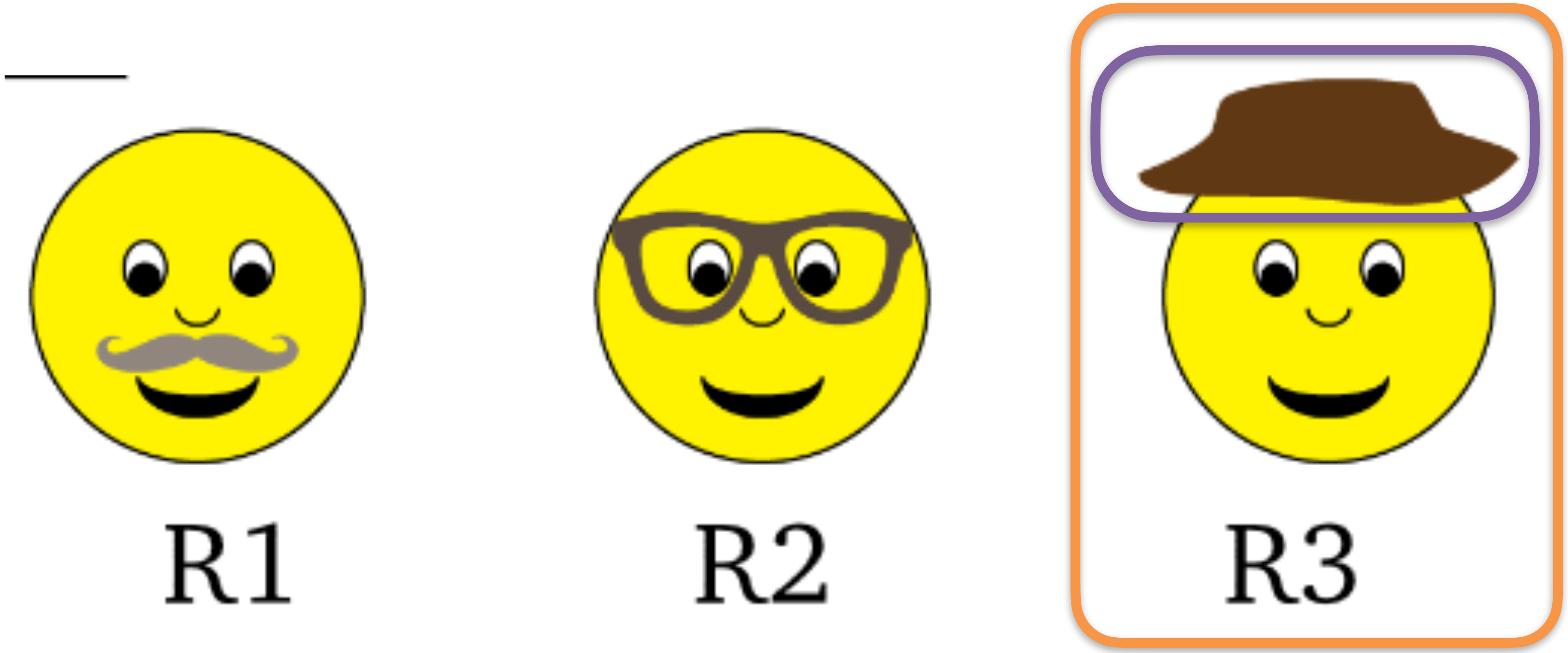
Vision and Language

Berkeley
NLP

slides from: Daniel Fried, Yonatan Bisk, L-P Morency

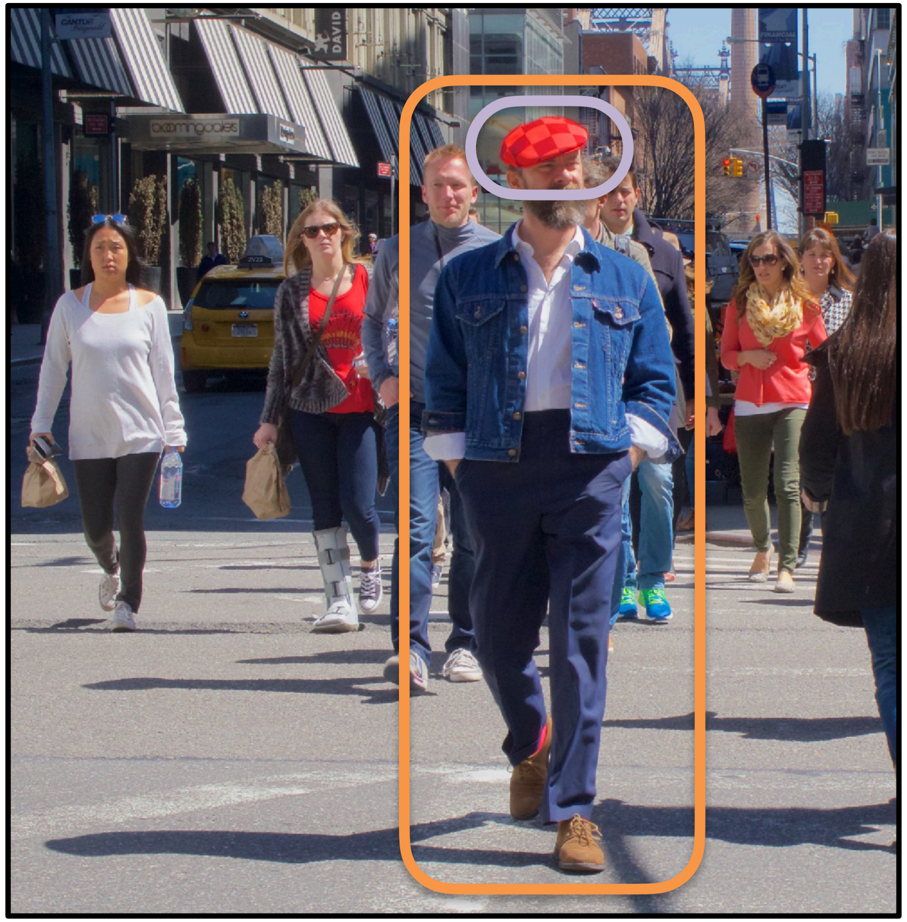
Language Grounding

“Bob is wearing a hat” $\longrightarrow$ $\exists\, x\, .\, \mathrm{hat}(x) \wedge \mathrm{wears}(\mathrm{bob}, x)$



Language Grounding

“Bob is wearing a hat” $\longrightarrow$ $\exists\, x \,.\, \text{hat}(x) \wedge \text{wears}(\text{bob}, x)$



x

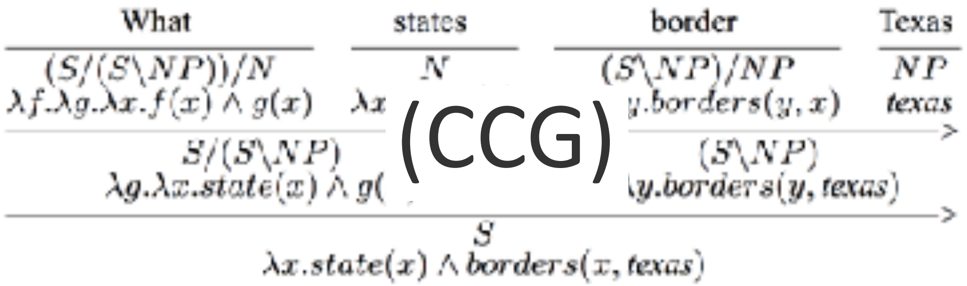
bob

Representation Learning

Representing language $\phi_l(x)$

“Bob is wearing a hat”

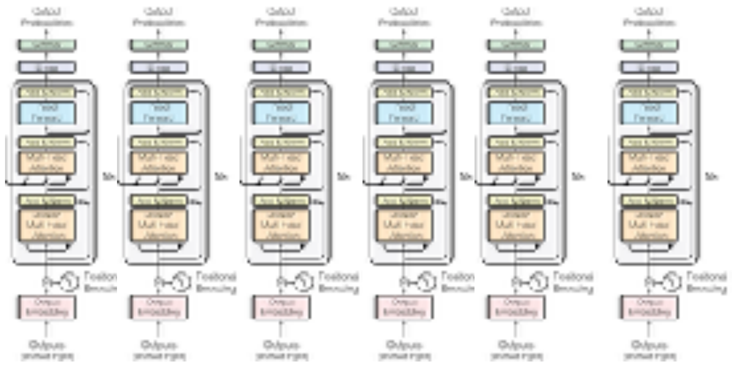
↓



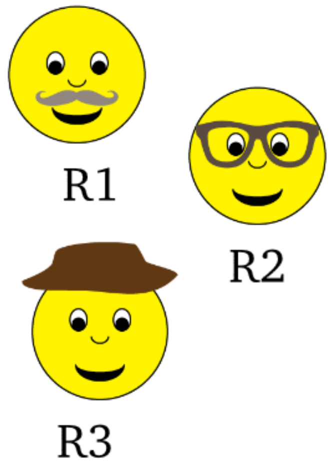
(CCG)

↓

∃ x . hat(x) ∧ wears(bob, x)

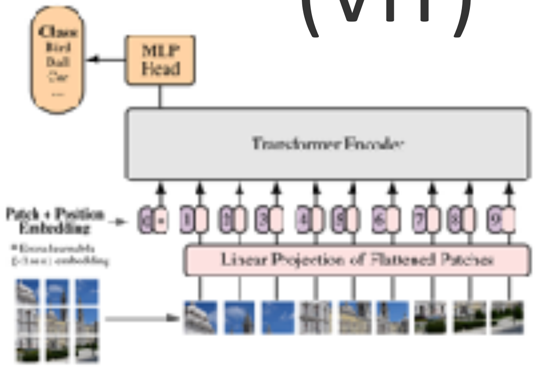


Representing the world $\phi_w(i)$



	mustache?	glasses?	hat?
R1	yes	no	no
R2	no	yes	no
R3	no	no	yes

(ViT)

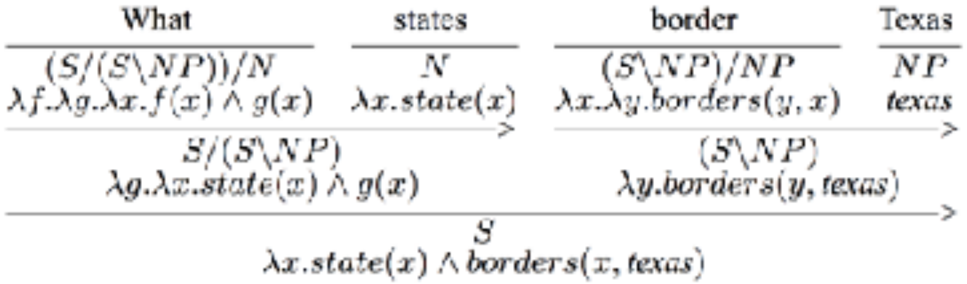

Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
Linear Projection of Flattened Patches

Grounding

Representing language $\phi_l(x)$

“Bob is wearing a hat”

↓



What: $(S/(S\backslash NP))/N$, $\lambda f.\lambda g.\lambda x.f(x) \land g(x)$

states: N, $\lambda x.state(x)$

border: $(S\backslash NP)/NP$, $\lambda x.\lambda y.borders(y,x)$

Texas: NP, $texas$

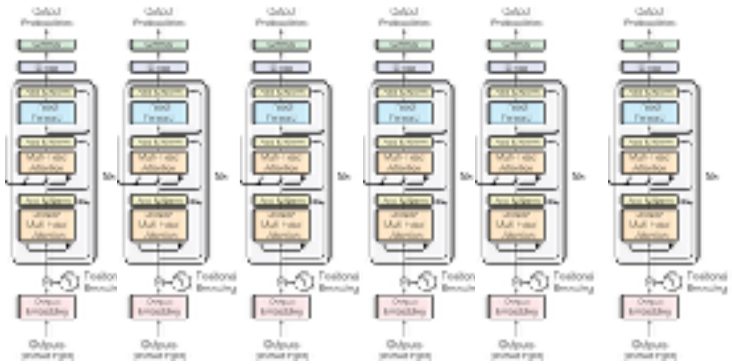
$\xrightarrow{>}$ $S/(S\backslash NP)$, $\lambda g.\lambda x.state(x) \land g(x)$

$\xrightarrow{>}$ $(S\backslash NP)$, $\lambda y.borders(y, texas)$

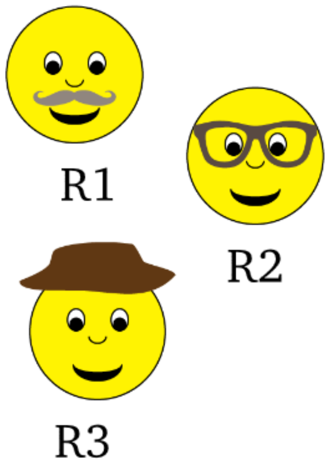
$\xrightarrow{>}$ S, $\lambda x.state(x) \land borders(x, texas)$

↓

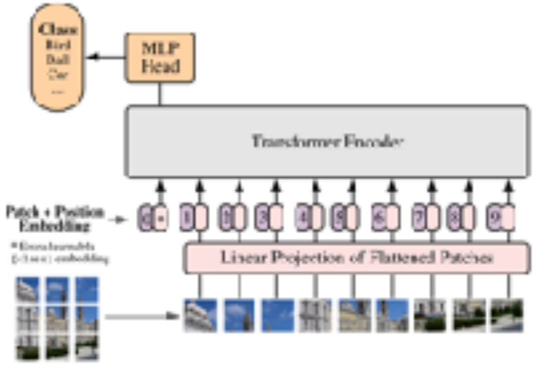
∃ x . hat(x) ∧ wears(bob, x)



Representing the world $\phi_w(i)$



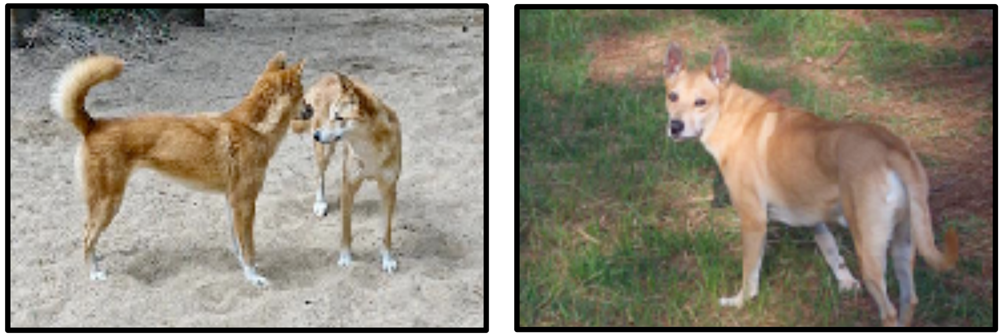
	mustache?	glasses?	hat?
R1	yes	no	no
R2	no	yes	no
R3	no	no	yes


Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
Linear Projection of Flattened Patches

Vision-Language Tasks

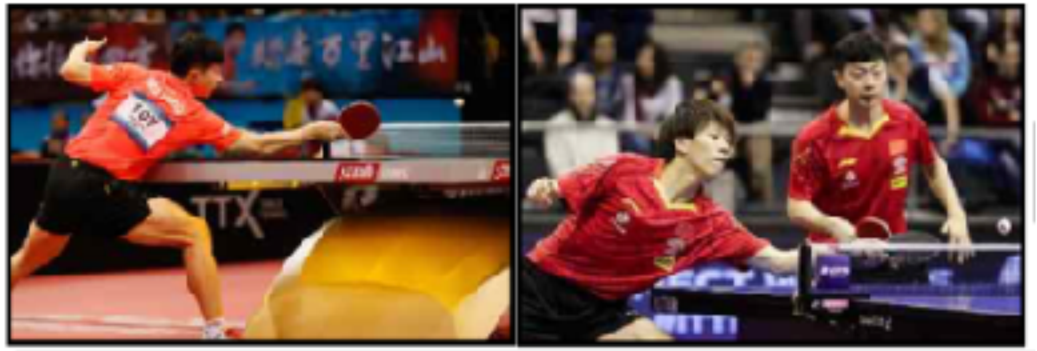
- Image-text entailment

$$\phi_l(x) \overset{?}{=} \phi_w(i)$$



The left image contains twice the number of dogs as the right image, and at least two dogs in total are standing.

