THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 22: Deep Reinforcement Learning (cont.)

Logistics

— This is our second to last lecture (**last lecture Tuesday**)

— **Paper presentations** due **tomorrow** (will post them over the weekend)

— **Final project presentations** are **December 13th,** noon-3pm

(I will ask you to submit slides 11:59pm on the December 12th)

I'll invite TAs possibly a few others

— **Final project write-ups** are due **December 20th**

Approaches to RL: Taxonomy

Model-free RL

Value-based RL

- — Estimate the optimal action-value function $Q^*(s,a)$
- — No policy (implicit)

Policy-based RL

- — Search directly for the optima policy π^*
- — No value function

Actor-critic RL

- — Value function
- — Policy function

Model-based RL

— Build a model of the world

— Plan (e.g., by look-ahead) using model

* slide from Dhruv Batra

Optimal Q Value Function

Optimal Q-function is the maximum achievable value

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = Q^{\pi^*}(s,a)$$

Once we have it, we can act optimally

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\ Q^*(s,a)$$

Optimal value maximizes over all future decisions

$$\begin{aligned} Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \ldots \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

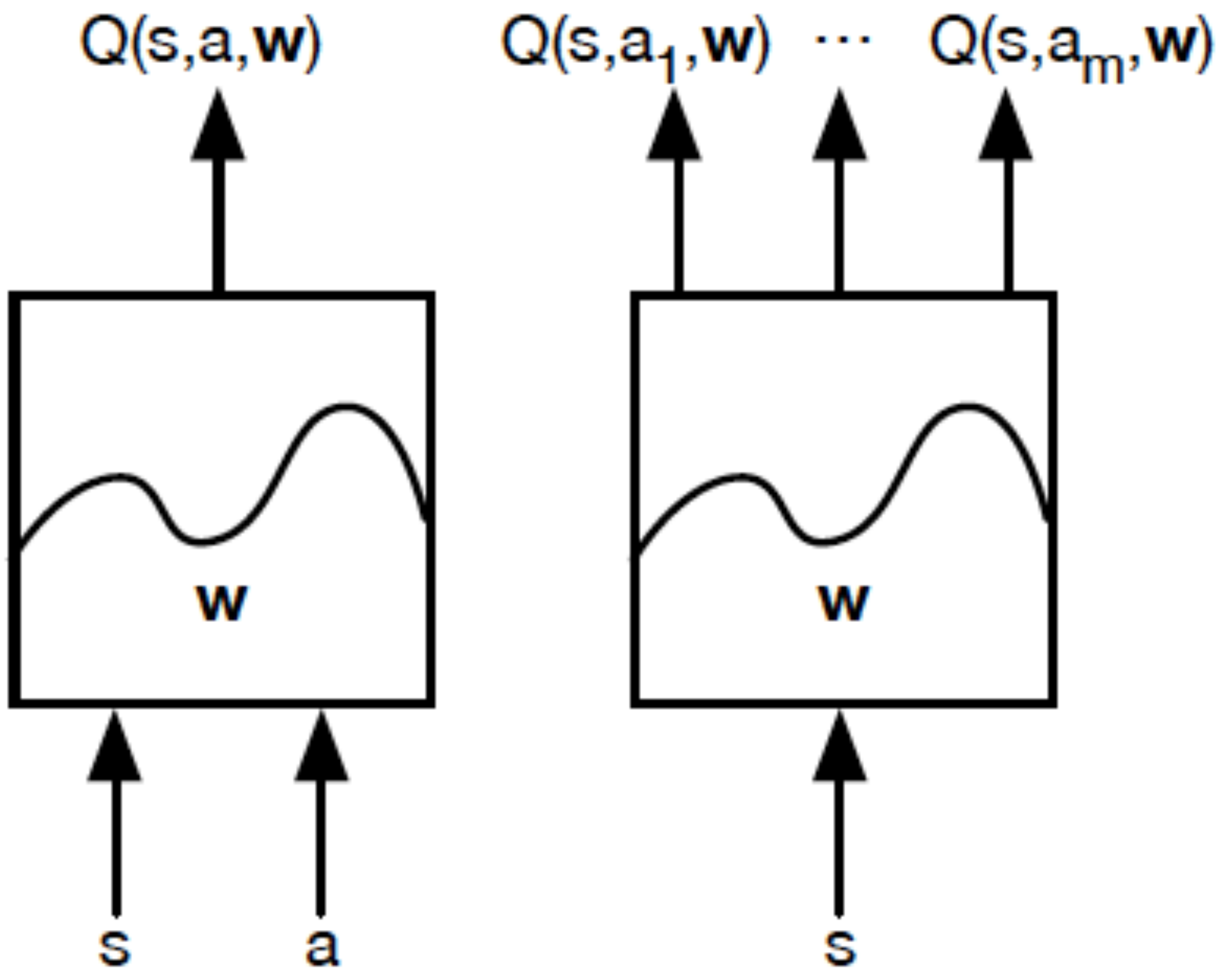
Formally, Q^* satisfied Bellman Equations

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a \right]$$

* slide from David Silver

Q-Networks

$$Q(s, a, \mathbf{w}) \approx Q^*(s, a)$$



* slide from David Silver

Q-Network Learning

Remember: want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

Forward Pass:

Loss function: $L_i(\theta_i) = \mathbb{E}\left[(y_i - Q(s,a;\theta_i)^2\right]$

where $y_i = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$

Backward Pass:

Gradient update (with respect to Q-function parameters θ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Q-Network Learning

Remember: want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

Forward Pass:

Loss function: $L_i(\theta_i) = \mathbb{E}\left[(y_i - Q(s,a;\theta_i))^2\right]$

where $y_i = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$

Iteratively try to make the Q-value close to the target value (y_i) it should have, if Q-function corresponds to optimal Q* (and optimal policy $\boldsymbol{\pi}^*$)

Backward Pass:

Gradient update (with respect to Q-function parameters θ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

Q-Network Learning

Remember: want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

Forward Pass:

Loss function: $$L_i(\theta_i) = \mathbb{E}\left[(y_i - Q(s,a;\theta_i))^2\right]$$

where $$y_i = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

Iteratively try to make the Q-value close to the target value (y_i) it should have, if Q-function corresponds to optimal Q* (and optimal policy π^*)

Backward Pass:

Gradient update (with respect to Q-function parameters θ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Q-Network Learning

Remember: want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

Forward Pass:

Loss function: $L_i(\theta_i) = \mathbb{E}\left[(y_i - Q(s,a;\theta_i))^2\right]$

where $y_i = \mathbb{E}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$

Iteratively try to make the Q-value close to the target value (y_i) it should have, if Q-function corresponds to optimal Q* (and optimal policy π^*)

Backward Pass:

Gradient update (with respect to Q-function parameters θ):

Need **tuples**: <s, a, r, s'>

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Training the Q-Network: **Experience Replay**

Learning from **batches of consecutive samples is problematic**:

— Samples are correlated => inefficient learning

— Current Q-network parameters determines next training samples (e.g. if maximizing action is to move left, training samples will be dominated by samples from left-hand size) => can lead to bad feedback loops

Address these problems using experience replay

— Continually update a replay memory table of transitions (s_t, a_t, r_t, s_{t+1}) as game (experience) episodes are played

— Train Q-network on random minibatches of transitions from the replay memory, instead of consecutive samples

Experience Replay

Experience Replay

To remove correlations, build data-set from agent's own experience

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\ldots$
$s_t, a_t, r_{t+1}, s_{t+1}$

$\rightarrow$ s, a, r, s'

$s_t, a_t, r_{t+1}, s_{t+1}$ $\rightarrow$

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights

Initialize replay memory, Q-network

for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Play M episodes (full games)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Initialize state (start geme screen pixes) at beggining of each episode

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

For each timestep T of the game
(T is max steps but can return early)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

With small probability take random action (explore)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Otherwise select greedy action from current policy (implicit in Q function)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Take action and observe the reward and next state

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Store transition replay in memory

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

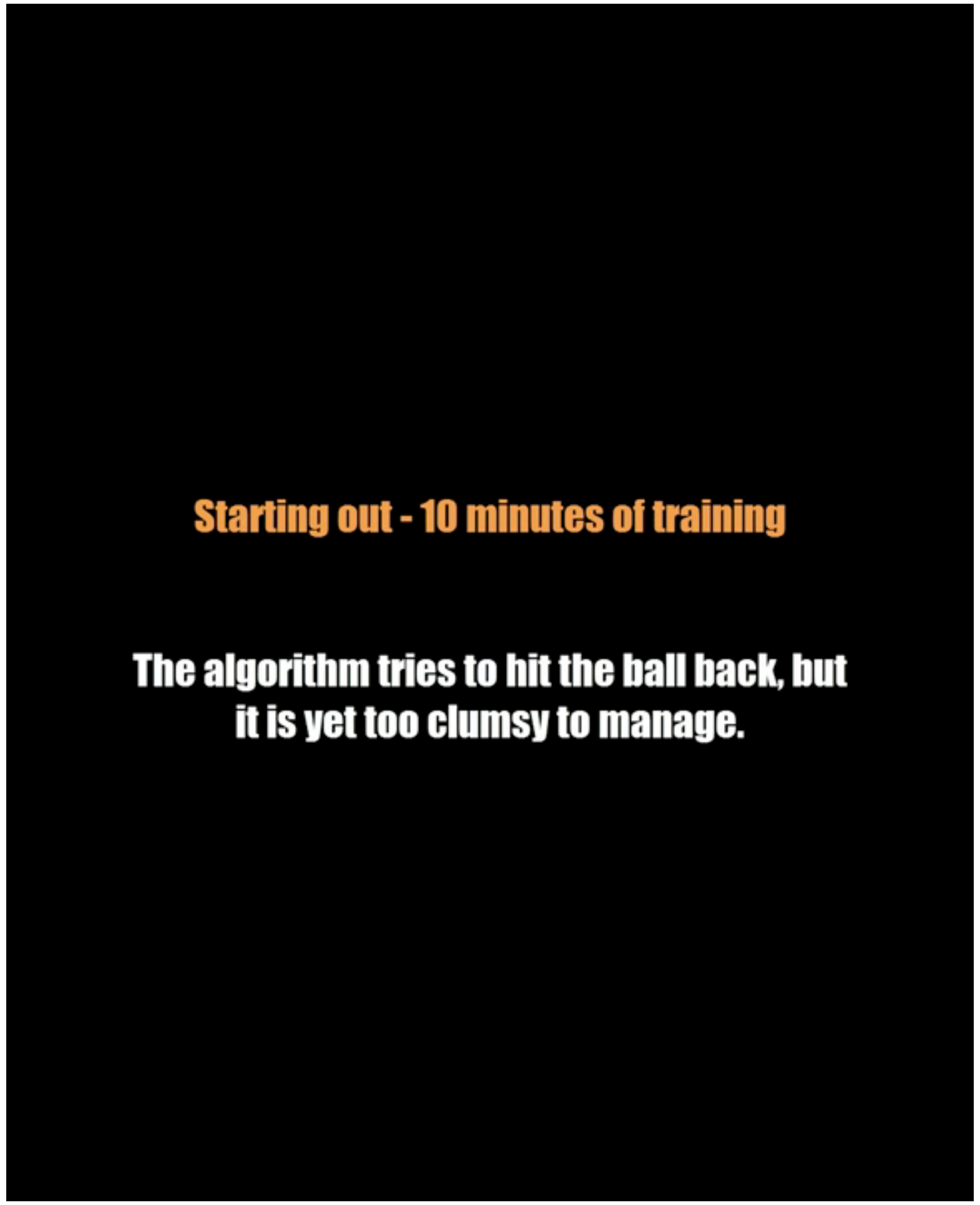
Putting it together: Deep Q-learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

