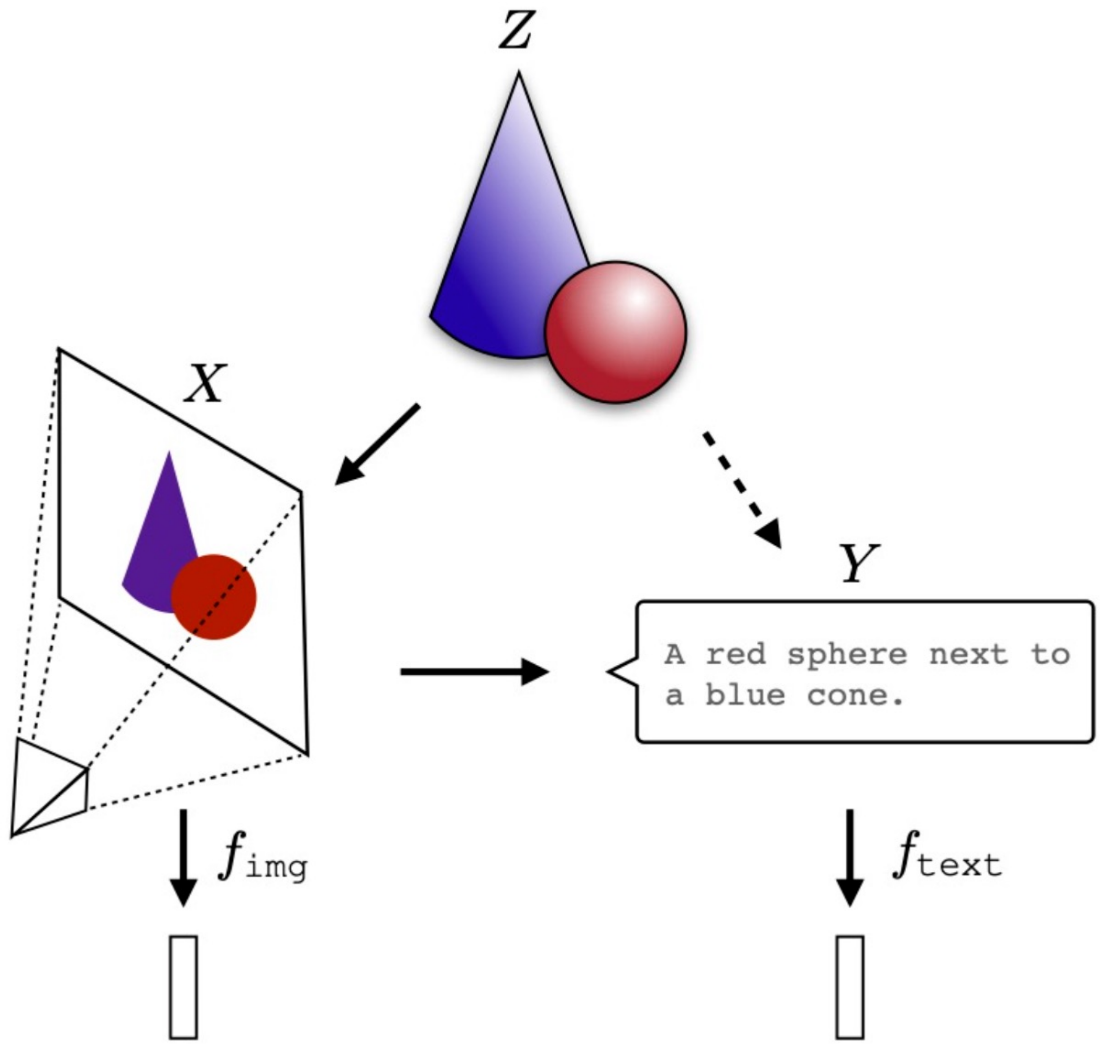
Vision and Language

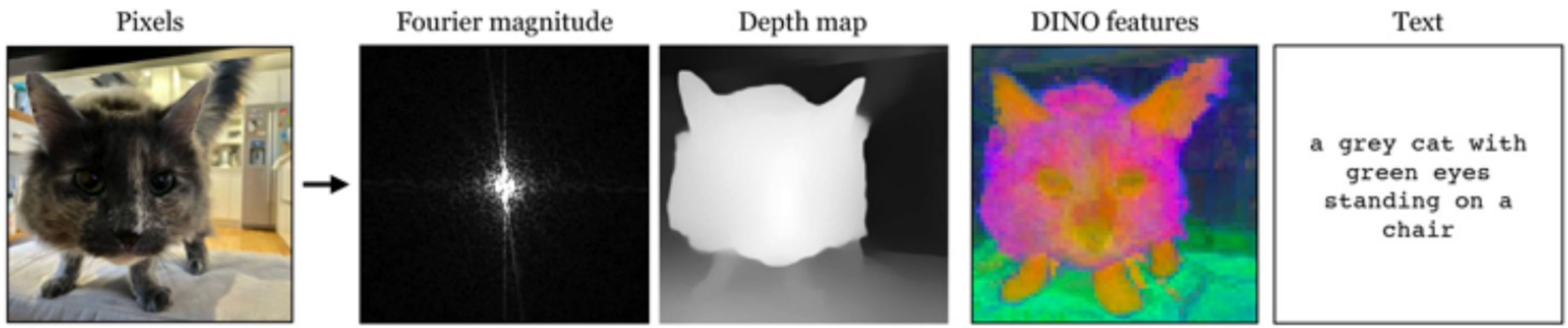
Jathushan Rajasegaran

14/04/2025



Z
X
Y
A red sphere next to a blue cone.
$f_{\texttt{img}}$
$f_{\texttt{text}}$

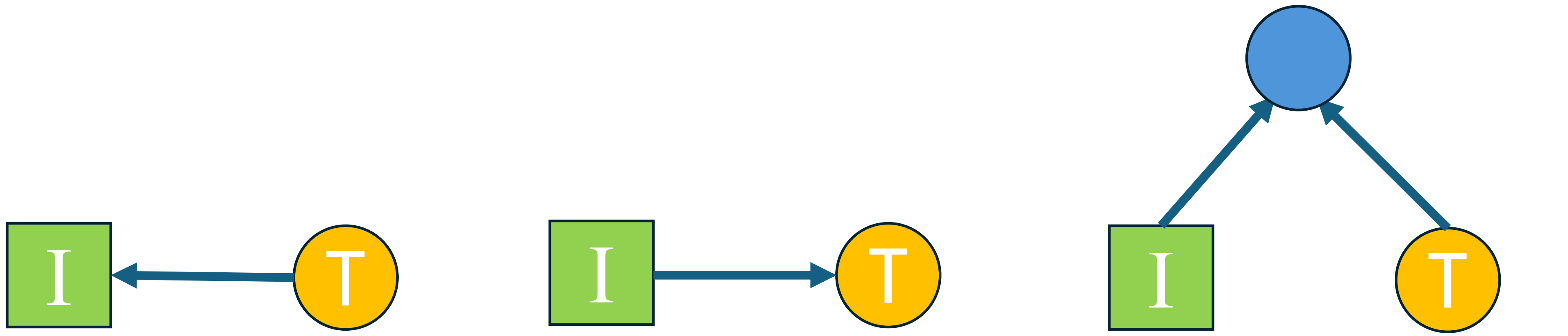
Text as a Visual Representation



Vision and Language

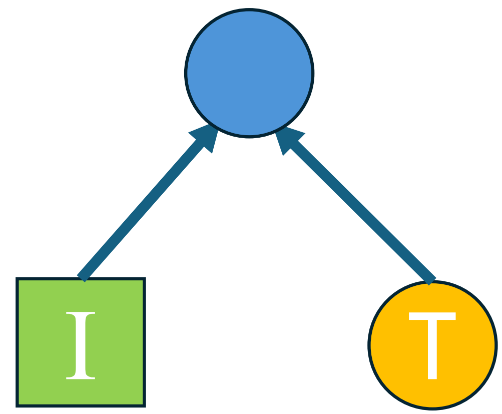
Vision and Language

Vision and Language

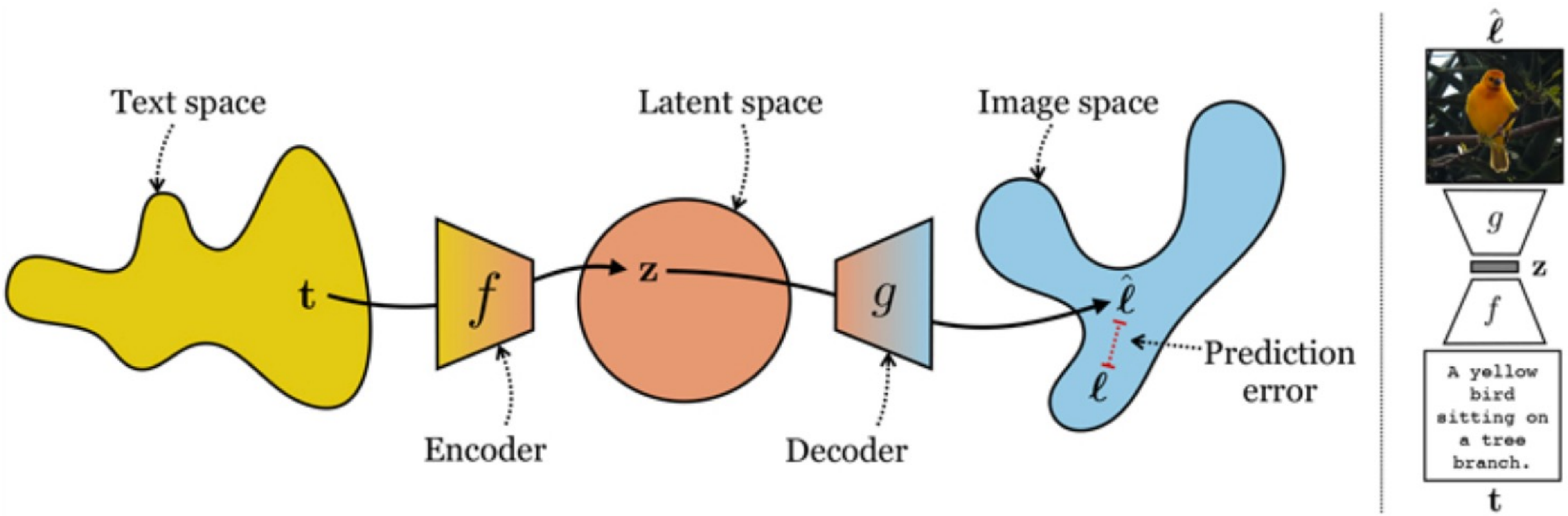


Vision and Language

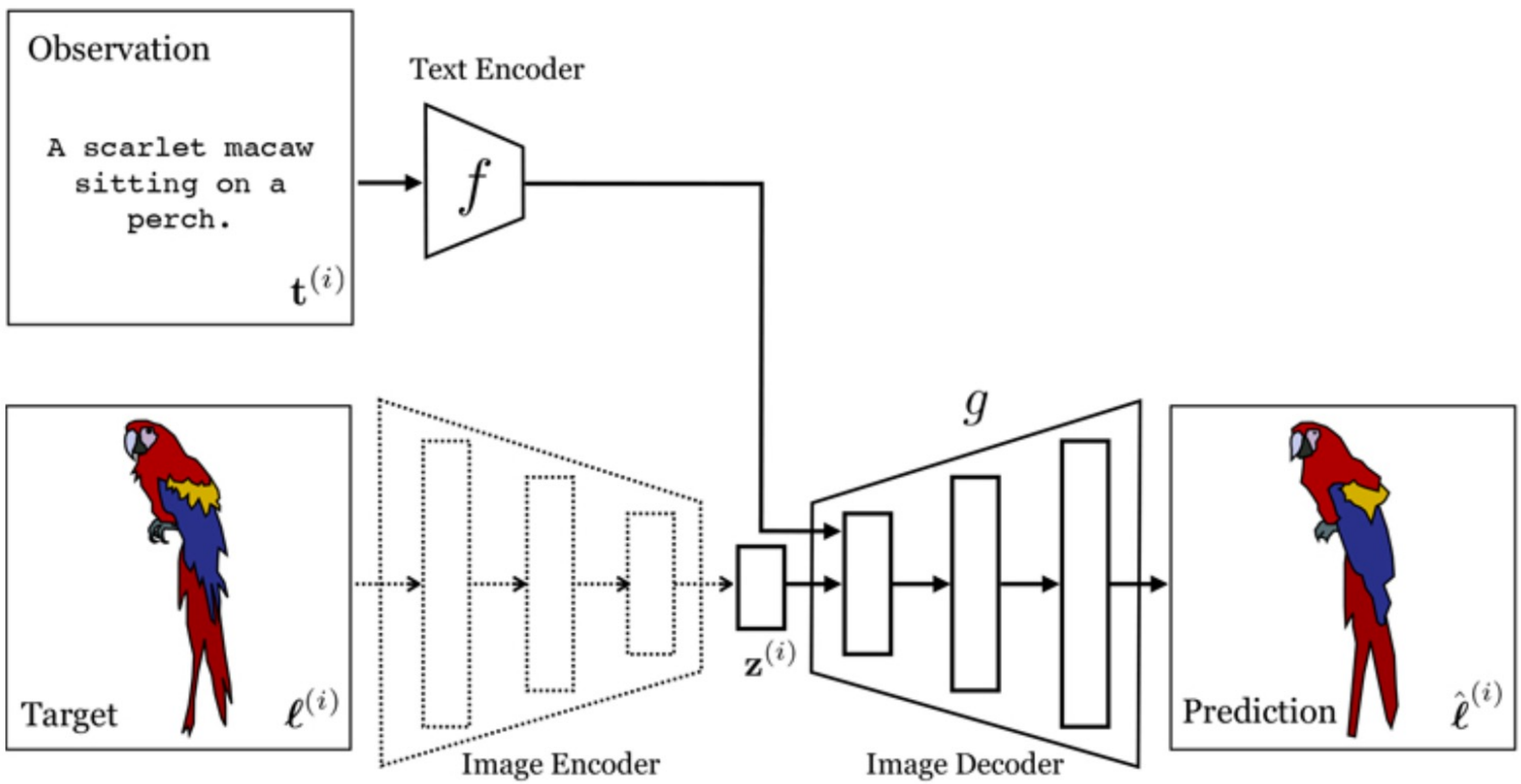
Stable Diffusion,
Dalle-1



Text-to-Image

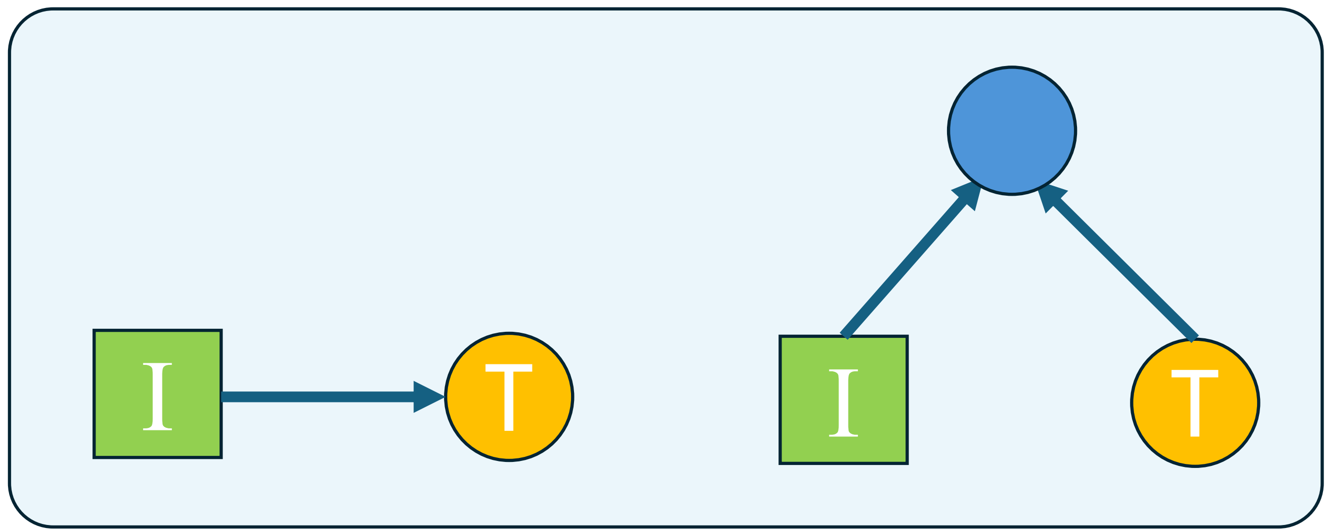


Text-to-Image



Vision and Language

Stable Diffusion,
Dalle-1



Learning Visual Representations from Language Supervision

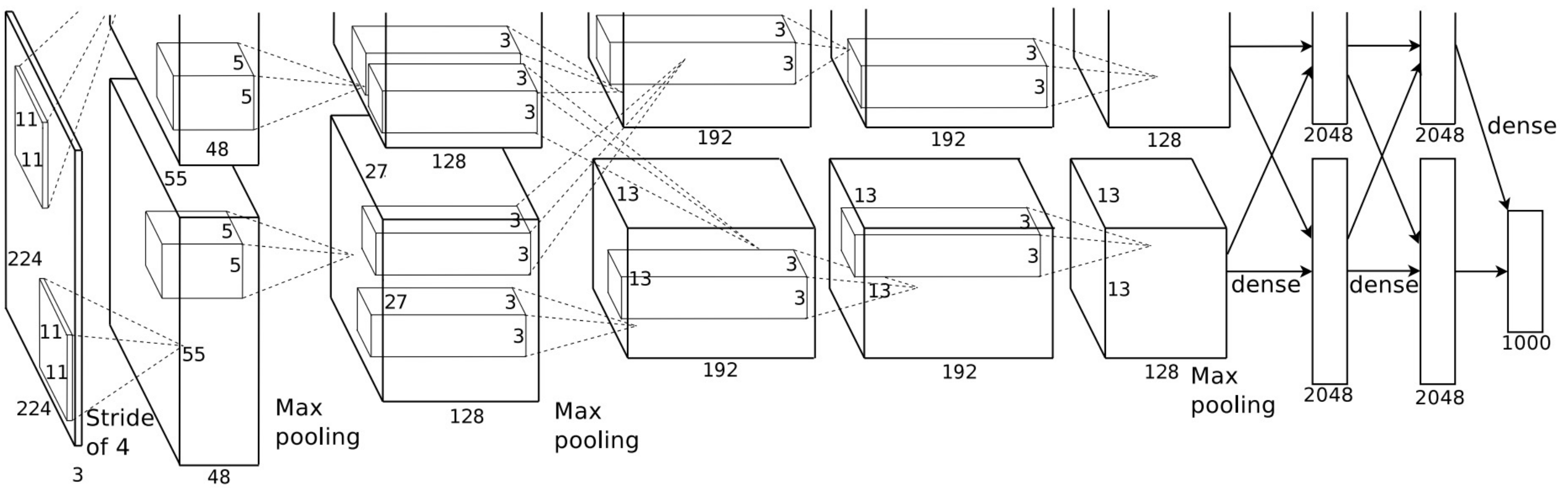
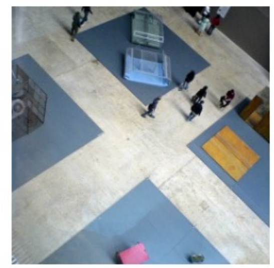


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Learning Visual Representations from Language Supervision

the veranda hotel portixol palma

plane approaching zrh avro regional jet rj

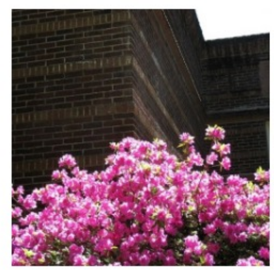


not as impressive as embankment that s for sure

student housing by lungaard tranberg architects in copenhagen click here to see where this photo was taken

Unusual is commonplace

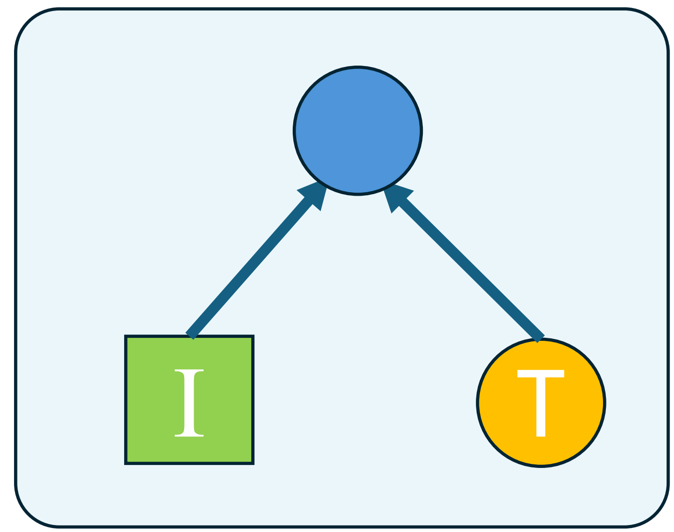
article in the local paper about all the unusual things found at otto s home

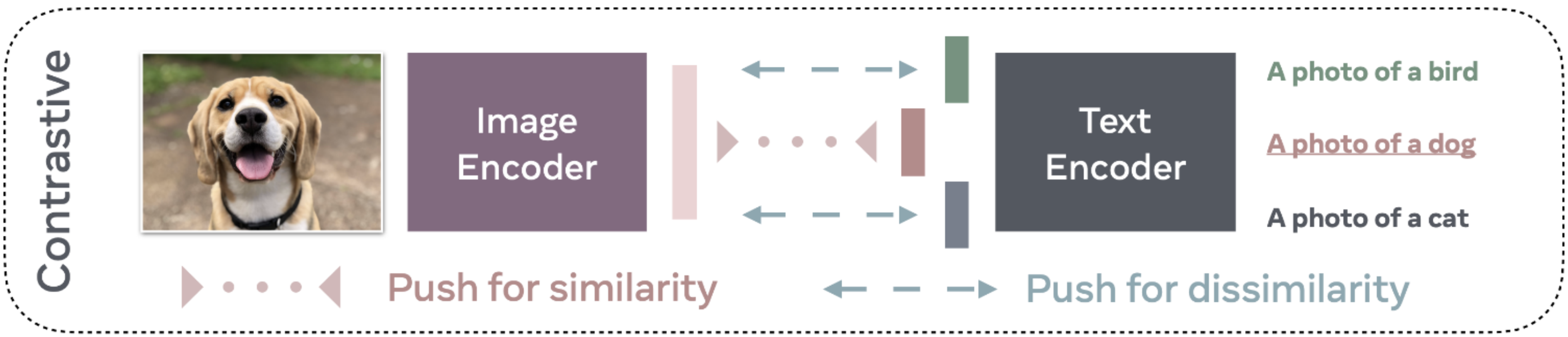


this was another one with my old digital camera i like the way it looks for some things though slow and lower resolution than new cameras another problem is that it s a bit of a brick to carry and is a pain unless you re carrying a bag with some room it s nearly x x and weighs ounces new one is x x and weighs ounces i underexposed this one a bit did exposure bracketing script underexposure on that camera looks melty yummy gold kodak film like