[NLVR2, Suhr et al. 2019]



右图中的人在发球，左图中的人在接球。

[MaRVL, Liu Fangyu et al. 2021]

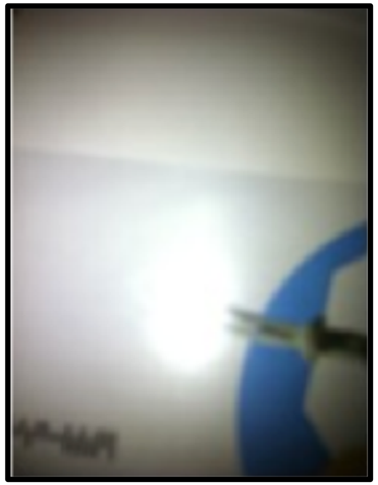
Vision-Language Tasks

- Image-text entailment
- Question answering

$$\mathrm{ask}(\phi_l(x), \phi_w(i))$$

Is this a vegetarian pizza?

[VQA, Antol et al. 2015]



Who is this mail for?

[VizWiz, Gurari et al. 2018]

Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning $\arg\max_x \mathrm{sim}(\phi_l(x), \phi_w(i))$

Concadia, Kreiss et al. 2023

How would you describe this image to someone who can't see it?

"Grocery store photo of several bunches of bananas."

Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning $\arg\max_{x} \mathrm{sim}(\phi_l(x), \phi_w(i))$

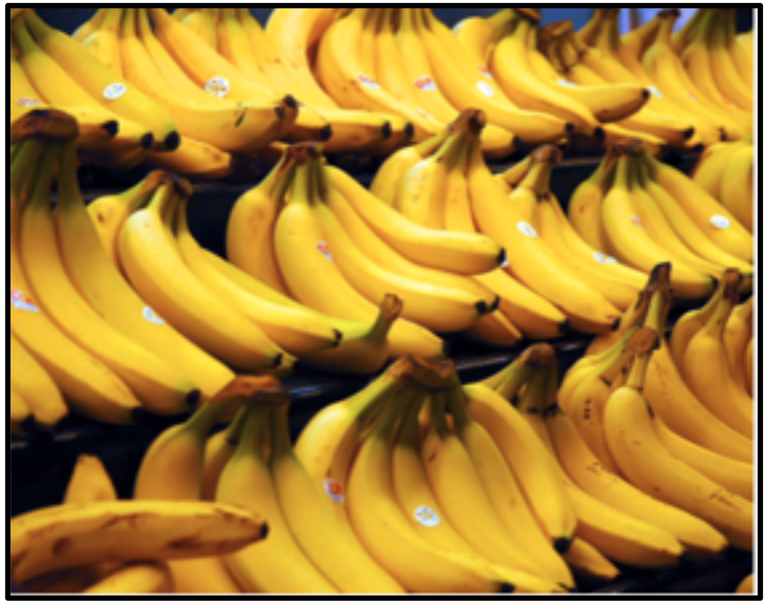
Concadia, Kreiss et al. 2023

What text should accompany this photo in a Wikipedia article about bananas?

“Grocery store photo of several bunches of bananas.”

Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning $\arg\max_{x} \text{sim}(\phi_l(x), \phi_w(i))$



Concadia, Kreiss et al. 2023

What text should accompany this photo in a Wikipedia article about bananas?

"Grocery store photo of several bunches of bananas."

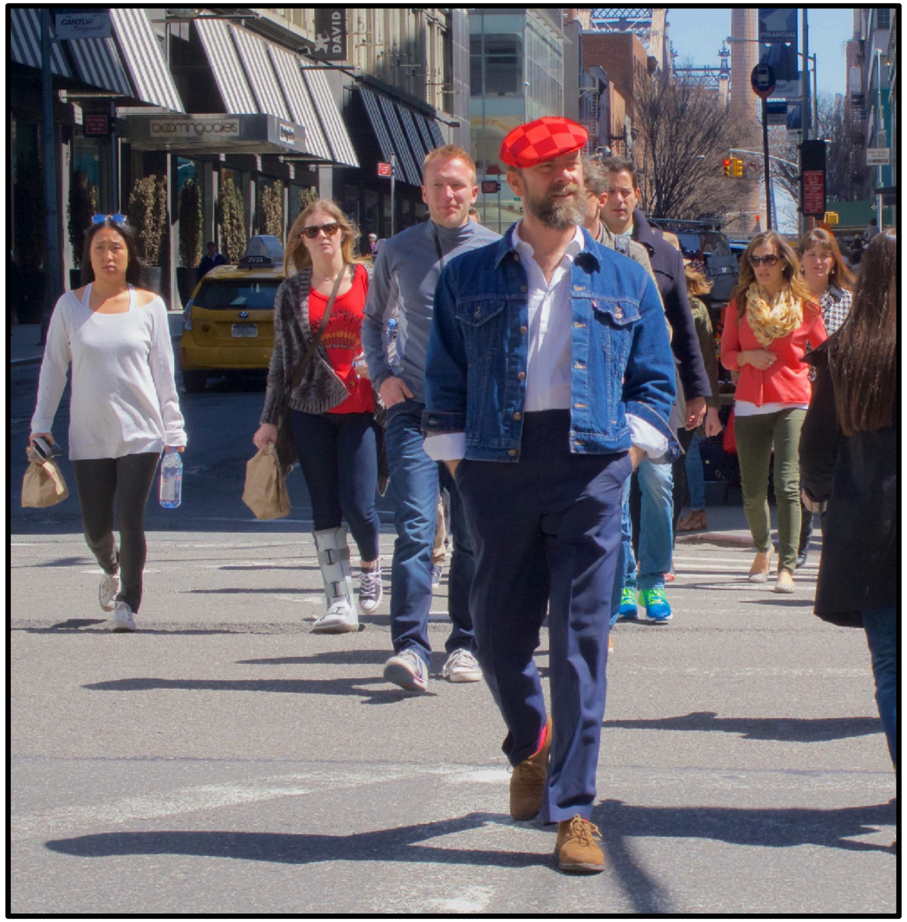
"Cavendish bananas are the main commercial banana cultivars sold in the world market."

Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning
- Referring expression resolution

$$\arg\max_{i' \in i} \operatorname{sim}(\phi_l(x), \phi_w(i'))$$

"The guy with the hat"

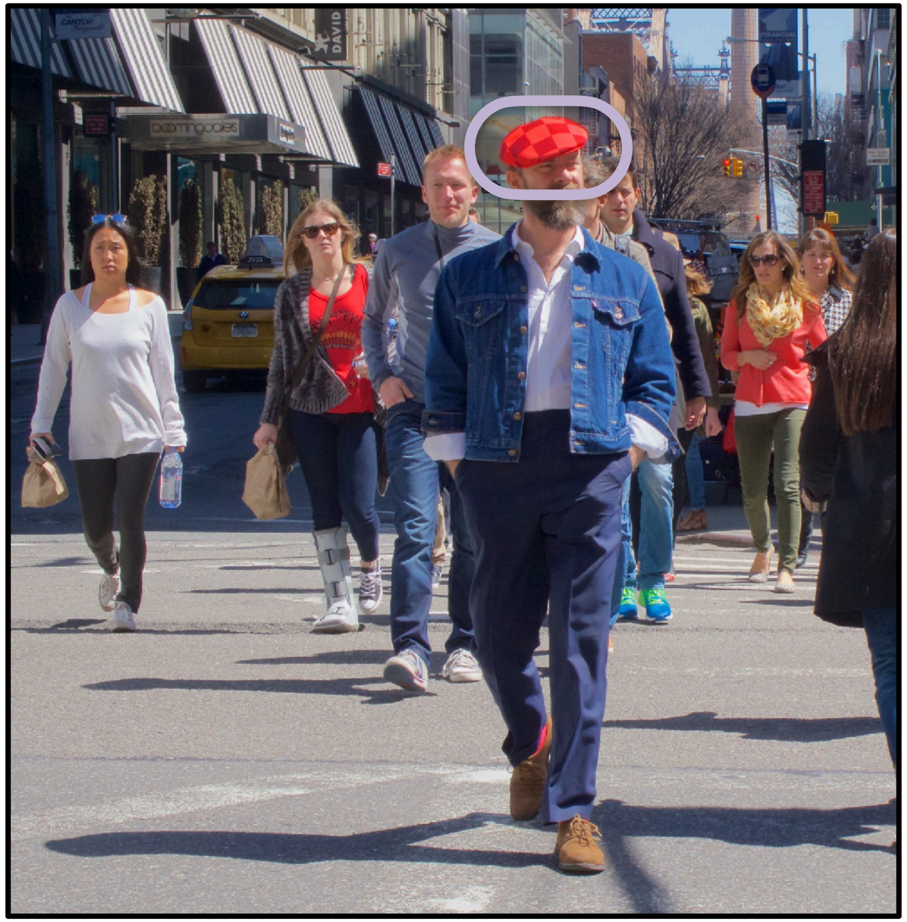


Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning
- Referring expression resolution

$$\arg\max_{i' \in i} \mathrm{sim}(\phi_l(x), \phi_w(i'))$$

"The guy with the hat"

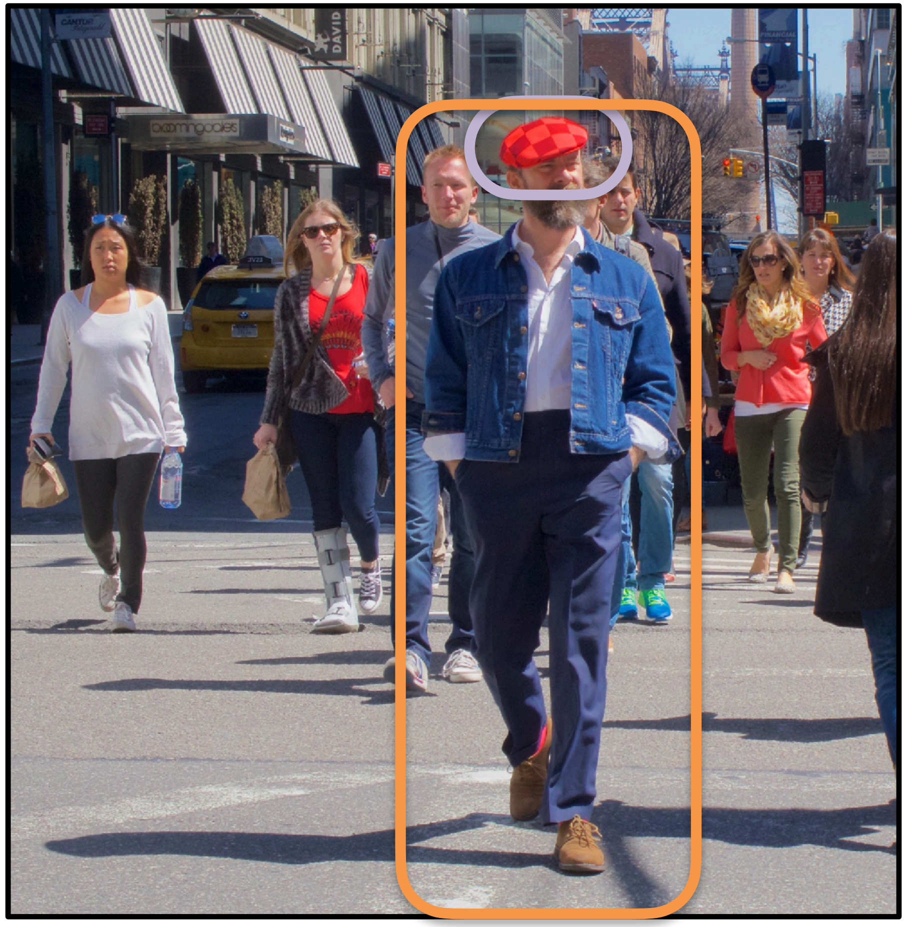


Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning
- Referring expression resolution

$$\arg\max_{i' \in i} \mathrm{sim}(\phi_l(x), \phi_w(i'))$$

"The guy with the hat"

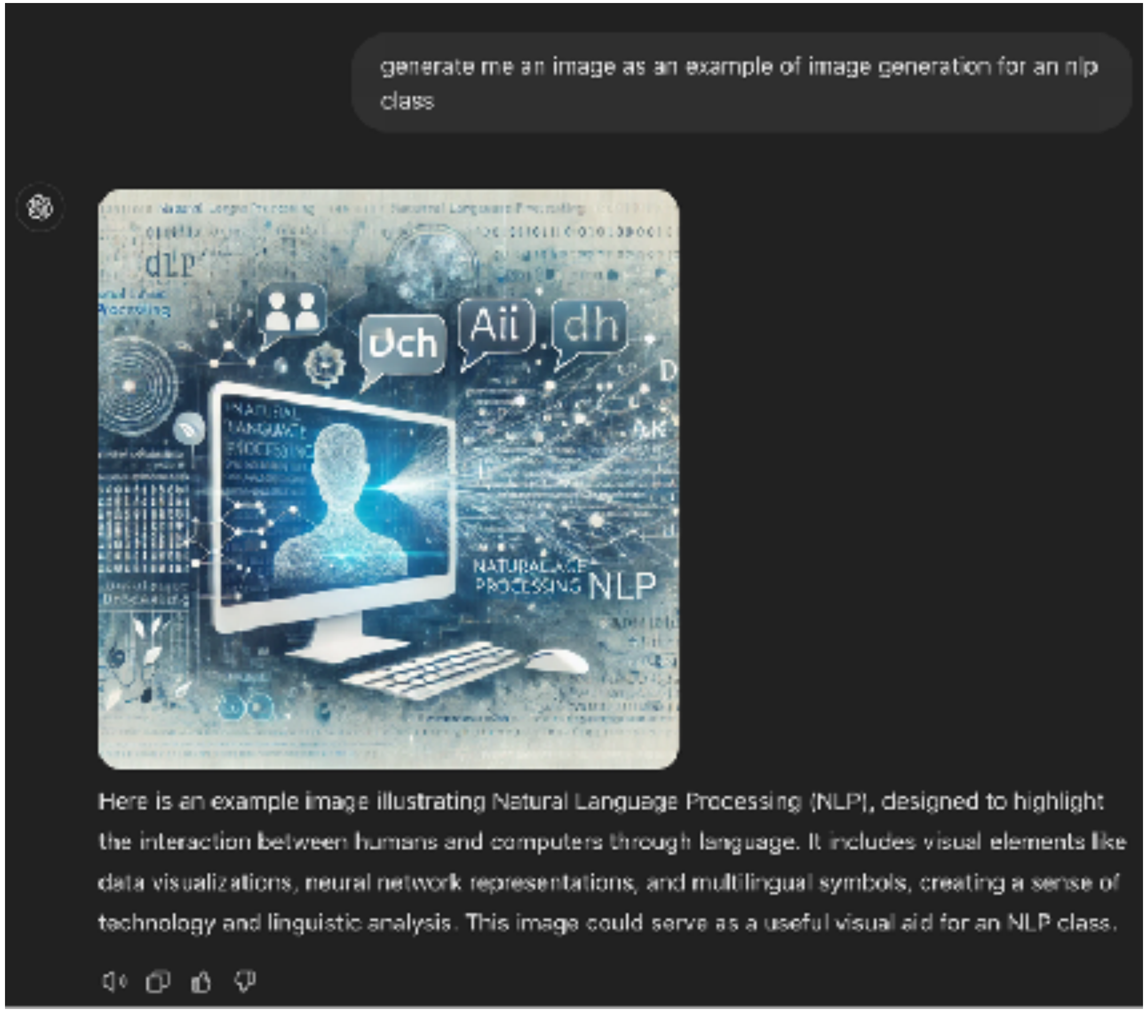


Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning
- Referring expression resolution
- Image generation

$$\arg\max_{i} \operatorname{sim}(\phi_l(x), \phi_w(i))$$

A visually appealing illustration representing Natural Language Processing (NLP). The image should include elements such as text flowing between a computer and a human figure, symbolizing communication between machines and people. The computer screen shows complex data visualizations, lines of text, and abstract nodes connected by lines, representing neural networks and linguistic analysis. A few floating speech bubbles with different languages and symbols can be present, emphasizing multilingual capabilities. The background should have a subtle digital theme with faint binary code and circuitry patterns, giving a sense of technology and AI. Use a modern, academic visual style with soft blue and white tones to convey a professional, educational atmosphere.



generate me an image as an example of image generation for an nlp class

Here is an example image illustrating Natural Language Processing (NLP), designed to highlight the interaction between humans and computers through language. It includes visual elements like data visualizations, neural network representations, and multilingual symbols, creating a sense of technology and linguistic analysis. This image could serve as a useful visual aid for an NLP class.