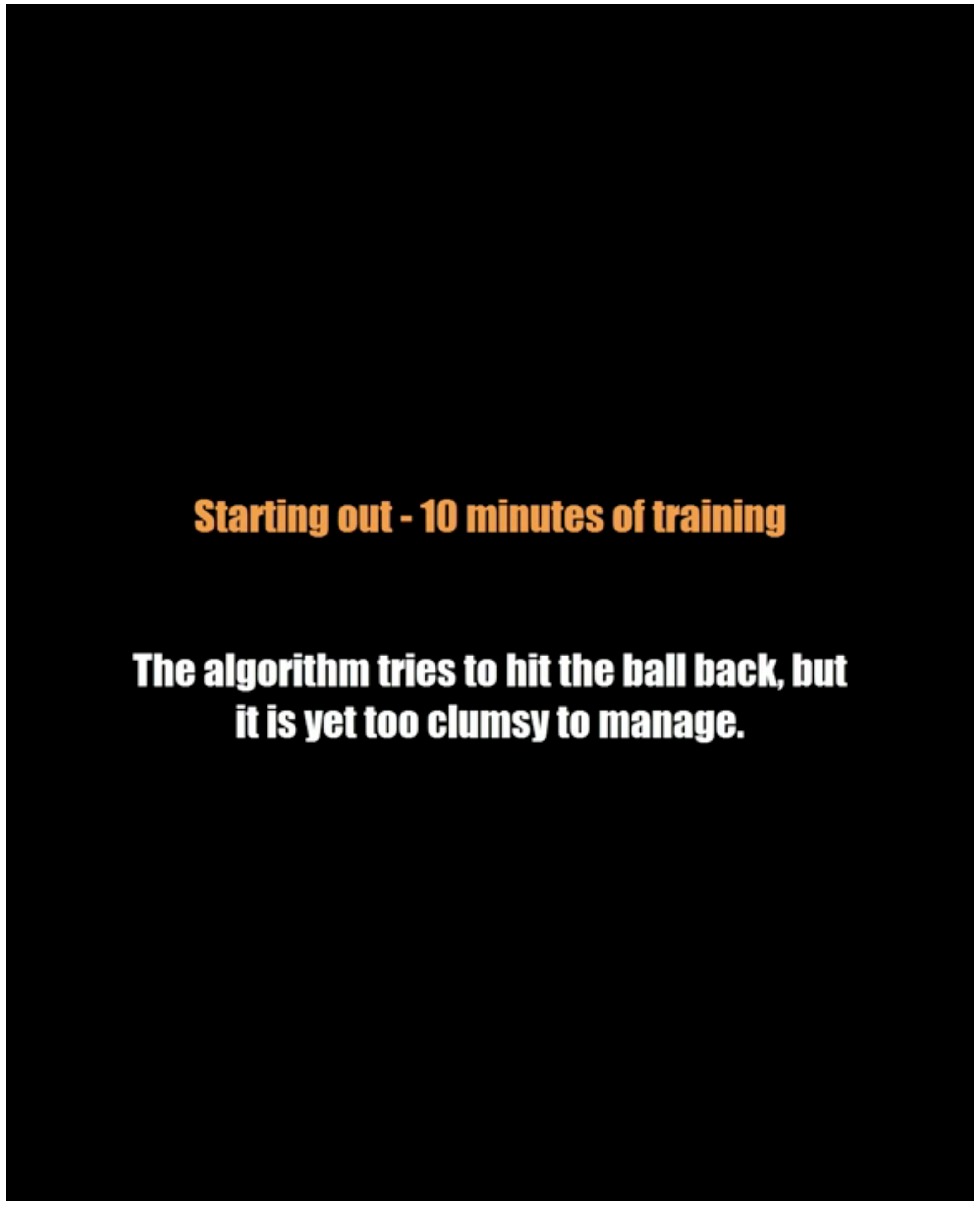
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Sample a random mini-batch from replay memory and perform a gradient descent step

Example: Atari Playing



Example: Atari Playing



Deep RL

Value-based RL

— Use neural nets to represent Q function $Q(s, a; \theta)$

$$Q(s, a; \theta^*) \approx Q^*(s, a)$$

Policy-based RL

— Use neural nets to represent the policy π_θ

$$\pi_{\theta^*} \approx \pi^*$$

Model-based RL

— Use neural nets to represent and learn the model

* slide from Dhruv Batra

Deep RL

Value-based RL

— Use neural nets to represent Q function $Q(s, a; \theta)$

$$Q(s, a; \theta^*) \approx Q^*(s, a)$$

Policy-based RL

— Use neural nets to represent the policy π_θ

$$\pi_{\theta^*} \approx \pi^*$$

Model-based RL

— Use neural nets to represent and learn the model

* slide from Dhruv Batra

Policy Gradients

Formally, let's define a class of parameterized policies:

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

Policy Gradients

Formally, let's define a class of parameterized policies:

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

Policy Gradients

Formally, let's define a class of parameterized policies:

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Policy Gradients

Formally, let's define a class of parameterized policies:

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Gradient ascent on policy parameters!

REINFORCE algorithm

Expected reward:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} [r(\tau)]$$
$$= \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Where r(τ) is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

REINFORCE algorithm

Expected reward:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Where r(τ) is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

Now let's differentiate this: $\nabla_\theta J(\theta) = \int_\tau r(\tau)\nabla_\theta p(\tau;\theta)\mathrm{d}\tau$

Intractable! Expectation of gradient is problematic when p depends on θ

REINFORCE algorithm

Expected reward:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_{\tau} r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Where r(τ) is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

Now let’s differentiate this: $\nabla_\theta J(\theta) = \int_{\tau} r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$

However, we can use a nice trick: $\nabla_\theta p(\tau;\theta) = p(\tau;\theta) \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta) \nabla_\theta \log p(\tau;\theta)$

If we inject this back:

$$\nabla_\theta J(\theta) = \int_{\tau} \left(r(\tau) \nabla_\theta \log p(\tau;\theta)\right) p(\tau;\theta) \mathrm{d}\tau$$
$$= \mathbb{E}_{\tau \sim p(\tau;\theta)}\left[r(\tau) \nabla_\theta \log p(\tau;\theta)\right]$$

Can estimate with Monte Carlo sampling

Intuition

Gradient estimator:

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Intuition

Gradient estimator:

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

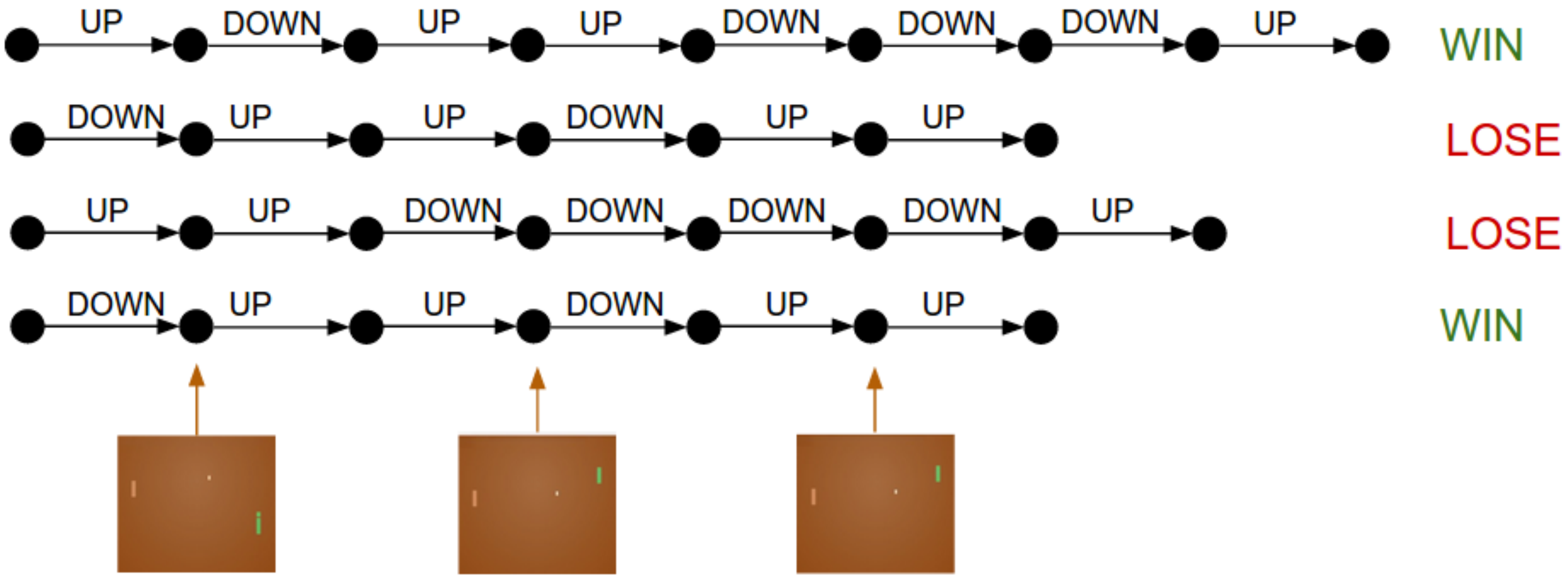
Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. **But in expectation, it averages out!**

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Intuition



* slide from Dhruv Batra

Intuition

Gradient estimator:

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. **But in expectation, it averages out!**

However, this also suffers from high variance because credit assignment is really hard. Can we help the estimator?

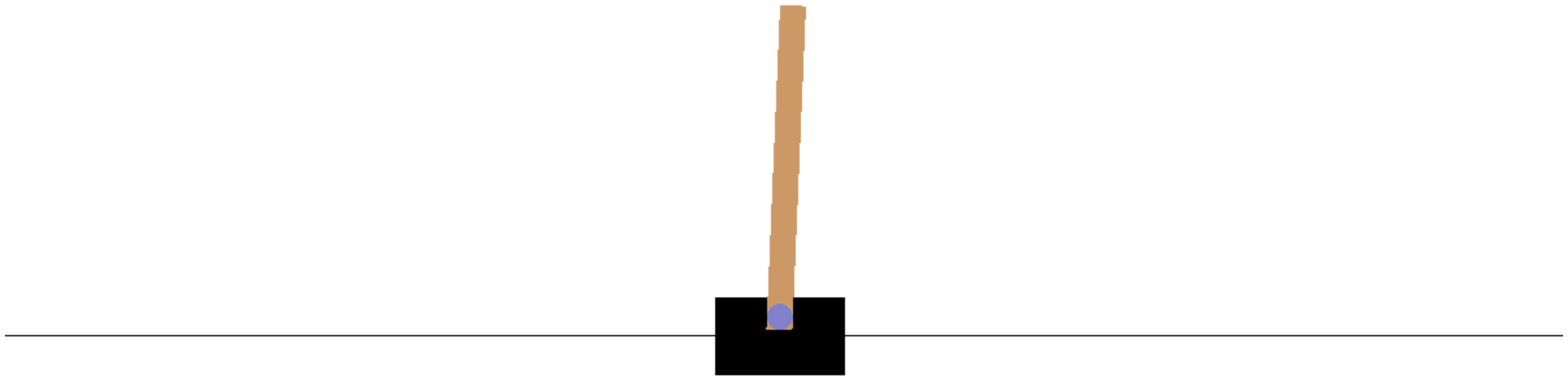
CartPole Environment

Unstable system. Poll will fall if left to own devices.

Goal: Keep the poll upright by applying +1 / -1 force (move cart left or right)

Reward: +1 for every frame for every time step pole remains upright

State: 4-D (position + velocity of cart, angle + velocity of pole)



https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57

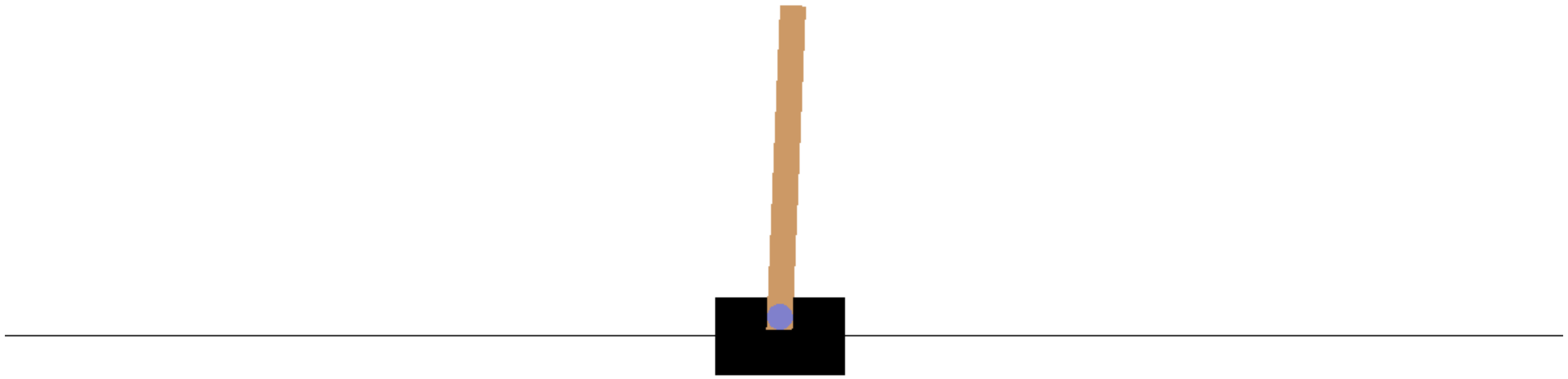
CartPole Environment

Unstable system. Poll will fall if left to own devices.

