survey of Vision-Language Pre-Trained Models”

| VL-PTM | Text encoder | Vision encoder | Fusion scheme | Pre-training tasks | Multimodal datasets for pre-training |
|---|---|---|---|---|---|
| **Fusion Encoder** | | | | | |
| VisualBERT [2019] | BERT | Faster R-CNN | Single stream | MLM+ITM | COCO |
| Uniter [2020] | BERT | Faster R-CNN | Single stream | MLM+ITM+WRA+MRFR+MRC | CC+COCO+VG+SBU |
| OSCAR [2020c] | BERT | Faster R-CNN | Single stream | MLM+ITM | CC+COCO+SBU+Flickr30k+VQA |
| InterBert [2020] | BERT | Faster R-CNN | Single stream | MLM+MRC+ITM | CC+COCO+SBU |
| ViLBERT [2019] | BERT | Faster R-CNN | Dual stream | MLM+MRC+ITM | CC |
| LXMERT [2019] | BERT | Faster R-CNN | Dual stream | MLM+ITM+MRC+MRFR+VQA | COCO+VG+VQA |
| VL-BERT [2019] | BERT | Faster R-CNN+ ResNet | Single stream | MLM+MRC | CC |
| Pixel-BERT [2020] | BERT | ResNet | Single stream | MLM+ITM | COCO+VG |
| Unified VLP [2020] | UniLM | Faster R-CNN | Single stream | MLM+seq2seq LM | CC |
| UNIMO [2020b] | BERT, RoBERTa | Faster R-CNN | Single stream | MLM+seq2seq LM+MRC+MRFR+CMCL | COCO+CC+VG+SBU |
| SOHO [2021] | BERT | ResNet + Visual Dictionary | Single stream | MLM+MVM+ITM | COCO+VG |
| VL-T5 [2021] | T5, BART | Faster R-CNN | Single stream | MLM+VQA+ITM+VG+GC | COCO+VG |
| XGPT [2021] | transformer | Faster R-CNN | Single stream | IC+MLM+DAE+MRFR | CC |
| Visual Parsing [2021] | BERT | Faster R-CNN + Swin transformer | Dual stream | MLM+ITM+MFR | COCO+VG |
| ALBEF [2021a] | BERT | ViT | Dual stream | MLM+ITM+CMCL | CC+COCO+VG+SBU |
| SimVLM [2021b] | ViT | ViT | Single stream | PrefixLM | C4+ALIGN |
| WenLan [2021] | RoBERTa | Faster R-CNN + EffcientNet | Dual stream | CMCL | RUC-CAS-WenLan |
| ViLT [2021] | ViT | Linear Projection | Single stream | MLM+ITM | CC+COCO+VG+SBU |
| **Dual Encoder** | | | | | |
| CLIP [2021] | GPT2 | ViT, ResNet | | CMCL | self-collected |
| ALIGN [2021] | BERT | EffcientNet | | CMCL | self-collected |
| DeCLIP [2021b] | GPT2, BERT | ViT, ResNet, RegNetY-64GF | | CMCL+MLM+CL | CC+self-collected |
| **Fusion Encoder+ Dual Encoder** | | | | | |
| VLMo [2021a] | BERT | ViT | Single stream | MLM+ITM+CMCL | CC+COCO+VG+SBU |
| FLAVA [2021] | ViT | ViT | Single stream | MMM+ITM+CMCL | CC+COCO+VG+SBU+RedCaps |

# Datasets

Trained on? Datasets with image-text pairs

| Dataset | Year | Num. of Image-Text Pairs | Language | Public |
|---|---|---|---|---|
| SBU Caption [92] [link] | 2011 | 1M | English | ✓ |
| COCO Caption [93] [link] | 2016 | 1.5M | English | ✓ |
| Yahoo Flickr Creative Commons 100 Million (YFCC100M) [94] [link] | 2016 | 100M | English | ✓ |
| Visual Genome (VG) [95] [link] | 2017 | 5.4 M | English | ✓ |
| Conceptual Captions (CC3M) [96] [link] | 2018 | 3.3M | English | ✓ |
| Localized Narratives (LN) [97] [link] | 2020 | 0.87M | English | ✓ |
| Conceptual 12M (CC12M) [98] [link] | 2021 | 12M | English | ✓ |
| Wikipedia-based Image Tex (WIT) [99] [link] | 2021 | 37.6M | 108 Languages | ✓ |
| Red Caps (RC) [100] [link] | 2021 | 12M | English | ✓ |
| LAION400M [28] [link] | 2021 | 400M | English | ✓ |
| LAION5B [27] [link] | 2022 | 5B | Over 100 Languages | ✓ |
| WuKong [101] [link] | 2022 | 100M | Chinese | ✓ |
| CLIP [14] | 2021 | 400M | English | ✗ |
| ALIGN [24] | 2021 | 1.8B | English | ✗ |
| FILIP [25] | 2021 | 300M | English | ✗ |
| WebLI [102] | 2022 | 12B | 109 Languages | ✗ |