Vision and Language

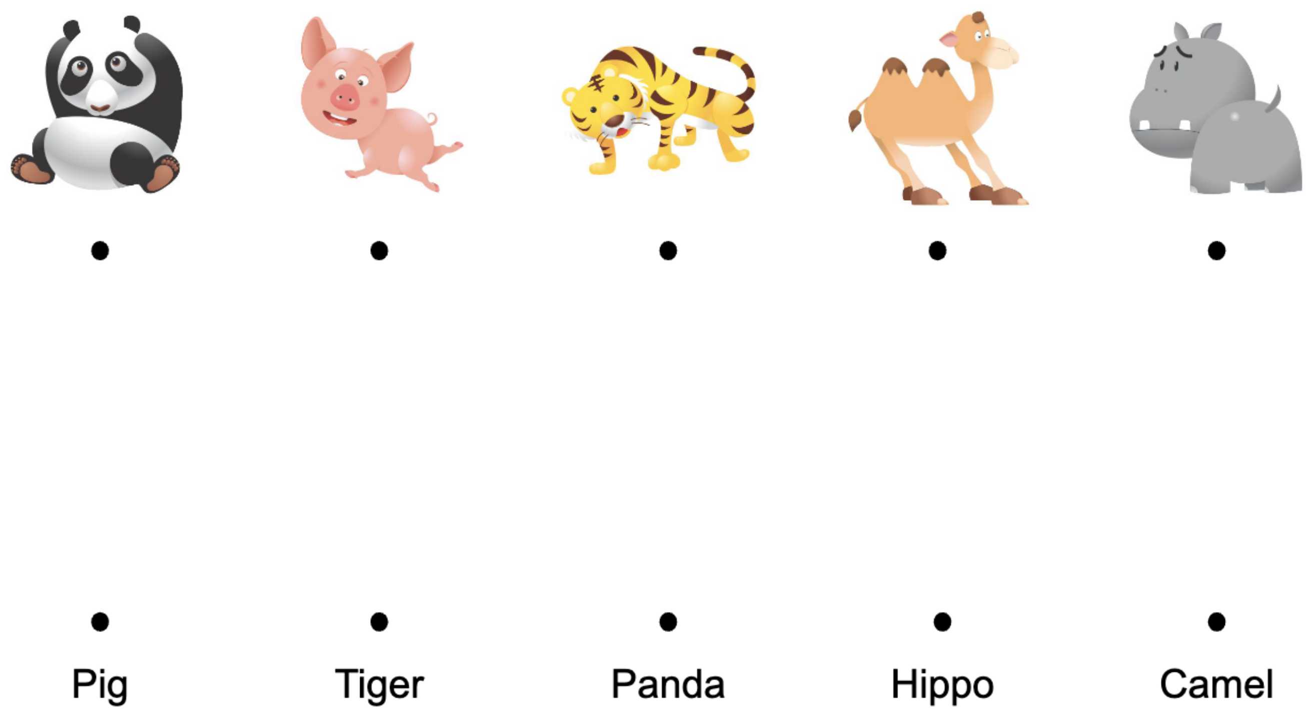
Stable Diffusion,
Dalle-1



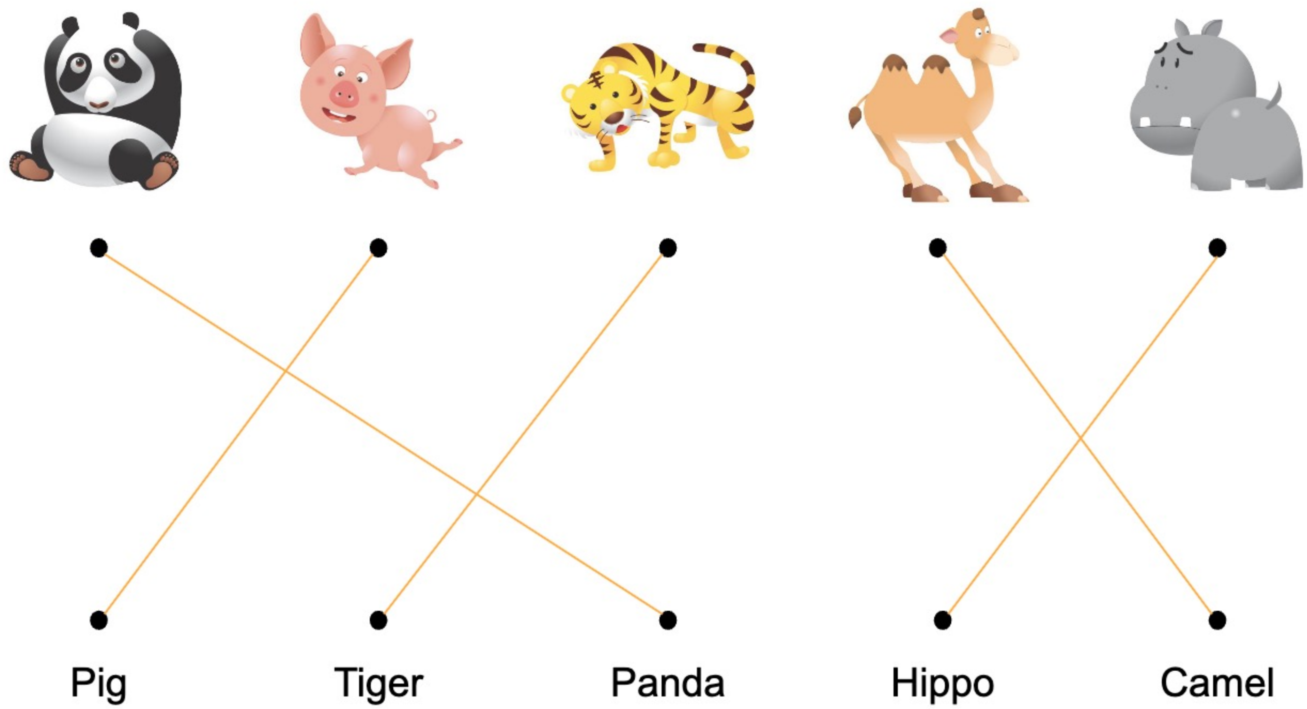

Contrastive
Image Encoder
Text Encoder
A photo of a bird
A photo of a dog
A photo of a cat
Push for similarity
Push for dissimilarity


CLIP
158
coffee isn't helping
get the jumper cables

Contrastive learning



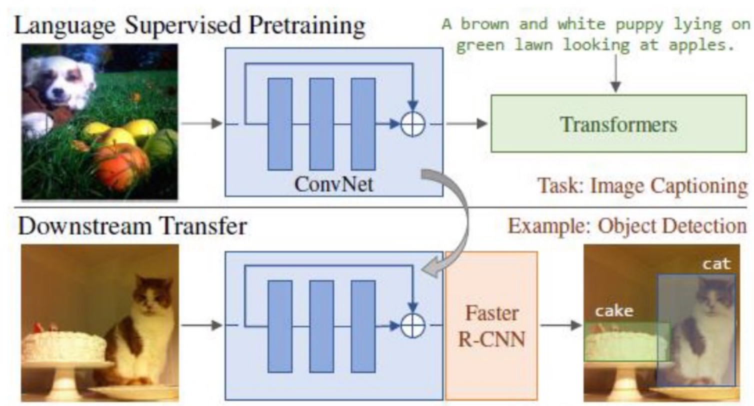
Contrastive learning



Related Works

- Visual N-Grams (2017)
  - First zero-shot transfer methodology
  - CNN to predict relevant words and n-grams (adjacent order) from images
  - “Unsupervised” training on 30 mil Flickr images (used comments)

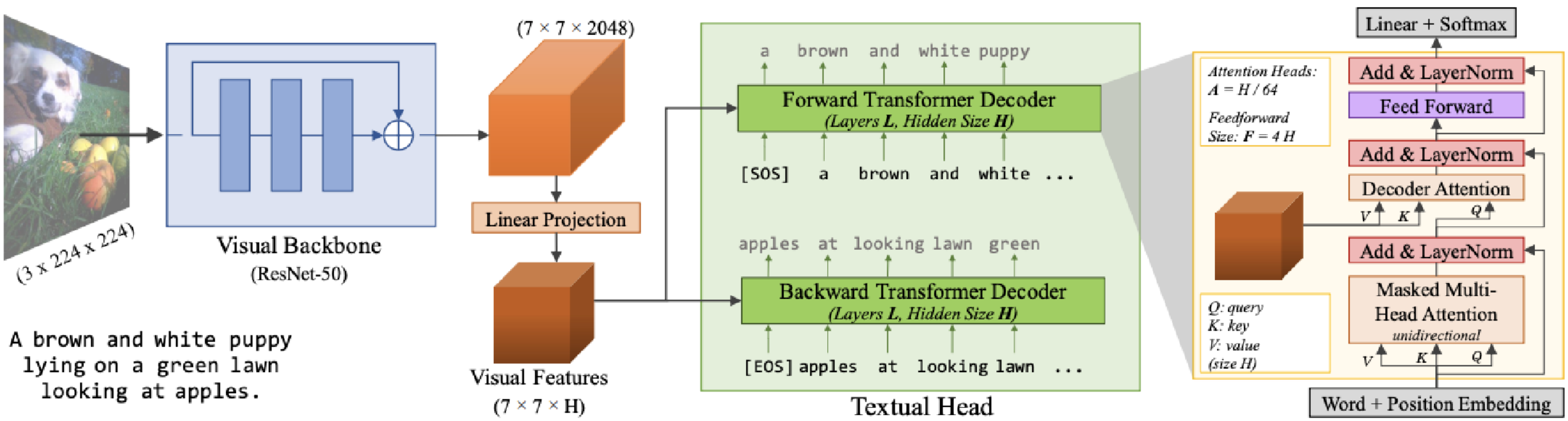
- VirTex (2020)
  - Transformer-based image captioning
  - CNN encoder + transformer decoder architecture
  - Caption generation for images lead to richer classifications, requires nuanced understanding
  - Better at downstream tasks like segmentation and object detection



Georgia Tech.

Related Works

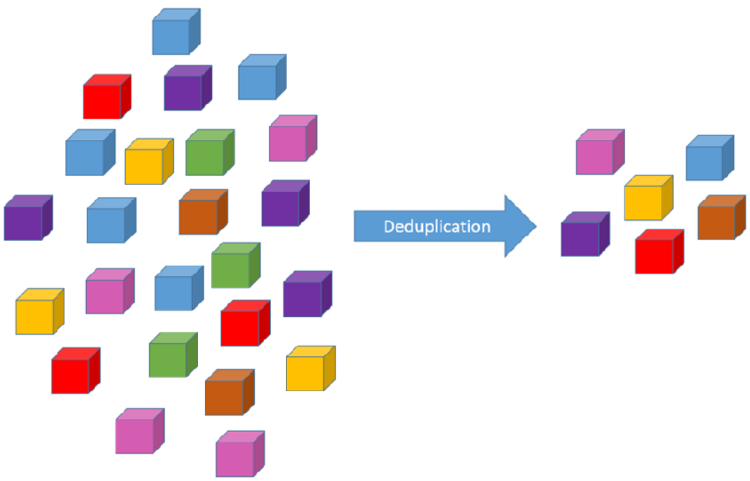
- VirTex continued (diagram was too good not to discuss)



Dual unidirectional Transformers for caption prediction bidirectional approach
Masked self-attention over captions and cross-attention with image features

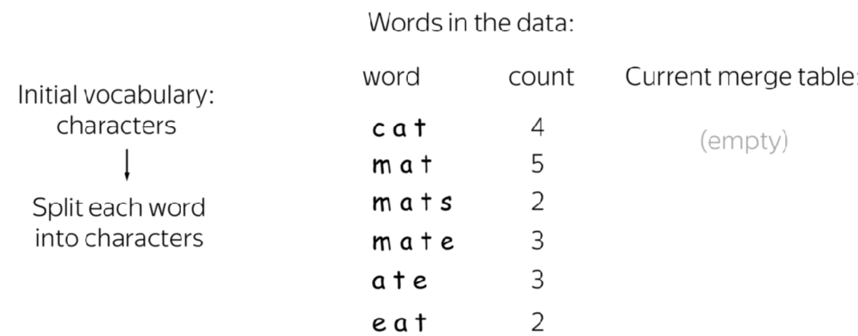
Approach (Data Collection)

- Raw web pairs aren't going to be perfect
  - Plenty of noise and even mismatches, abstract pairs
  - Either way → CLIP gets stronger with weird stuff
- CLIP filtering
  - 500,000 unique internet queries to cover all domains
  - Pulled in captions, descriptions, comments any kind of data paired with images
  - 1 query could produce max most relevant 20k image-text pairs, ensuring diversity
- De-duplication
  - Image text pairs underwent de-duplication which just ensures overlap is minimal
  - Each sample should ideally be unique
  - Also lowers overlap with benchmarking datasets, → real evaluation and generalization capabilities



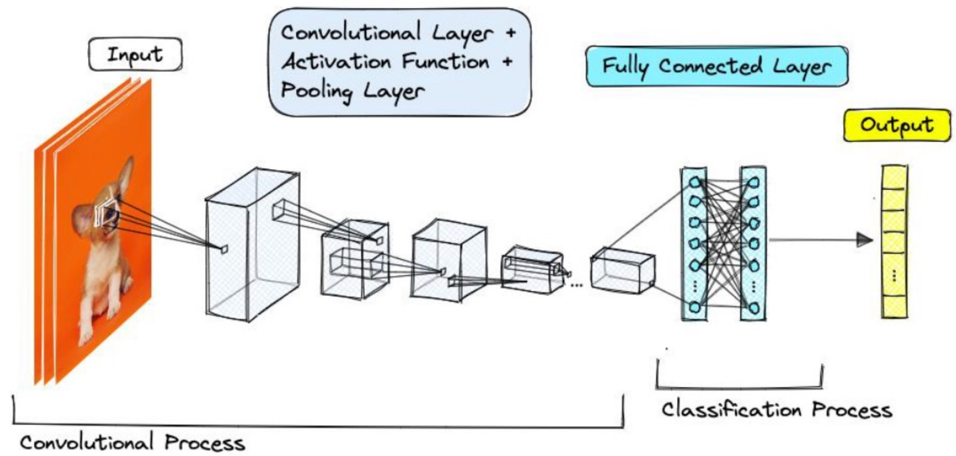
Approach (Tokenization)

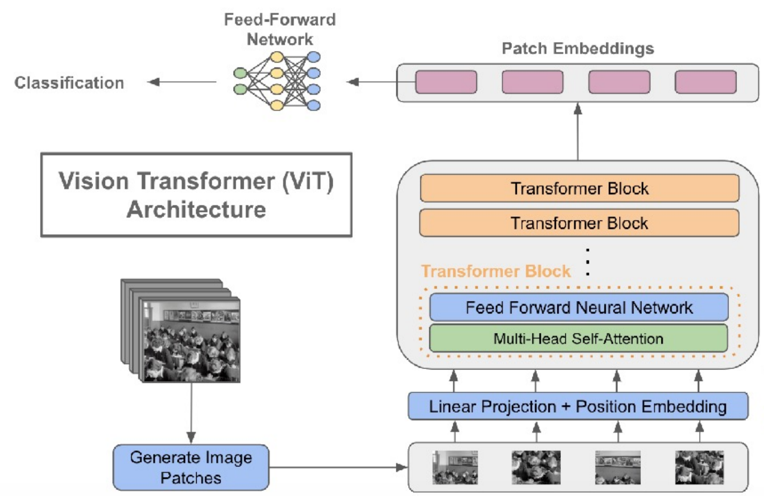
- Text Tokenizer: Byte Pair Encoding (BPE)
  - Relate each word in the text as character sequence
  - Words also get end of word tokens (e.g., ”fork" → ”f o r k </w>")
  - **Frequencies**: Count up common adjacency pairs
  - **Merging:** Merge common pairs, add to vocabulary
- **Why?**
  - Common words and subwords are tokenized well
  - Great for zero-shot tasks
- No non-linear projection
  - Other contrastive learning methods use a non-linear projection between the representation and embedded space



Approach (Image Encodings)

- ResNet encoder
  - CNN architecture, conv layers + pooling → feature vector
  - Linear layer for final embedding, L2 normed for ease of similarity
- ViT encoder
  - Patches over image, flattened and projected into embedding (like with text)
  - Positional encodings for those patches, multi-head self attention + feedforward neural nets are strong
  - A classification token is added onto the patch embeddings sequence, then normalized too