Goal: Keep the poll upright by applying +1 / -1 force (move cart left or right)

Reward: +1 for every frame for every time step pole remains upright

State: 4-D (position + velocity of cart, angle + velocity of pole)



https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57

CartPole Environment

$$R + \gamma R + \gamma^2 R + \gamma^3 R + \ldots = \frac{R}{1-\gamma}$$

Note: we can focus on short-term horizon policy by setting gamma = 0

on long-term horizon policy by setting gamma close to 1

CartPole Environment

$$R + \gamma R + \gamma^2 R + \gamma^3 R + \ldots = \frac{R}{1-\gamma}$$

Note: we can focus on short-term horizon policy by setting gamma = 0

on long-term horizon policy by setting gamma close to 1

What happens if we delayed our reward, e.g., only receive 1 if pole is upright after 500 time steps?

CartPole Environment

$$R + \gamma R + \gamma^2 R + \gamma^3 R + \ldots = \frac{R}{1-\gamma}$$

Note: we can focus on short-term horizon policy by setting gamma = 0

on long-term horizon policy by setting gamma close to 1

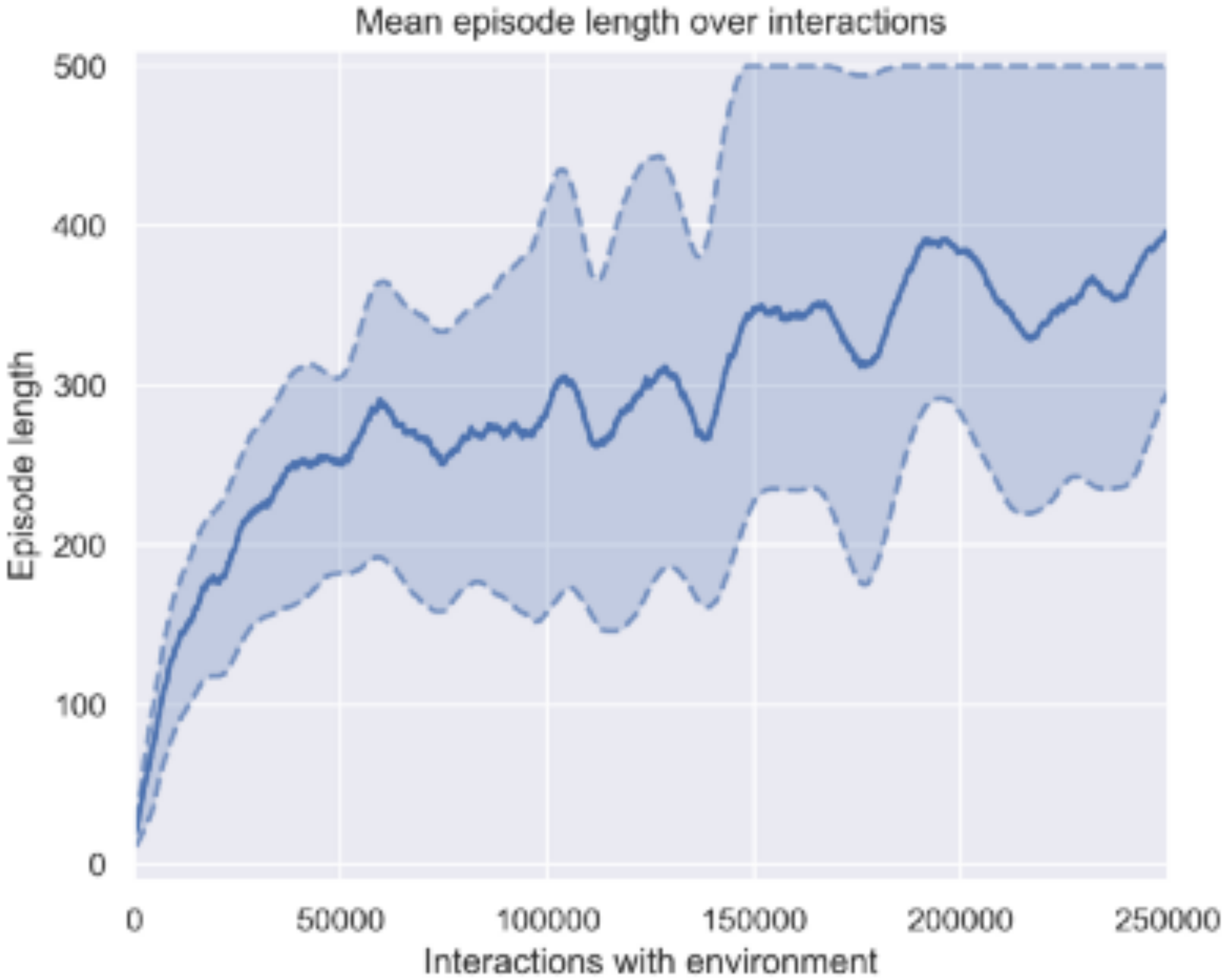
What happens if we delayed our reward, e.g., only receive 1 if pole is upright after 500 time steps?

$$\gamma^{499} R$$

REINFORCE with **Whitening Baseline**

Subtract mean over rewards in a rollout and divide by the standard deviation

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t) \qquad r(\tau) = \sum \gamma^t r_t \qquad r(\tau) = \frac{\sum \gamma^t r_t - \mu_{r_t}}{\sigma_{r_t}}$$



1 iteration = 1 episode + gradient update step

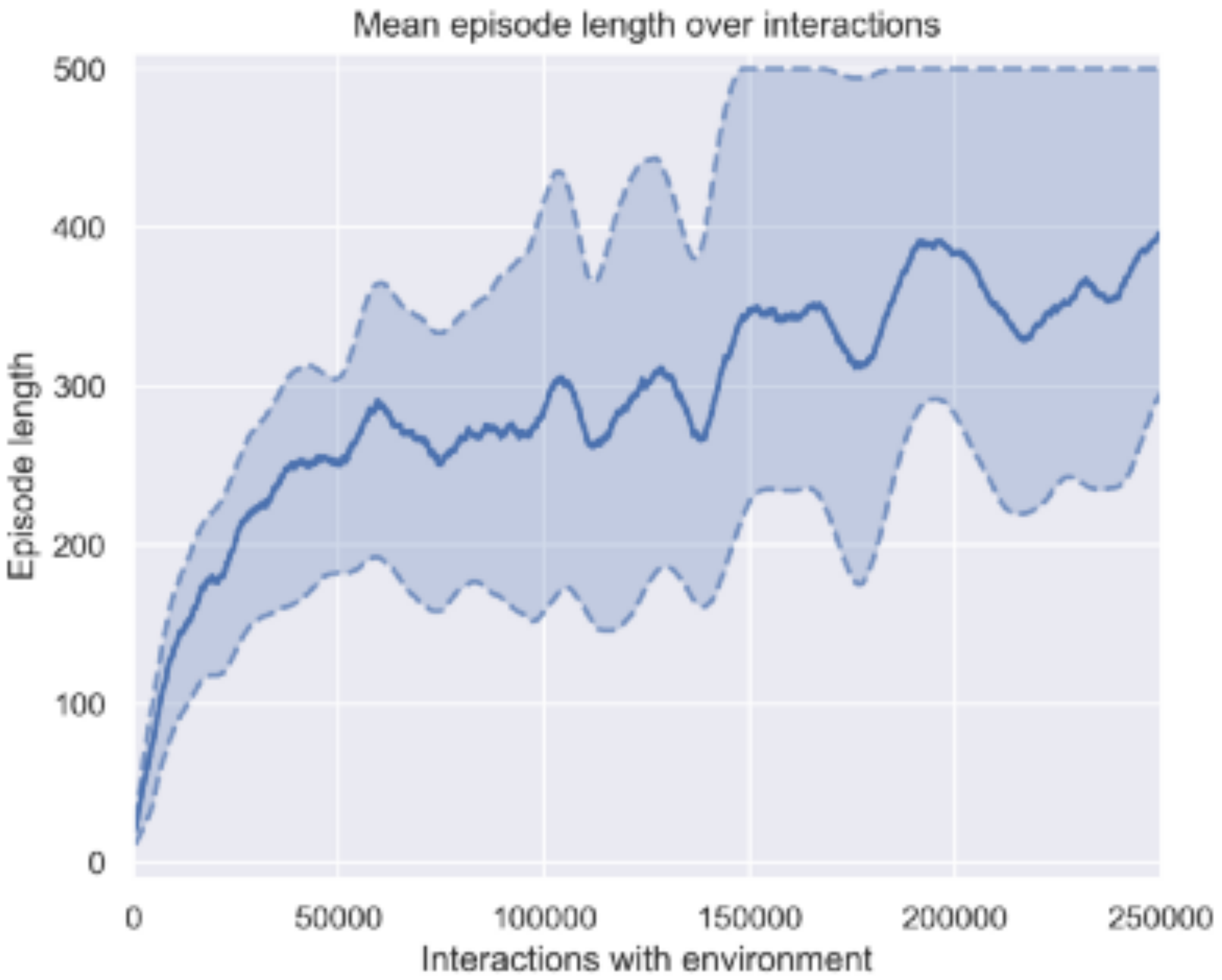
1 interaction = 1 action taken in the environment

REINFORCE with **Whitening Baseline**

Does not solve a game, even after 1000 iterations!!

Algorithm unstable (variance is high)

1 iteration = 1 episode + gradient update step

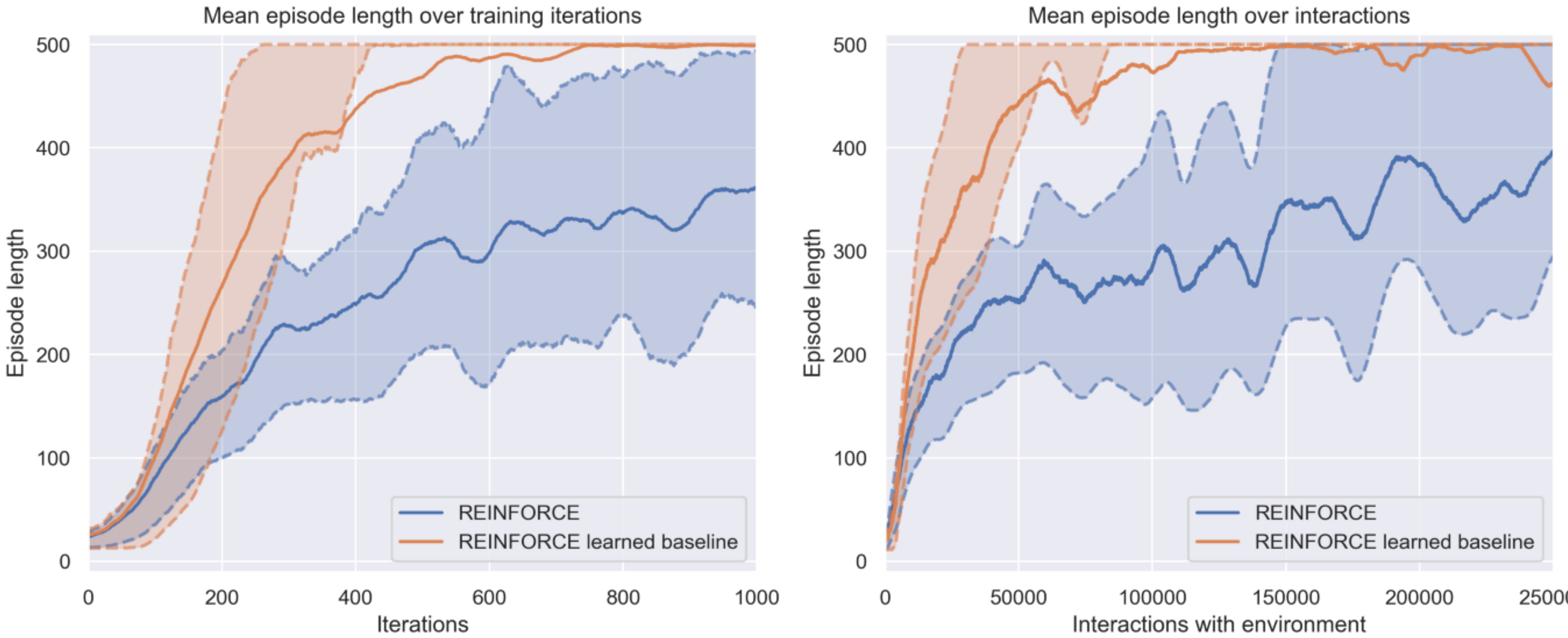


1 interaction = 1 action taken in the environment

REINFORCE with **Learned Baseline** (self-critic)

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$r(\tau) = \sum \gamma^t r_t - V_\phi(s_t)$$