Vision-Language Tasks

- Image-text entailment
- Question answering
- Image captioning
- Referring expression resolution
- Image generation
- Conversational question answering
- Video question answering, captioning
- Instruction following

Methods

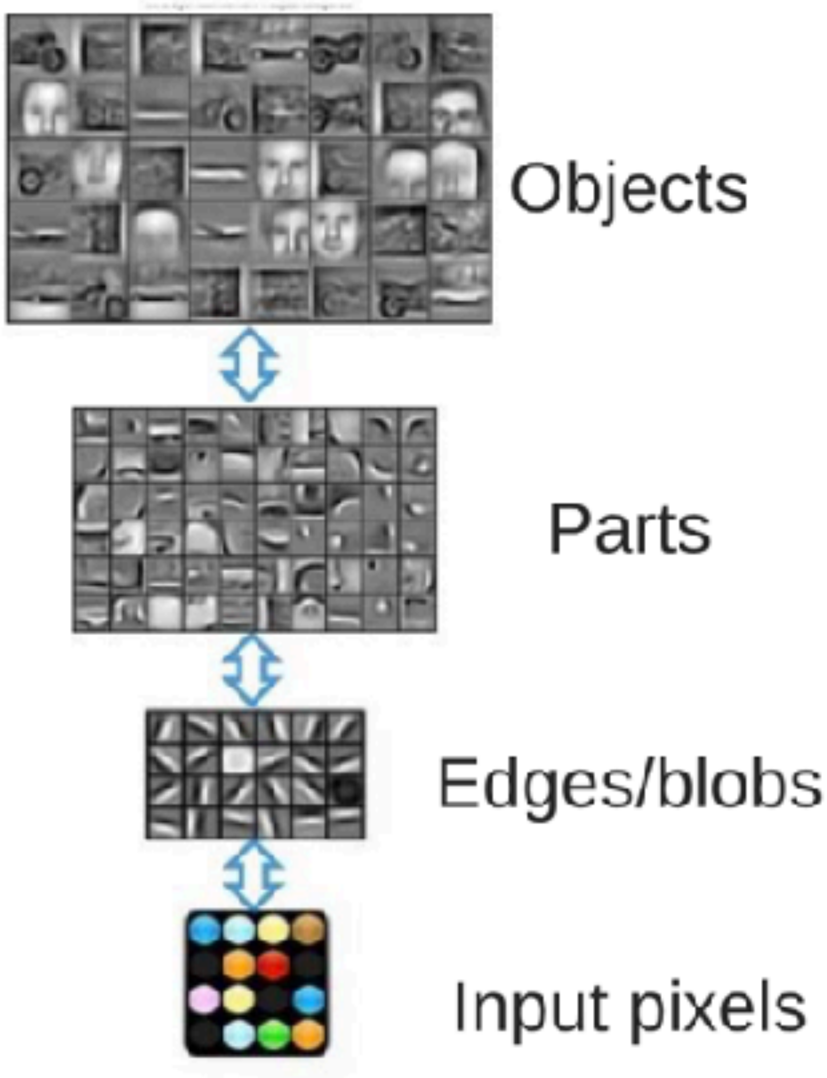
- Language representations
- Image representations
- Joint embedding spaces
- Multimodal transformers
- Neurosymbolic / code as a bottleneck

Image Representation: CNNs

Goal: building more abstract, hierarchical visual representations

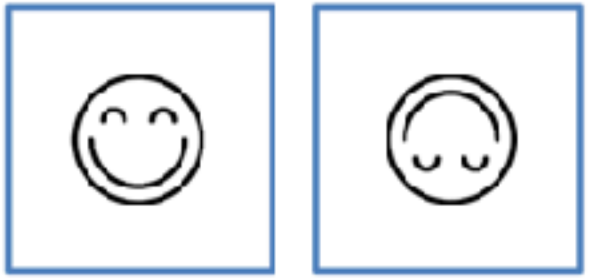
Key advantages:

1) Inspired from visual cortex
2) Encourages visual abstraction
3) Exploits *translation invariance*
4) Kernels/templates are learned
5) Fewer parameters than MLP



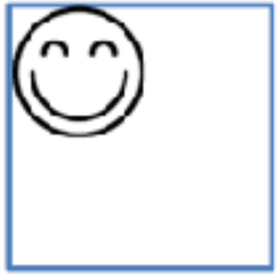
Slides from LP Morency

Image Representation: CNNs



2 Data Points – Which one is up?

- ➢ MLP can easily learn this task (possibly with only 1 neuron!)



What happens if the face is slightly translated?

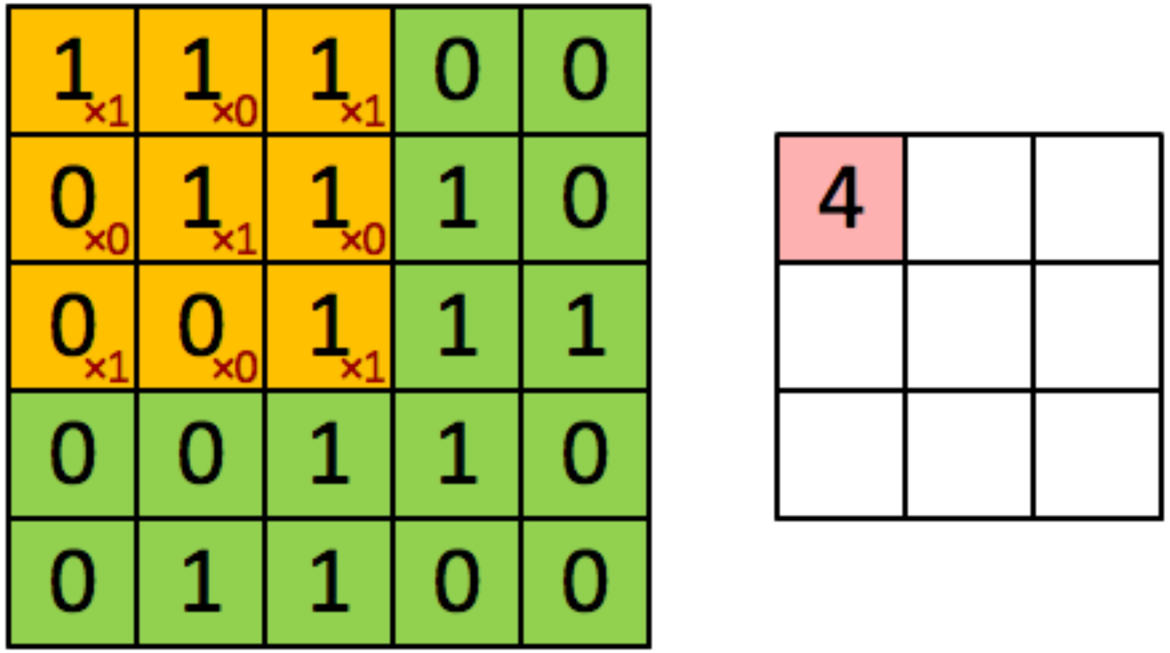
- ➢ The model should still be able to classify it

Conventional MLP models are not translation invariant!

- ➢ But CNNs are kernel-based, which helps with translation invariance and reduce number of parameters

Slides from LP Morency

CNNs: Convolution



1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

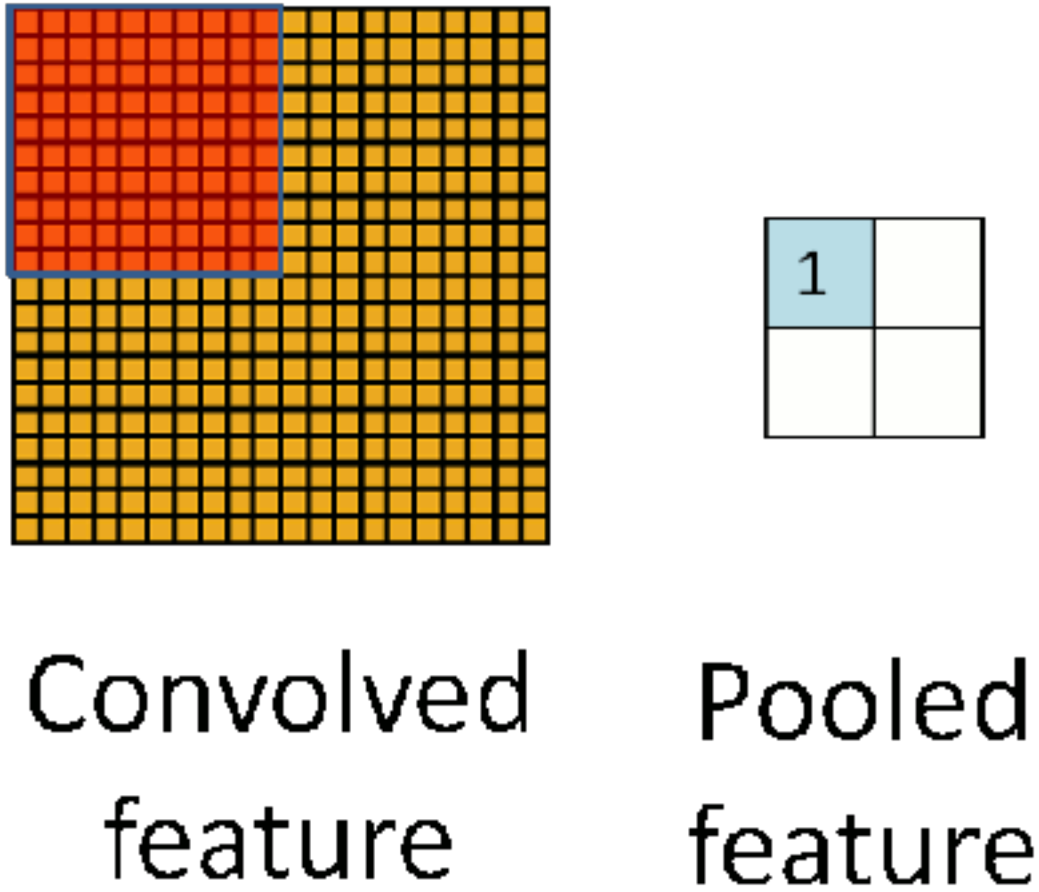
Convolved Feature

Filter:

1 0 1
0 1 0
1 0 1

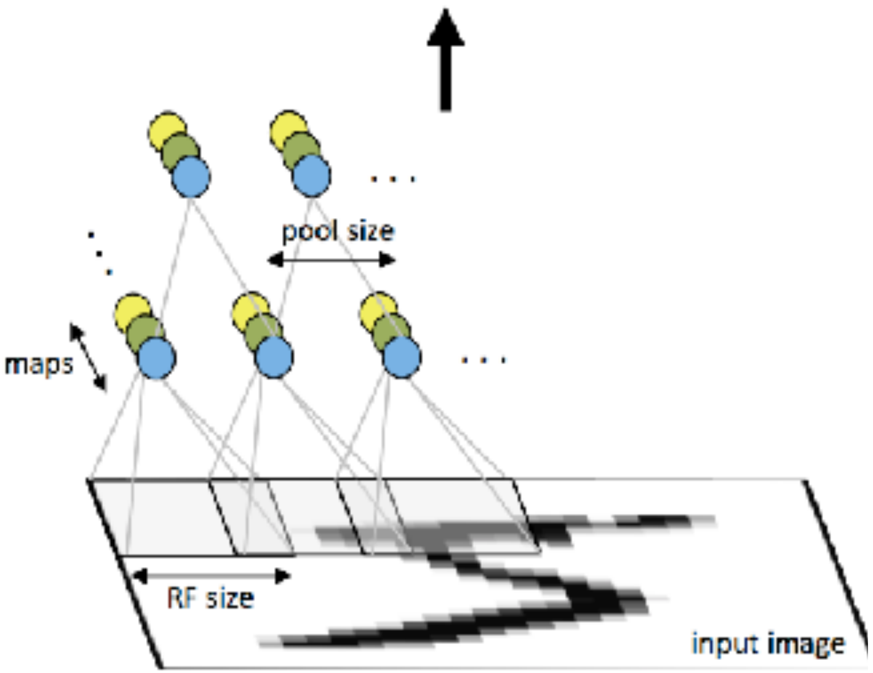
http://ufldl.stanford.edu/tutorial/

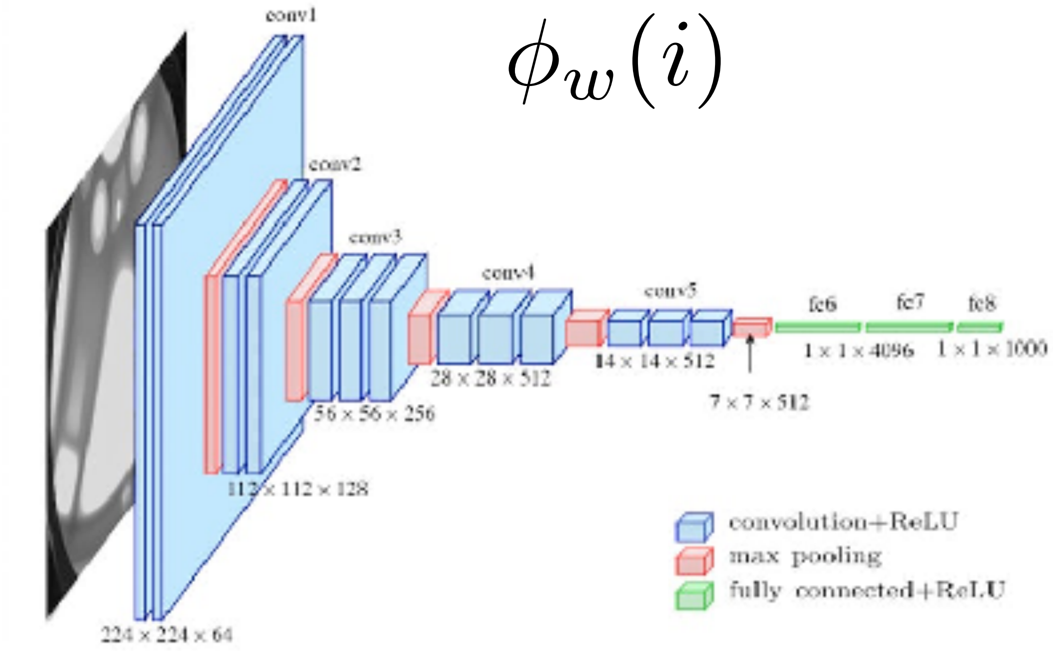
CNNs: Pooling



http://ufldl.stanford.edu/tutorial/

Multilayer CNNs





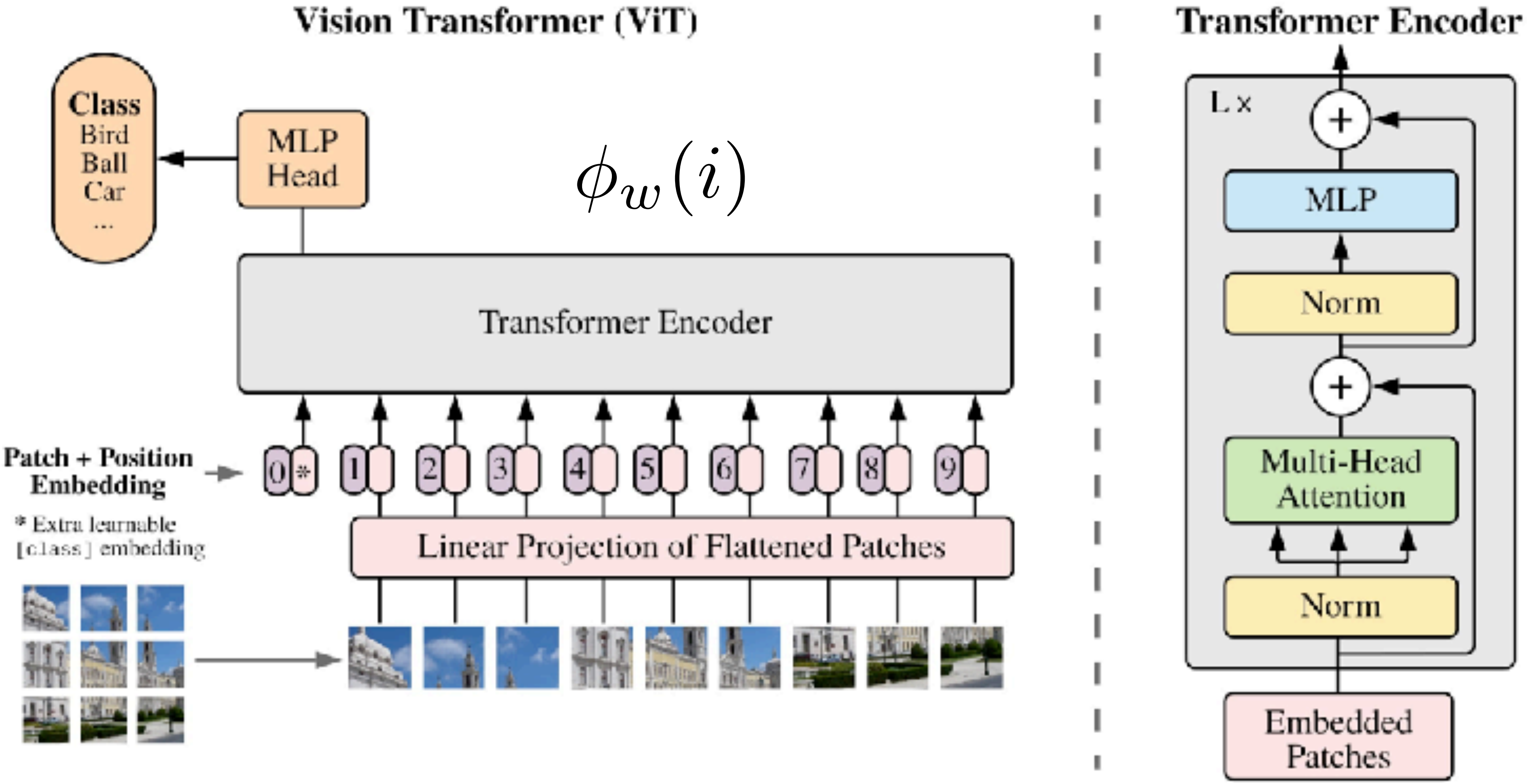
$$\phi_w(i)$$

http://ufldl.stanford.edu/tutorial/ , VGGNet (Simonyan et al. 2014)

CNN Explainer

https://poloclub.github.io/cnn-explainer/

Vision Transformer



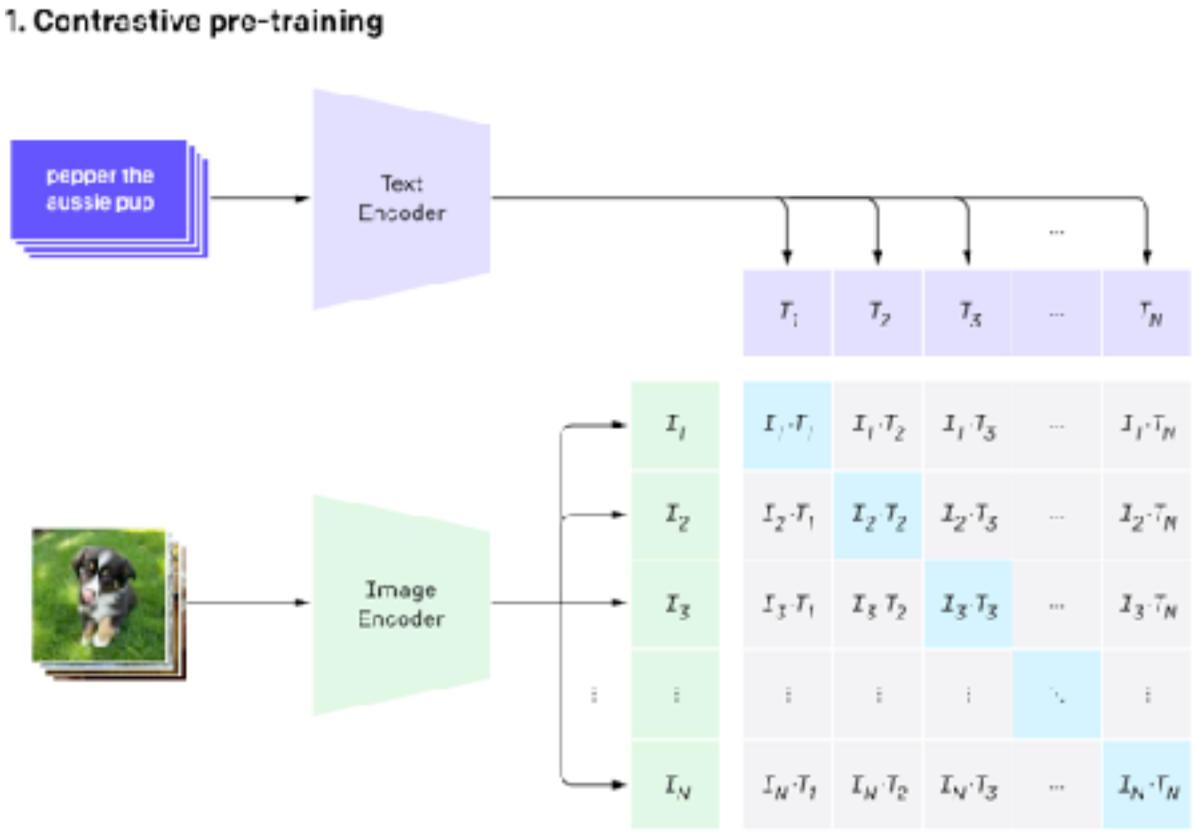
$$\phi_w(i)$$

ViT, Dosovitskiy et al. 2021

Image-Text Matching

- Idea: learn a shared representational space of images and text
- I.e., representation of sentence *x* paired with image *i* should be similar to one another $\phi_l(x) \approx \phi_w(i)$
- CLIP (Contrastive Language-Image Pre-Training), Radford et al. 2021