Zhang et al ‘23

# Break & Questions

# Outline

- **Finish up from last time**
  - Flash attention, non-attention based models
- **Multimodal Models Intro + One-Encoder Models**
  - Short history, adapting models to incorporate multiple modalities, BERT-like vision-language models, ViTs
- **VLM Variations and Types**
  - Multi-encoder setups, contrastive training, CLIP, joint training, few-shot models, visual instructions

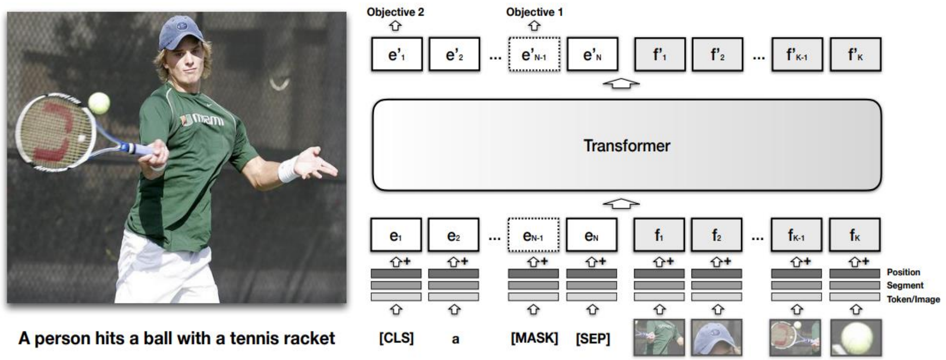
# Contrastive Vision-Language Models

So far, trained the modalities together

- I.e., text and images were both inputs to a transformer
- This is "fusion", but we could do it **later**…

- I.e., produce two representations separately, then produce some means of connecting/tying them together

- **Contrastive** approach




Li et al '19

# VLMs: **Contrastive Training**

Training approach: contrastive

- Loss example: InfoNCE (noise contrastive estimation) loss:

$$\mathcal{L}_I^{\text{InfoNCE}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\left(z_i^I \cdot z_+^I/\tau\right)}{\sum_{j=1, j\neq i}^{B+1} \exp(z_i^I \cdot z_j^I/\tau)}$$

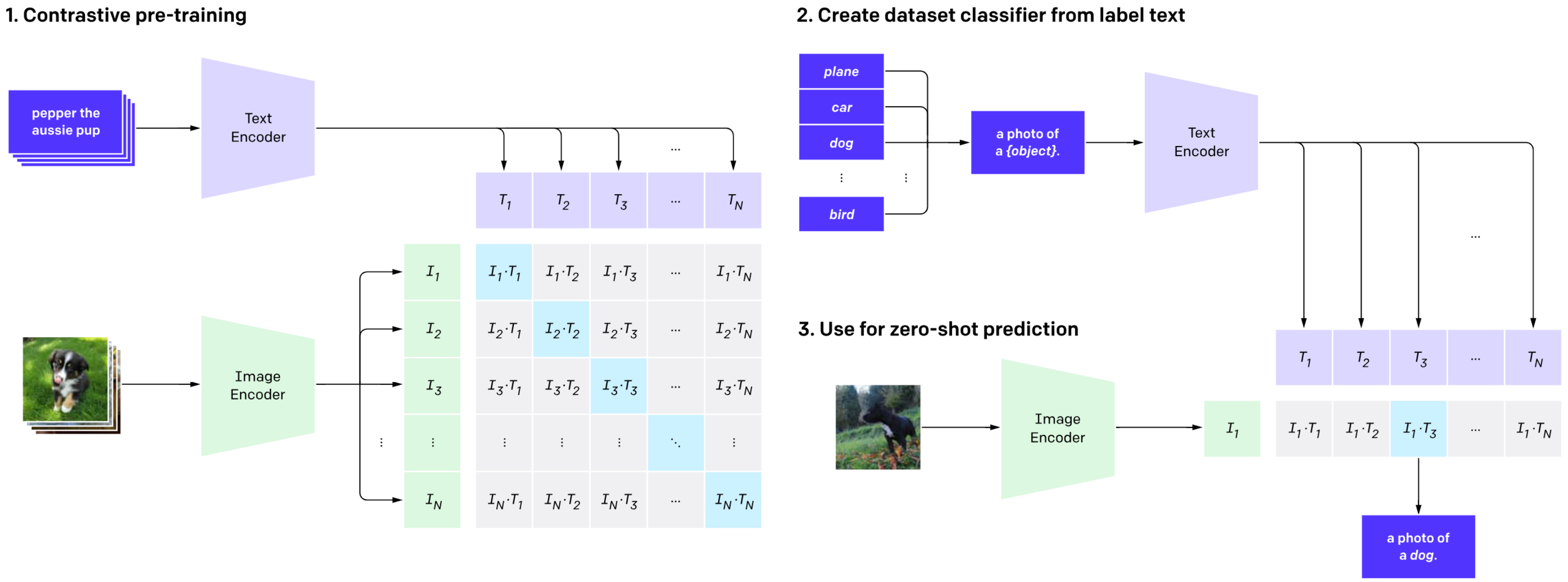
- To train a text and image encoder simultaneously, symmetrize:

$$\mathcal{L}_{I\to T} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\left(z_i^I \cdot z_i^T/\tau\right)}{\sum_{j=1}^{B} \exp(z_i^I \cdot z_j^T/\tau)}$$

$$\mathcal{L}_{T\to I} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\left(z_i^T \cdot z_i^I/\tau\right)}{\sum_{j=1}^{B} \exp(z_i^T \cdot z_j^I/\tau)}$$

# VLMs: **CLIP**

A simple but easily scalable constrastive VLM

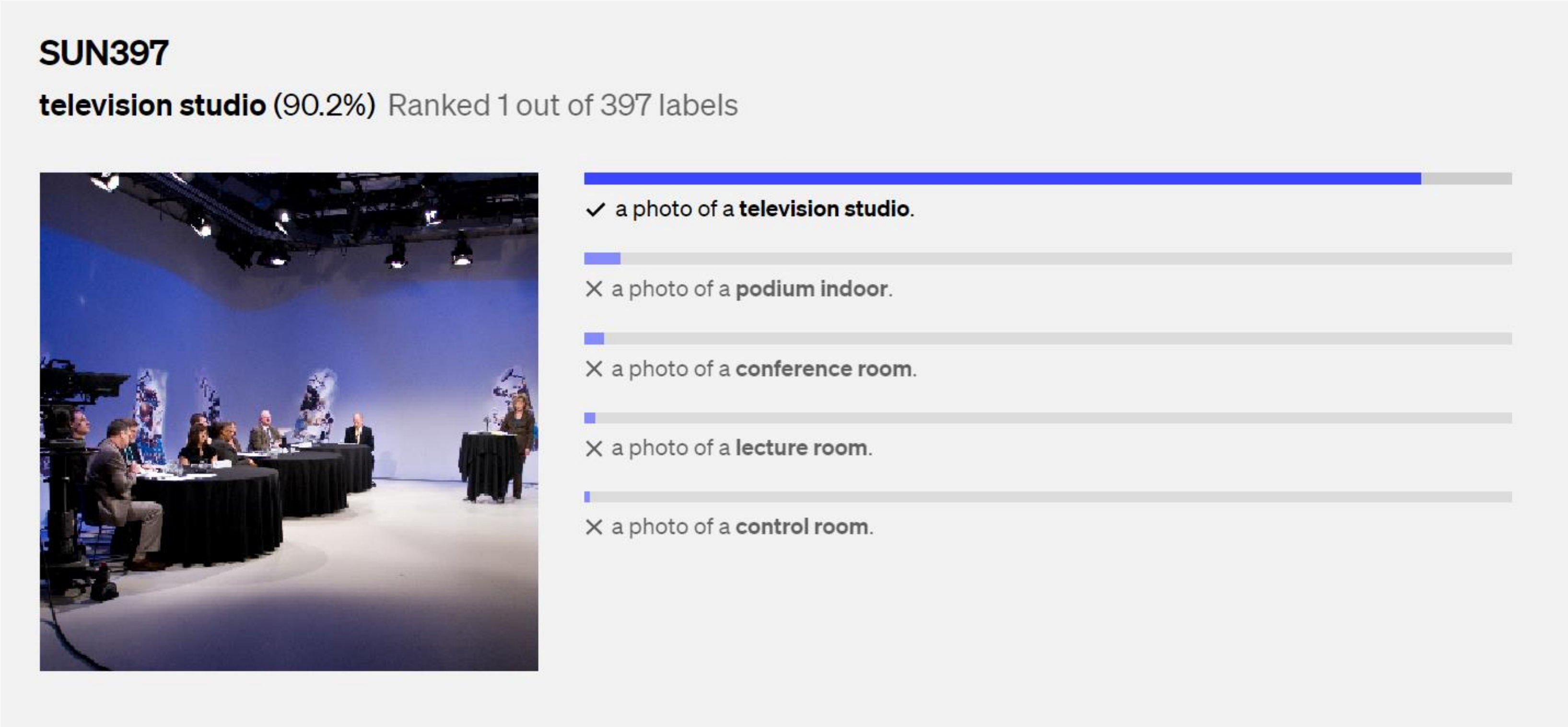



OpenAI

# How to use CLIP?

Standard way: use pre-defined templates

- E.g., “a photo of a [X]”

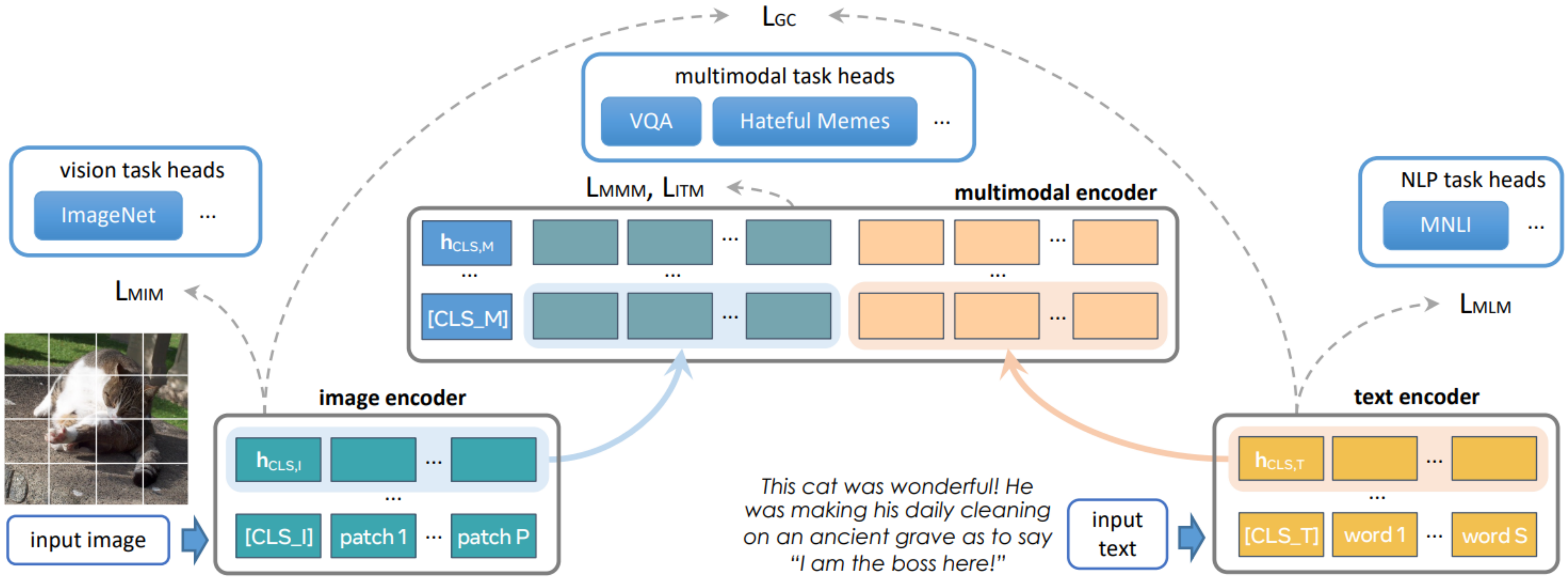



OpenAI

# VLMs: **FLAVA**

Foundational Language And Vision Alignment Model (FLAVA)