1 iteration = 1 episode + gradient update step

1 interaction = 1 action taken in the environment

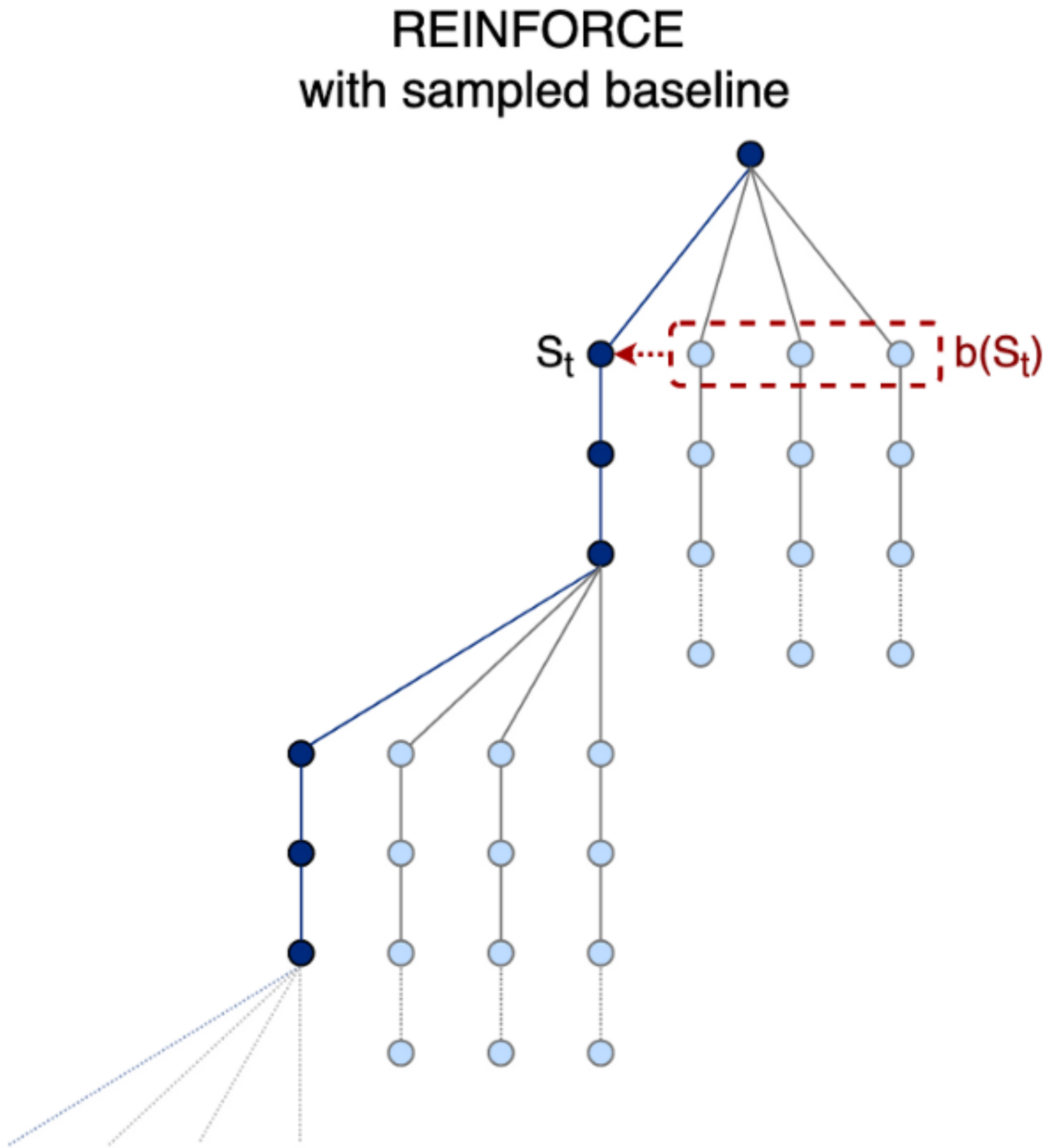
REINFORCE with **Sampled Baseline**

https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57

$$\hat{v}(S_t) = \frac{1}{N_b}\sum_{b=1}^{N_b} G_t^{(b)} \quad \text{(sample rollouts)}$$

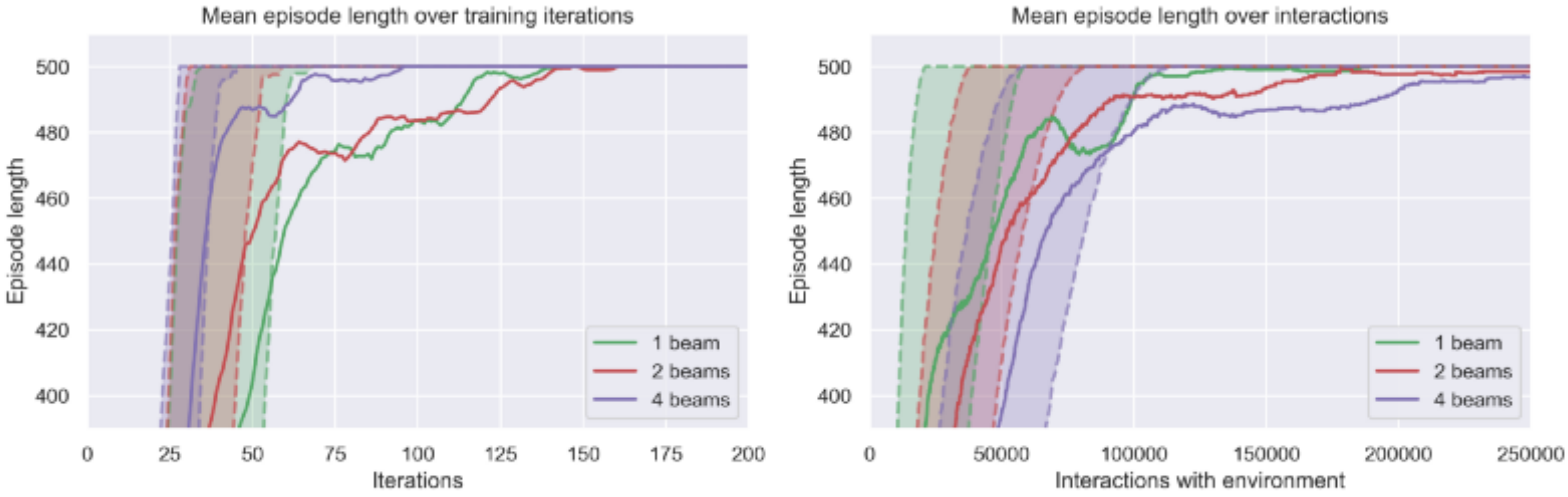
$$\hat{v}(S_t) = G_t^{(greedy)} \quad \text{(greedy rollout)}$$

$$\theta_{t+1} = \theta_t + \alpha\left(G_t - \hat{v}(S_t)\right)\nabla \log \pi(A_t|S_t, \theta)$$



REINFORCE with **Sampled Baseline**

https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57



1 iteration = 1 episode + gradient update step

1 interaction = 1 action taken in the environment

REINFORCE with **Sampled Baseline**

https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57

1 iteration = 1 episode + gradient update step

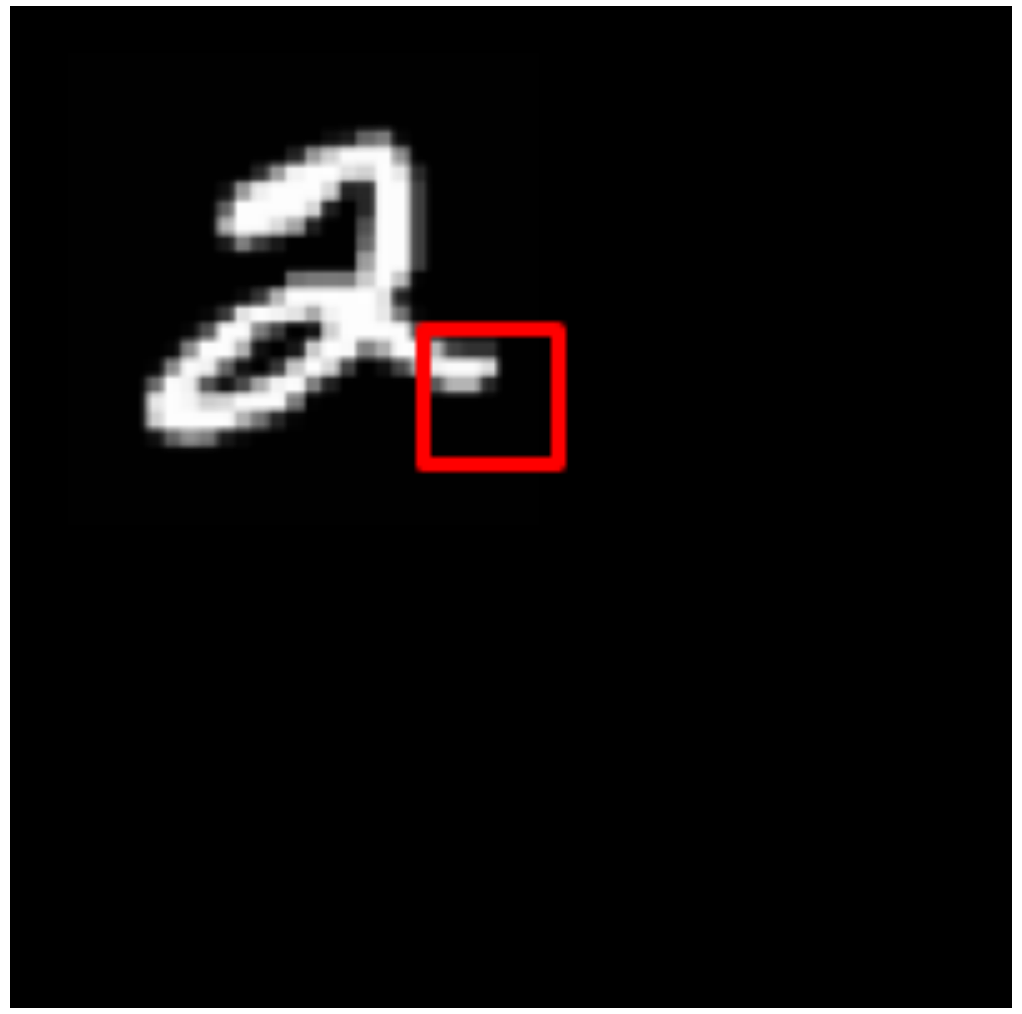
1 interaction = 1 action taken in the environment

REINFORCE in Action: **Recurrent Attention Model** (REM)

Objective: Image Classification

Take a sequence of “glimpses” selectively focusing on regions of the image, to predict class

- Inspiration from human perception and eye movements
- Saves computational resources => scalability
- Able to ignore clutter / irrelevant parts of image



glimpse

[Mnih *et al.*, 2014]

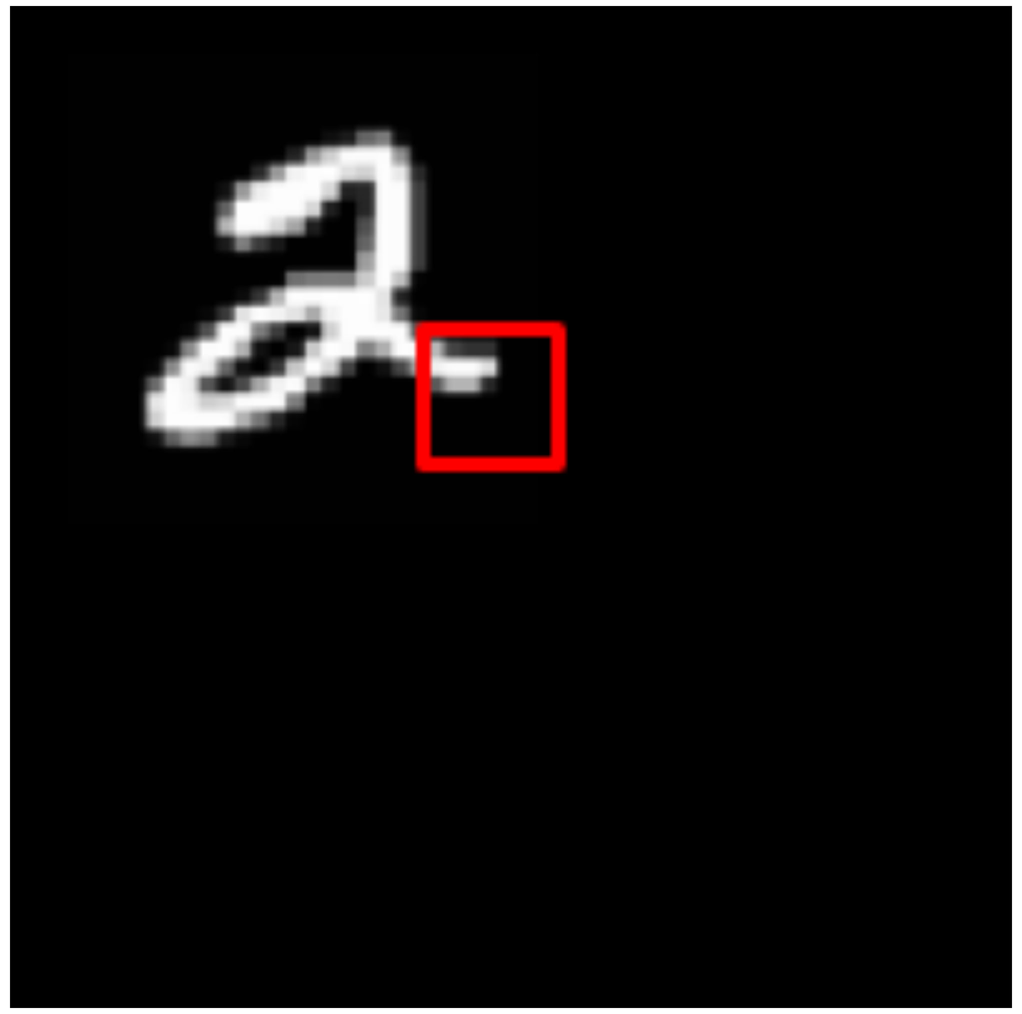
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