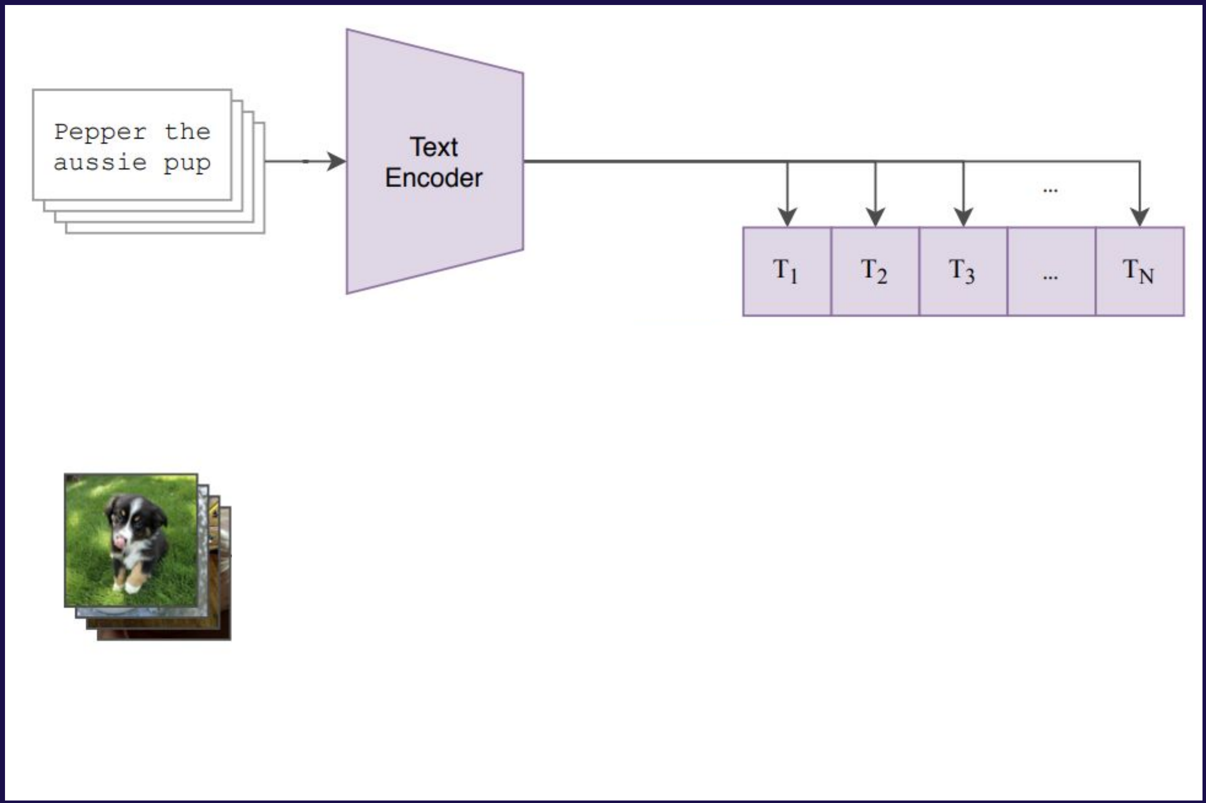




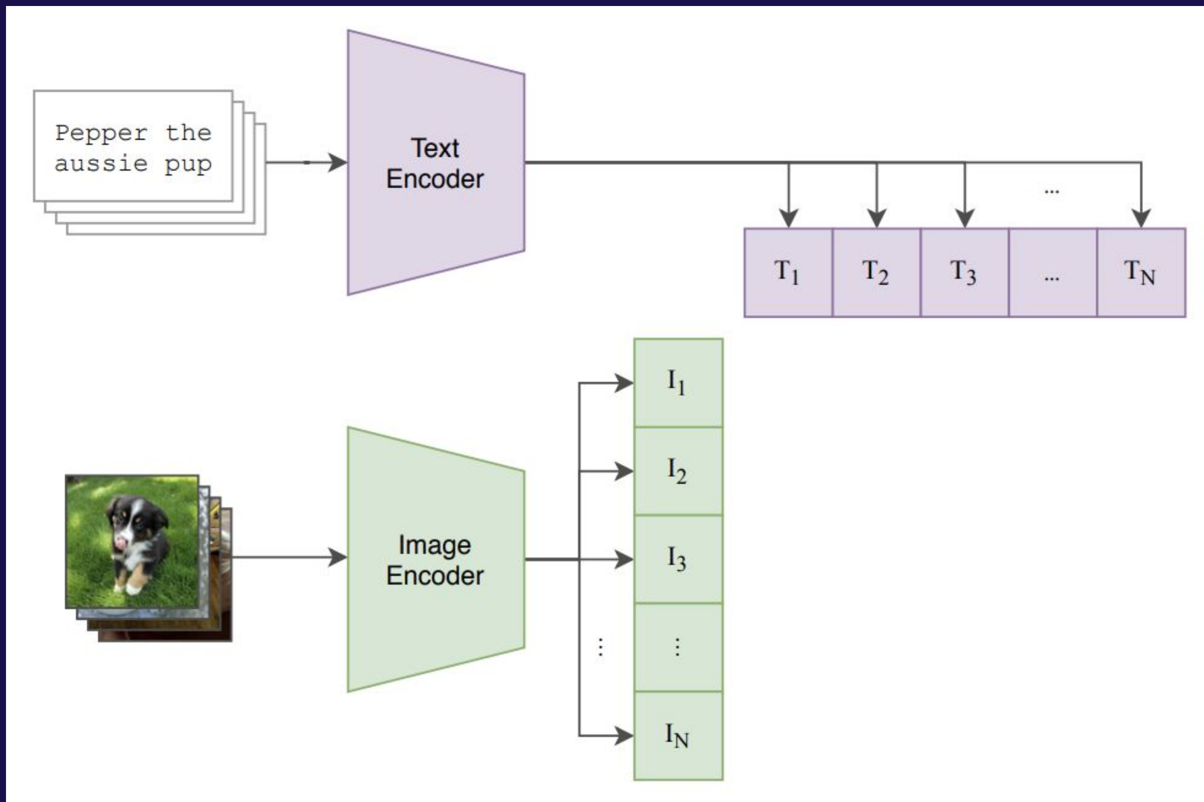
Georgia Tech

CLIP: Contrastive Language-Image Pre-training

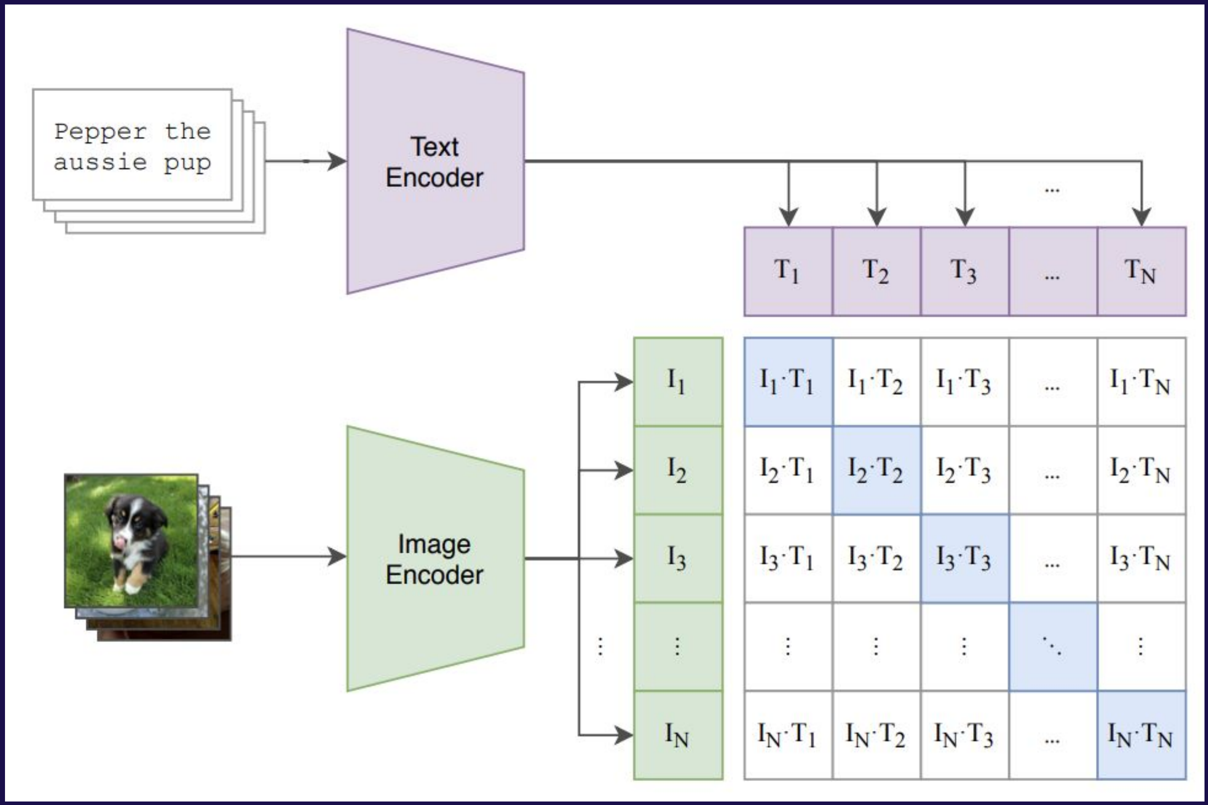
CLIP: Contrastive Language-Image Pre-training



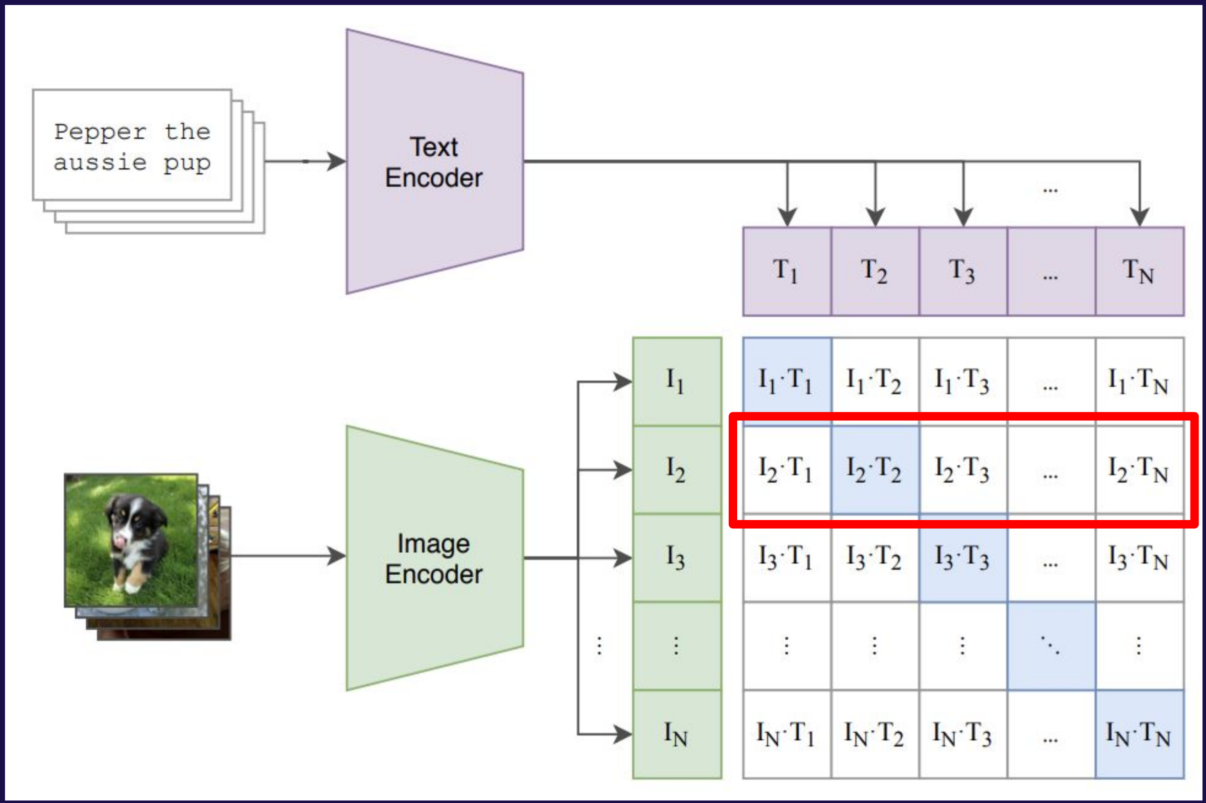
CLIP: Contrastive Language-Image Pre-training



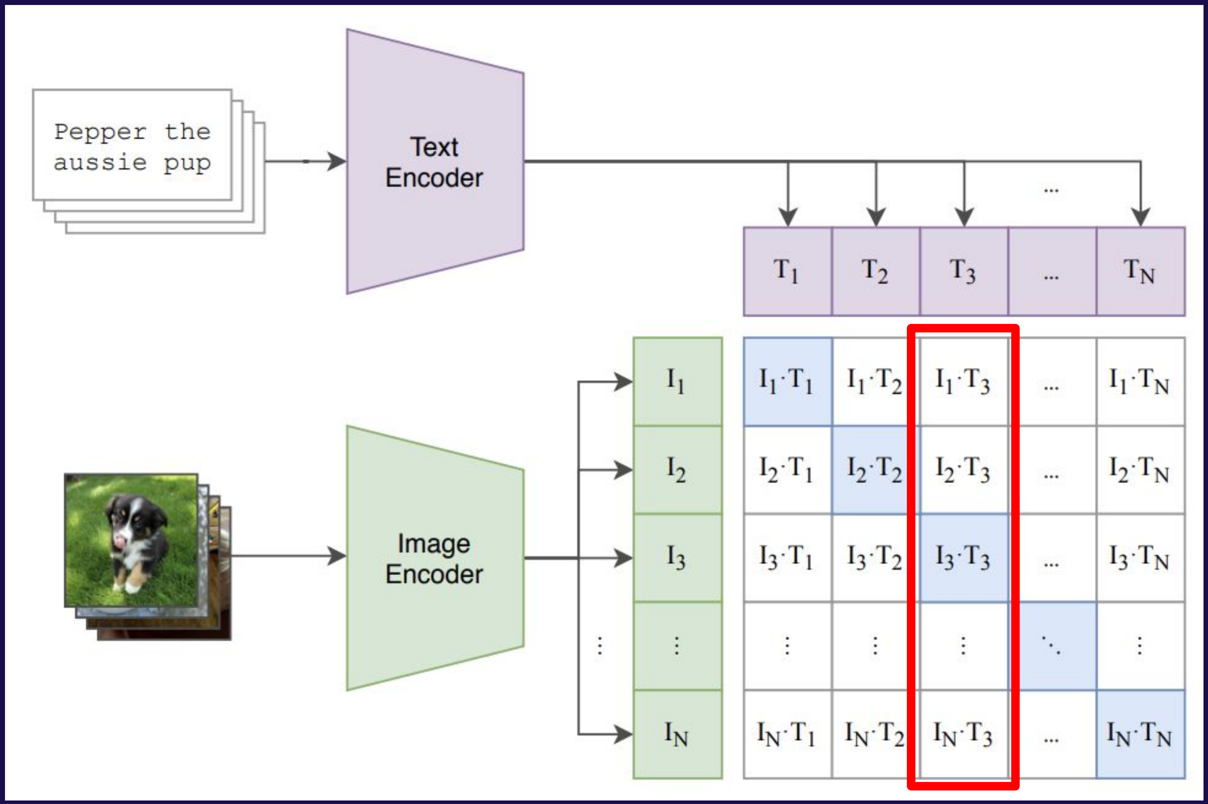
CLIP: Contrastive Language-Image Pre-training



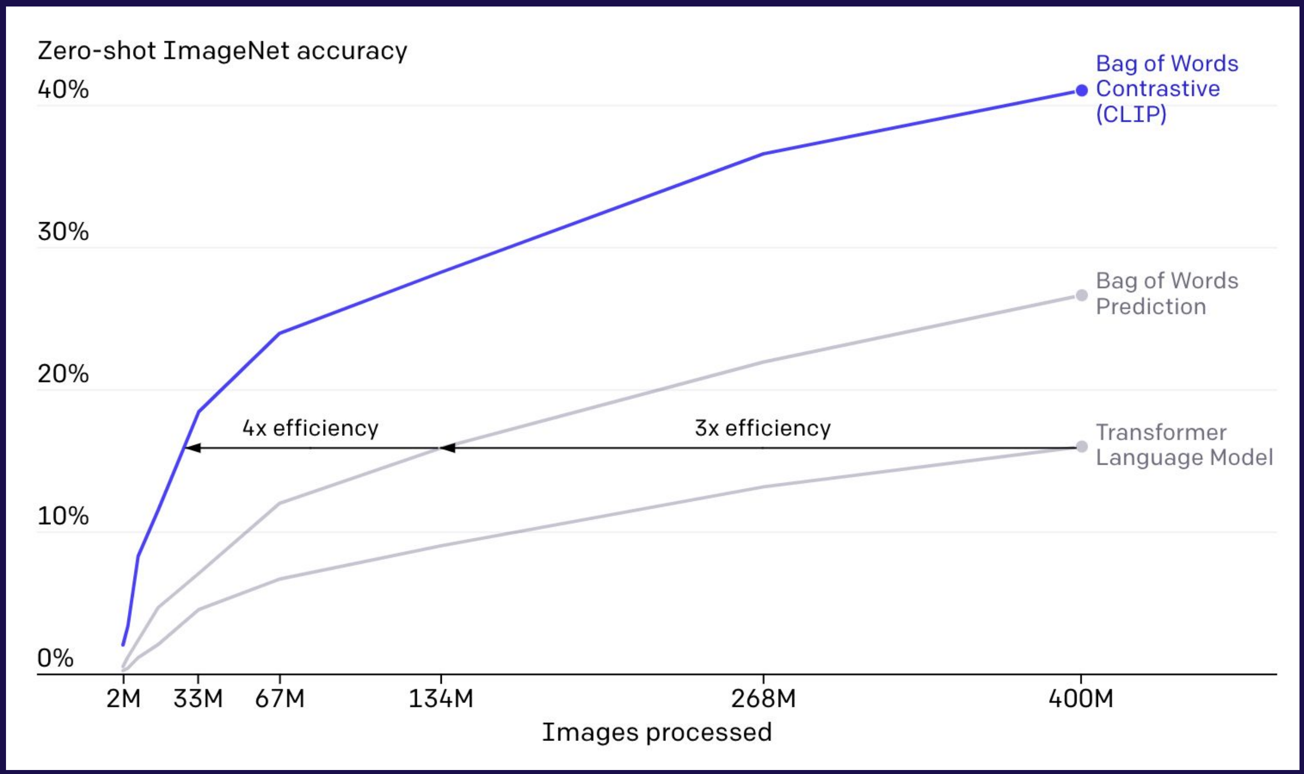
CLIP: Contrastive Language-Image Pre-training



CLIP: Contrastive Language-Image Pre-training



Why contrastive?



Some CLIP details

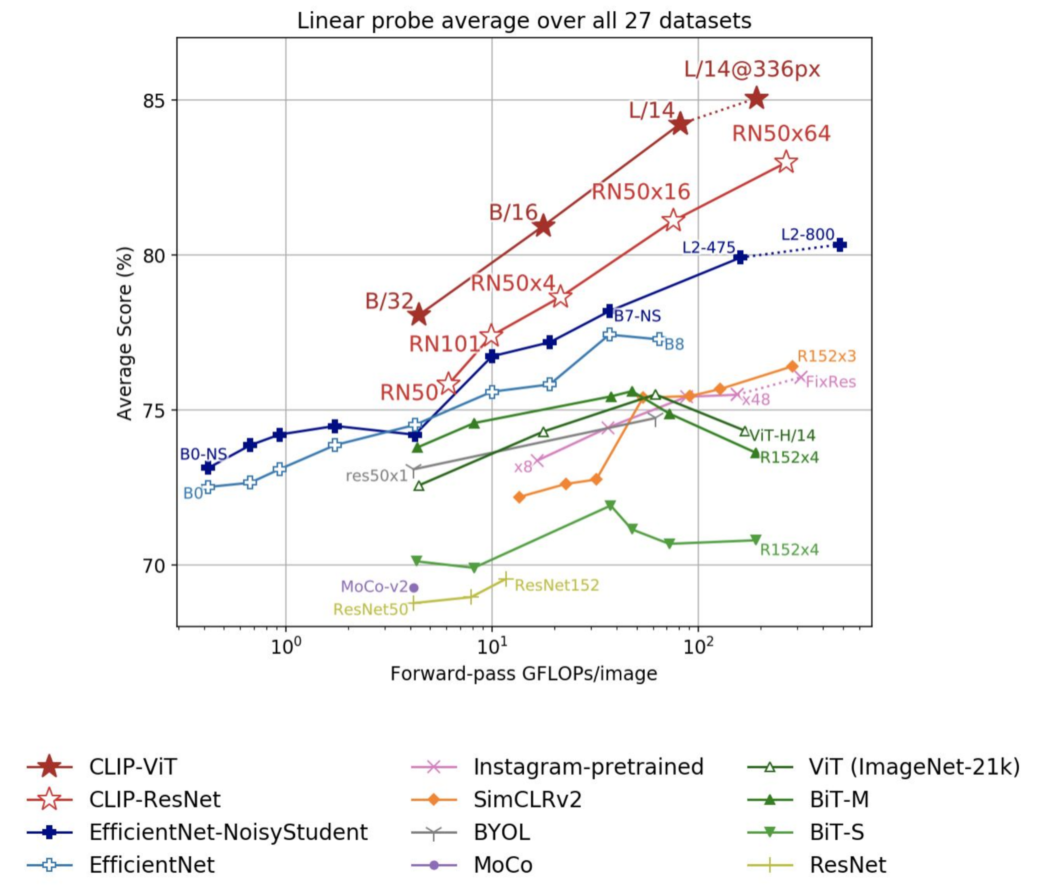
Training

- Trained on 400M image-text pairs from the internet
- Batch size of 32,768
- 32 epochs over the dataset
- Cosine learning rate decay

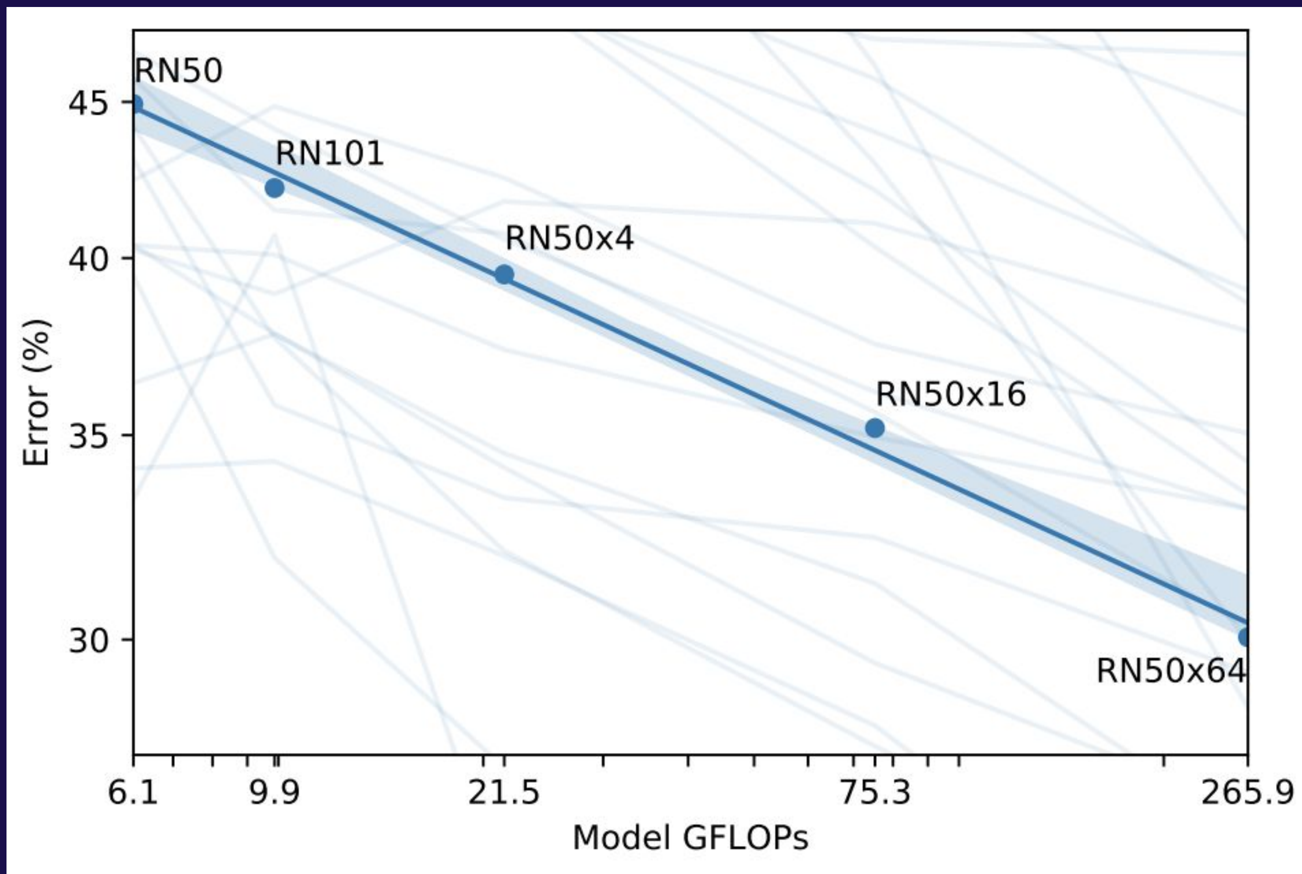
Architecture

- ResNet-based or ViT-based image encoder
- Transformer-based text encoder

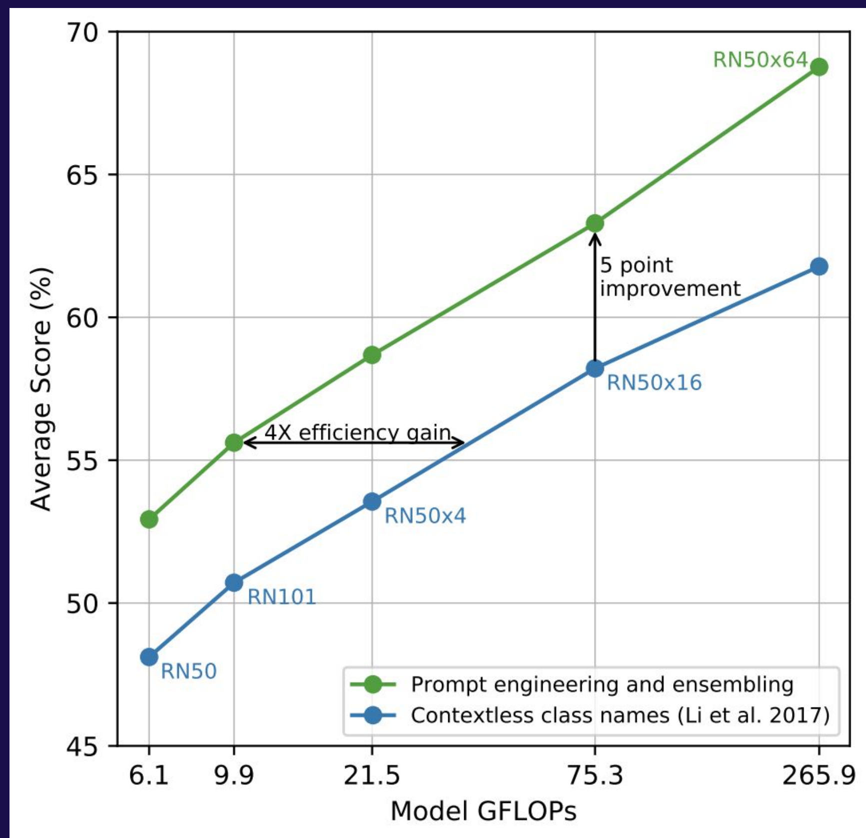
Linear probe performance vs SOTA vision models