- Combines everything
- Pretrain **separately** and **jointly**



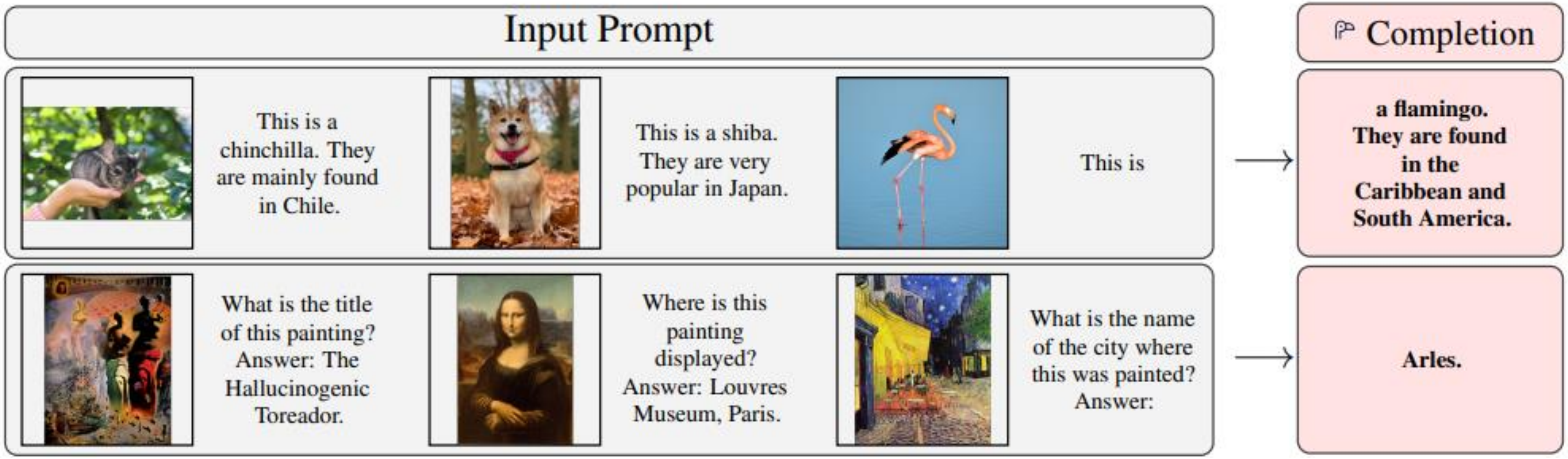

Singh et al '22

# **Few-Shot** VLMs

The models we've talked about are either meant to

- Do zero-shot prediction, OR
- Be fine-tuned for a particular task

- What about **few-shot** (like in LLMs) for VLMs?