REINFORCE in Action: **Recurrent Attention Model** (REM)

Objective: Image Classification

Take a sequence of “glimpses” selectively focusing on regions of the image, to predict class

- Inspiration from human perception and eye movements
- Saves computational resources => scalability
- Able to ignore clutter / irrelevant parts of image



glimpse

State: Glimpses seen so far
Action: (x,y) coordinates (center of glimpse) of where to look next in image
Reward: 1 at the final timestep if image correctly classified, 0 otherwise

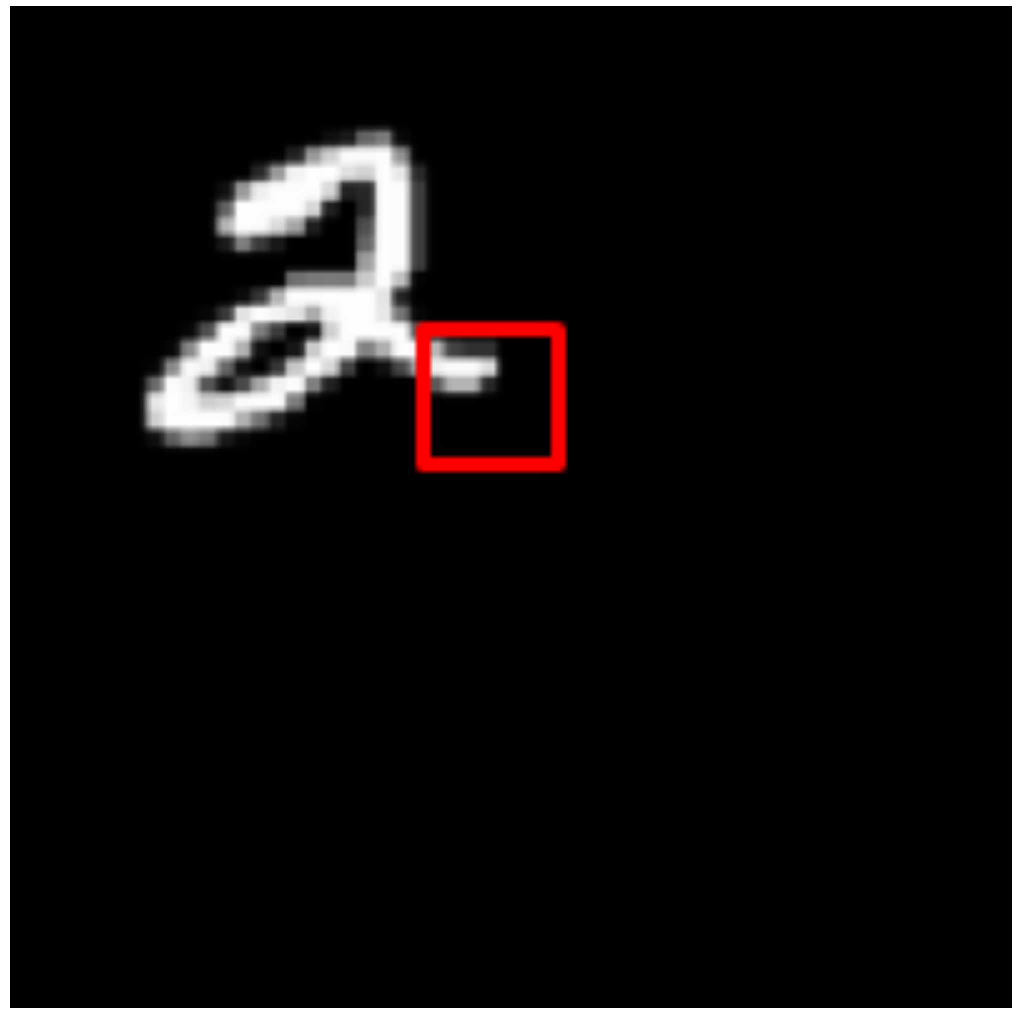
[Mnih *et al.*, 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

REINFORCE in Action: **Recurrent Attention Model** (REM)

Objective: Image Classification

Take a sequence of “glimpses” selectively focusing on regions of the image, to predict class
- Inspiration from human perception and eye movements
- Saves computational resources => scalability
- Able to ignore clutter / irrelevant parts of image



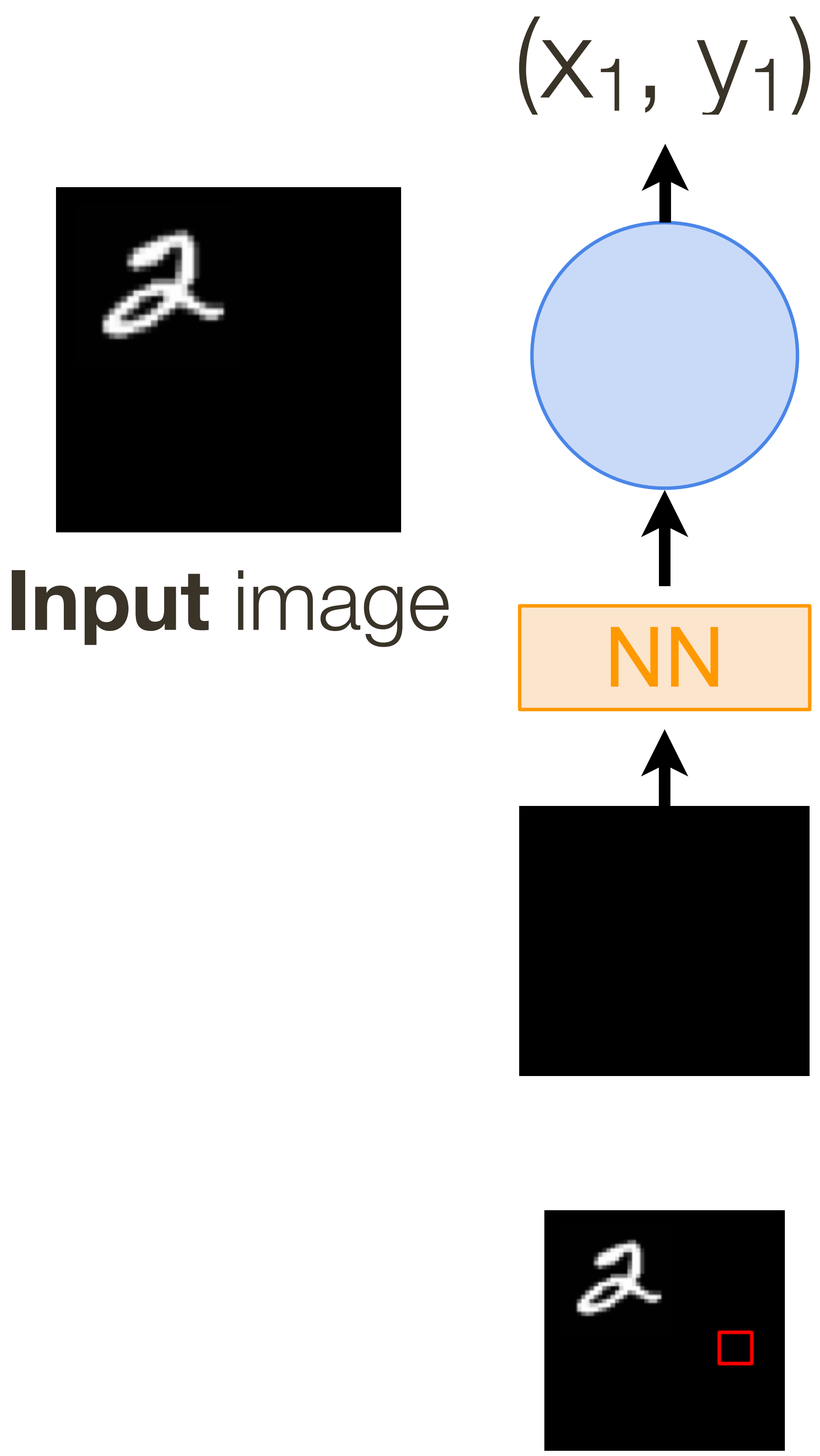
glimpse

State: Glimpses seen so far
Action: (x,y) coordinates (center of glimpse) of where to look next in image
Reward: 1 at the final timestep if image correctly classified, 0 otherwise

Glimpsing is a **non-differentiable operation** => learn policy for how to take glimpse actions using REINFORCE
Given state of glimpses seen so far, use RNN to model the state and output next action

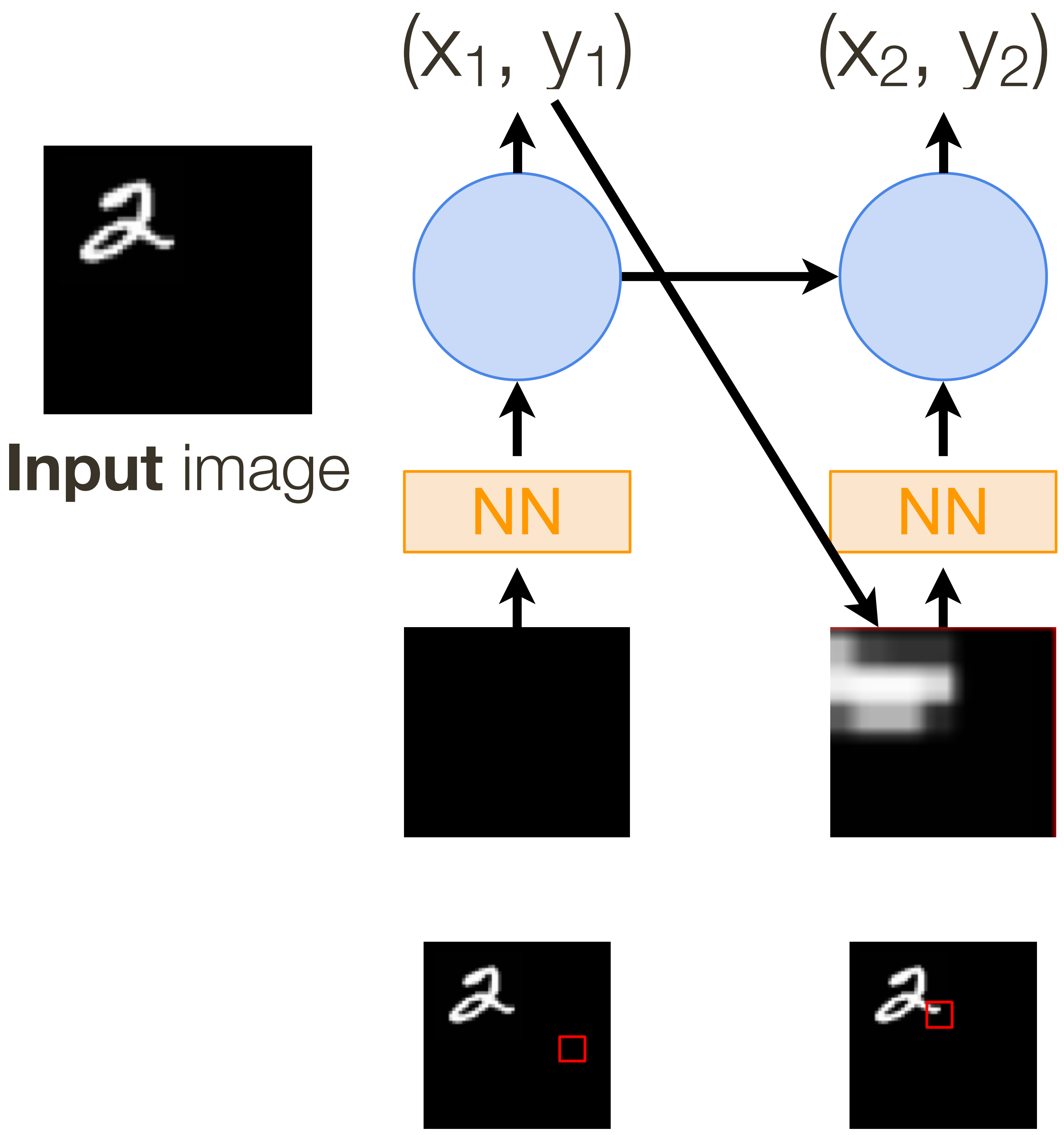
[Mnih *et al.*, 2014]