CLIP

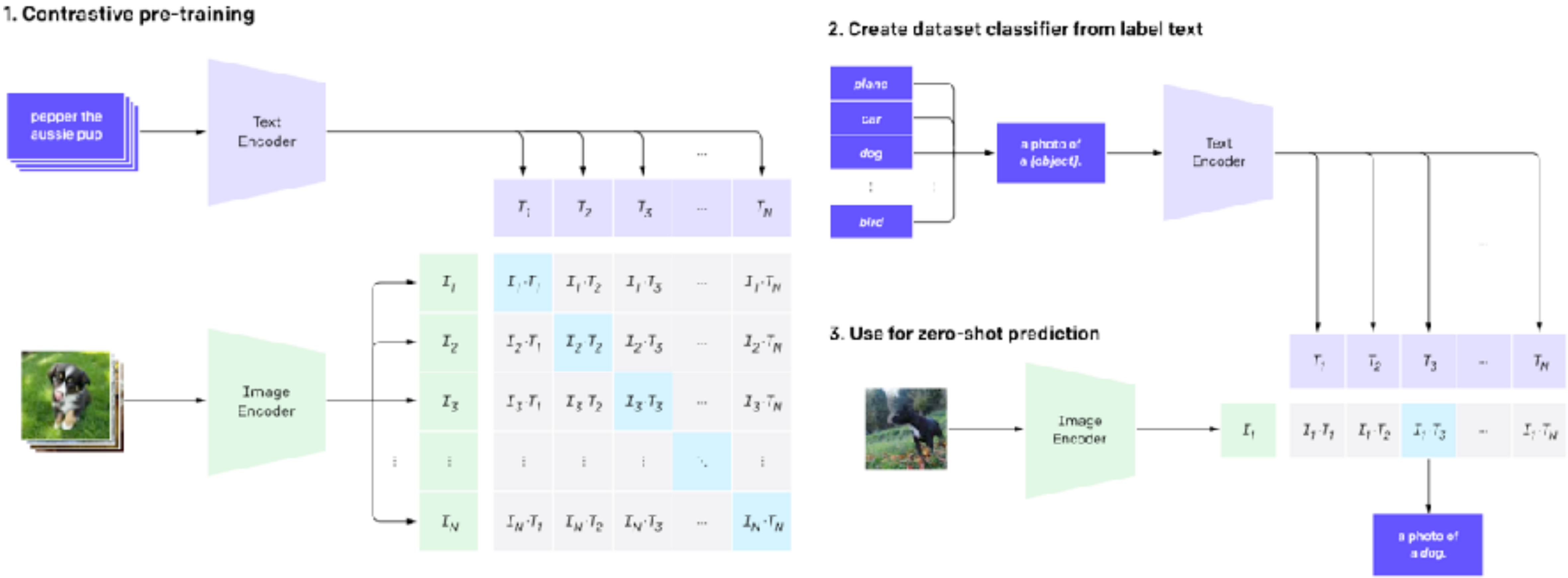


	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$	...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$	...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	⋱	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t) / 2
```

Radford et al. 2021

CLIP



1. Contrastive pre-training

2. Create dataset classifier from label text

3. Use for zero-shot prediction

	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$	...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$	...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	⋱	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

Radford et al. 2021

CLIP Applied: Image-Text Retrieval

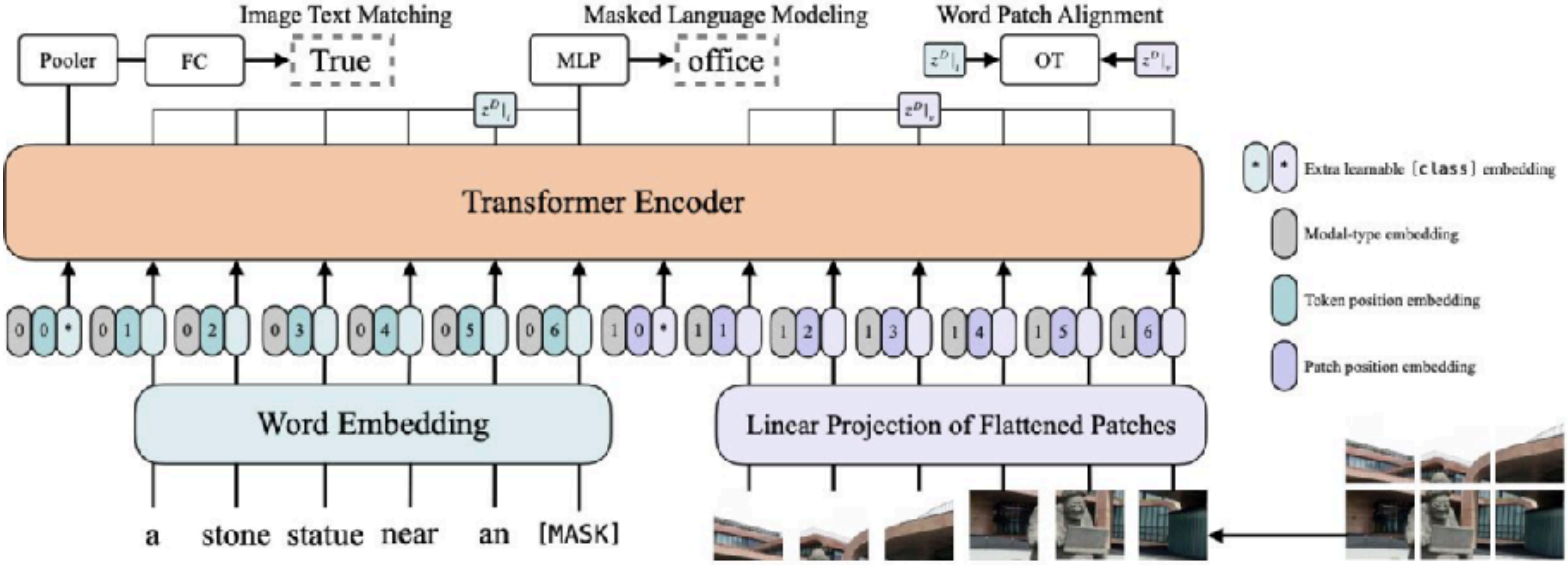
E.g., retrieve an image given a piece of text:

$$\arg\max_{i \in \mathcal{I}} \phi_l(x) \cdot \phi_w(i)$$

https://josephrocca.github.io/clip-image-sorter/

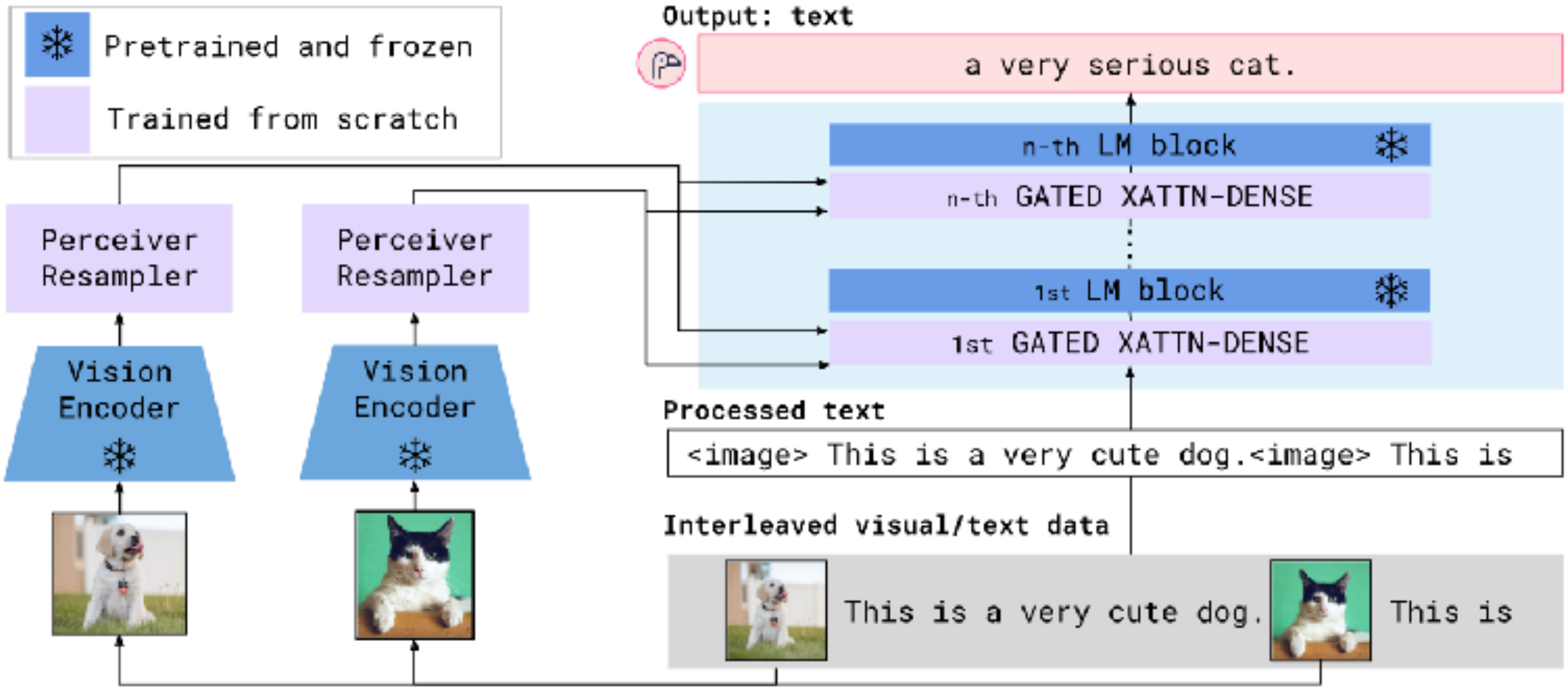
Joint Encoding: Multimodal Transformers

ViLT (Kim et al. 2021), encoder-only model (like BERT)



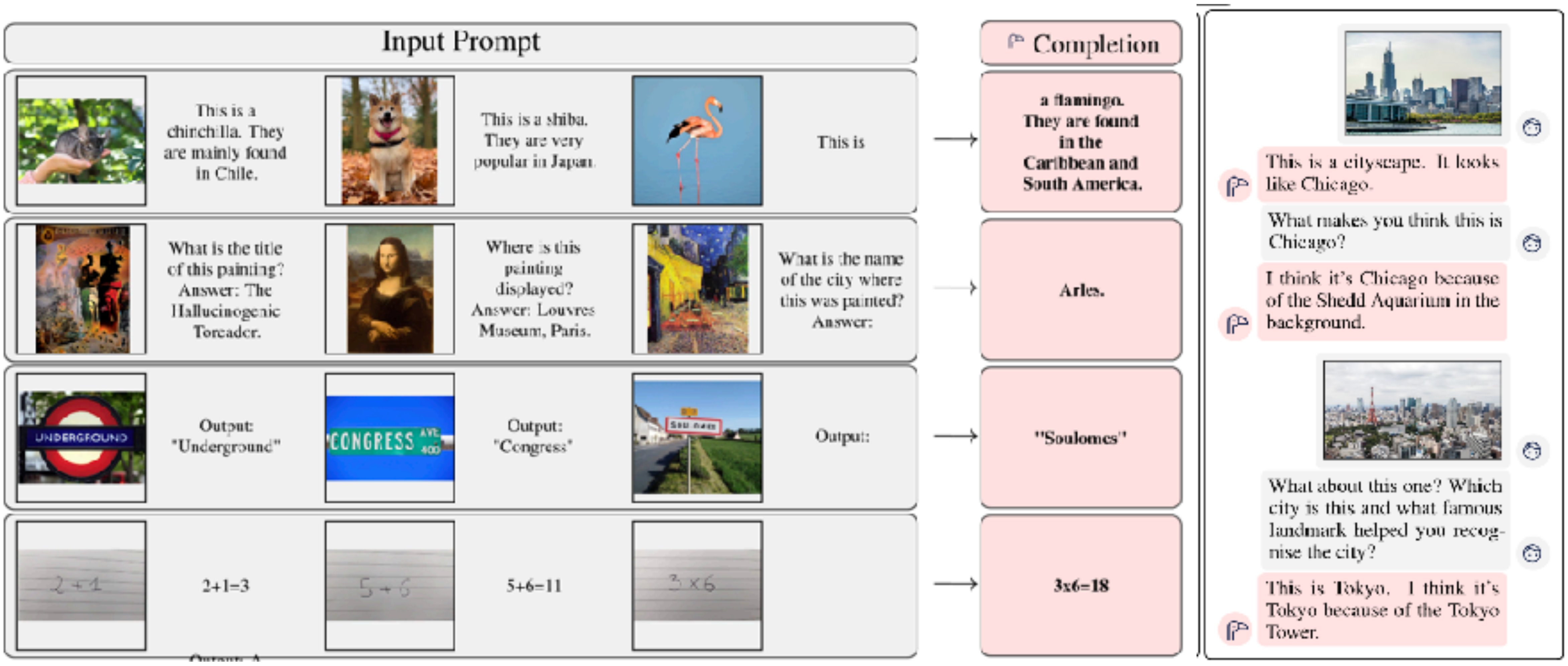
Joint Encoding: Multimodal Transformers

Flamingo, Alayrac et al. 2022



Joint Encoding: Multimodal Transformers

Flamingo, Alayrac et al. 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

This is a cityscape. It looks like Chicago.

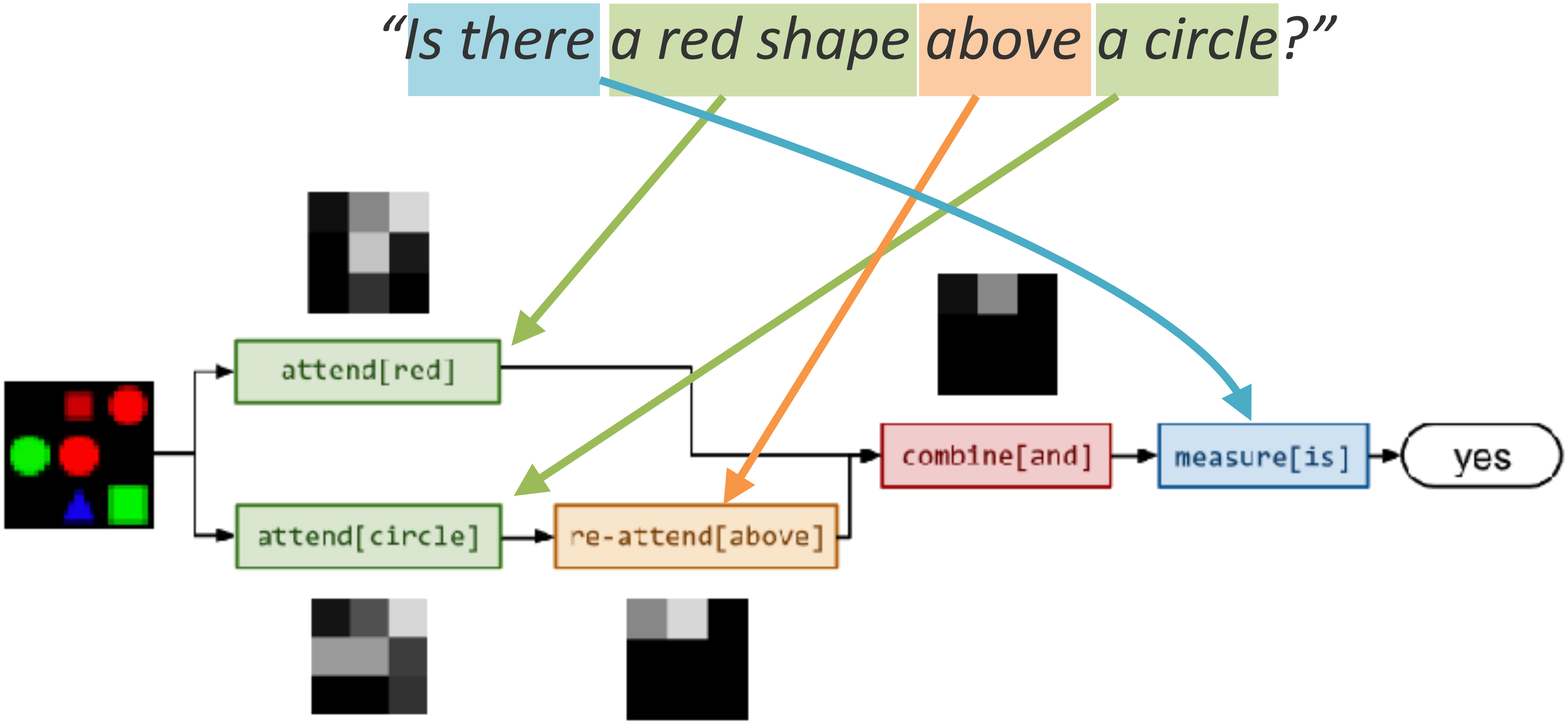
What makes you think this is Chicago?

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.

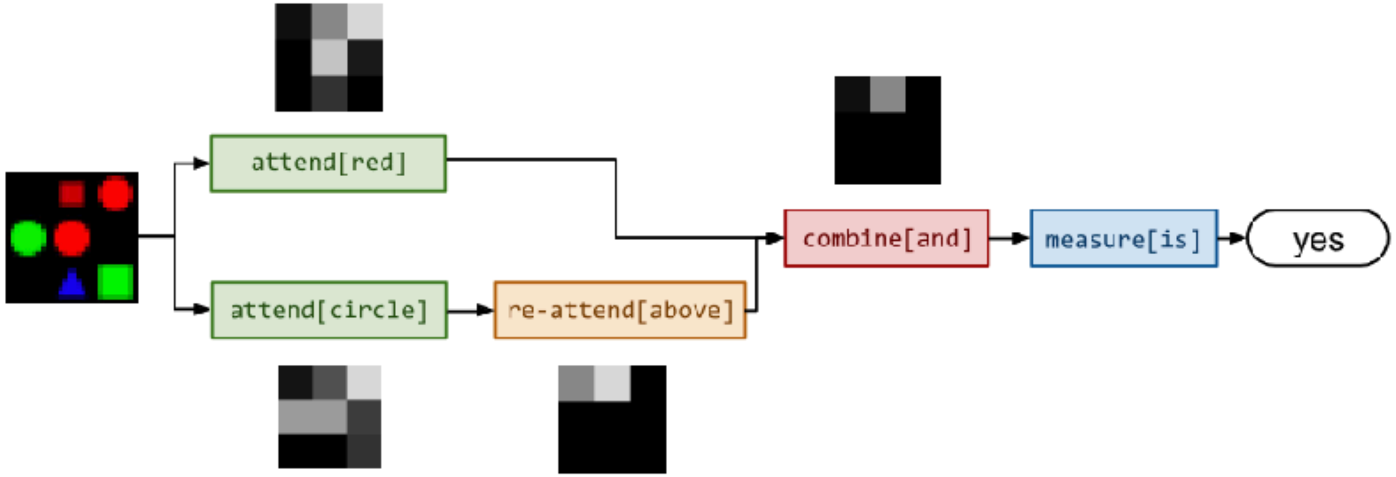
Neuromodular Approaches



Neural Module Networks, Andreas et al. 2017

Neuromodular Approaches

"Is there a red shape above a circle?"

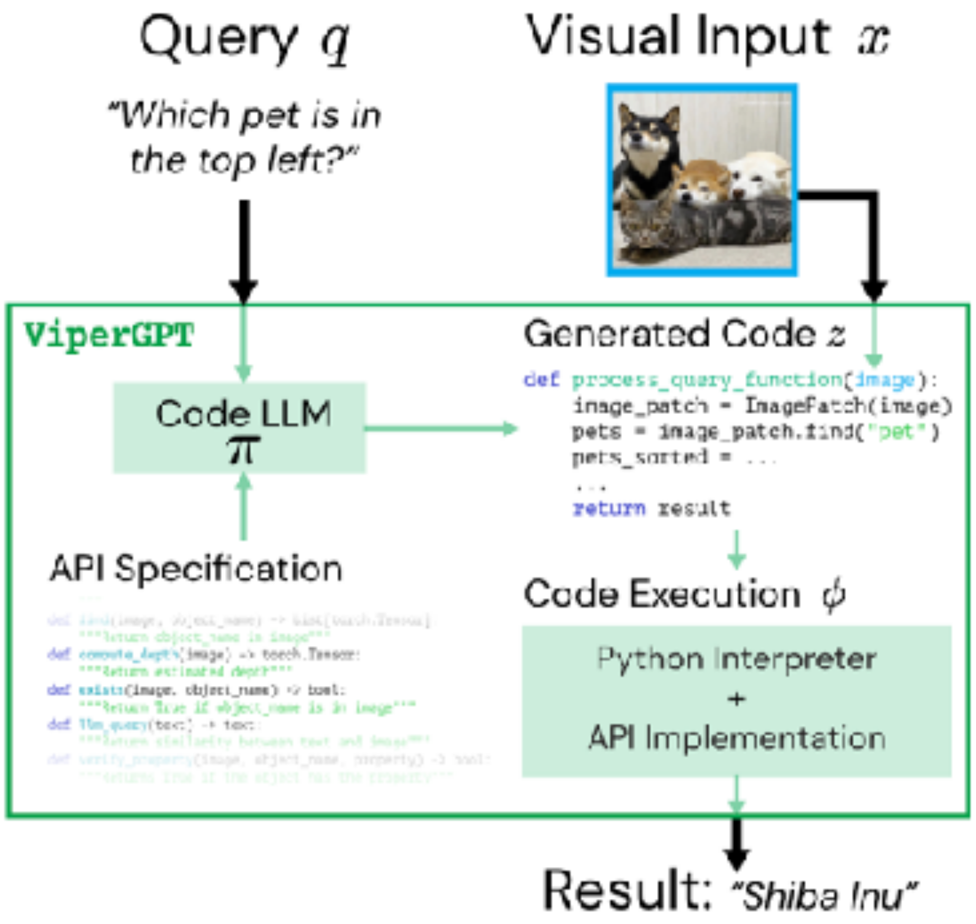


- Map *x* to some structured representation $\phi_l(x)$
- Manipulate image $\phi_w(i)$ according to components of this structured representation

Neural Module Networks, Andreas et al. 2017

Neuromodular Approaches

- Text representation: executable python code
- Image representation: pixels (also assume access to some computer vision algorithms)
- Grounding: executing python code on image representations

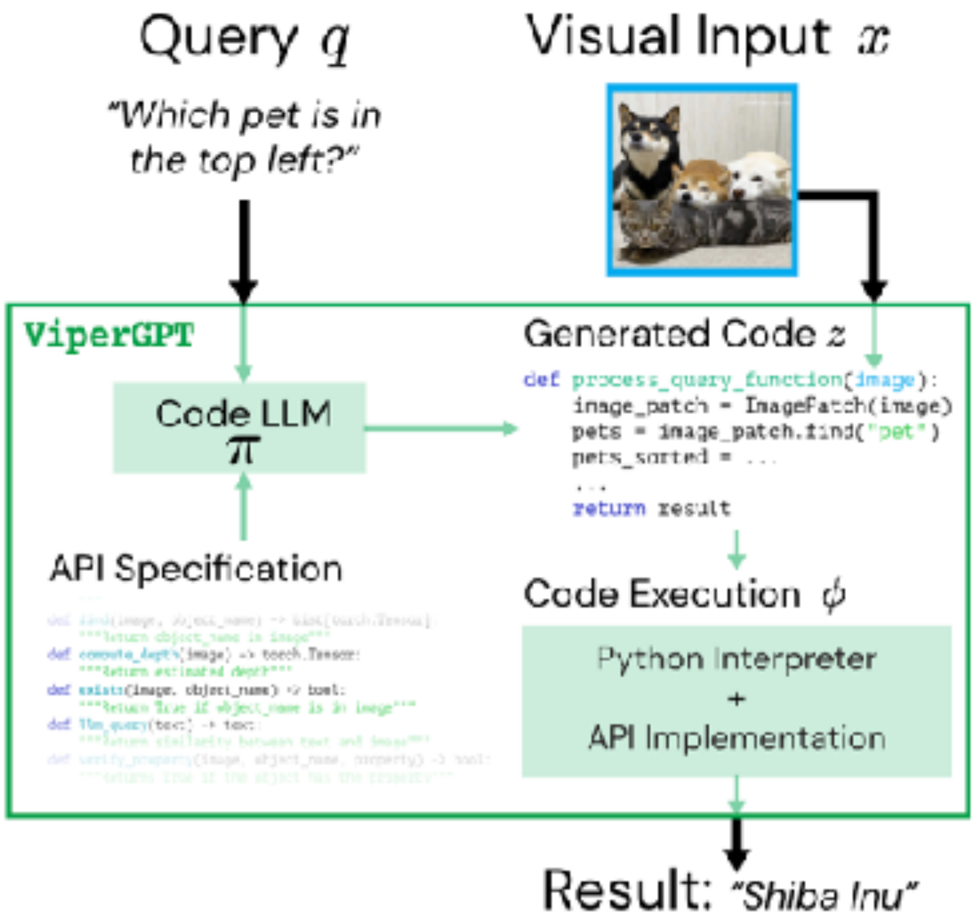