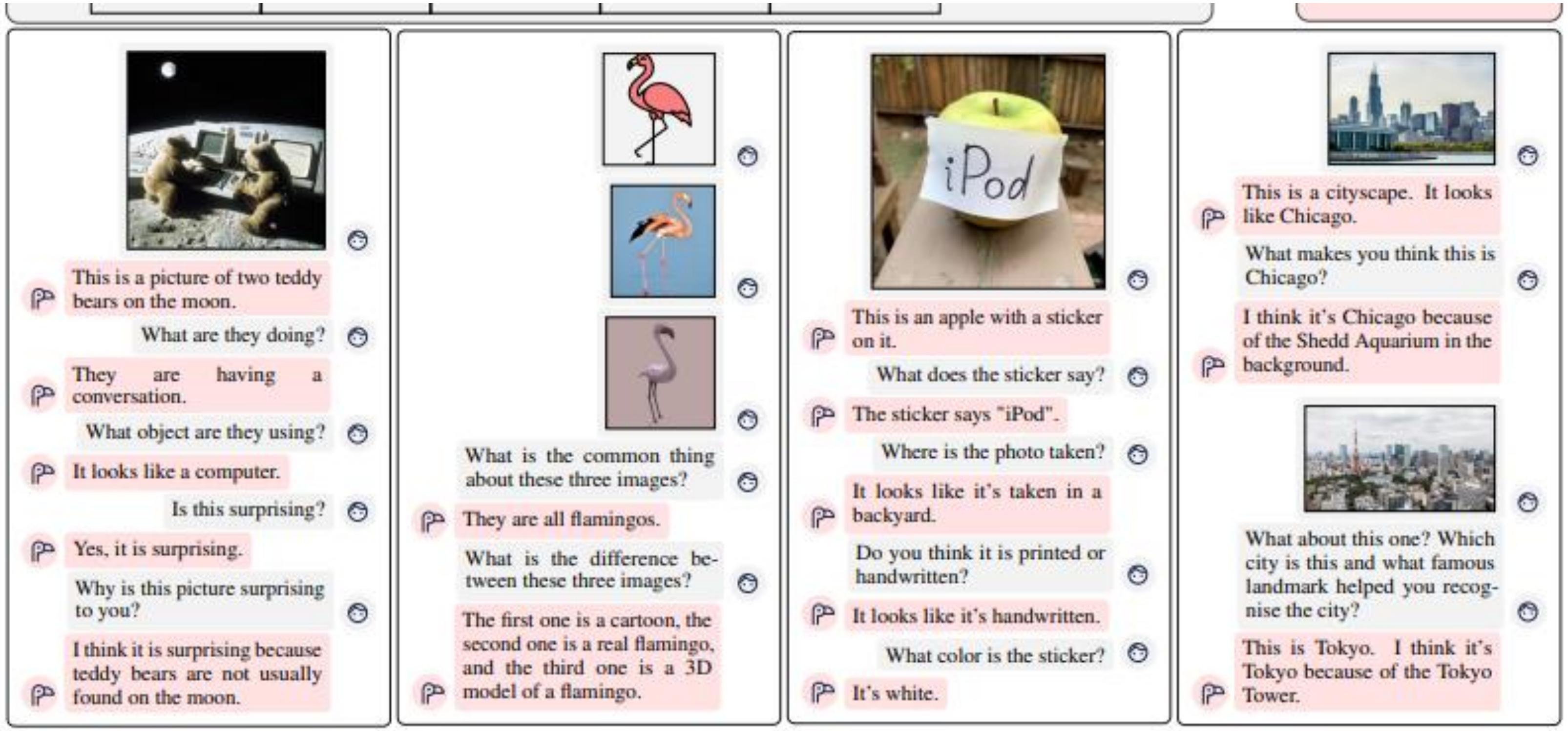



Alayrac et al '22

# Few-Shot VLMs: **Flamingo**

Flamingo: 80B parameter model (based on an LLM)