REINFORCE in Action: **Recurrent Attention Model** (REM)



[Mnih *et al.*, 2014]

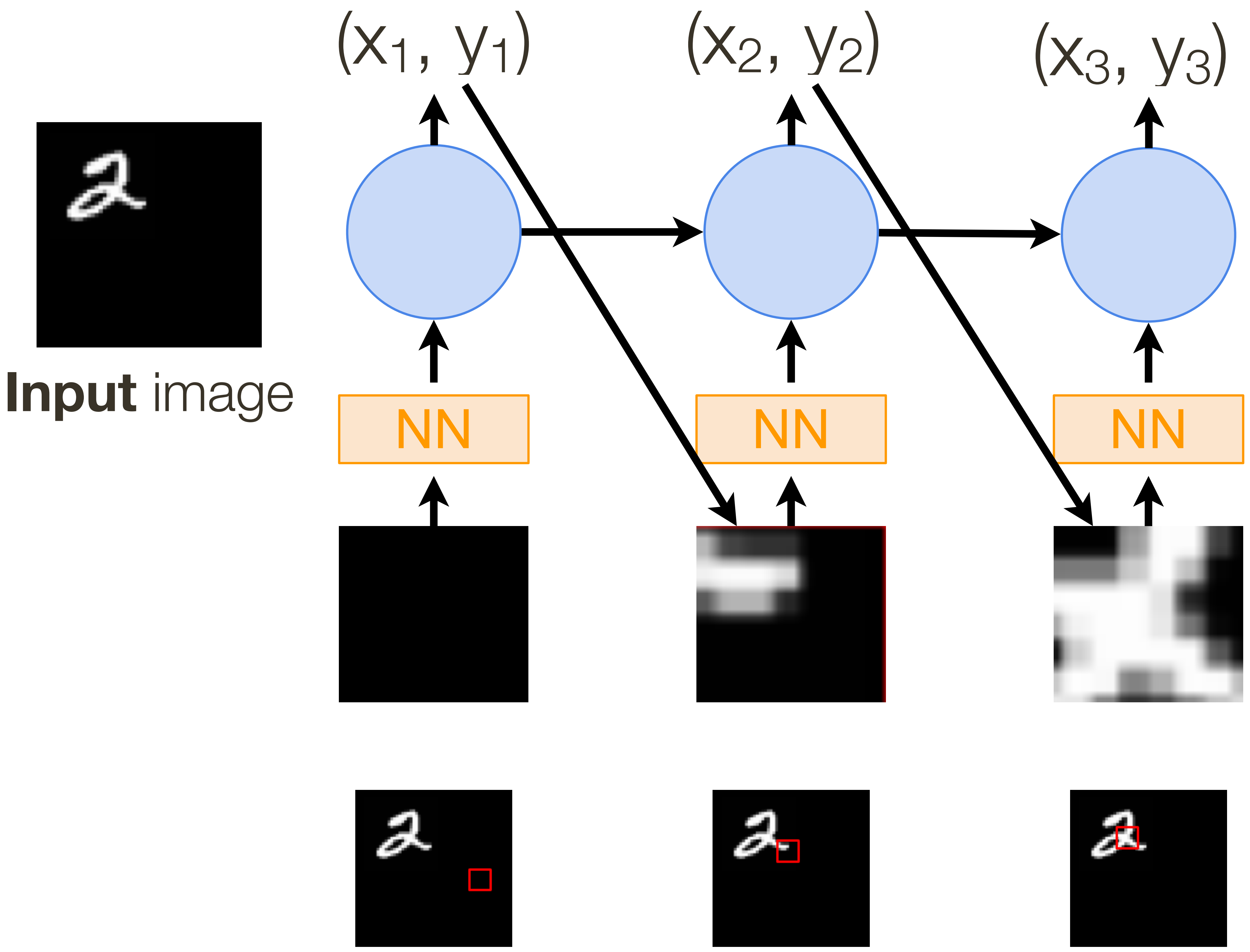
REINFORCE in Action: **Recurrent Attention Model** (REM)



[Mnih *et al.*, 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

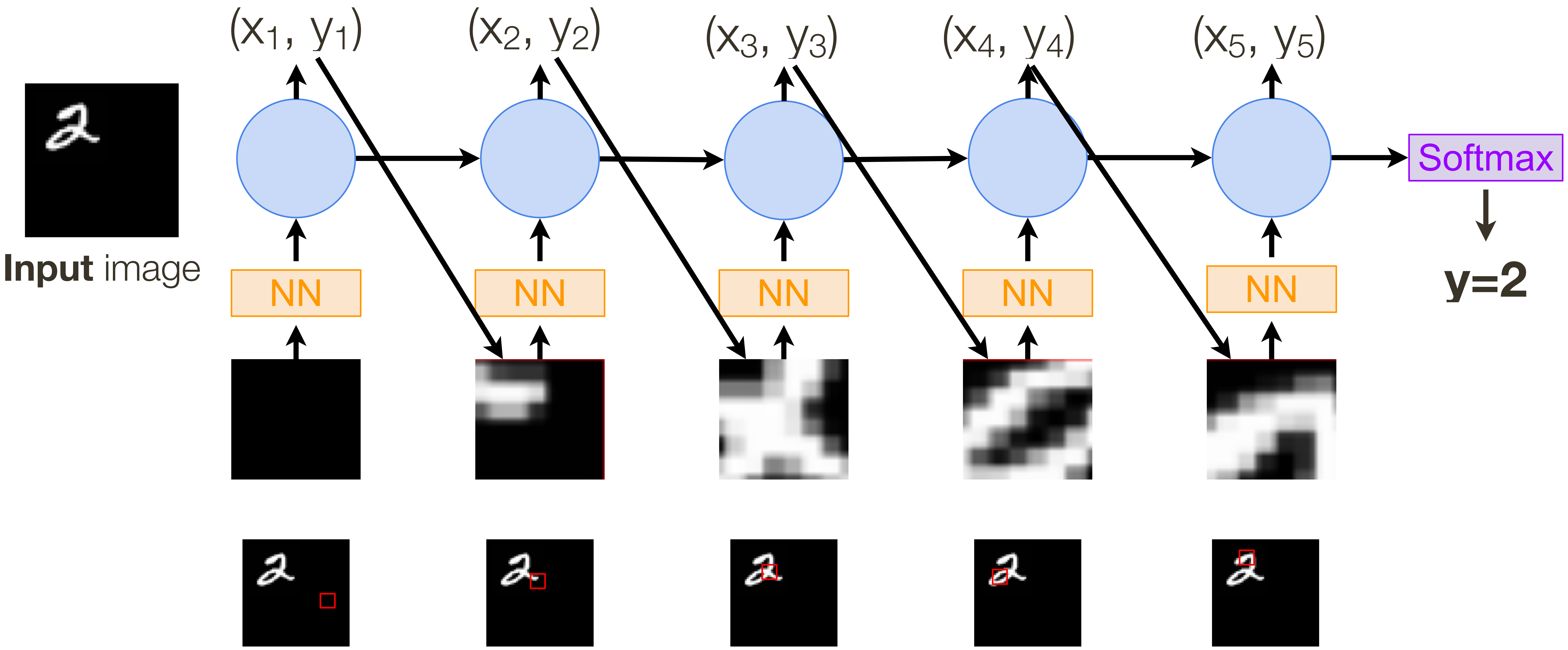
REINFORCE in Action: **Recurrent Attention Model** (REM)



[Mnih *et al.*, 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

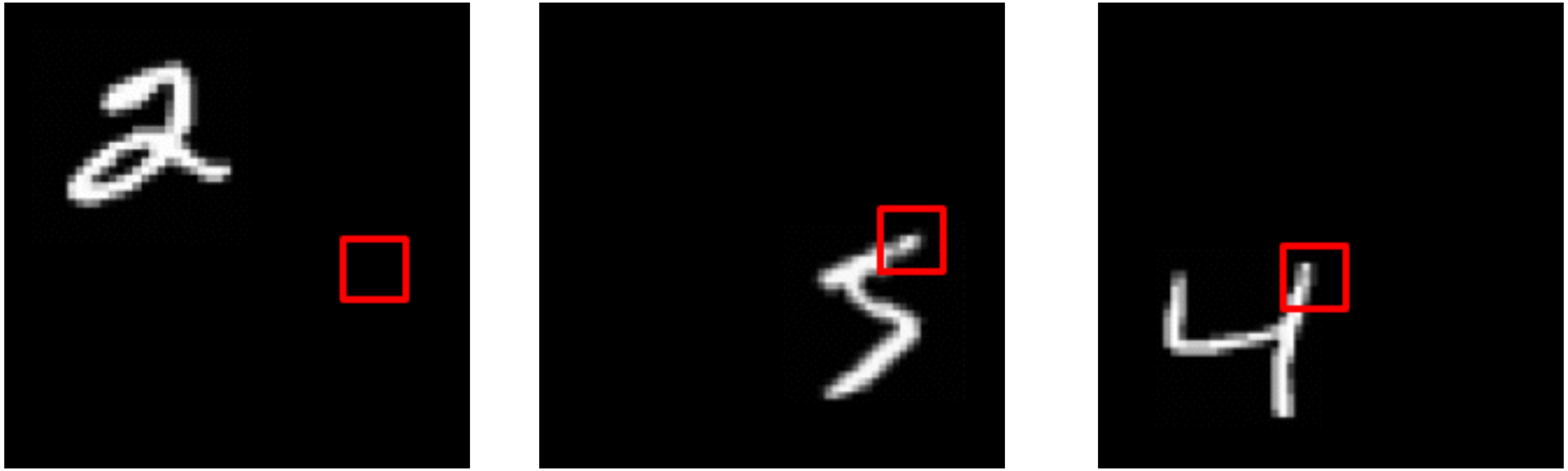
REINFORCE in Action: **Recurrent Attention Model** (REM)



[Mnih *et al.*, 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

REINFORCE in Action: **Recurrent Attention Model** (REM)

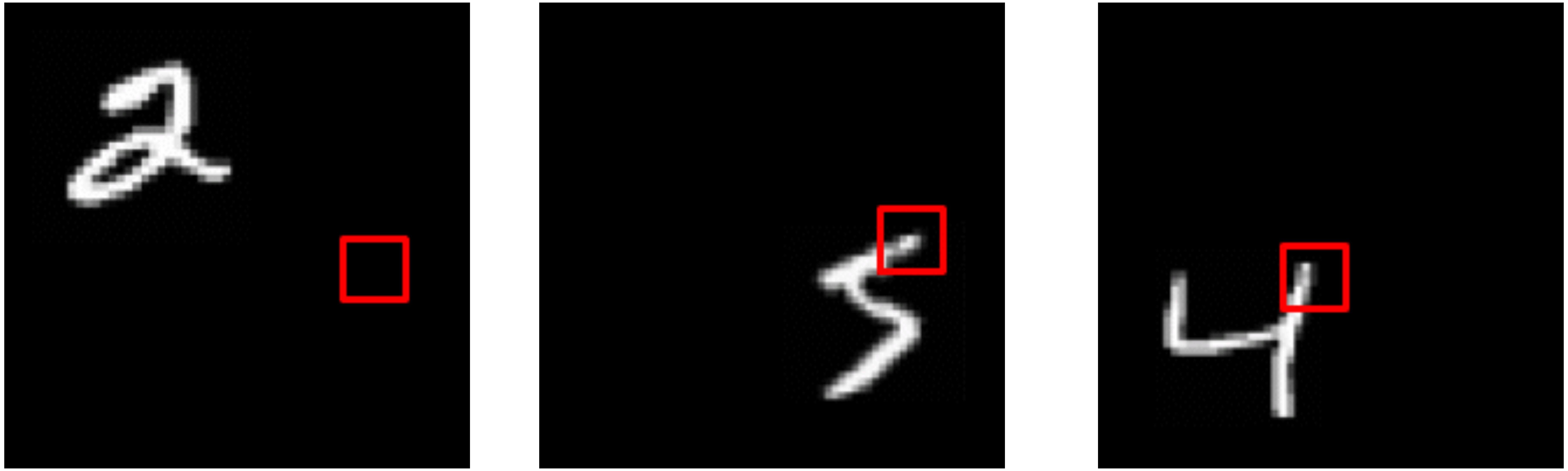


Has also been used in many other tasks including fine-grained image recognition, image captioning, and visual question-answering!