Zero-shot performance vs model size



Prompt engineering

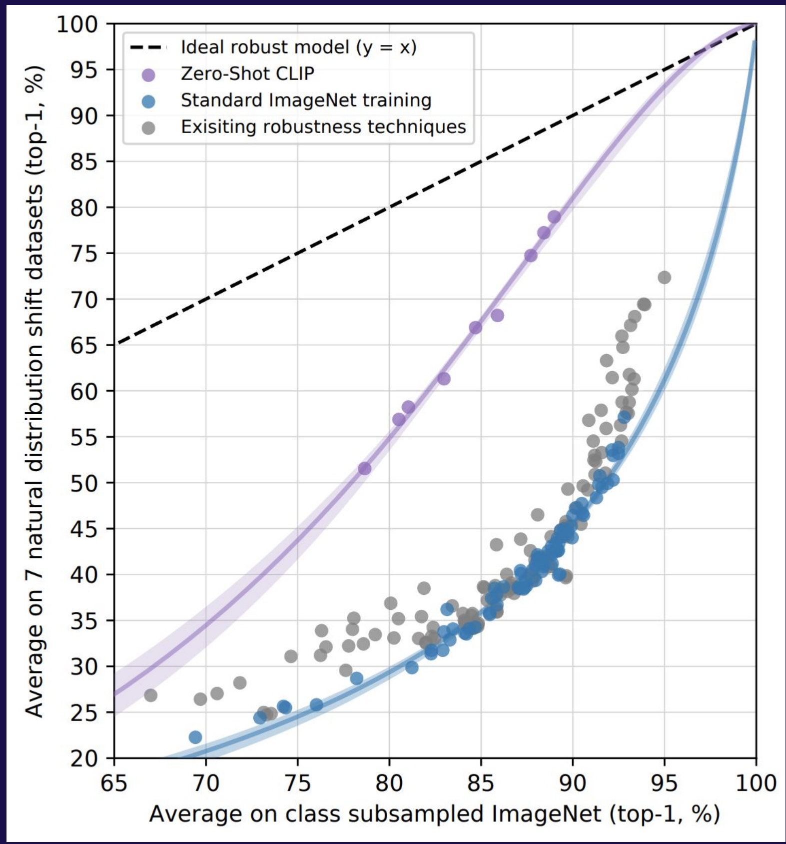


Robustness to natural distribution shift

Zero-Shot CLIP is much more robust!

7 ImageNet-like Datasets (Taori et al.)

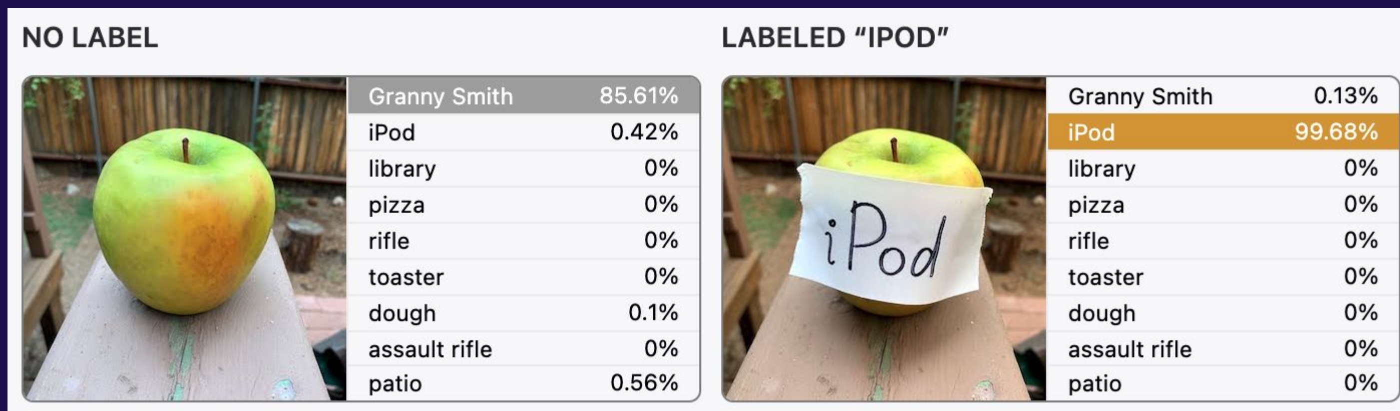
- ImageNetV2
- ImageNet-A
- ImageNet-R
- ImageNet Sketch
- ObjectNet
- ImageNet Vid
- Youtube-BB

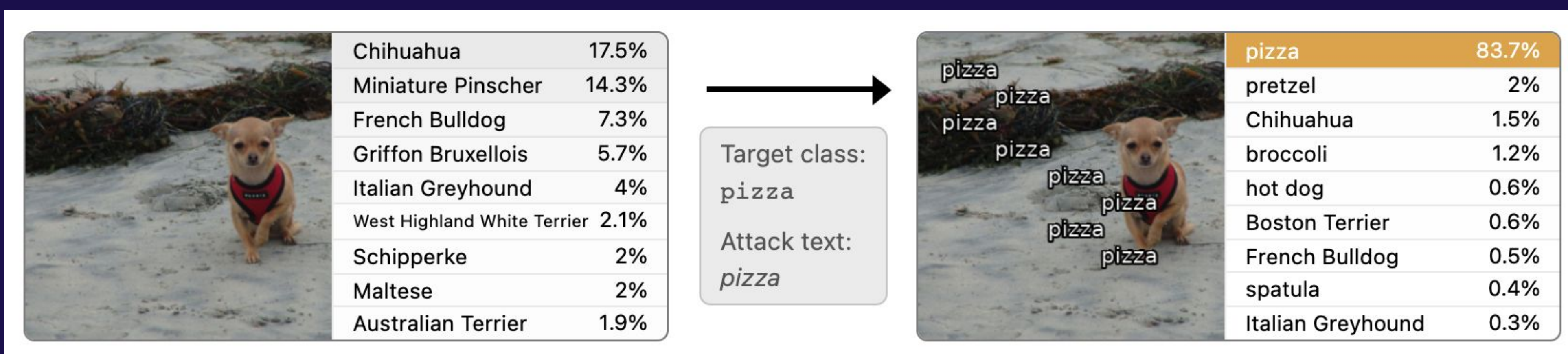


Limitations of CLIP

- Zero-shot performance is well below the SOTA
- Especially weak on abstract tasks such as counting
- Poor on out-of-distribution data such as MNIST
- Susceptible to adversarial attacks
- Dataset selection in the eval suite, use of large validation sets for prompt engineering
- Social biases

Typographic Attacks

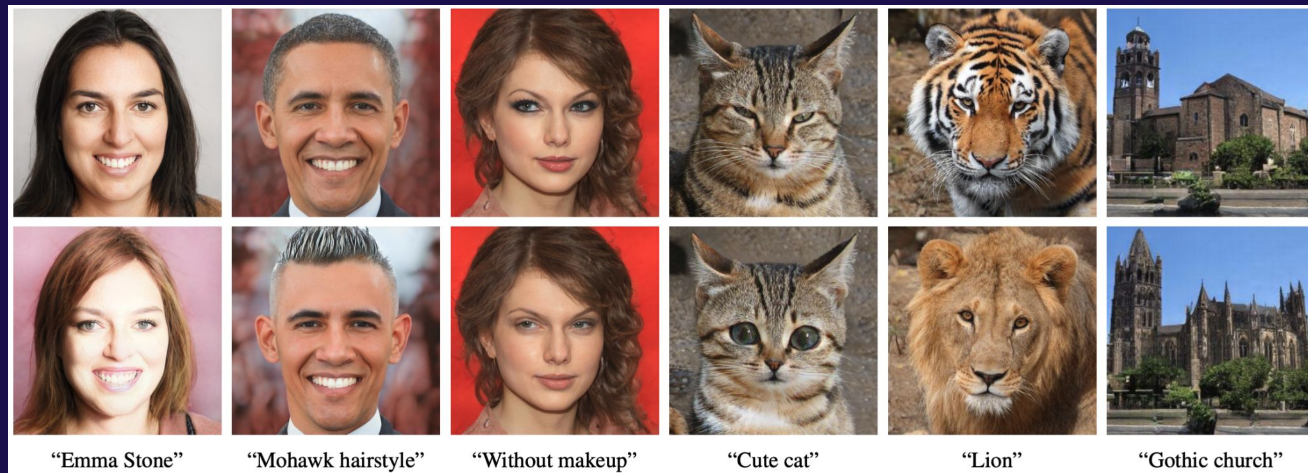




Applications of CLIP

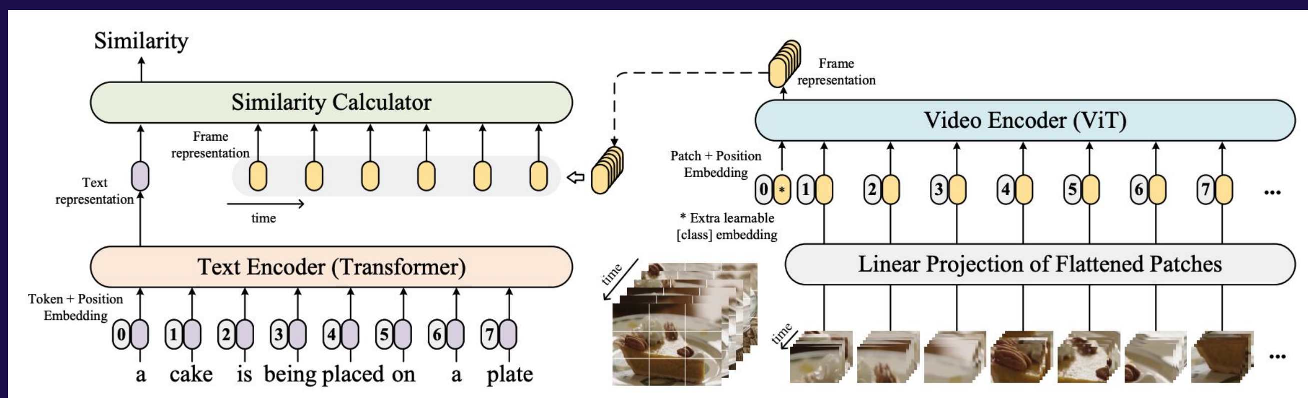
StyleCLIP
(Patashnik et al.)

Steering a GAN Using CLIP



CLIP4Clip
(Luo & Ji, et al.)

Video retrieval using
CLIP features



More text-based image generations using CLIP

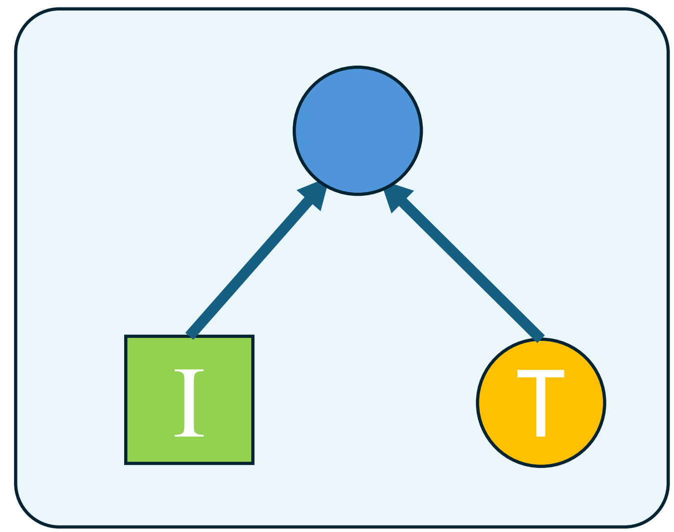
“A banquet hall”

“Geoffrey Hinton”

“Dogs playing poker”

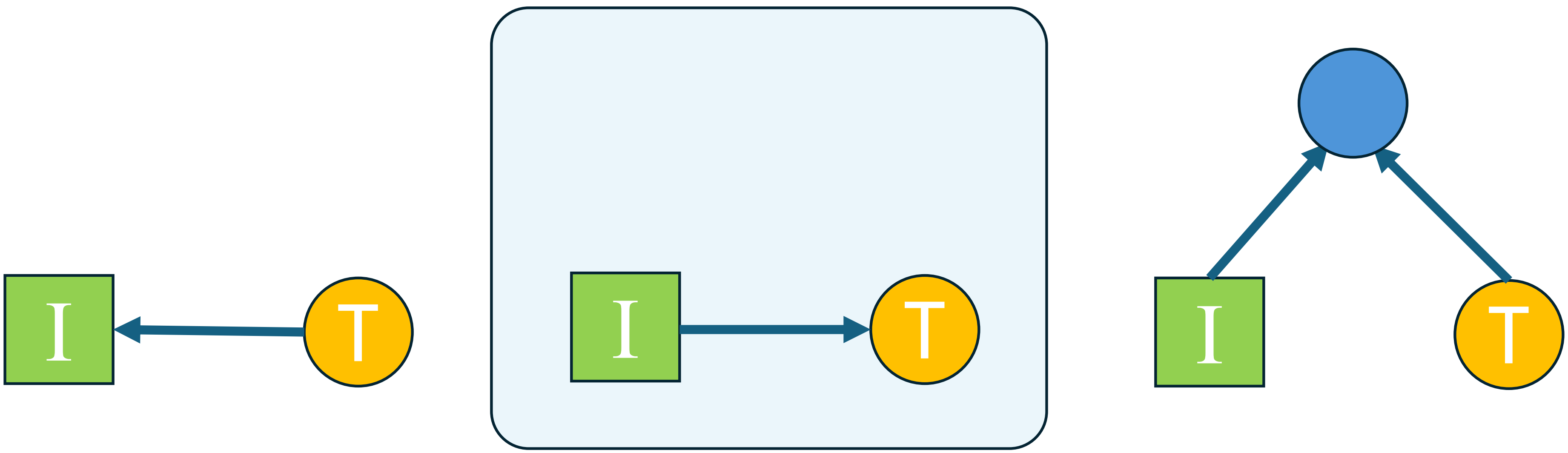
Vision and Language

Stable Diffusion,
Dalle-1



CLIP

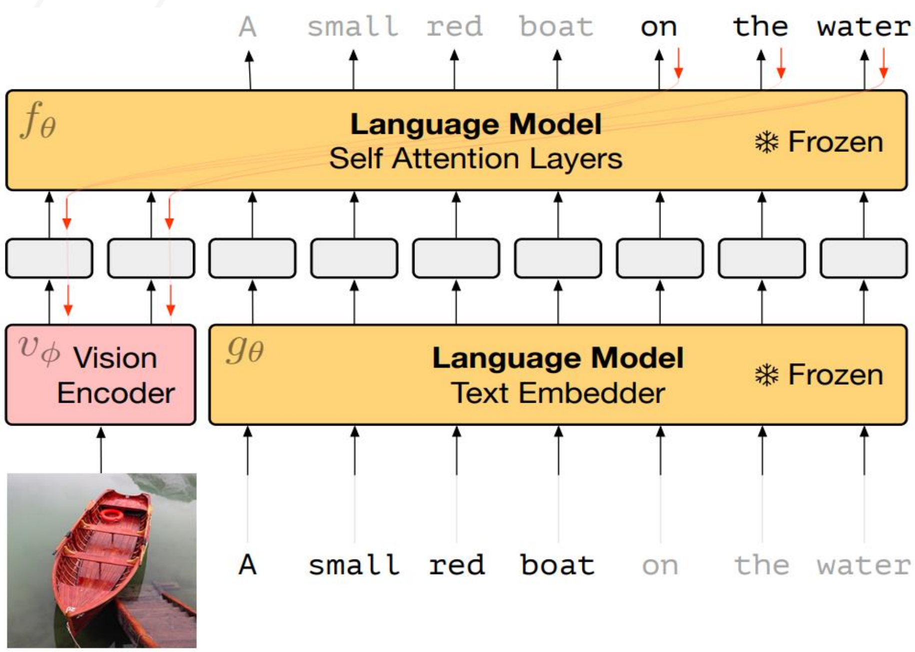
Vision and Language



Stable Diffusion,
Dalle-1

CLIP

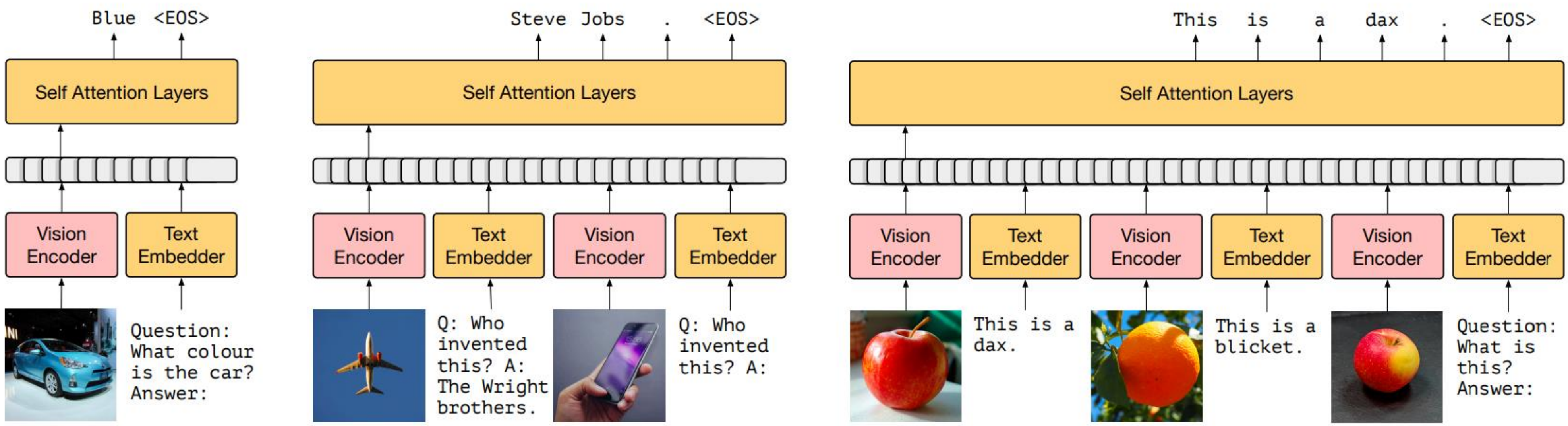
Frozen



A simple architecture : a completely frozen LLM, conversion of the image w/ Resnet into 2 tokens (~prefix tuning). Gradient flows through LLM

- fine-tuning θ hurts generalization (because the LM datasets size >> text/image coupled datasets)
- Modularity : plug-n-play any LLM !
- Proof on concept : small scale (7B model), but enough to show interesting properties for few-shot
- Training objective : for only one image ! But at inference multiple images supported (thanks to relative pos. enc.)

Example inference



(a) **0-shot VQA**

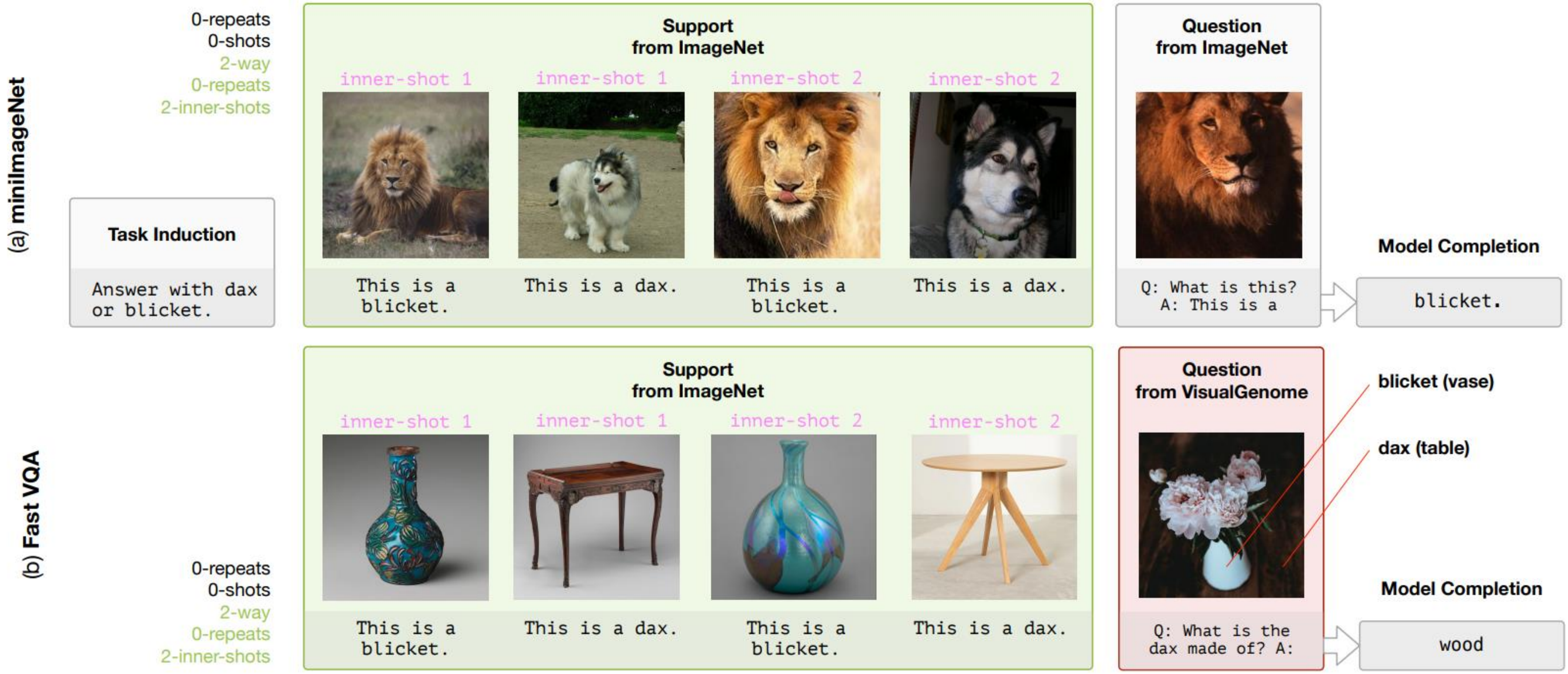
(b) **1-shot outside-knowledge VQA**

(c) **Few-shot image classification**

Possible thanks to Position encodings !