- Multi-image!
- More complex interleaved architecture

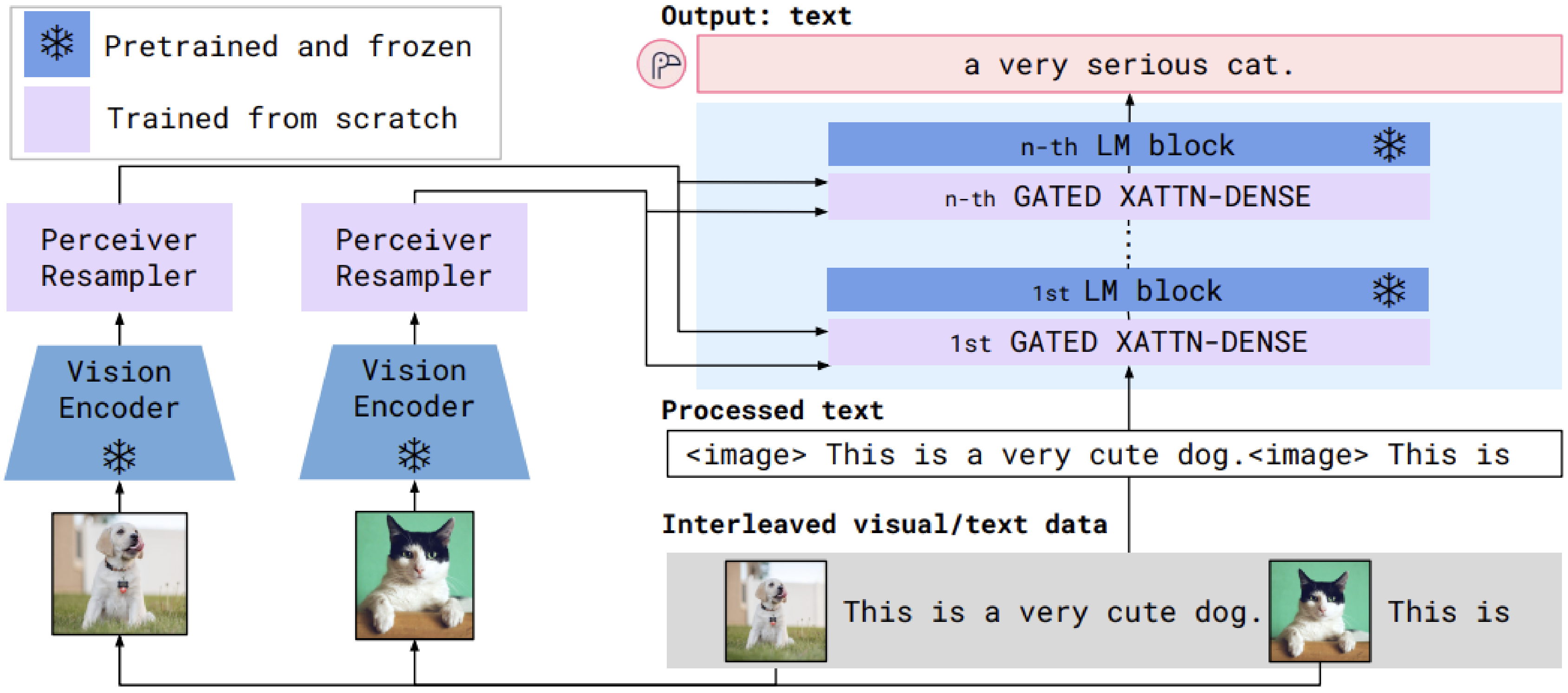



Alayrac et al '22

# Few-Shot VLMs: **Flamingo**

Flamingo: 80B parameter model (based on an LLM)

- Multi-image!
- More complex interleaved architecture



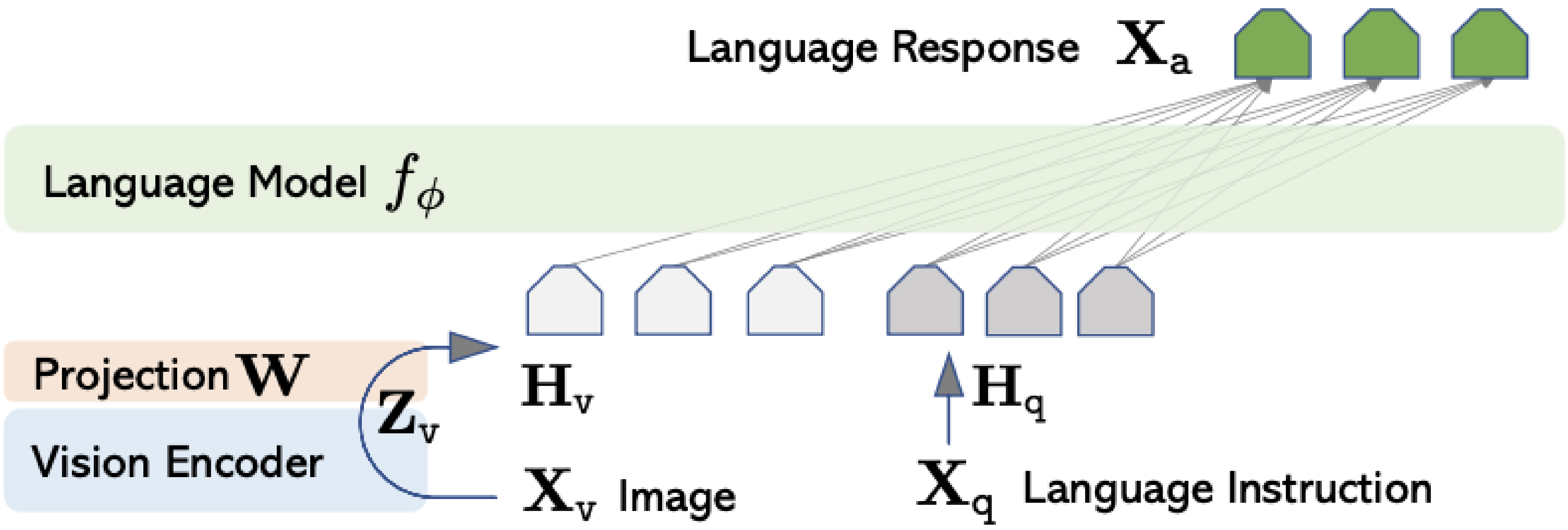

Alayrac et al '22

# Visual Instructions: **Llava**

Visual instruction tuning

**Goal**: achieve close to GPT-4's multimodal performance

- Simple, scalable architecture:




Liu et al '23

# Visual Instructions: **Llava**

## Example conversation

**Note**: challenging out-of-distribution example

Model designed to handle multi-round conversation

Source: https://www.barnorama.com/wp-content/uploads/2016/12/03-Confusing-Pictures.jpg

| | |
|---|---|
| User | What is unusual about this image? |
| LLaVA | The unusual aspect of this image is a man ironing clothes on the back of a minivan or van. This is not a typical place to perform this activity, as one would usually iron clothes in a more stationary and safe location, such as a home, using a regular ironing board. The scene depicted in the image is peculiar as it involves a makeshift ironing setup on a vehicle, which can be both unsafe and unconventional. Additionally, it is not clear how the man is able to maintain balance and stability while ironing clothes in such an unstable environment. |
| | [*Start a new conversation, and clear the history*] |
| User | What's happening in the scene? |
| LLaVA | The image depicts a man standing on top of a yellow SUV in a busy city street. He is holding a portable ladder, seemingly ironing clothes while standing on the vehicle. Around the scene, there are other cars, a traffic light, and a fire hydrant. Additionally, there are two other people in the vicinity, one close to the man on the car and another further away. The man on the ladder appears to be attracting attention due to his unconventional choice of ironing his clothes on top of a moving car. The city street around him is bustling with activity, adding to the unique nature of the scene. |
| User | What is unusual about this image? |
| GPT-4 [36] | The unusual thing about this image is that a man is ironing clothes on an ironing board attached to the roof of a moving taxi. |
| User | What is unusual about this image? |
| BLIP-2 | a man is sitting on the back of a yellow cab |
| User | What is unusual about this image? |
| OpenFlamingo | The man is drying his clothes on the hood of his car. |

Liu et al '23

# Closed-source Multimodal Models

Now, back to closed-source models.

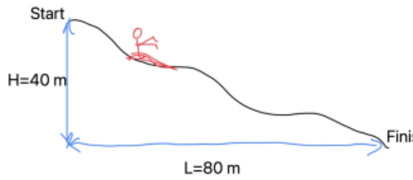
Similar capabilities, and can be more powerful because they are trained on all modalities **jointly**.

Downside: **expensive**!

**Prompt**

Here's a solution to a Physics problem by a student.

23. A skier slides down the frictionless slope as shown. What is the skier's speed at the bottom?




Solution:

The conservation of energy means that the energy of the skier is same at the beginning and at the end.

Start: $E = mgL$

End: $E = \frac{1}{2} mv^2$

$\Rightarrow mgL = \frac{1}{2} mv^2$

$\Rightarrow v = \sqrt{2gL} = \sqrt{2 \times 9.81 \times 80} = 39.6\ m.s^{-1}$

Try to reason about the question step by step.

Did the student get the correct answer? If the solution is wrong, please explain what is wrong and solve the problem.

Make sure to use LaTeX for math and round off the final answer to two decimal places.

**Response**

The student did not get the correct answer. The student made a mistake in the calculation of the potential energy at the start of the slope. The potential energy at the start of the slope is given by $mgh$, where $m$ is the mass of the skier, $g$ is the acceleration due to gravity, and $h$ is the height of the slope. The student used the length of the slope, $L$, instead of the height, $H$, in the calculation of the potential energy.

The correct solution is:

The conservation of energy means that the total energy of the skier is the same at the beginning and at the end of the slope.

At the start of the slope, the skier has only potential energy, which is given by $mgh$.

At the end of the slope, the skier has only kinetic energy, which is given by $\frac{1}{2} mv^2$

Therefore, we can write:

$$mgh = \frac{1}{2} mv^2$$

Canceling out the mass $m$ on both sides, we get:

$$gh = \frac{1}{2} v^2$$

Solving for $v$, we get:

$$v = \sqrt{2gh}$$

Substituting the given values, we get:

$$v = \sqrt{2 \times 9.81 \times 40} = 28.01 \text{ m/s}$$

Therefore, the skier's speed at the bottom of the slope is 28.01 m/s.

Gemini Team Google:

# Thank You!