[Mnih *et al.*, 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

REINFORCE in Action: **Recurrent Attention Model** (REM)

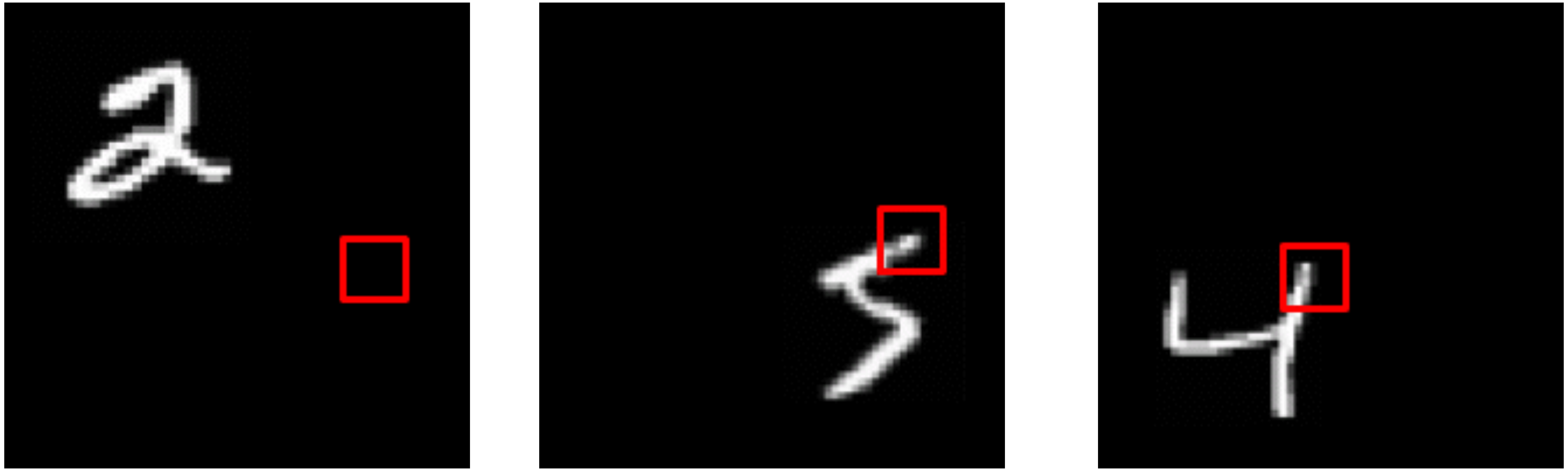


Has also been used in many other tasks including fine-grained image recognition, image captioning, and visual question-answering!

[Mnih *et al.*, 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

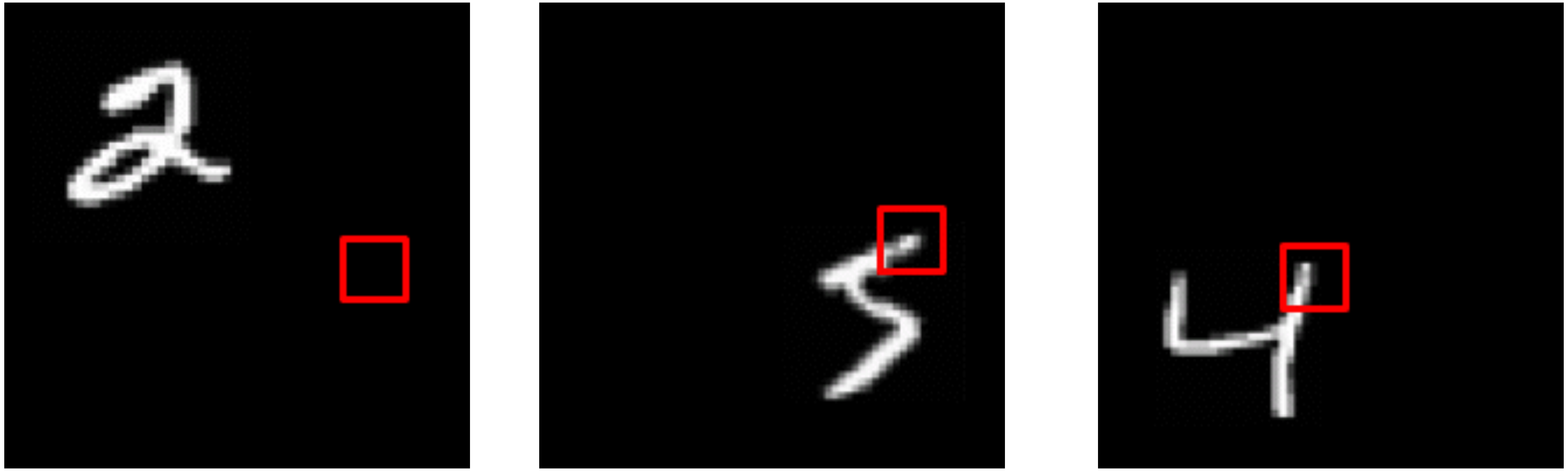
REINFORCE in Action: **Recurrent Attention Model** (REM)



Has also been used in many other tasks including fine-grained image recognition, image captioning, and visual question-answering!

[Mnih *et al.*, 2014]

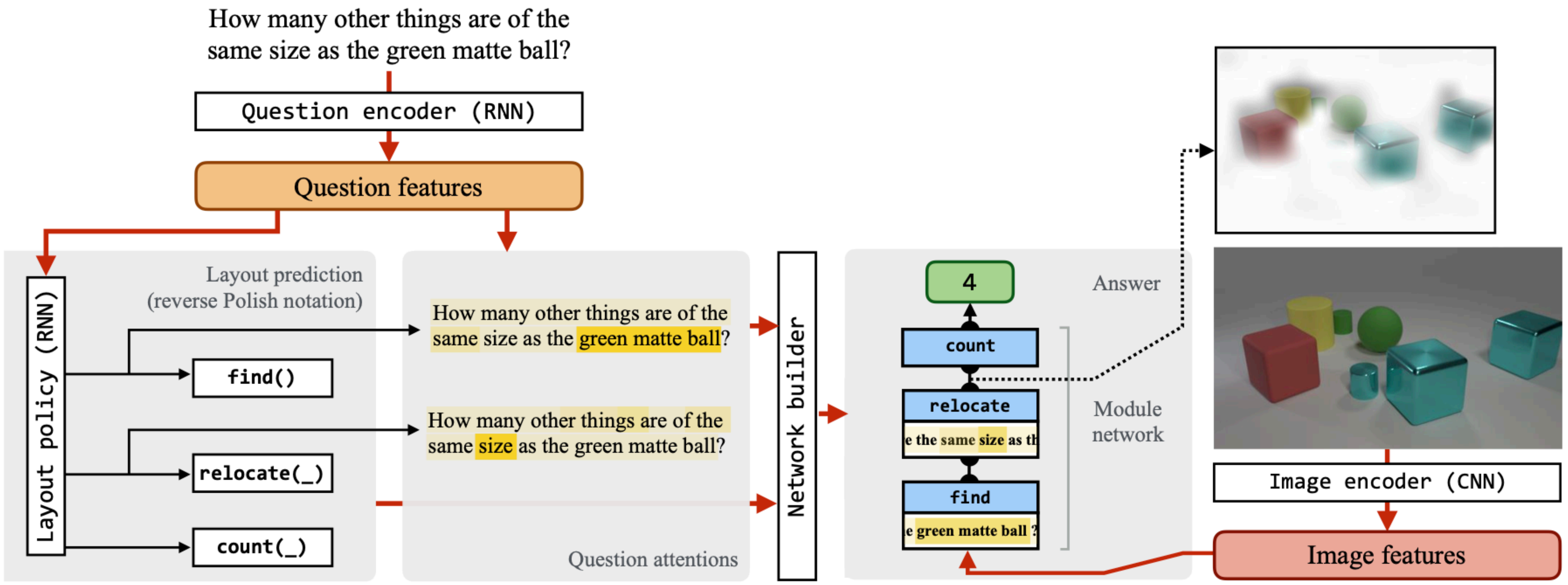
REINFORCE in Action: **Recurrent Attention Model** (REM)



Has also been used in many other tasks including fine-grained image recognition, image captioning, and visual question-answering!

[Mnih *et al.*, 2014]

Learning To Reason: End-to-End Module Networks for VQA

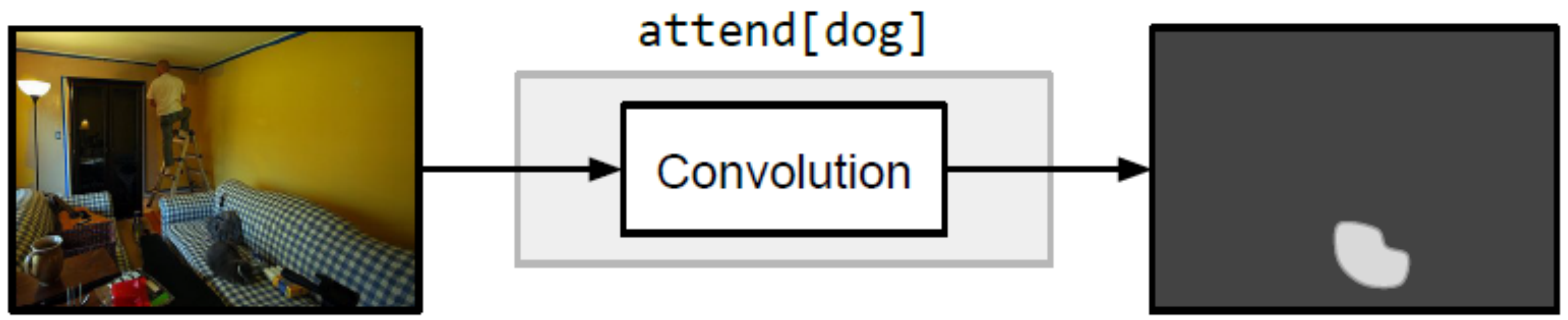


[Hu et al., 2017]

Learning To Reason: End-to-End Module Networks for VQA

[Hu et al., 2017]

$\texttt{attend} : \textit{Image} \rightarrow \textit{Attention}$

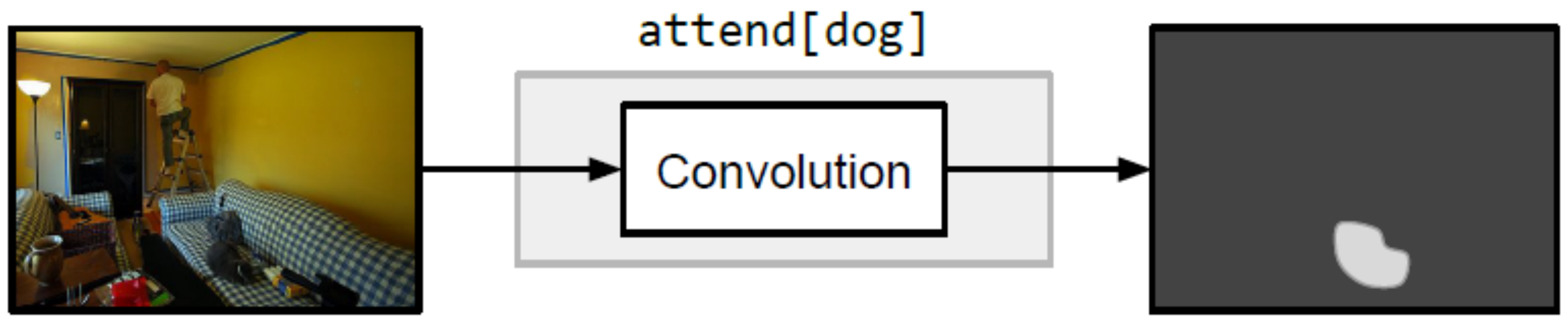


Takes image and outputs attention map, conditions on the [label], i.e., find()