Approach - continued

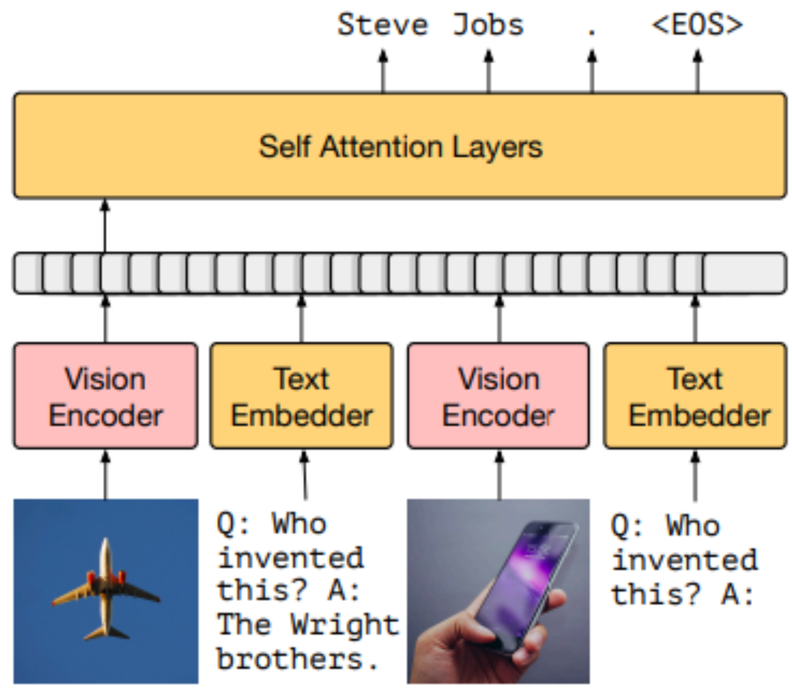
MiniImageNet : benchmark used to measure few-shot capabilities (from *Matching Networks for One Shot Learning*, 2016)
New task : Fast VQA from ImageNet and VisualGenome (vs. Real-Fast VQA)
k-shots / k – repeats …



Encyclopedic Knowledge and OKVQA

n-shot Acc.	n=0	n=1	n=4	$\mathcal{T}$
Frozen	5.9	9.7	12.6	✗
Frozen 400mLM	4.0	5.9	6.6	✗
Frozen finetuned	4.2	4.1	4.6	✗
Frozen train-blind	3.3	7.2	0.0	✗
Frozen VQA	19.6	–	–	✗
Frozen VQA-blind	12.5	–	–	✗
MAVEx [42]	39.4	–	–	✓

Table 2: Transfer from Conceptual Captions to OKVQA. The $\mathcal{T}$ column indicates if a model uses training data from the OKVQA training set. *Frozen* does not train on VQAv2 except in the baseline row, and it never trains on OKVQA.



(b) **1-shot outside-knowledge VQA**

Fast Concept Binding Examples:

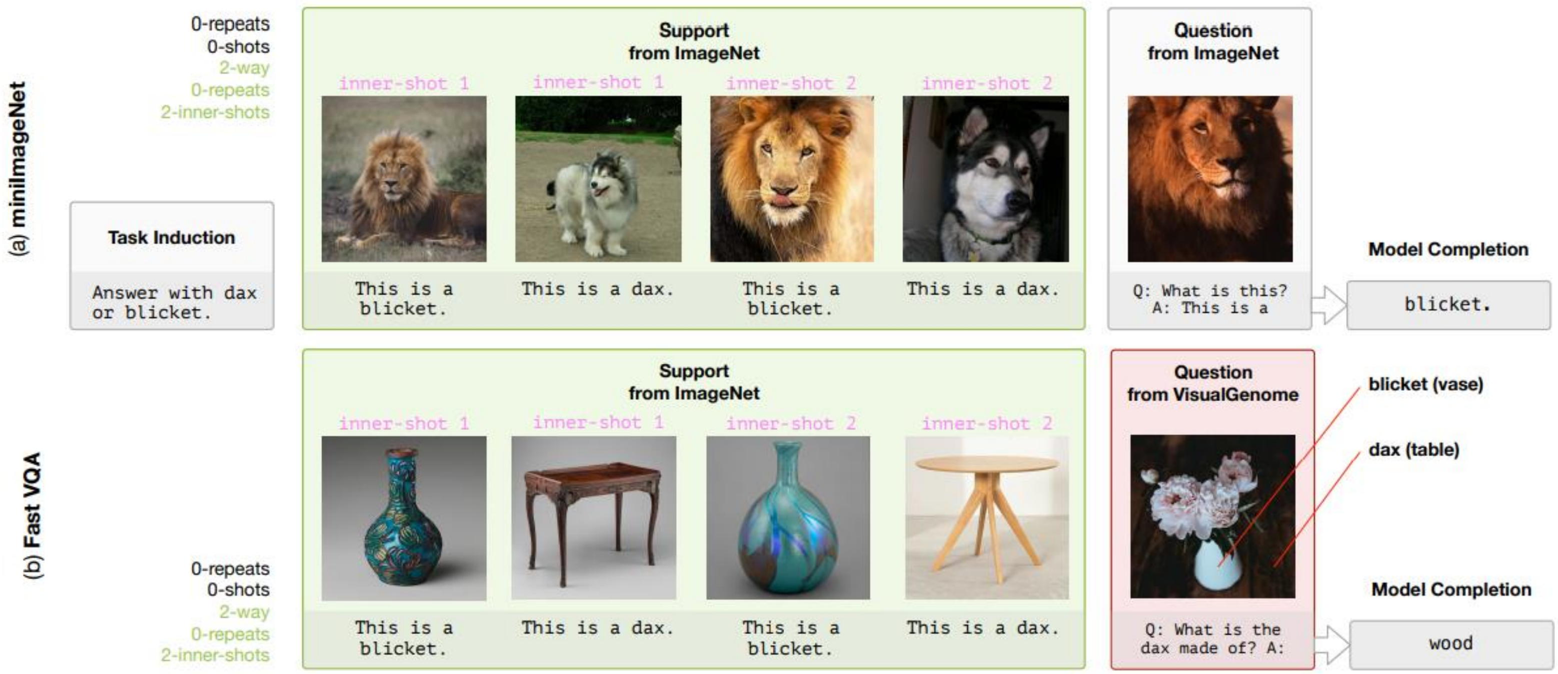
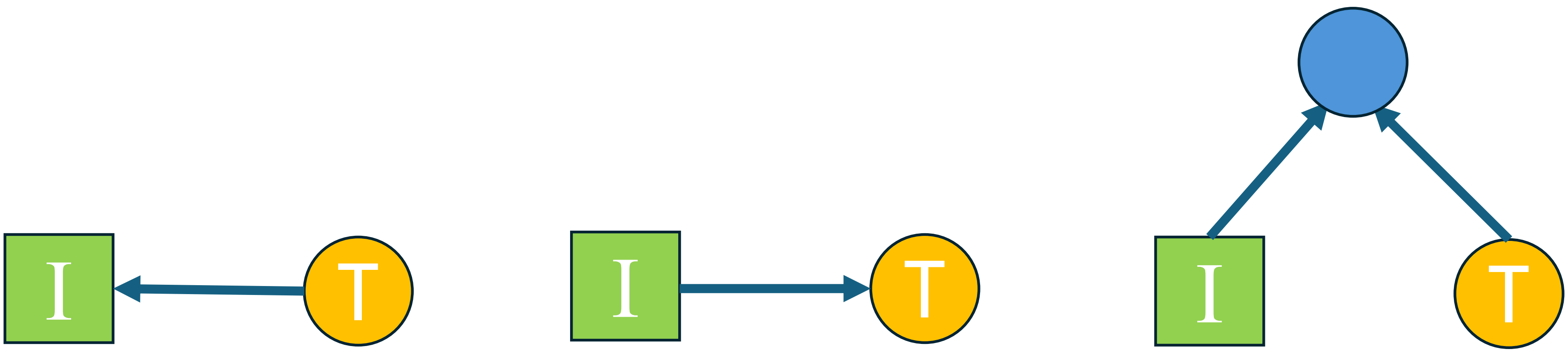


Figure 4: Examples of (a) the Open-Ended miniImageNet evaluation (b) the Fast VQA evaluation.

Vision and Language

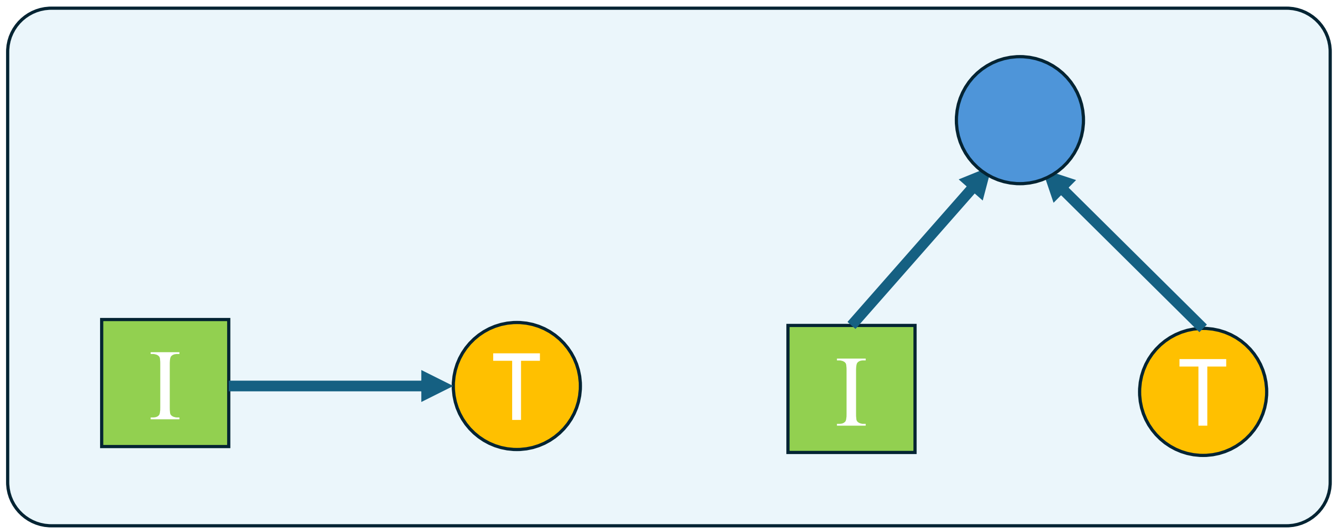


Stable Diffusion, Dalle-1

FROZEN

CLIP

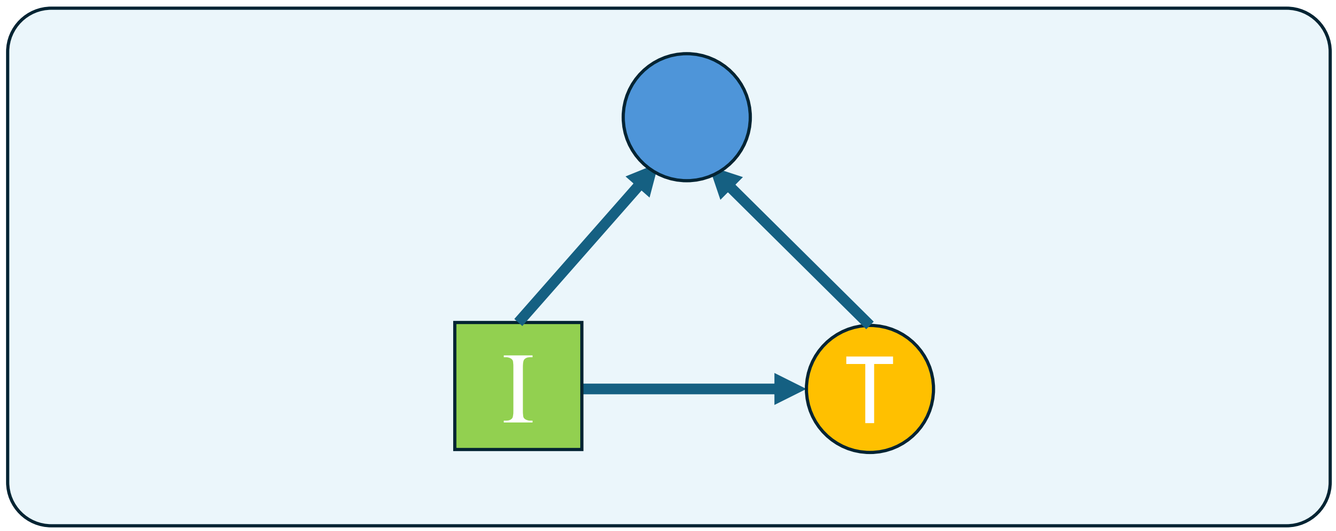
Vision and Language



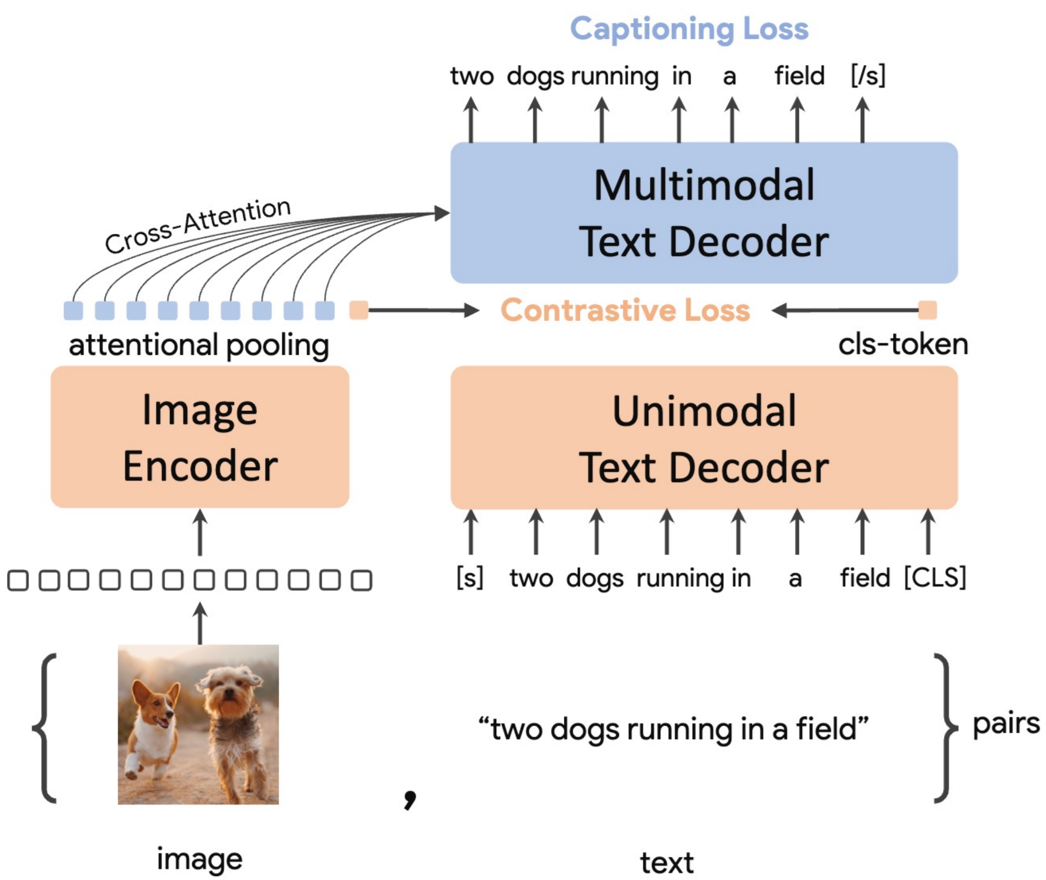
Stable Diffusion,
Dalle-1

Vision and Language

Stable Diffusion,
Dalle-1

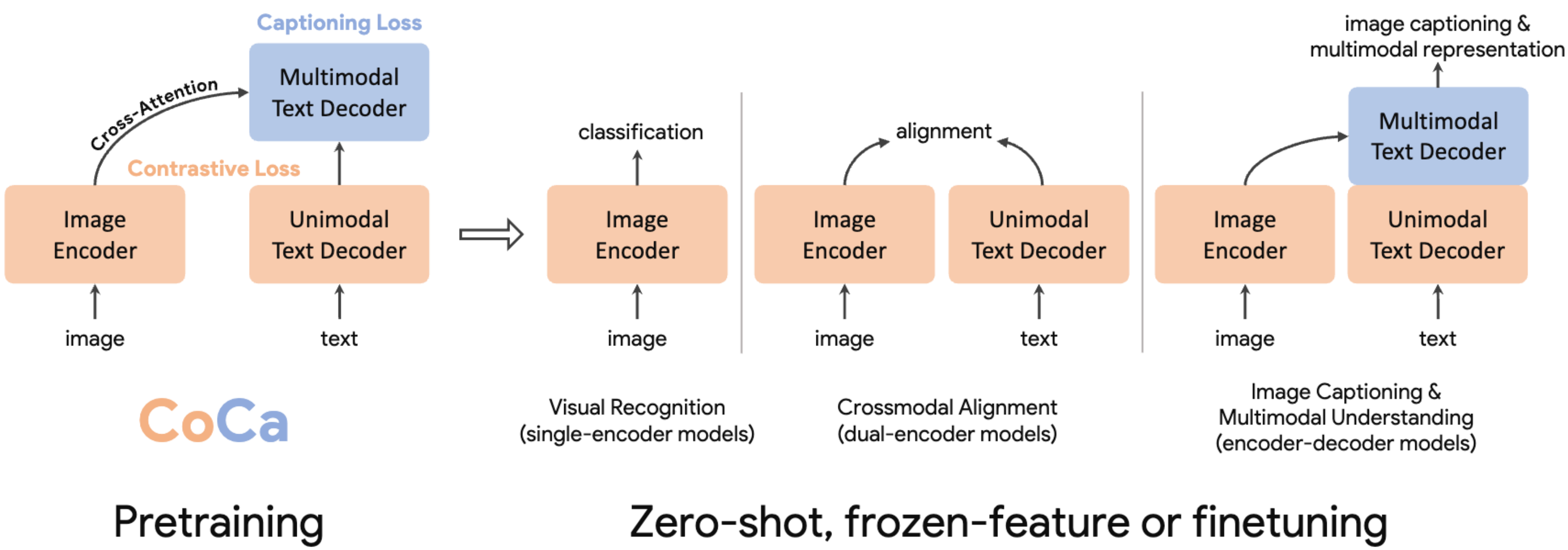


CoCa

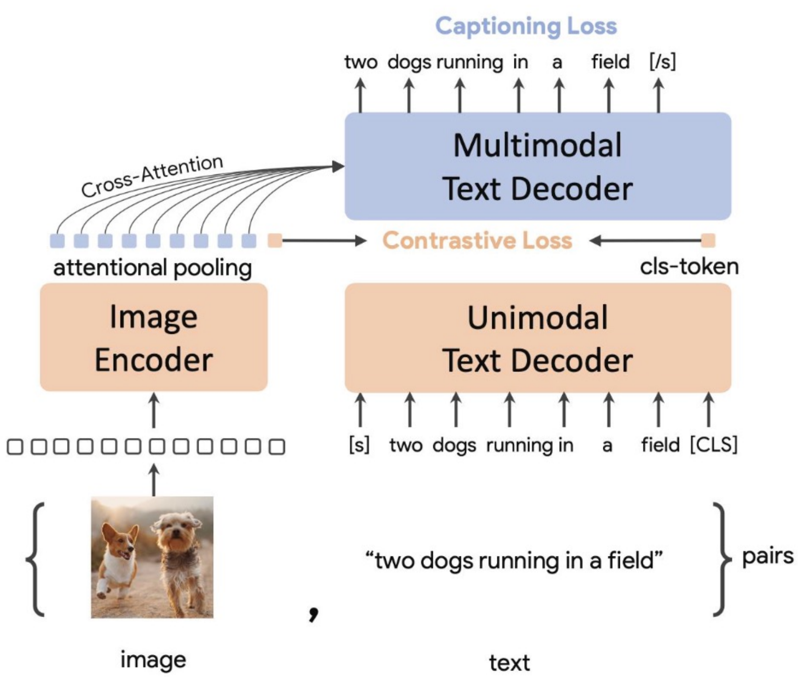


1. Image encoder produces unimodal image representation.
2. Unimodal **decoder** produces unimodal text representation
3. Contrastive loss between unimodal image and text representations.
4. Unimodal representations get fed into multimodal decoder (cross attention).
5. Captioning loss between predicted caption and actual caption (autoregressive).