```python
"""
def find(image, object_name) -> List[torch.Tensor]:
    """Return object_name in image"""
def compute_depth(image) -> torch.Tensor:
    """Return estimated depth"""
def exists(image, object_name) -> bool:
    """Return True if object_name is in image"""
def llm_query(text) -> text:
    """Return similarity between text and image"""
def verify_property(image, object_name, property) -> bool:
    """Returns True if the object has the property"""
```

```python
def process_query_function(image):
    image_patch = ImagePatch(image)
    pets = image_patch.find("pet")
    pets_sorted = ...
    ...
    return result
```

Neuromodular Approaches

With sufficiently powerful code LLMs (e.g., Codex) and access to an API that can operate on top of images (or other modalities), no domain-specific or multimodal training is necessary



```
"""
def find(image, object_name) -> List[torch.Tensor]:
    """Return object_name in image"""
def compute_depth(image) -> torch.Tensor:
    """Return estimated depth"""
def exists(image, object_name) -> bool:
    """Return True if object_name is in image"""
def llm_query(text) -> text:
    """Return similarity between text and image"""
def verify_property(image, object_name, property) -> bool:
    """Returns True if the object has the property"""
```

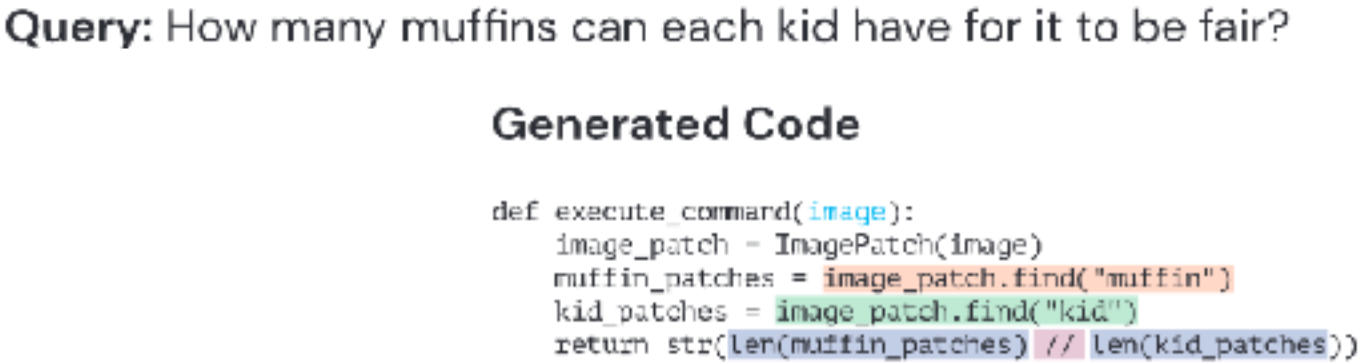
```
def process_query_function(image):
    image_patch = ImagePatch(image)
    pets = image_patch.find("pet")
    pets_sorted = ...
    ...
    return result
```

Neuromodular Approaches

Query: How many muffins can each kid have for it to be fair?

ViperGPT, Surís et al. 2023

Neuromodular Approaches



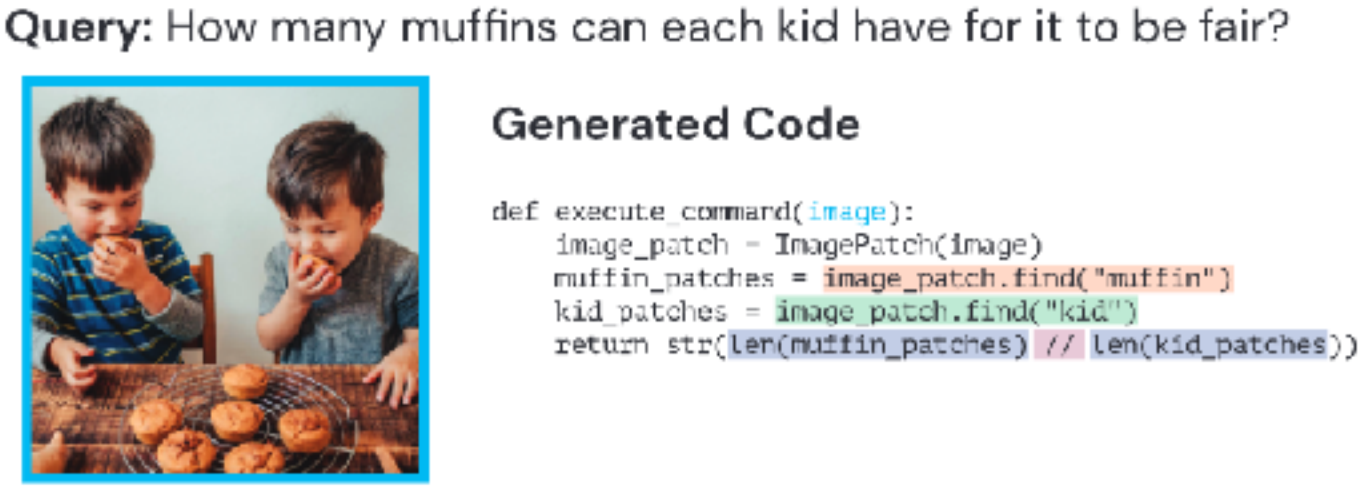
Query: How many muffins can each kid have for it to be fair?

Generated Code

```
def execute_command(image):
    image_patch = ImagePatch(image)
    muffin_patches = image_patch.find("muffin")
    kid_patches = image_patch.find("kid")
    return str(len(muffin_patches) // len(kid_patches))
```

ViperGPT, Surís et al. 2023

Neuromodular Approaches



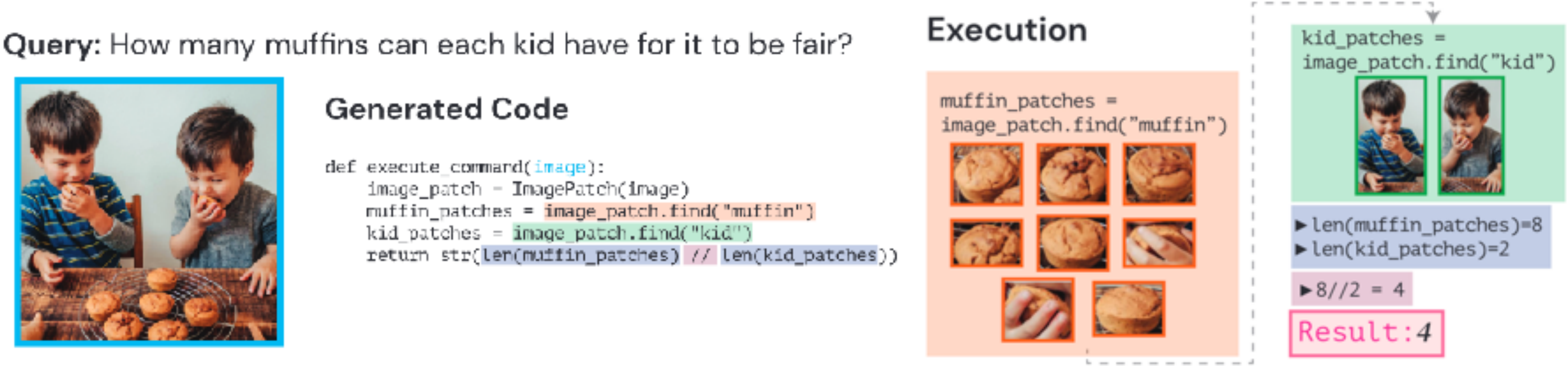
Query: How many muffins can each kid have for it to be fair?

Generated Code

```
def execute_command(image):
    image_patch = ImagePatch(image)
    muffin_patches = image_patch.find("muffin")
    kid_patches = image_patch.find("kid")
    return str(len(muffin_patches) // len(kid_patches))
```

ViperGPT, Surís et al. 2023

Neuromodular Approaches



Query: How many muffins can each kid have for it to be fair?

Generated Code

```
def execute_command(image):
    image_patch = ImagePatch(image)
    muffin_patches = image_patch.find("muffin")
    kid_patches = image_patch.find("kid")
    return str(len(muffin_patches) // len(kid_patches))
```

Execution

```
muffin_patches =
image_patch.find("muffin")
```

```
kid_patches =
image_patch.find("kid")
```

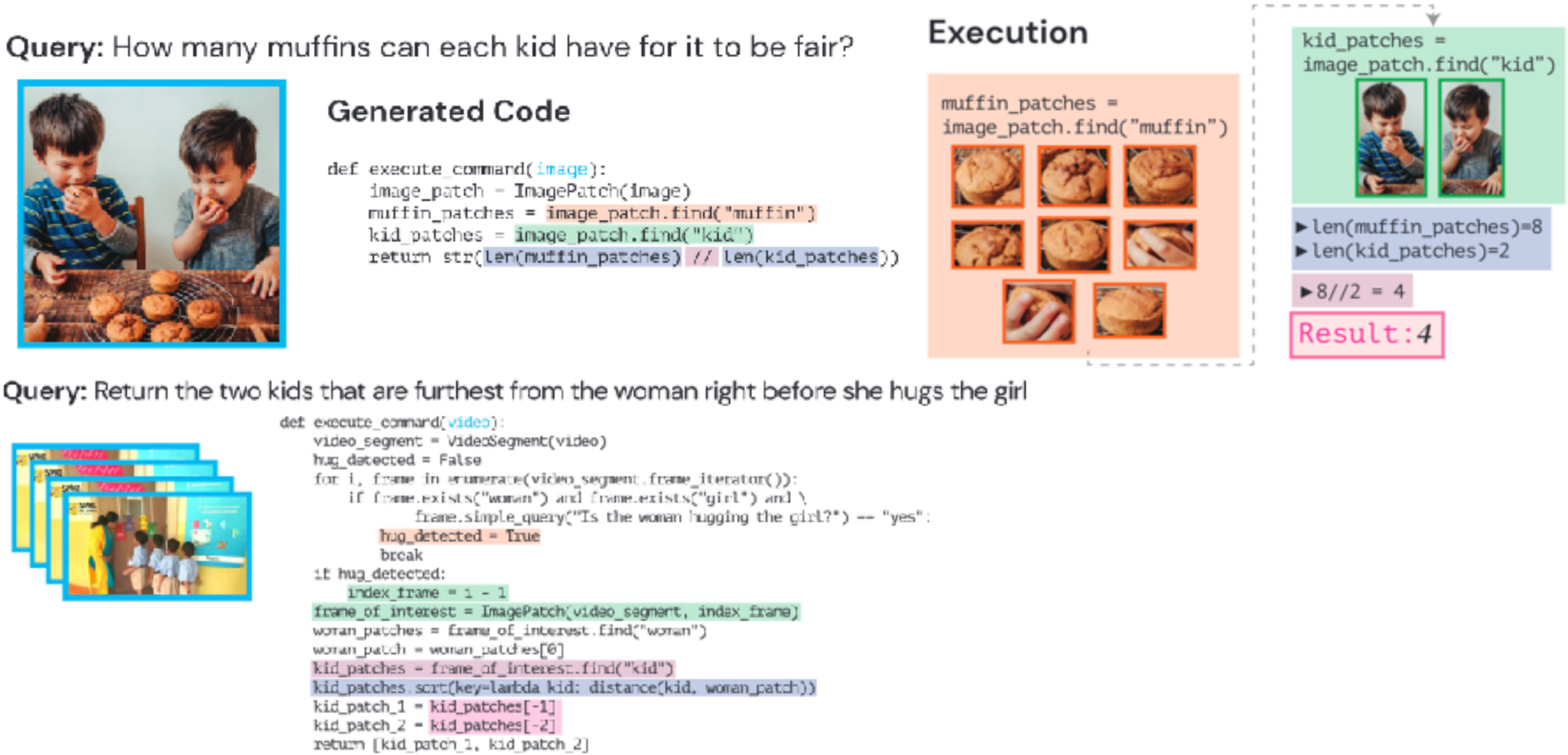
```
▶ len(muffin_patches)=8
▶ len(kid_patches)=2
```

```
▶8//2 = 4
```

```
Result:4
```

ViperGPT, Surís et al. 2023

Neuromodular Approaches



Query: How many muffins can each kid have for it to be fair?

Generated Code

```
def execute_command(image):
    image_patch = ImagePatch(image)
    muffin_patches = image_patch.find("muffin")
    kid_patches = image_patch.find("kid")
    return str(len(muffin_patches) // len(kid_patches))
```

Execution

```
muffin_patches =
image_patch.find("muffin")
```

```
kid_patches =
image_patch.find("kid")
```

```
▶len(muffin_patches)=8
▶len(kid_patches)=2
```

```
▶8//2 = 4
```

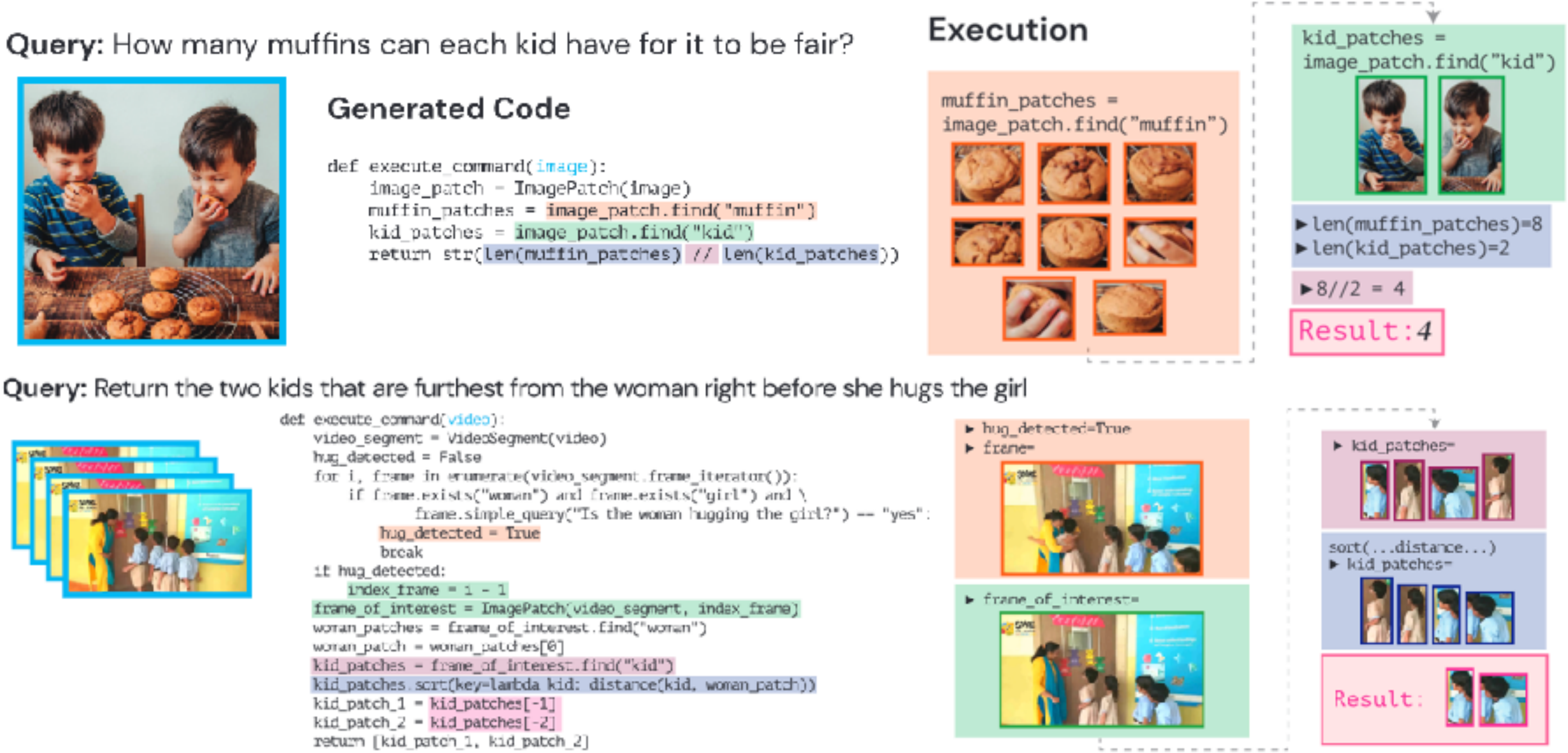
```
Result:4
```

Query: Return the two kids that are furthest from the woman right before she hugs the girl