CoCa

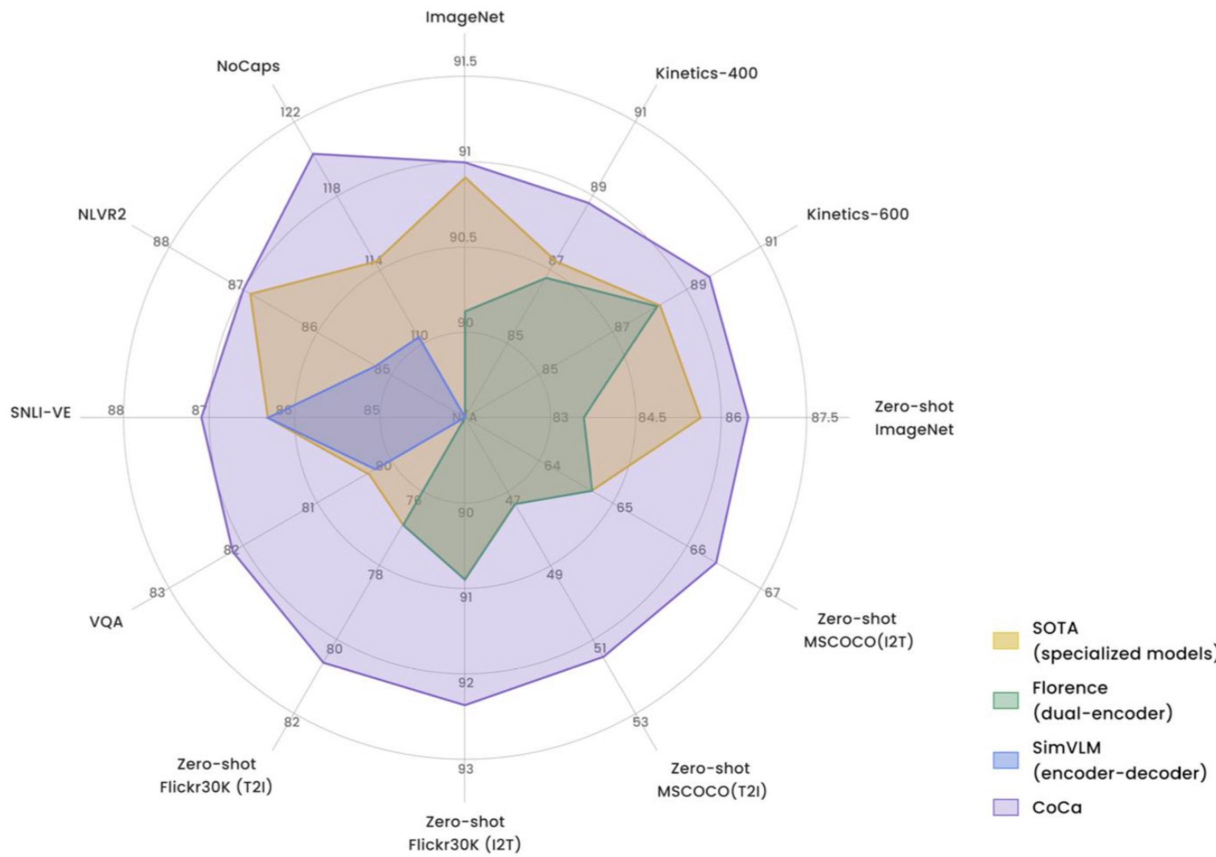

Captioning Loss
Multimodal Text Decoder
Cross-Attention
Contrastive Loss
Image Encoder
Unimodal Text Decoder
image
text
CoCa
classification
Image Encoder
image
alignment
Image Encoder
Unimodal Text Decoder
image
text
image captioning & multimodal representation
Multimodal Text Decoder
Image Encoder
Unimodal Text Decoder
image
text
Visual Recognition (single-encoder models)
Crossmodal Alignment (dual-encoder models)
Image Captioning & Multimodal Understanding (encoder-decoder models)
Pretraining
Zero-shot, frozen-feature or finetuning

CoCa: Contrastive Captioners are Image-Text Foundation Models



- Uses fine grain image representation (256 image tokens) + unimodal text representation.
- Ignores CLS.
- Uses cross attention.
- Obtain unified image-text representation used to predict probability distribution of the vocabulary through autoregression.

CoCa: Contrastive Captioners are Image-Text Foundation Models

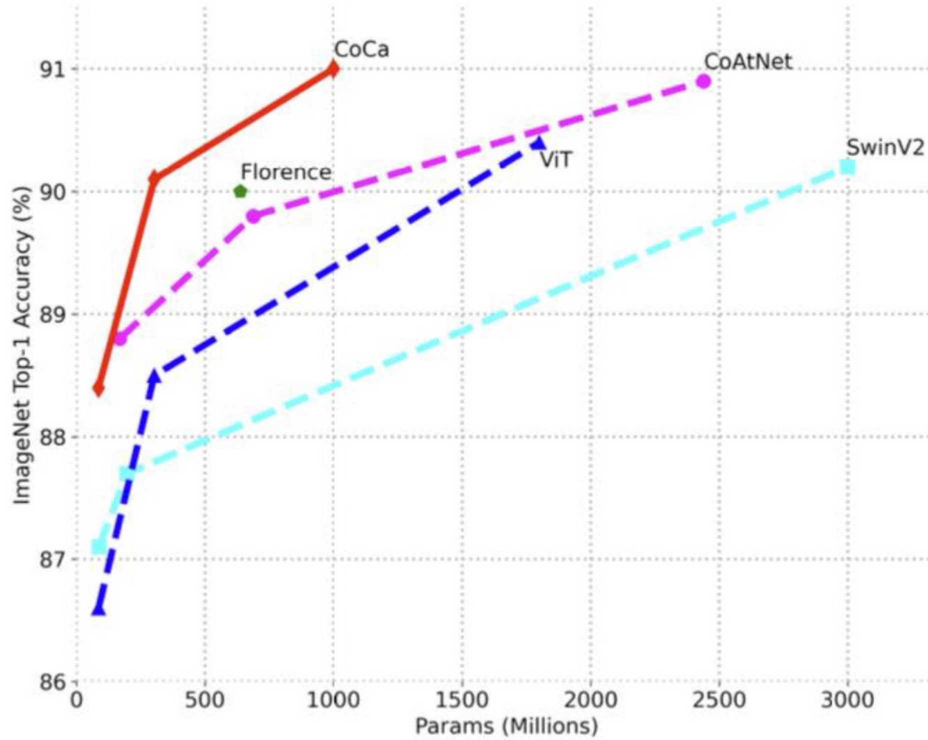


CoCa: Contrastive Captioners are Image-Text Foundation Models

Model	ImageNet	M
ALIGN[a]	88.6	V
Florence[b]	90.1	M
MetaPseudoLabels[c]	90.2	V
CoAtNet[d]	90.9	F
ViT-G[e]	90.5	M
+ Model Soups[f]	90.9	C
CoCa (frozen)	90.6	C
CoCa (finetuned)	**91.0**	C

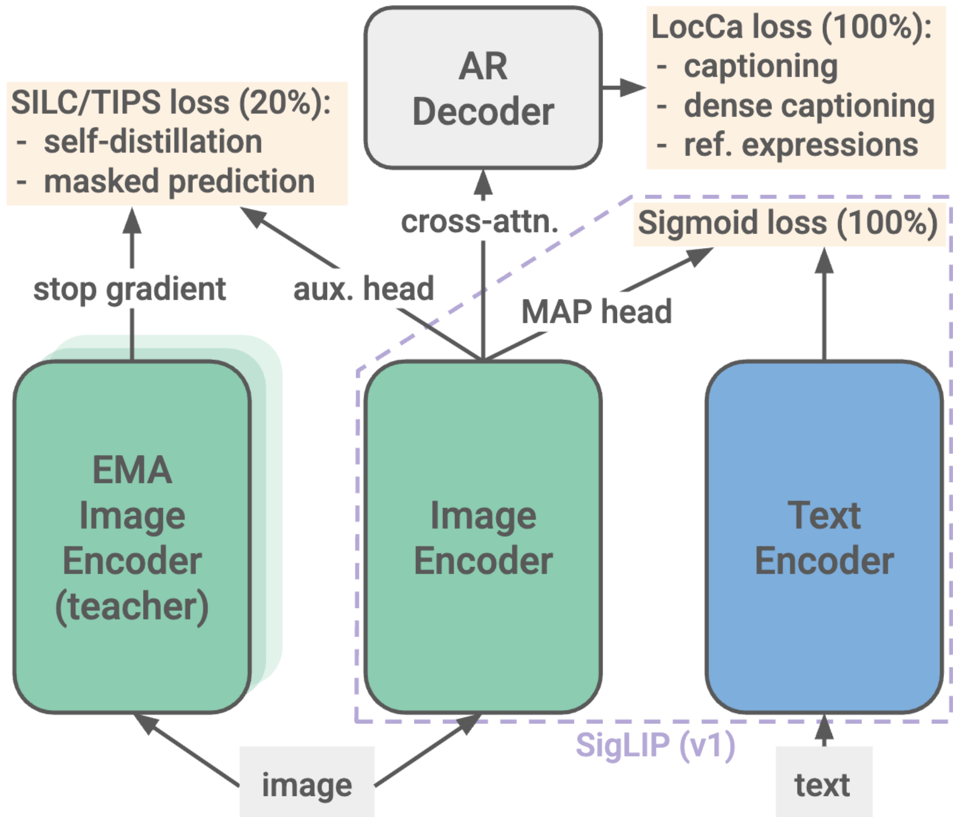
Table 2: Image classification and video acti
reference: [a](Jia et al., 2021) [b](Yuan et al., 20
[g](Wortsman et al., 2022) [g](Arnab et al., 202
2021) [l](Zhang et al., 2021a).

Is visual encoder finetuning that relevant?

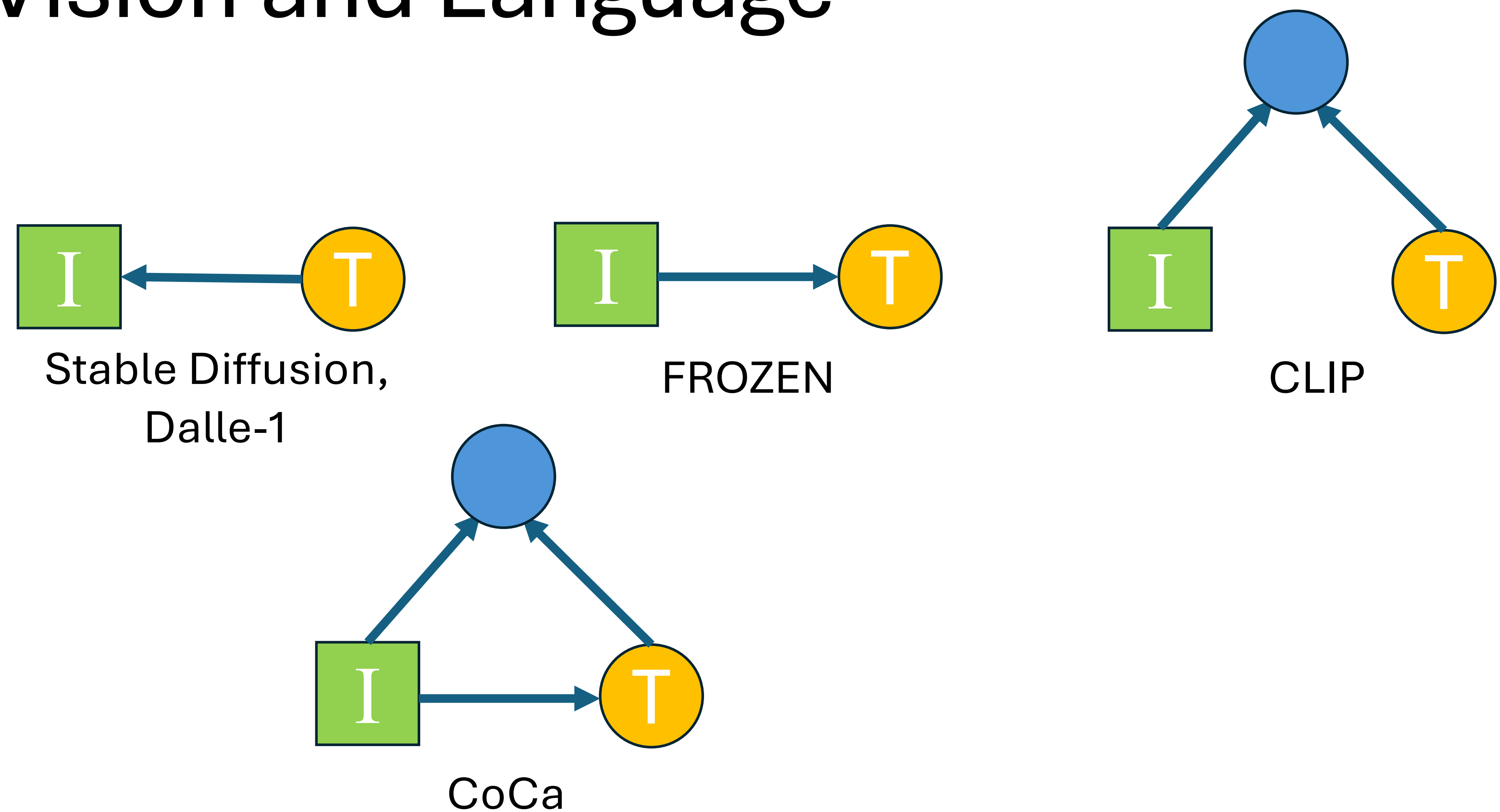


(a) Finetuned ImageNet Top-1 Accuracy.

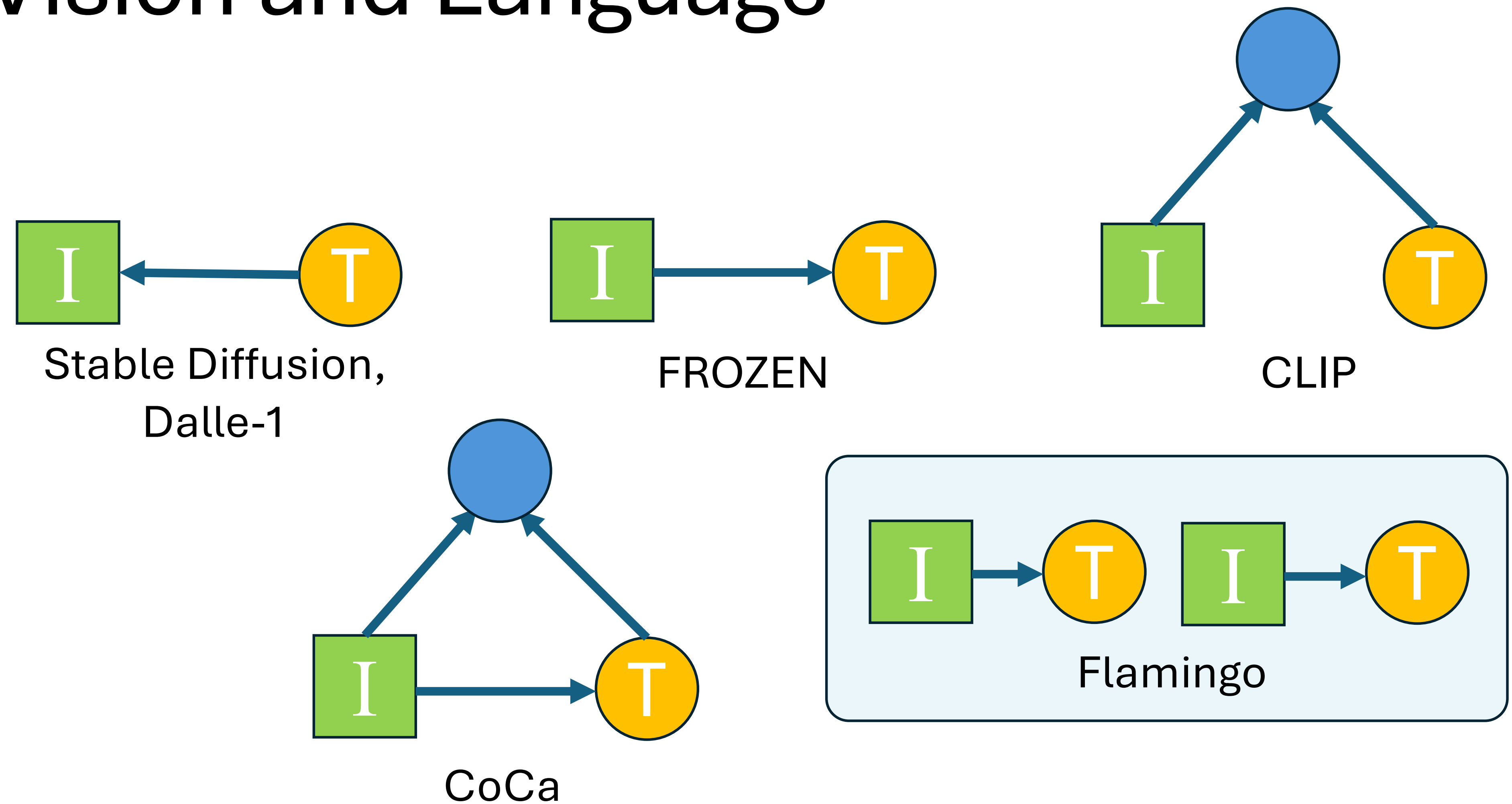
SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features



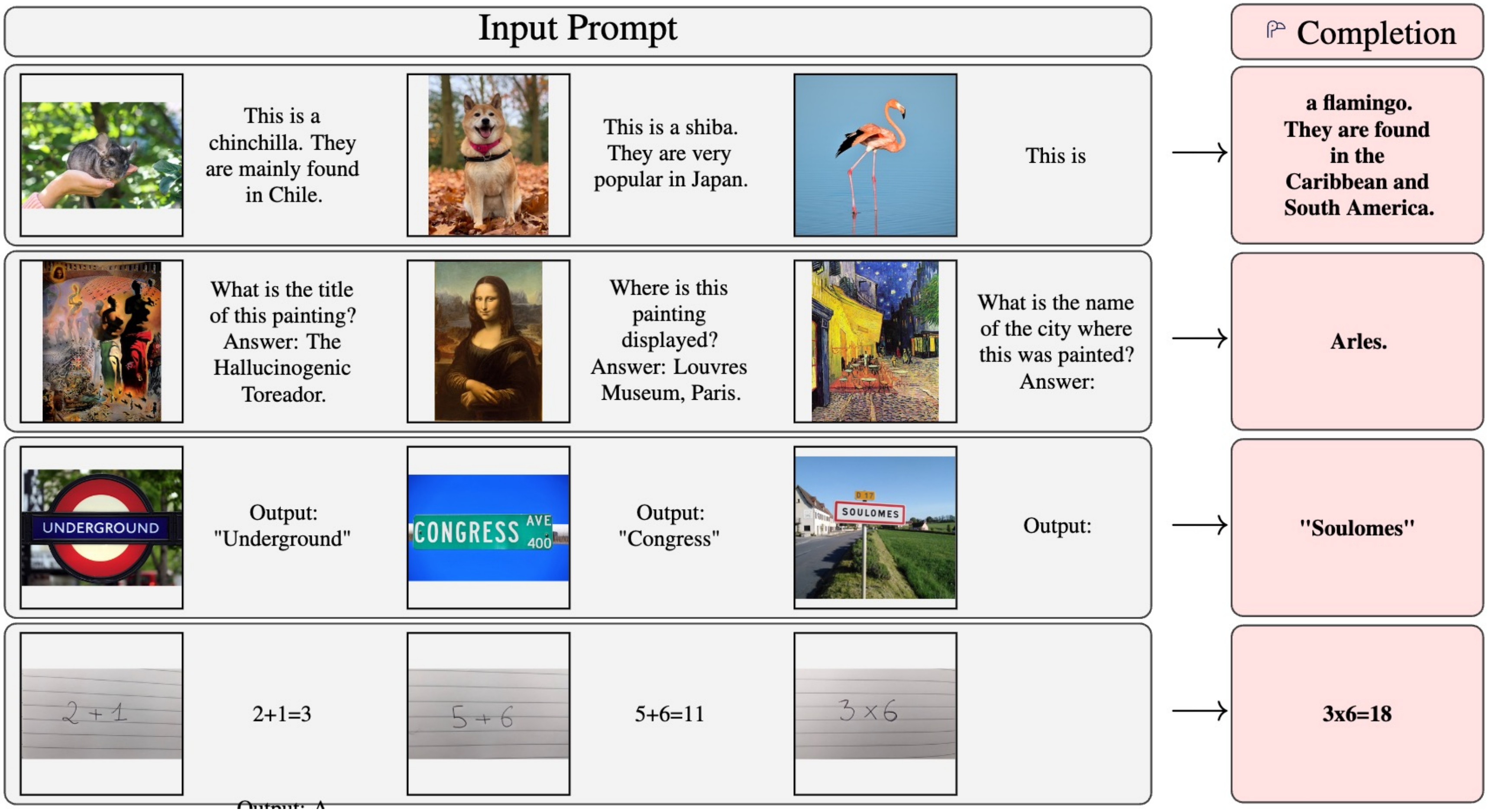
Vision and Language



Vision and Language

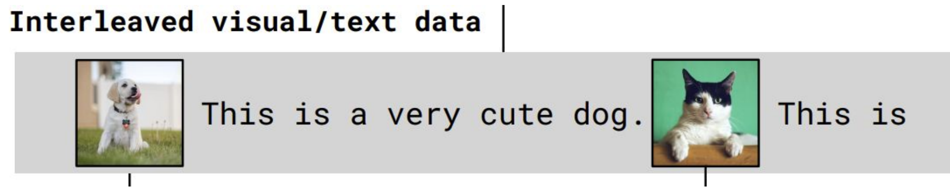


Flamingo: a Visual Language Model for Few-Shot Learning

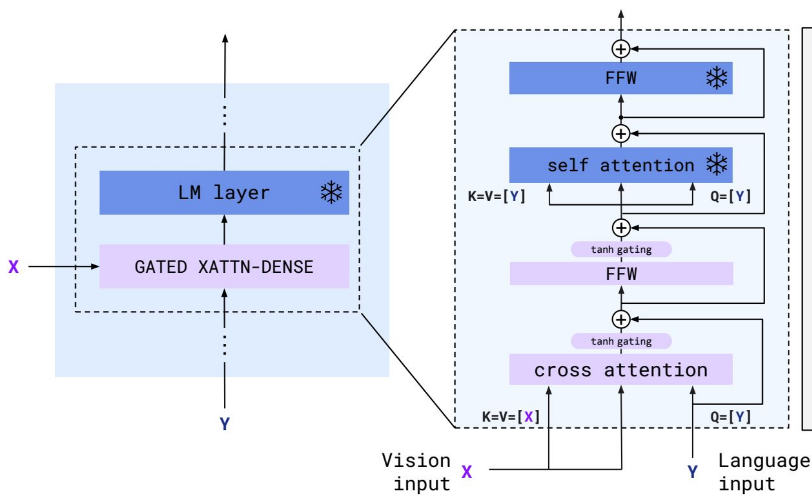


Approach

Text input interleaved with image



Visually-conditioned autoregressive text generation




```python
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
```

Use of tanh and initialized to zero: to have no effect at training beginning

Georgia Tech

Approach

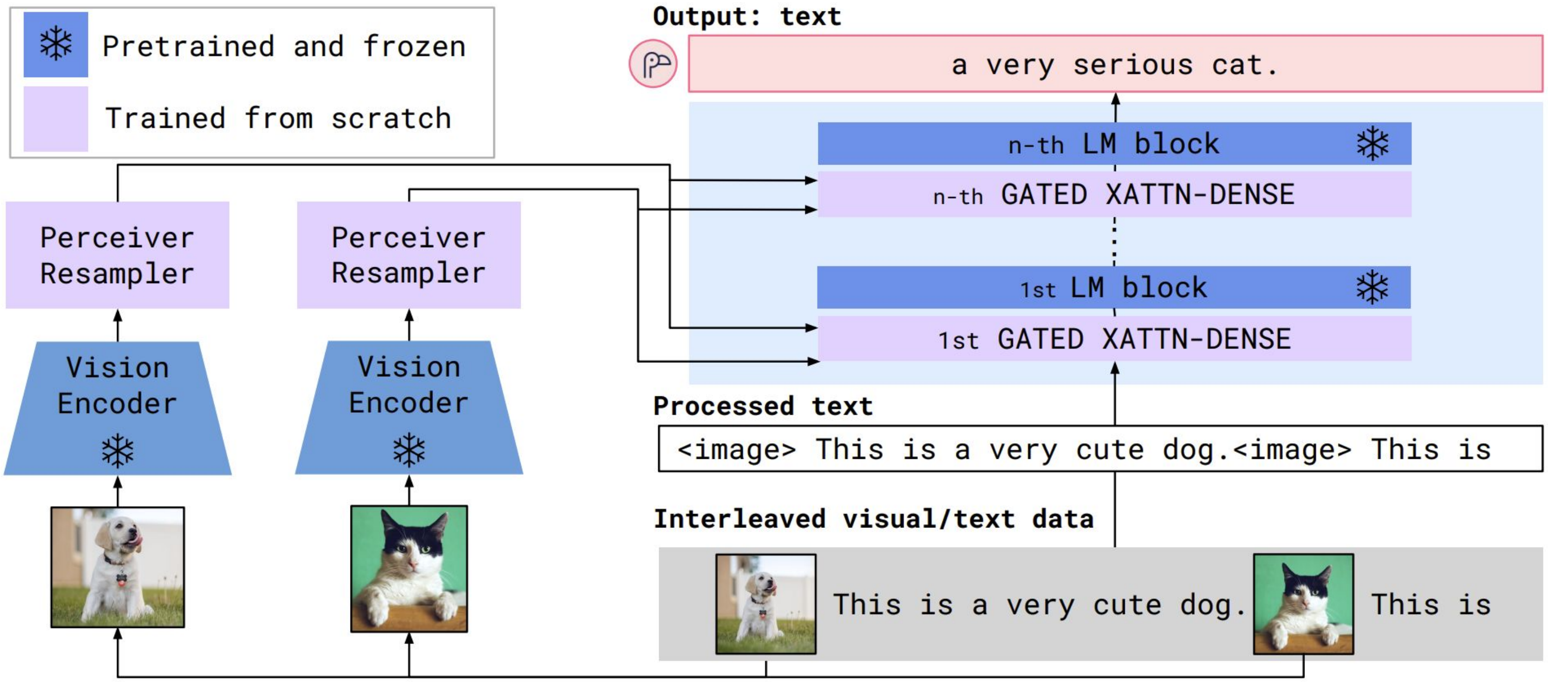


Figure 3: **Flamingo architecture overview.** Flamingo is a family of visual language models (VLMs) that take as input visual data interleaved with text and produce free-form text as output.

Approach

Vision Encoder: From pixels to features

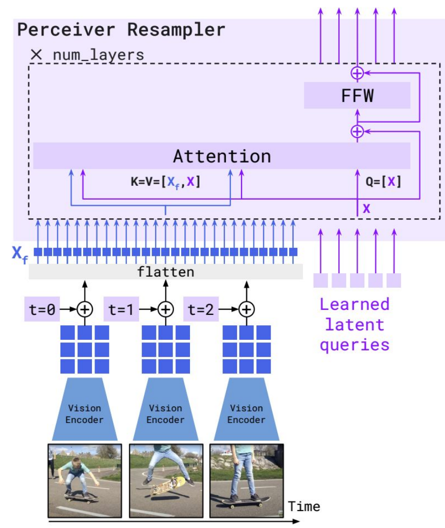
Architecture:

- Normalizer Free ResNet (NFNet)

Trained on:

- Datasets of image and text pairs, using the two-term contrastive loss from Radford et al.

Perceiver Resampler: From varying-size large feature maps to few visual tokens.




```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
```

Figure 5: **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. 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.

Georgia Tech

Approach

Training on a mixture of vision and language datasets

- Datasets
  - M3W:Interleaved image and text dataset.
  - ALIGN: 1.8B text-to-image
  - LTIP: 312M long-text and image
  - VTP: 27M short-video and text

Figure 9: **Training datasets.** Mixture of training datasets of different formats. N corresponds to the number of visual inputs for a single example. For paired image (or video) and text datasets, $N = 1$. T is the number of video frames ($T = 1$ for images). H, W, and C are height, width and color channels.

- Multi-objective training and optimisation strategy.
  - Tuning the per-dataset weights λm is key to performance.
  - Below weights were obtained empirically at a small model scale and kept fixed afterwards.

Dataset	M3W	ALIGN	LTIP	VTP
λm	1.0	0.2	0.2	0.03

Experiments and Results

Zero/Few-shot Performance

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)	-

Table 1: **Comparison to the state of the art.** A *single* Flamingo model reaches the state of the art on a wide array of image **(I)** and video **(V)** understanding tasks with few-shot learning, significantly outperforming previous best zero- and few-shot methods with as few as four examples. More importantly, using only 32 examples and without adapting any model weights, Flamingo *outperforms* the current best methods – fine-tuned on thousands of annotated examples – on seven tasks. Best few-shot numbers are in **bold**, best numbers overall are underlined.

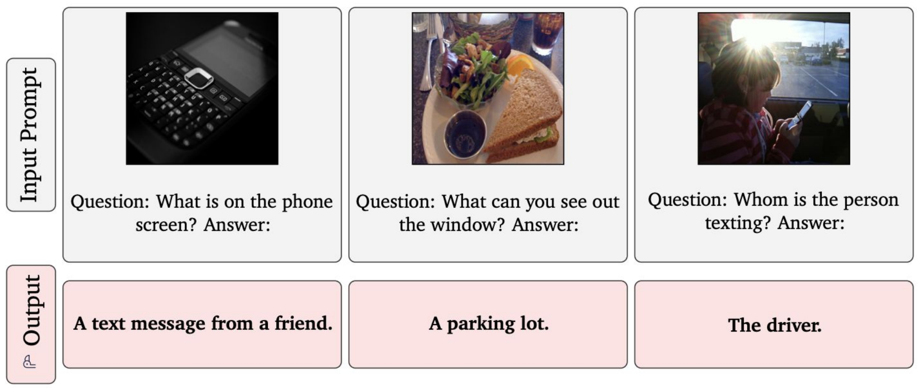
Limitations

Functional Limitations

- Hallucinations (Q)
- Poor generalization for long sequences
- Worse than contrastive models in classification
- Sensitivity to examples

Practical Limitations

- Text interface inconvenient for some tasks
- Expensive to train



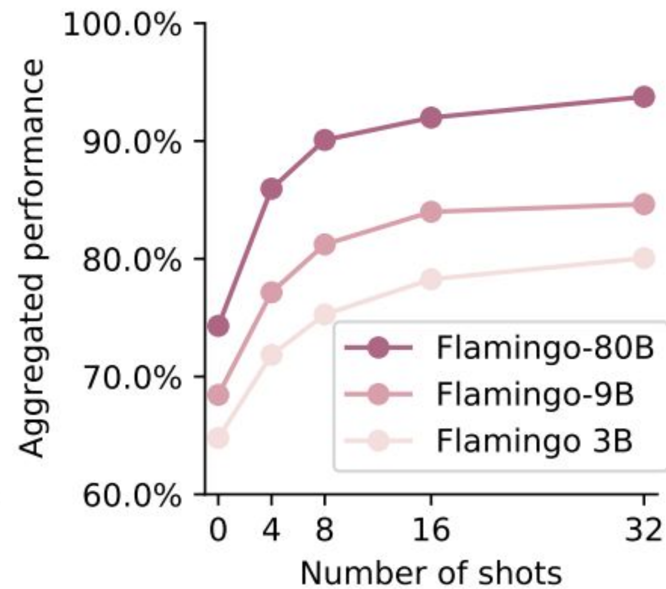
Q: Is the model simply inferring answers through the prompts without using images?

Georgia Tech

Limitations

Learning new task or identifying trained task?

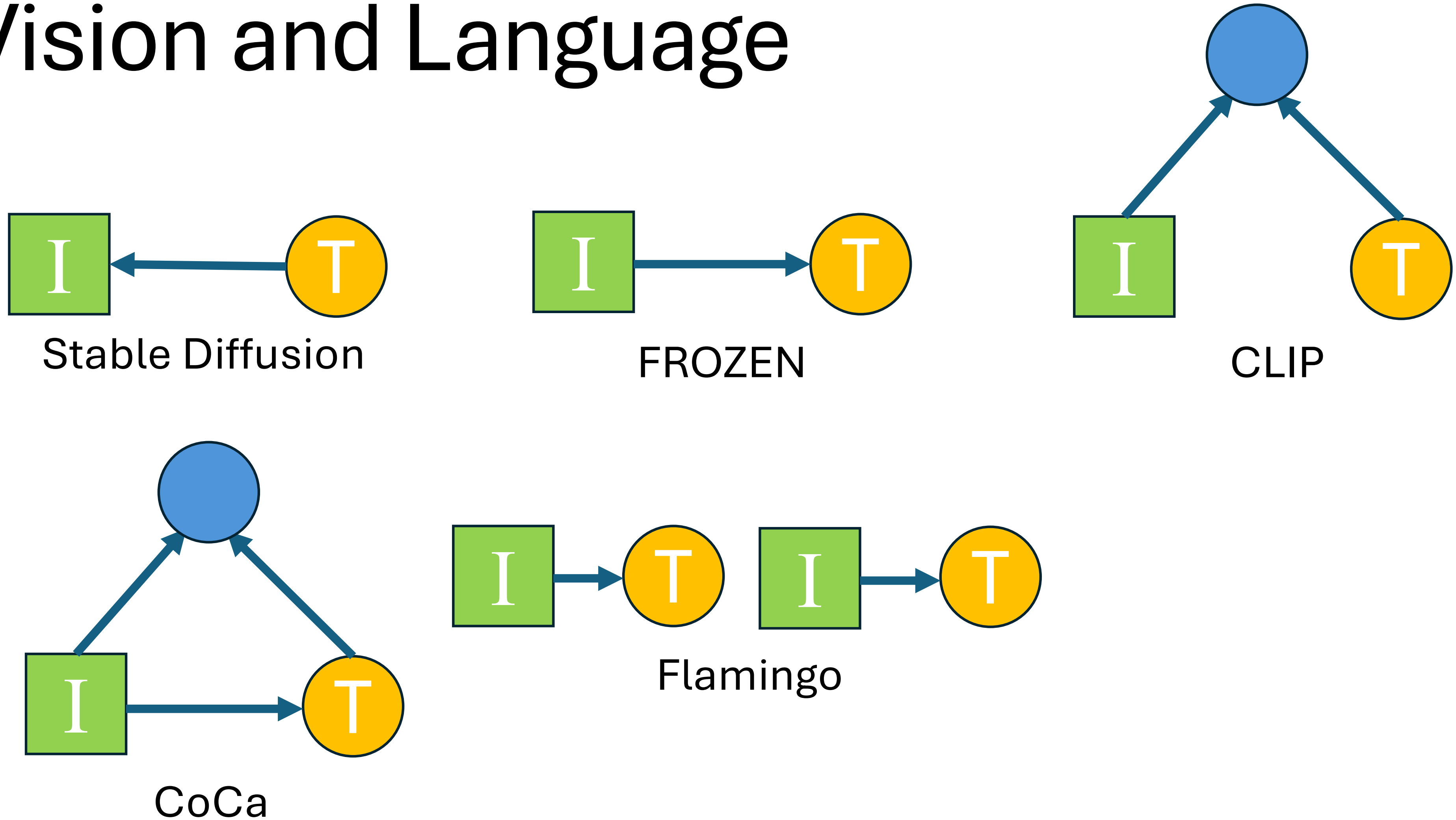
- Performance plateaus as number of examples reach 32
- Non-trivial performance without images (Q)
- Examples may be locating task in memory (Q)
  - "Task Location" [8]



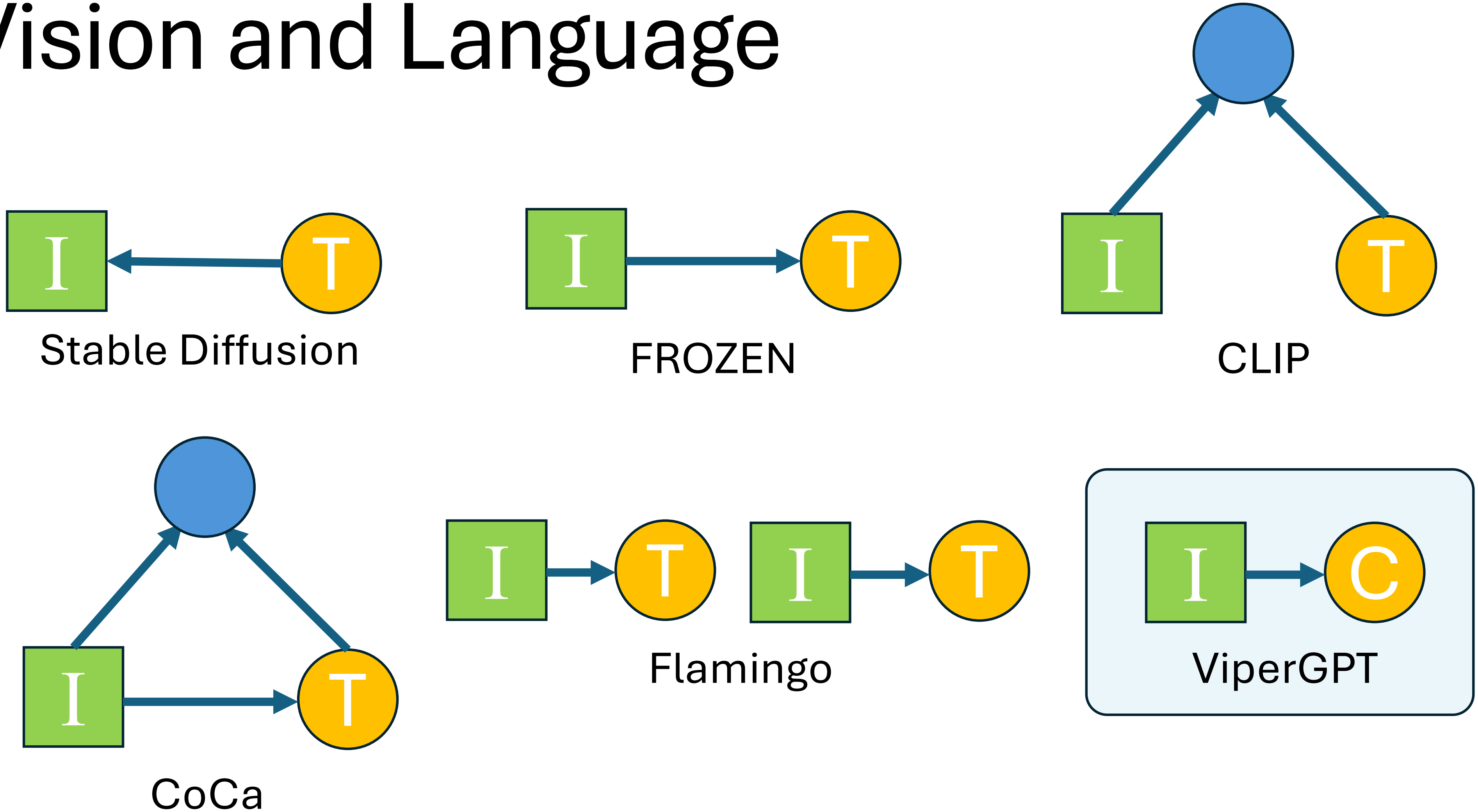
Q: Is the model learning a new task at inference or just identifying a task learned during training?

Q: Is it possible that the model's success is just due to the capabilities of the LM?

Vision and Language



Vision and Language



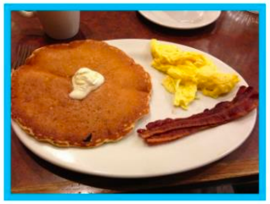
ViperGPT: Visual Inference via Python Execution for Reasoning

Problem Statement: VLM Reasoning Tasks

- Visual Grounding
  - Identifying the bounding box in an image that corresponds best to a given query.

Query: pizza front

- Compositional Image Question Answering
  - Decomposing complex questions into simpler tasks.



Query: Does that pancake look brown and round?

- External Knowledge-dependent Image Question Answering
  - Many questions about images can only be answered correctly by integrating outside knowledge about the world.

Query: The real live version of this toy does what in the winter?

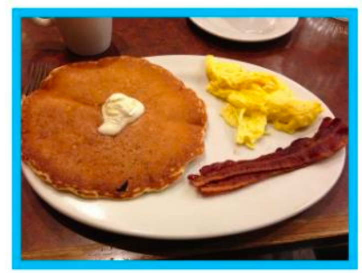
Georgia Tech

Problem Statement: VLM Reasoning Tasks

- Visual Grounding
  - Identifying the bounding box in an image that corresponds best to a given query.

Query: pizza front

- Compositional Image Question Answering
  - Decomposing complex questions into simpler tasks.

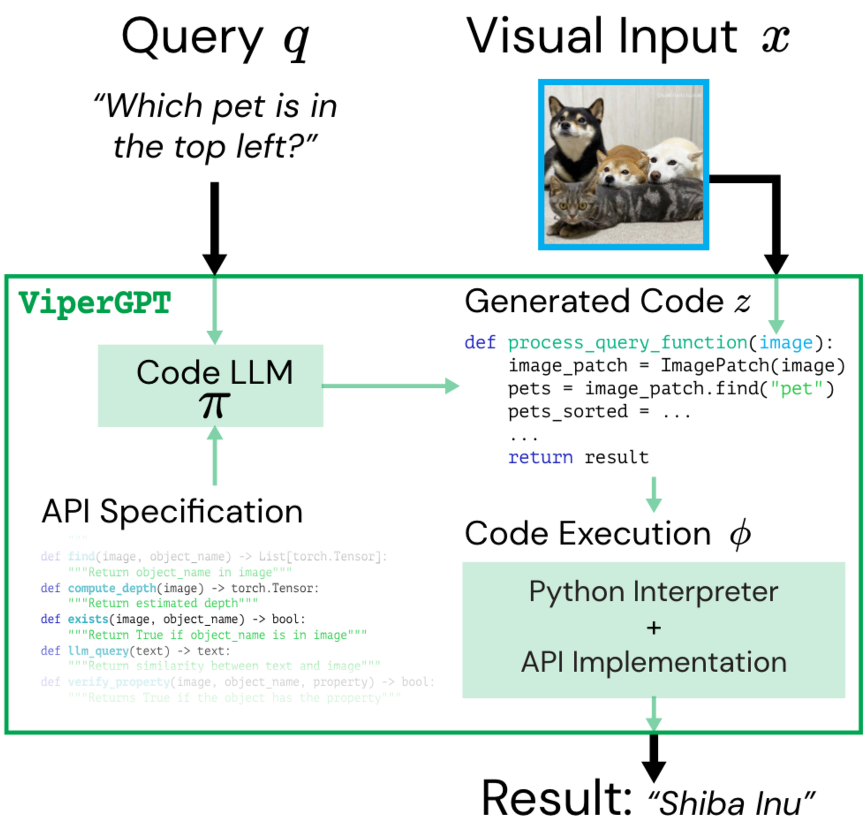


Query: Does that pancake look brown and round?

- External Knowledge-dependent Image Question Answering
  - Many questions about images can only be answered correctly by integrating outside knowledge about the world.

Query: The real live version of this toy does what in the winter?

Approach: Overview

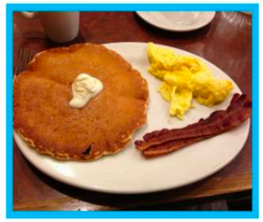


- ViperGPT is a framework for solving complex visual queries programmatically.
- Inputs
  - Visual input x: image / videos
  - Textual query q: questions or descriptions
- Output r: any type (e.g., text / image crops)
- Program generator π: $z = \pi(q)$
  - π: LLMs
  - z : Python code
- Execution engine ϕ: $r = \phi(x, z)$
  - Python Interpreter
  - API Implementation

Approach: Program Generation

Query: Does that pancake look brown and round?

In:



Generated code

```
def execute_command(image):
    image_patch = ImagePatch(image)
    pancake_patches = image_patch.find("pancake")
    is_brown = pancake_patches[0].verify_property("pancake", "brown")
    is_round = pancake_patches[0].verify_property("pancake", "round")
    return bool_to_yesno(is_brown and is_round)
```

Query: Are there water bottles to the right of the bookcase that is made of wood?