```
def execute_command(video):
    video_segment = VideoSegment(video)
    hug_detected = False
    for i, frame in enumerate(video_segment.frame_iterator()):
        if frame.exists("woman") and frame.exists("girl") and \
                frame.simple_query("Is the woman hugging the girl?") == "yes":
            hug_detected = True
            break
    if hug_detected:
        index_frame = i - 1
    frame_of_interest = ImagePatch(video_segment, index_frame)
    woman_patches = frame_of_interest.find("woman")
    woman_patch = woman_patches[0]
    kid_patches = frame_of_interest.find("kid")
    kid_patches.sort(key=lambda kid: distance(kid, woman_patch))
    kid_patch_1 = kid_patches[-1]
    kid_patch_2 = kid_patches[-2]
    return [kid_patch_1, kid_patch_2]
```

ViperGPT, Surís et al. 2023

Neuromodular Approaches



Query: How many muffins can each kid have for it to be fair?

Generated Code

```
def execute_command(image):
    image_patch = ImagePatch(image)
    muffin_patches = image_patch.find("muffin")
    kid_patches = image_patch.find("kid")
    return str(len(muffin_patches) // len(kid_patches))
```

Execution

```
muffin_patches =
image_patch.find("muffin")
```

```
kid_patches =
image_patch.find("kid")
```

```
▶len(muffin_patches)=8
▶len(kid_patches)=2
```

```
▶8//2 = 4
```

Result:4

Query: Return the two kids that are furthest from the woman right before she hugs the girl

```
def execute_command(video):
    video_segment = VideoSegment(video)
    hug_detected = False
    for i, frame in enumerate(video_segment.frame_iterator()):
        if frame.exists("woman") and frame.exists("girl") and \
                frame.simple_query("Is the woman hugging the girl?") == "yes":
            hug_detected = True
            break
    if hug_detected:
        index_frame = i - 1
    frame_of_interest = ImagePatch(video_segment, index_frame)
    woman_patches = frame_of_interest.find("woman")
    woman_patch = woman_patches[0]
    kid_patches = frame_of_interest.find("kid")
    kid_patches.sort(key=lambda kid: distance(kid, woman_patch))
    kid_patch_1 = kid_patches[-1]
    kid_patch_2 = kid_patches[-2]
    return [kid_patch_1, kid_patch_2]
```

```
▶ hug_detected=True
▶ frame=
```

```
▶ frame_of_interest=
```

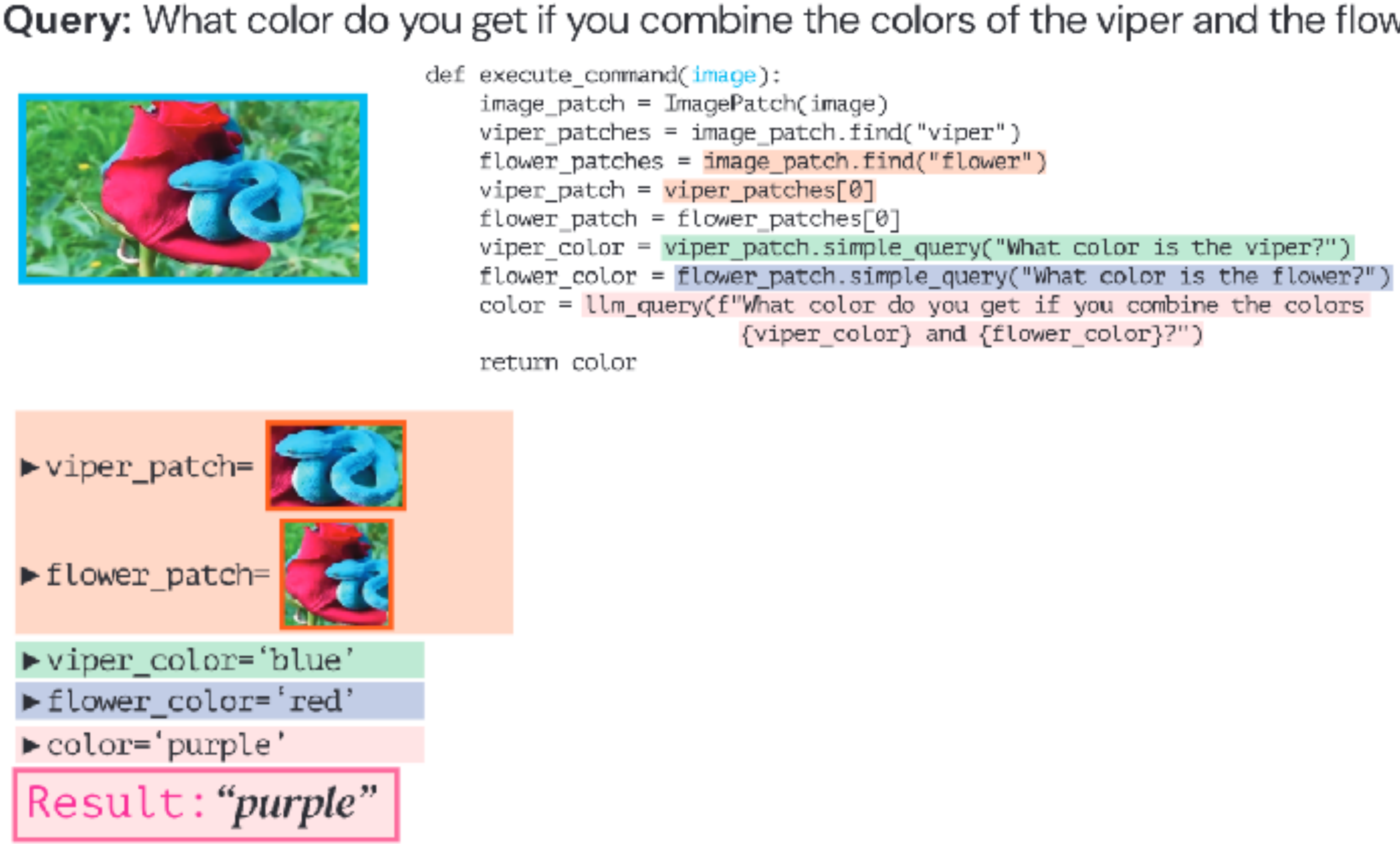
```
▶ kid_patches=
```

```
sort(...distance...)
▶ kid_patches=
```

Result:

ViperGPT, Surís et al. 2023

Neuromodular Approaches



Query: What color do you get if you combine the colors of the viper and the flower?

```
def execute_command(image):
    image_patch = ImagePatch(image)
    viper_patches = image_patch.find("viper")
    flower_patches = image_patch.find("flower")
    viper_patch = viper_patches[0]
    flower_patch = flower_patches[0]
    viper_color = viper_patch.simple_query("What color is the viper?")
    flower_color = flower_patch.simple_query("What color is the flower?")
    color = llm_query(f"What color do you get if you combine the colors
                        {viper_color} and {flower_color}?")
    return color
```

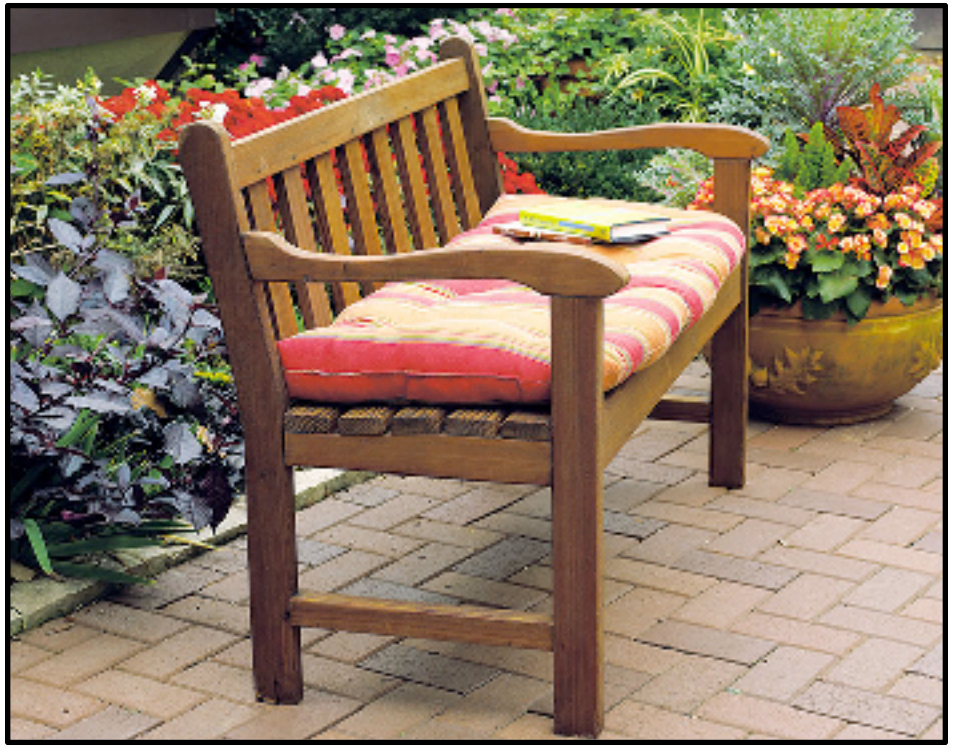
```
▶viper_patch=
▶flower_patch=
▶viper_color='blue'
▶flower_color='red'
▶color='purple'
```

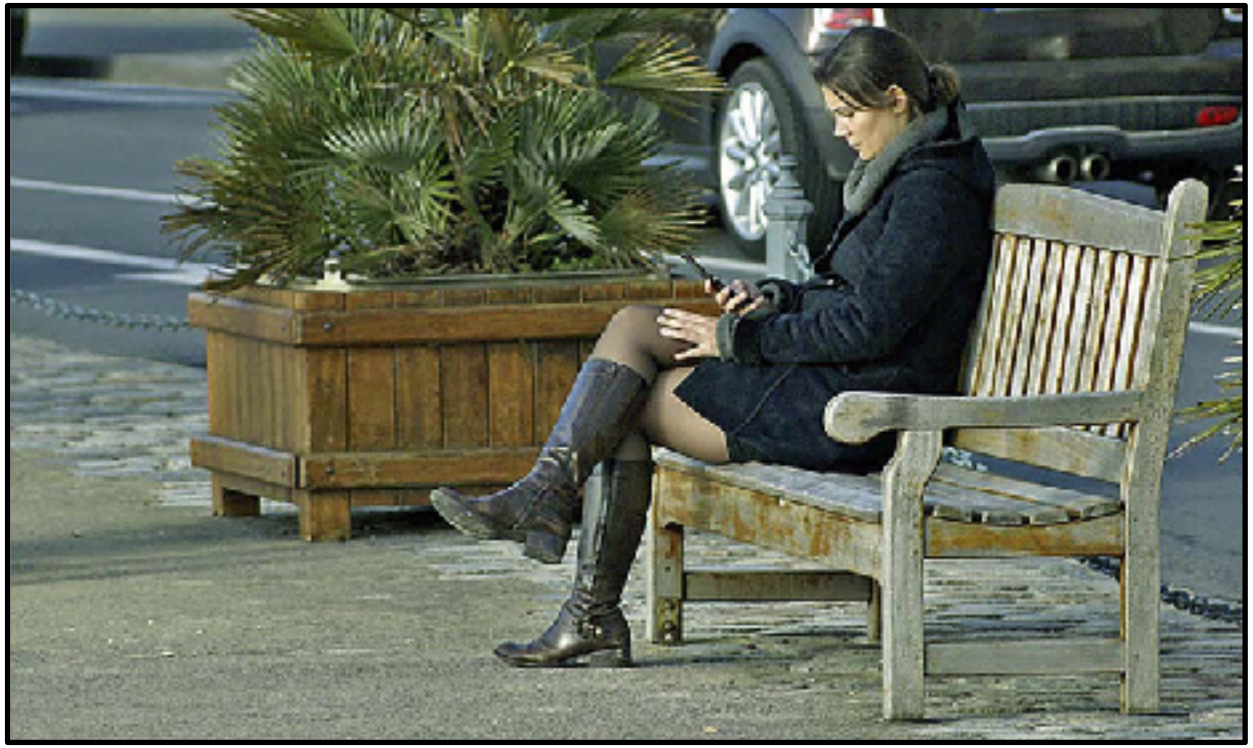
Result: *"purple"*

ViperGPT, Surís et al. 2023

Drawback: Context-Dependence

“Is the potted plant to the right of the bench?”



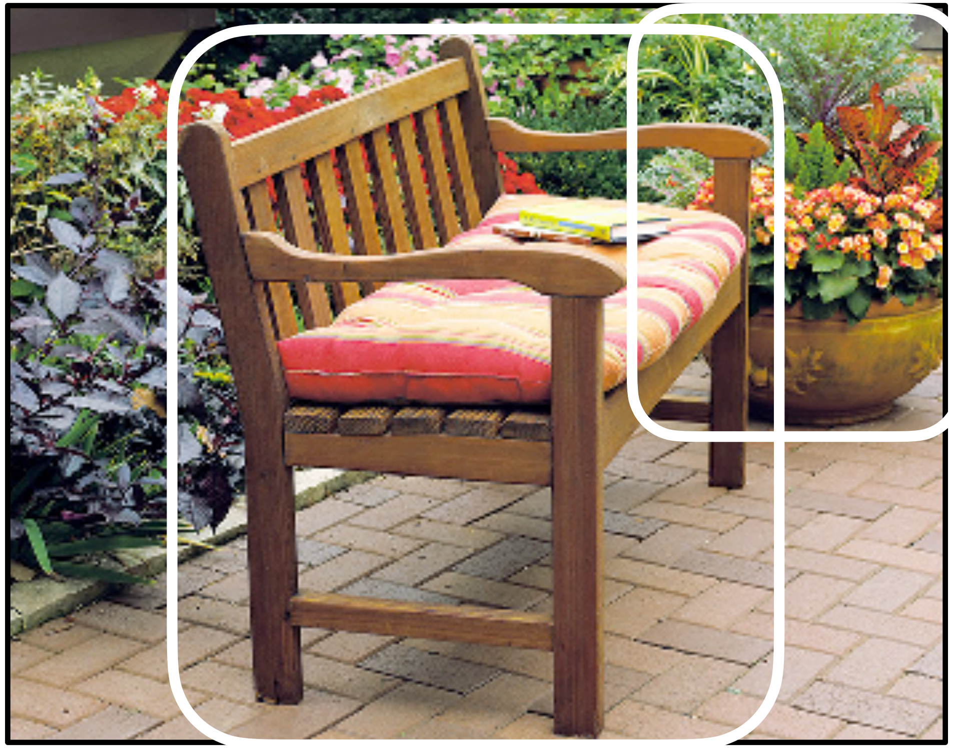


Visual Spatial Reasoning, Liu Fangyu et al. 2023

Drawback: Context-Dependence

“Is the potted plant to the right of the bench?”