In:

Generated code

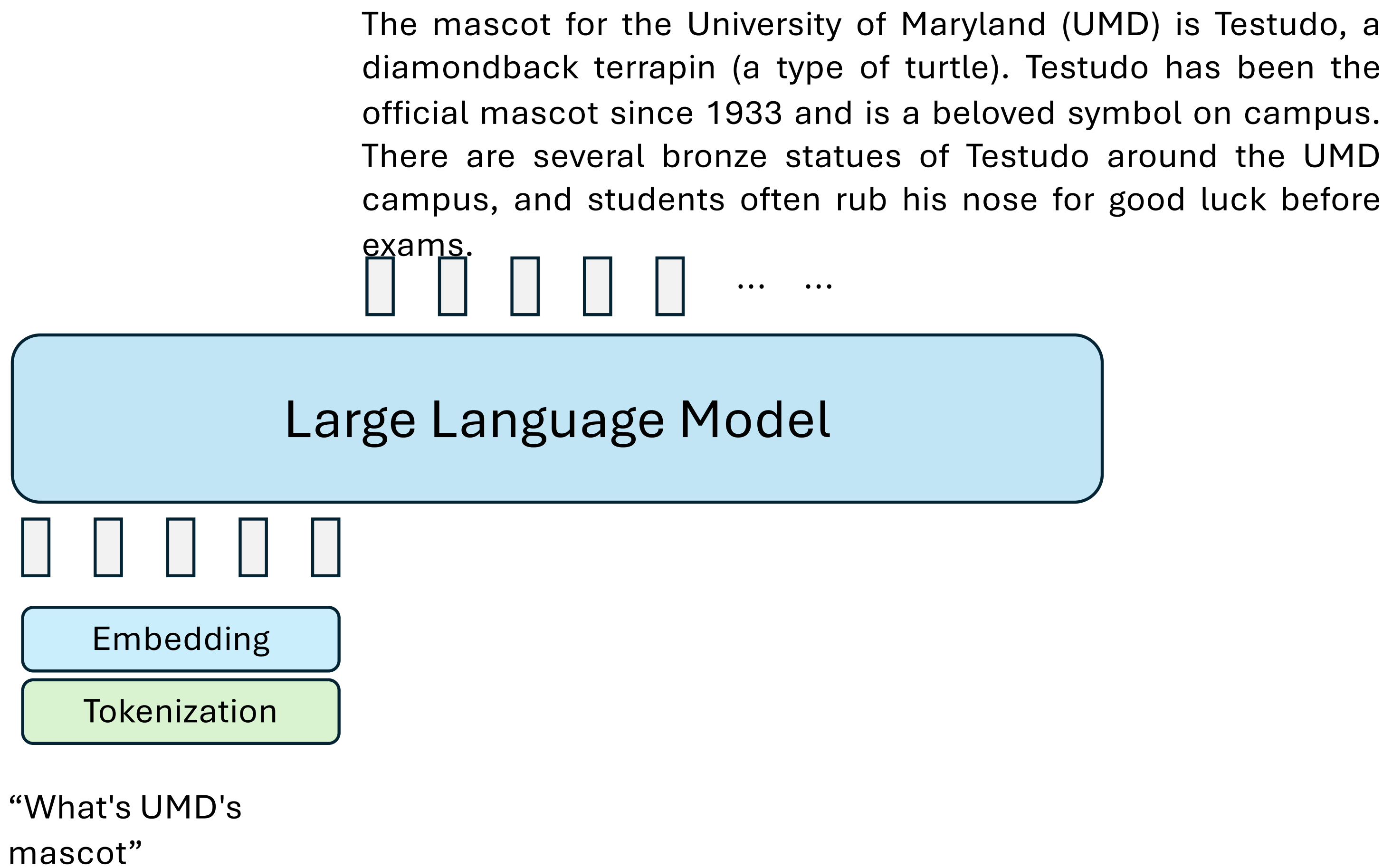
```
def execute_command(image):
    image_patch = ImagePatch(image)
    bookcase_patches = image_patch.find("bookcase")
    for bookcase_patch in bookcase_patches:
        is_wood = bookcase_patch.verify_property("bookcase", "wood")
        if is_wood:
            water_bottle_patches = image_patch.find("water bottle")
            for water_bottle_patch in water_bottle_patches:
                if water_bottle_patch.horizontal_center > \
                   bookcase_patch.horizontal_center:
                    return "yes"
            return "no"
    return "no"
```

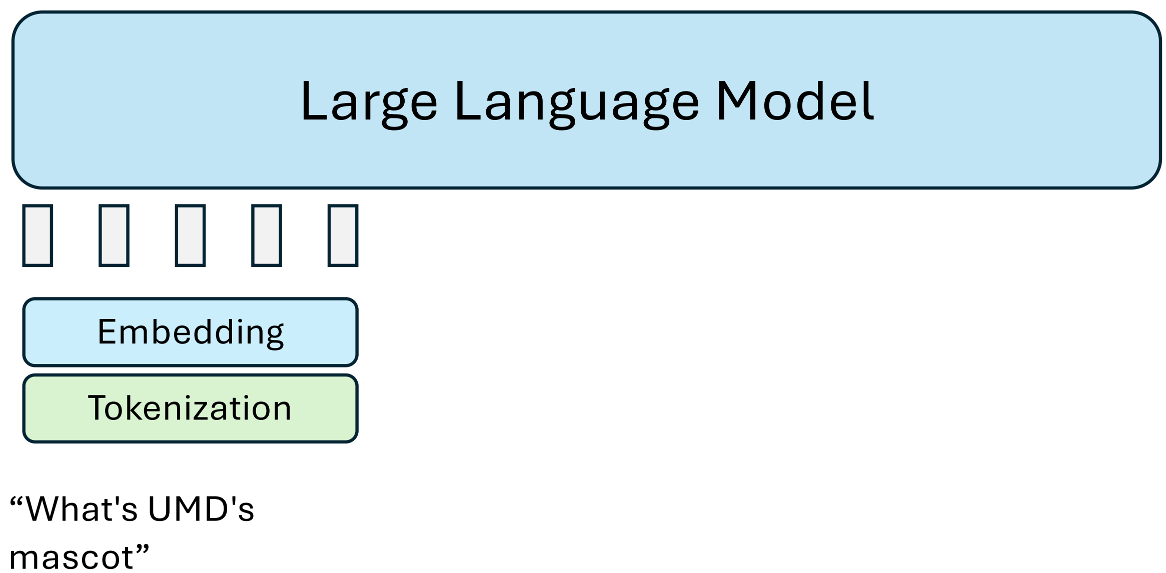
- Program Generator: GPT-3 Codex
  - Obviates the need for task-specific training for program generation.
- Input: a sequence of code text
  - Prompt: API specification
  - Query for the sample under consideration
- Output: Python function definition as a string.

Visual Grounding

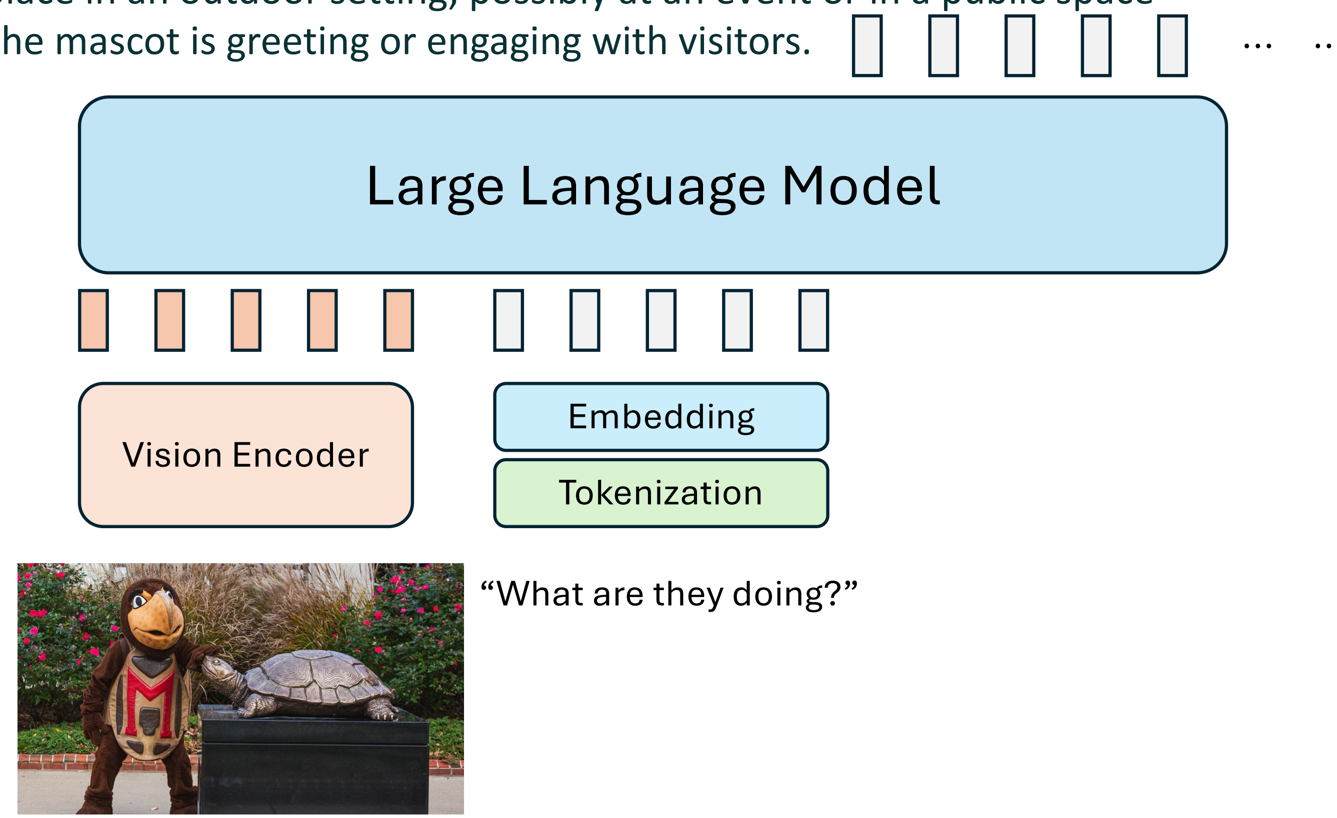
- Requires spatial reasoning and object identification
- Modules provided:
  - **Find, exists, verify_property, best_image_match, compute_depth, distance**
- Evaluated on RefCOCO and RefCOCO+
- Takeaways:
  - Clearly outperforms zero-shot methods
  - Still far behind fine-tuned models
  - Expected result since this task focuses on visual understanding instead of reasoning

		IoU (%) ↑	
		RefCOCO	RefCOCO+
Sup.	MDETR [53]	90.4	85.5
	OFA [53]	94.0	91.7
ZS	OWL-ViT [38]	30.3	29.4
	GLIP [31]	55.0	52.2
	ReCLIP [49]	58.6	60.5
	`ViperGPT` (ours)	**72.0**	**67.0**

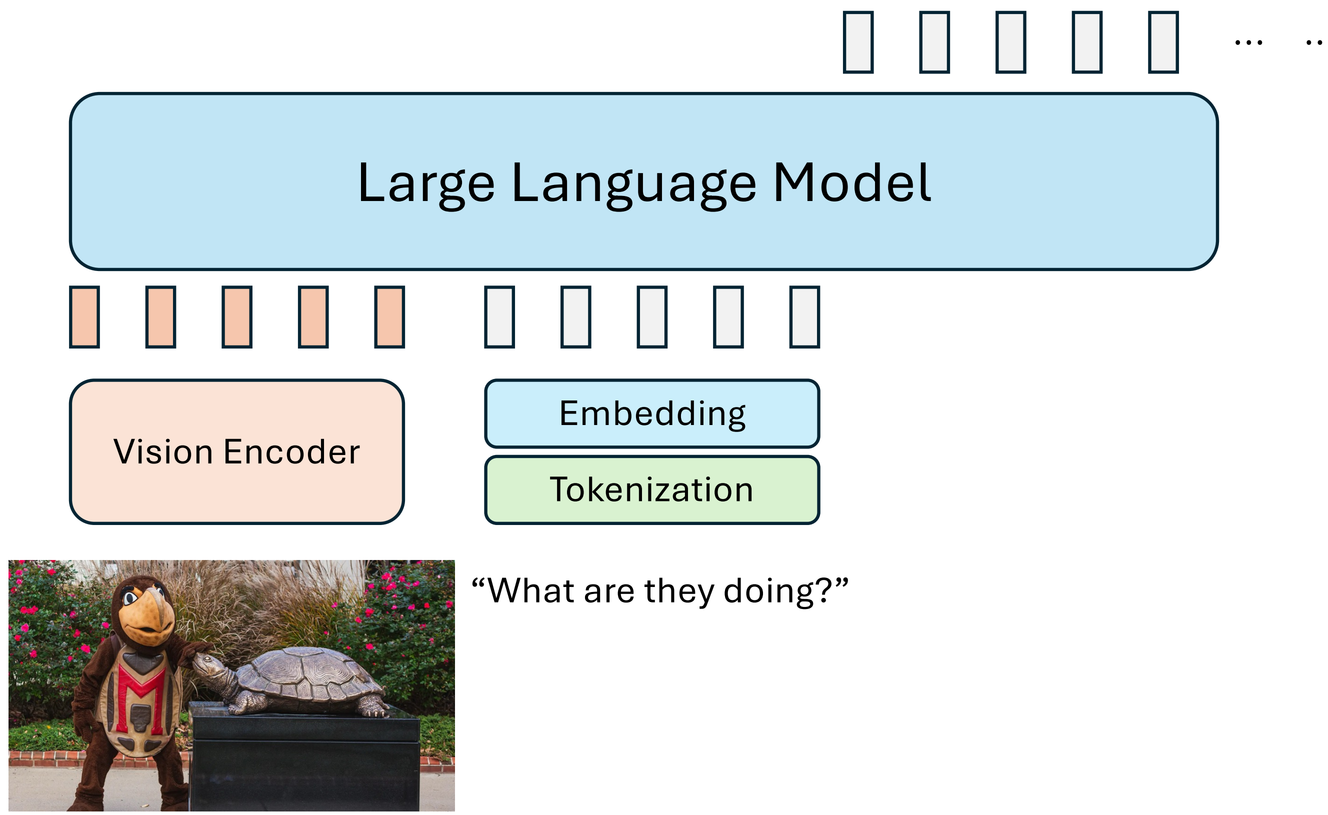

The mascot for the University of Maryland (UMD) is Testudo, a diamondback terrapin (a type of turtle). Testudo has been the official mascot since 1933 and is a beloved symbol on campus. There are several bronze statues of Testudo around the UMD campus, and students often rub his nose for good luck before exams.
… …
Large Language Model
Embedding
Tokenization
“What's UMD's mascot”


Large Language Model
Embedding
Tokenization
“What's UMD's mascot”

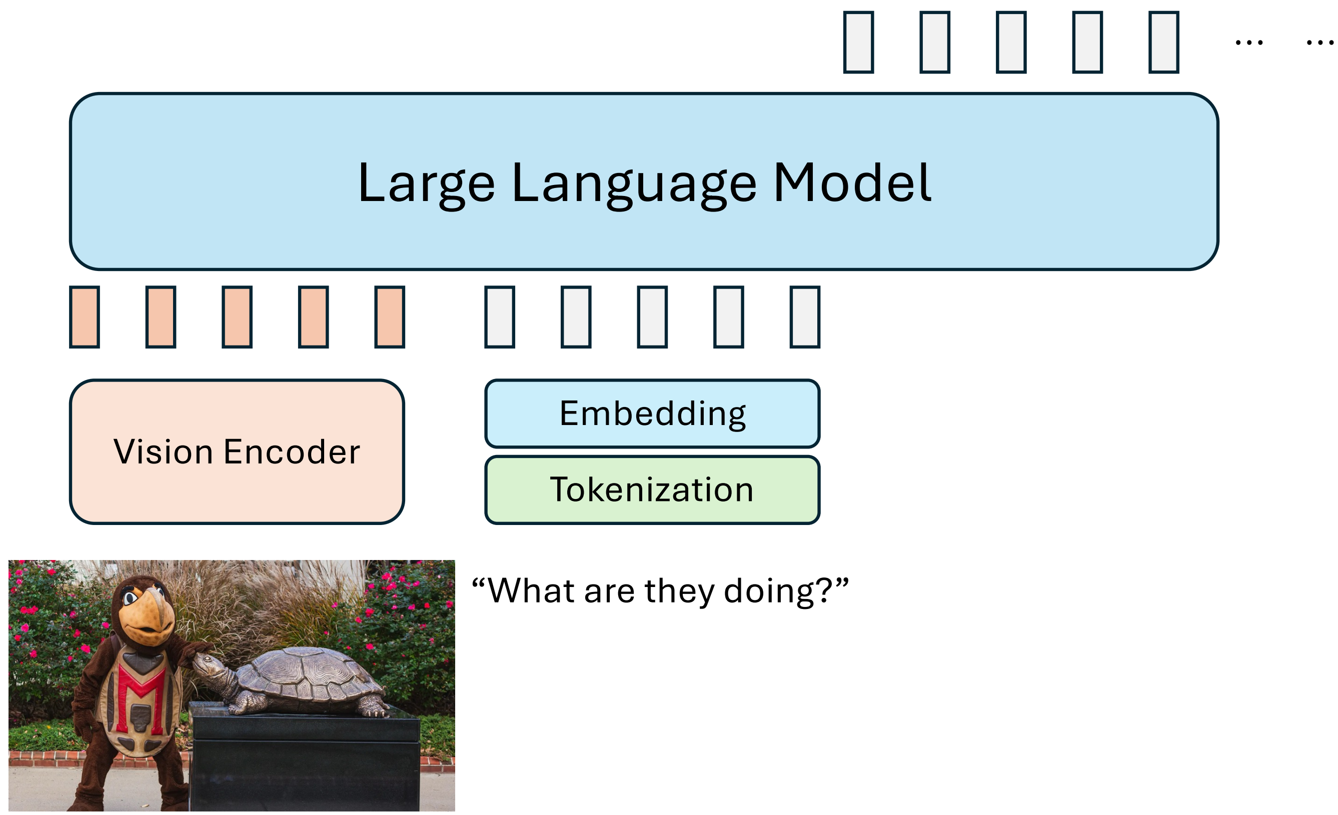
The eagle mascot is interacting with a bronze turtle statue. The mascot has its right arm extended, gently touching the nose of the turtle. This creates a friendly and engaging scene, with the mascot appearing to be in conversation with the statue. The bronze turtle is positioned on a black pedestal, adding an interesting contrast to the mascot's costume. This interaction seems to be taking place in an outdoor setting, possibly at an event or in a public space where the mascot is greeting or engaging with visitors.



https://molmo.allenai.org/

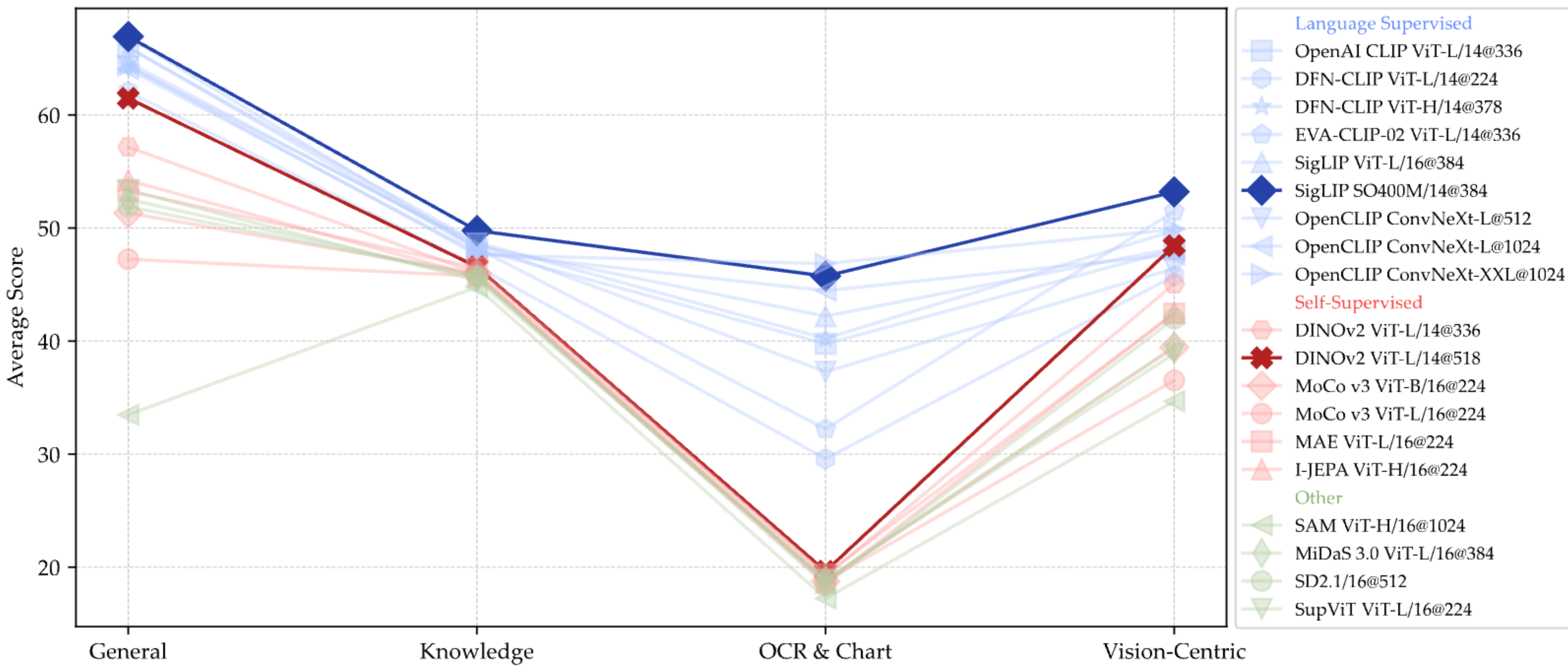


https://molmo.allenai.org/



https://molmo.allenai.org/

Vision Encoders for MLLMs



Vision Encoders for MLLMs