```
bbox_plant = detect(image, “potted plant”)
bbox_bench = detect(image, “bench”)
return bbox_plant.x > bbox_bench.x
```

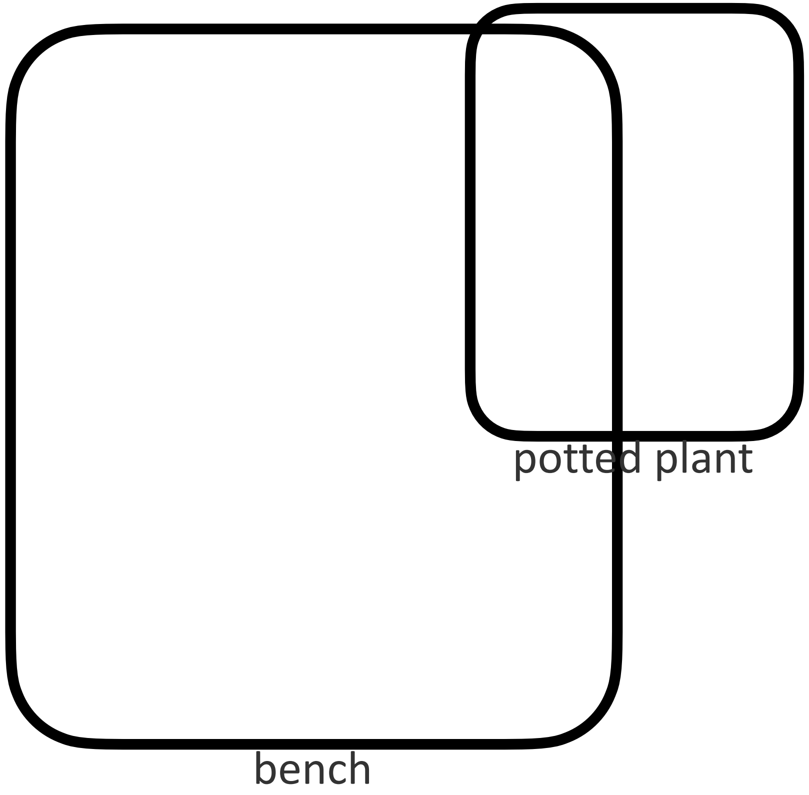


Visual Spatial Reasoning, Liu Fangyu et al. 2023

Drawback: Context-Dependence

“Is the potted plant to the right of the bench?”

```
bbox_plant = detect(image, “potted plant”)
bbox_bench = detect(image, “bench”)
return bbox_plant.x > bbox_bench.x
```



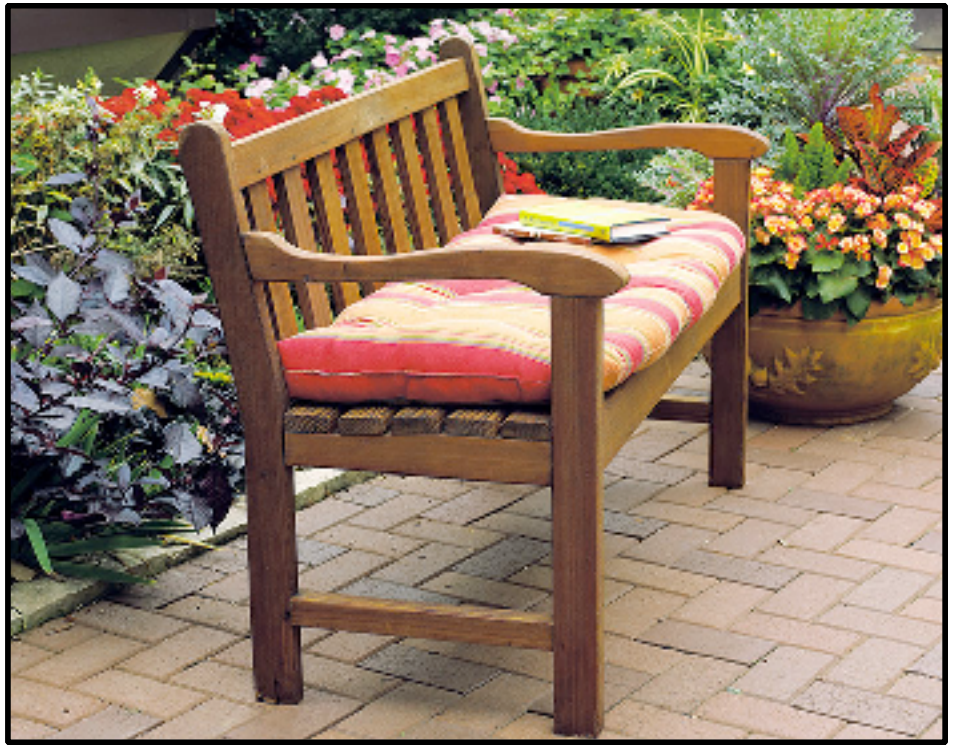
✓

Visual Spatial Reasoning, Liu Fangyu et al. 2023

Drawback: Context-Dependence

"Is the potted plant to the right of the bench?"

```
bbox_plant = detect(image, "potted plant")
bbox_bench = detect(image, "bench")
return bbox_plant.x > bbox_bench.x
```



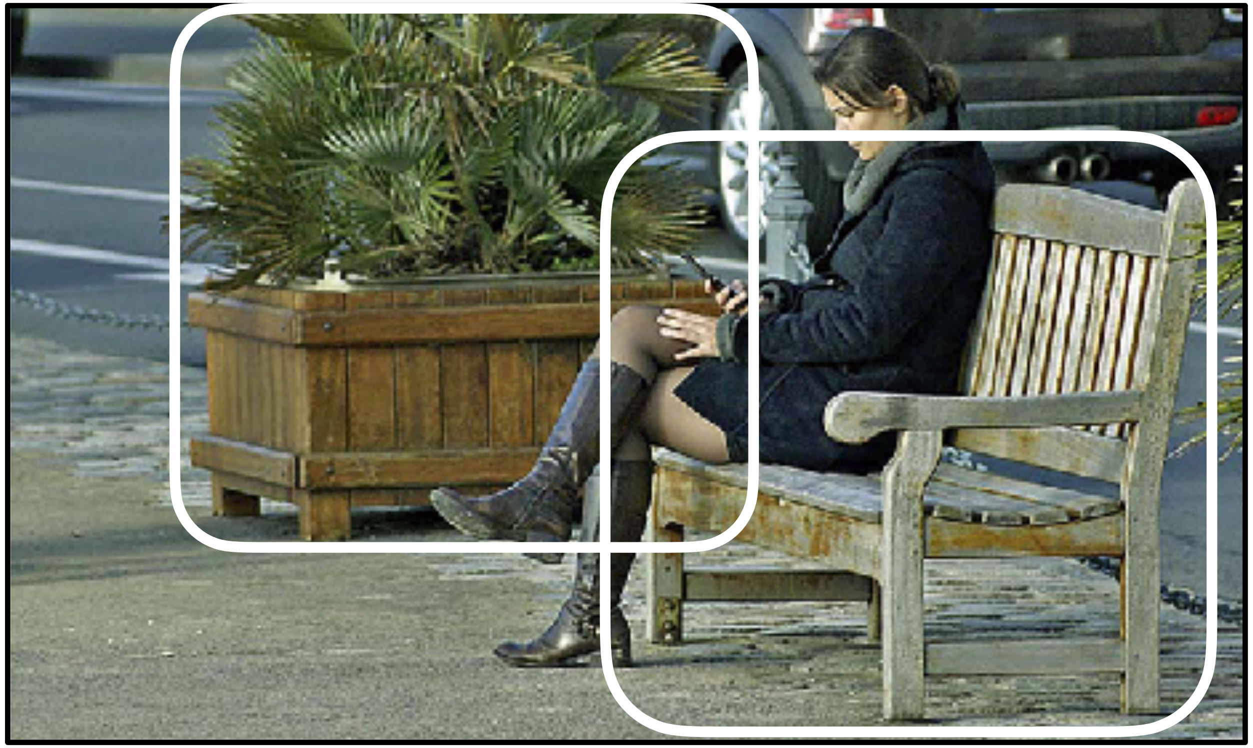
✗

Visual Spatial Reasoning, Liu Fangyu et al. 2023

Drawback: Context-Dependence

“Is the potted plant to the right of the bench?”

```
bbox_plant = detect(image, “potted plant”)
bbox_bench = detect(image, “bench”)
return bbox_plant.x > bbox_bench.x
```

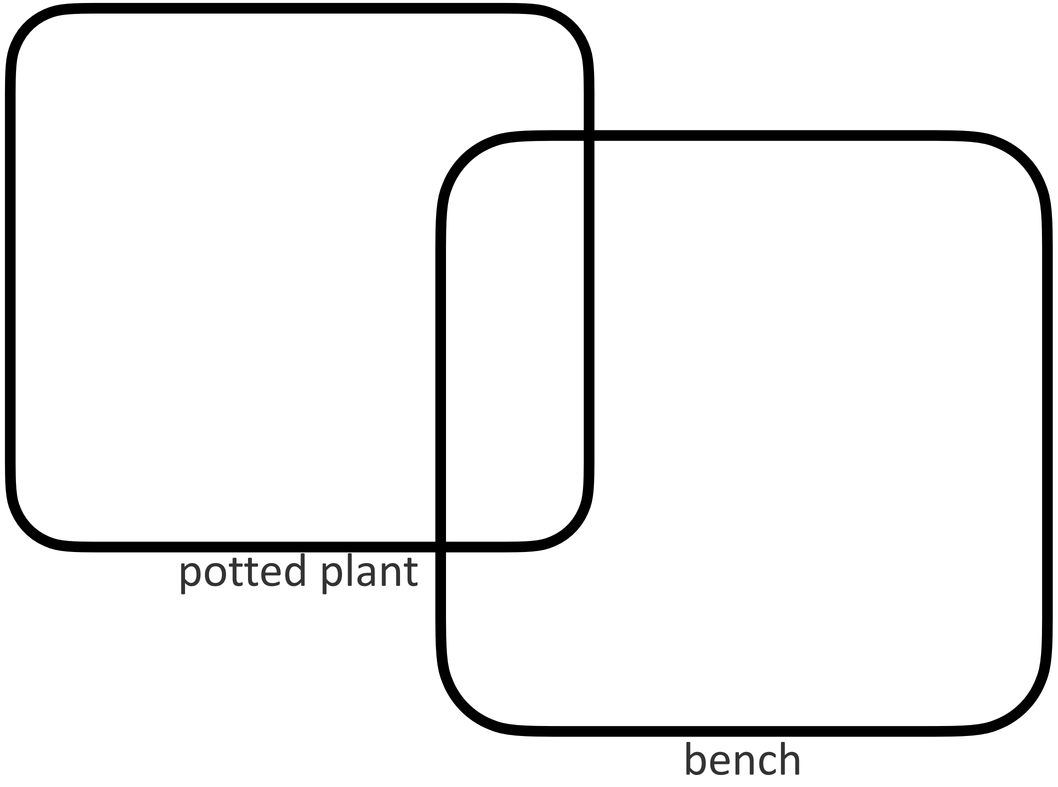


Visual Spatial Reasoning, Liu Fangyu et al. 2023

Drawback: Context-Dependence

“Is the potted plant to the right of the bench?”

```
bbox_plant = detect(image, “potted plant”)
bbox_bench = detect(image, “bench”)
return bbox_plant.x > bbox_bench.x
```



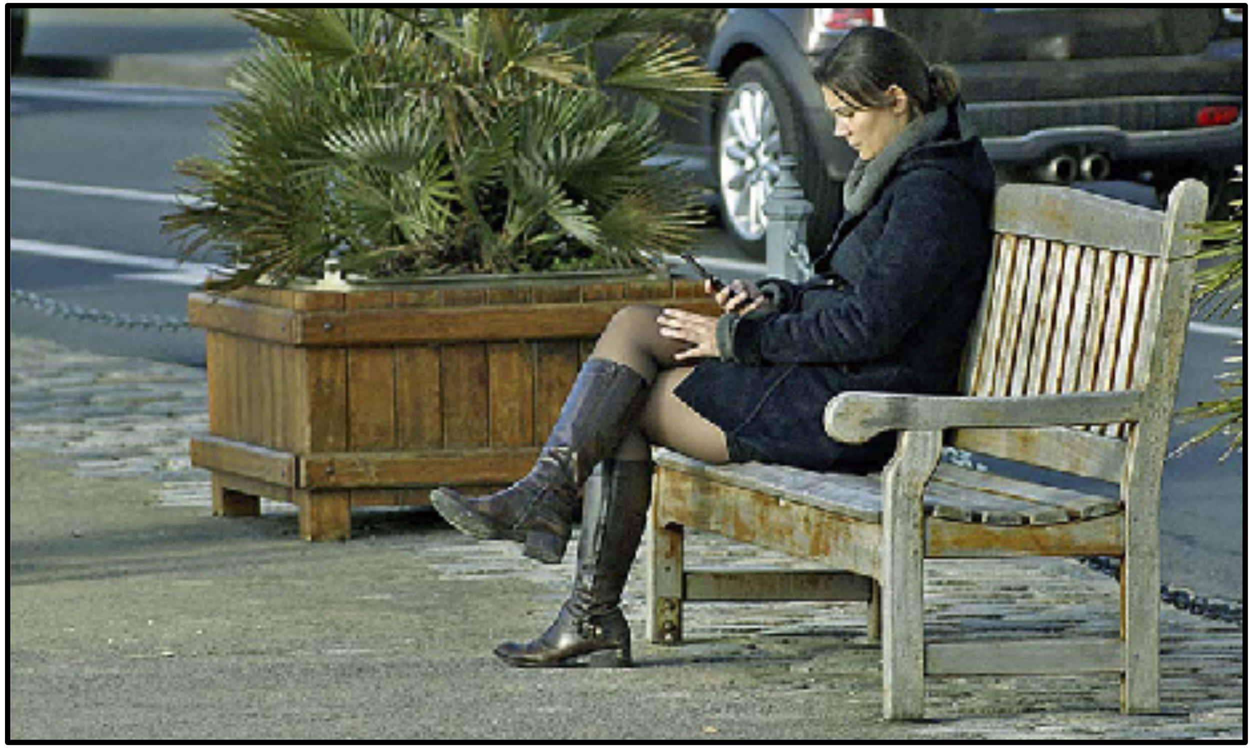
✗

Visual Spatial Reasoning, Liu Fangyu et al. 2023

Drawback: Context-Dependence

“Is the potted plant to the right of the bench?”

```
bbox_plant = detect(image, “potted plant”)
bbox_bench = detect(image, “bench”)
return bbox_plant.x > bbox_bench.x
```



✓

Visual Spatial Reasoning, Liu Fangyu et al. 2023