Learning To Reason: End-to-End Module Networks for VQA

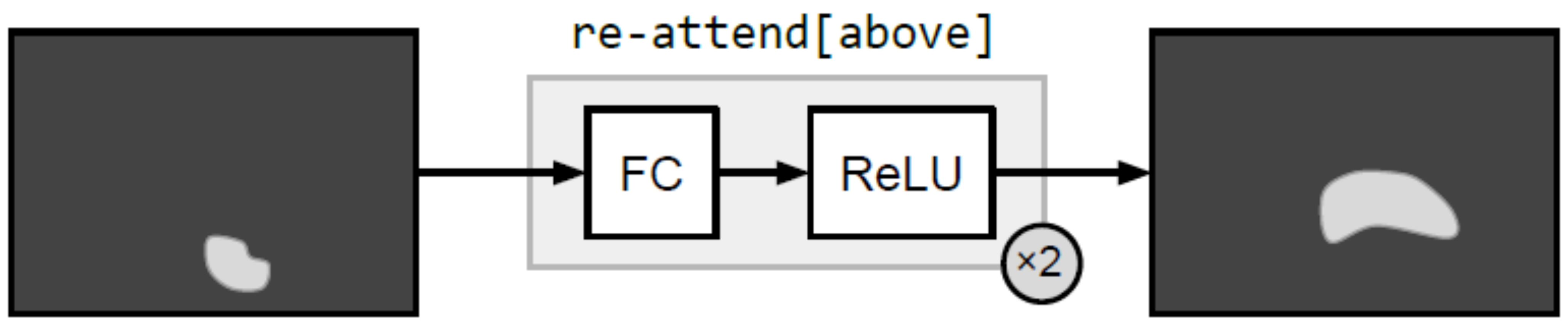
[Hu et al., 2017]

attend : $Image \rightarrow Attention$



Takes image and outputs attention map, conditions on the [label], i.e., find()

re-attend : $Attention \rightarrow Attention$

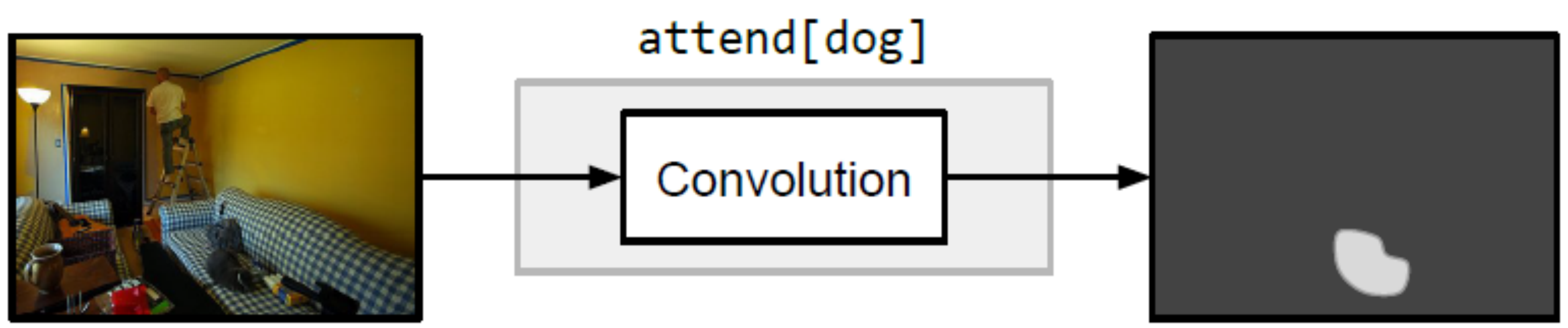


Shifts a attention based on logical relationship (e.g. above)

Learning To Reason: End-to-End Module Networks for VQA

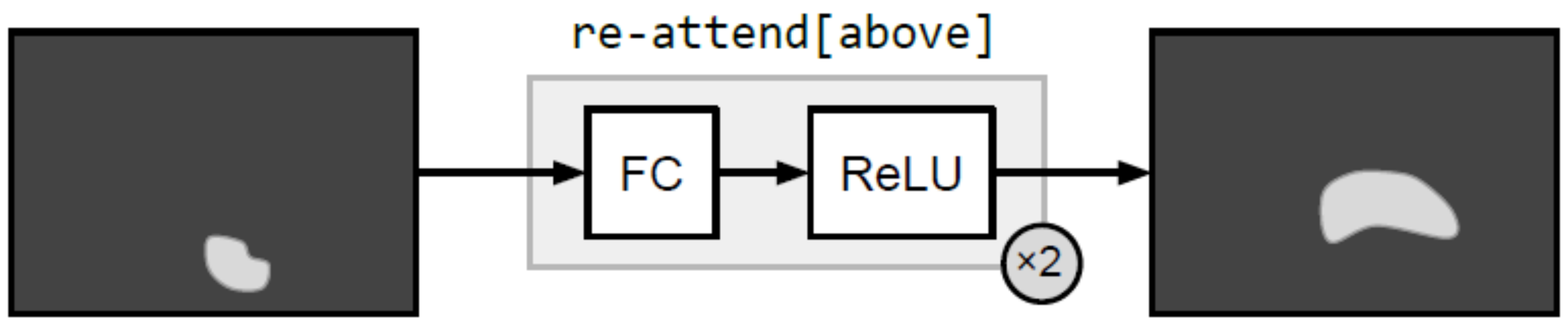
[Hu et al., 2017]

$\texttt{attend} : \mathit{Image} \rightarrow \mathit{Attention}$



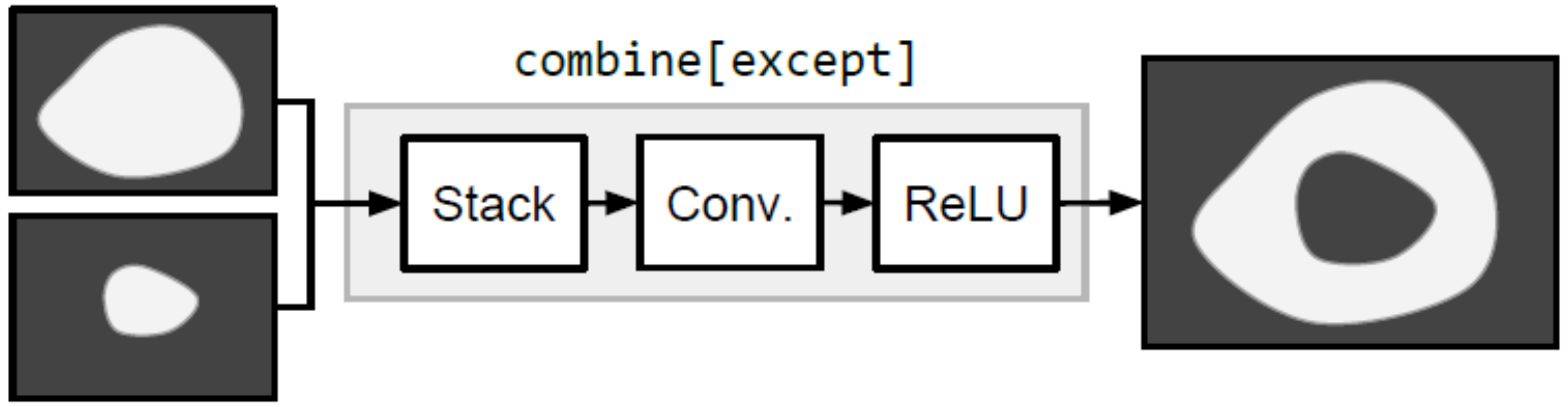
Takes image and outputs attention map, conditions on the [label], i.e., find()

$\texttt{re-attend} : \mathit{Attention} \rightarrow \mathit{Attention}$



Shifts a attention based on logical relationship (e.g. above)

$\texttt{combine} : \mathit{Attention} \times \mathit{Attention} \rightarrow \mathit{Attention}$

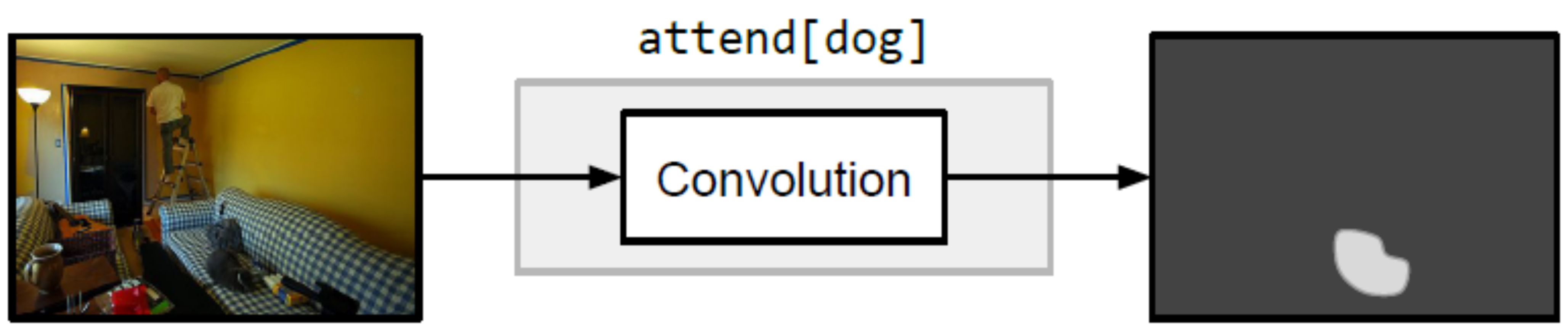


Logical relations on attention (e.g., and/or)

Learning To Reason: End-to-End Module Networks for VQA

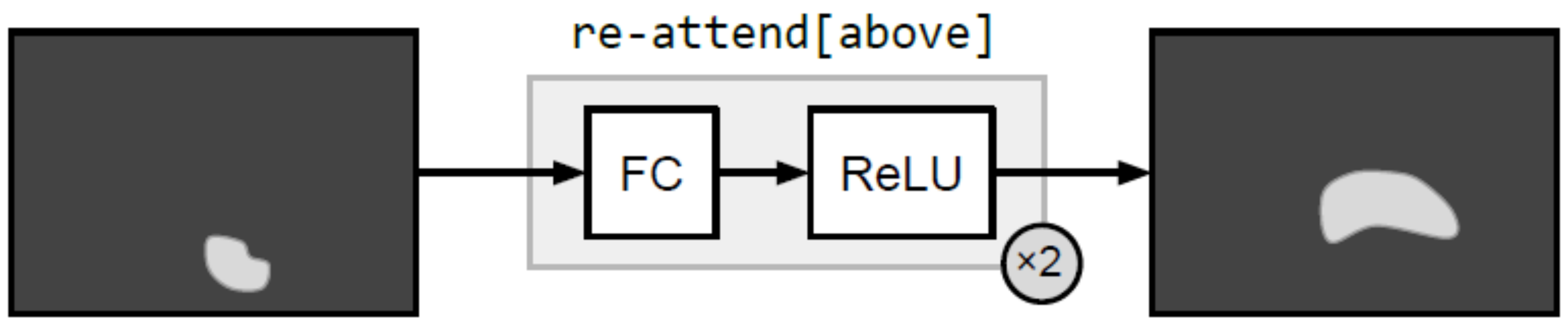
[Hu et al., 2017]

attend : $Image \rightarrow Attention$



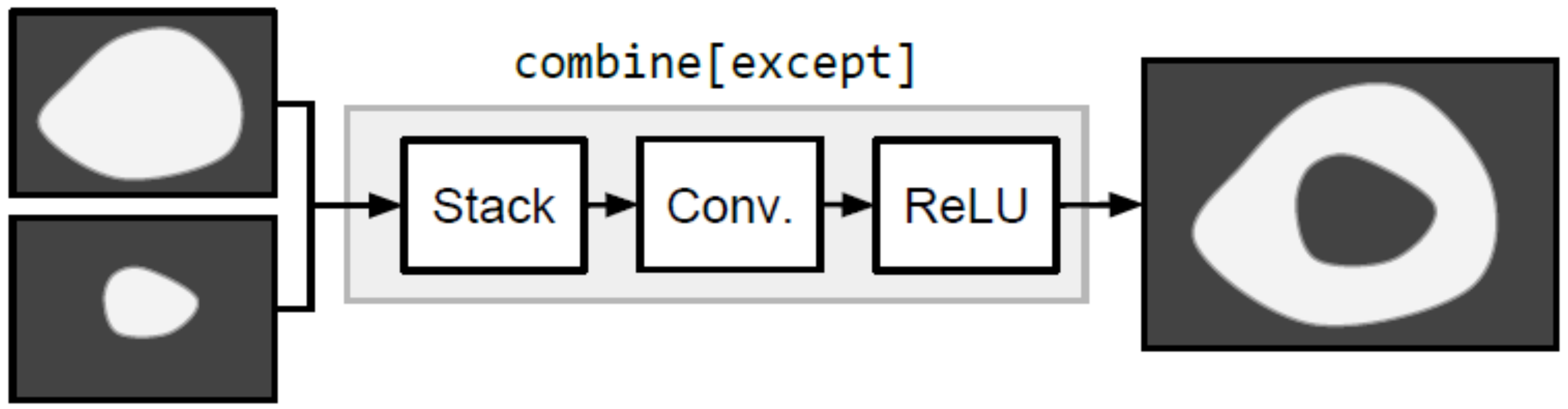
Takes image and outputs attention map, conditions on the [label], i.e., find()

re-attend : $Attention \rightarrow Attention$



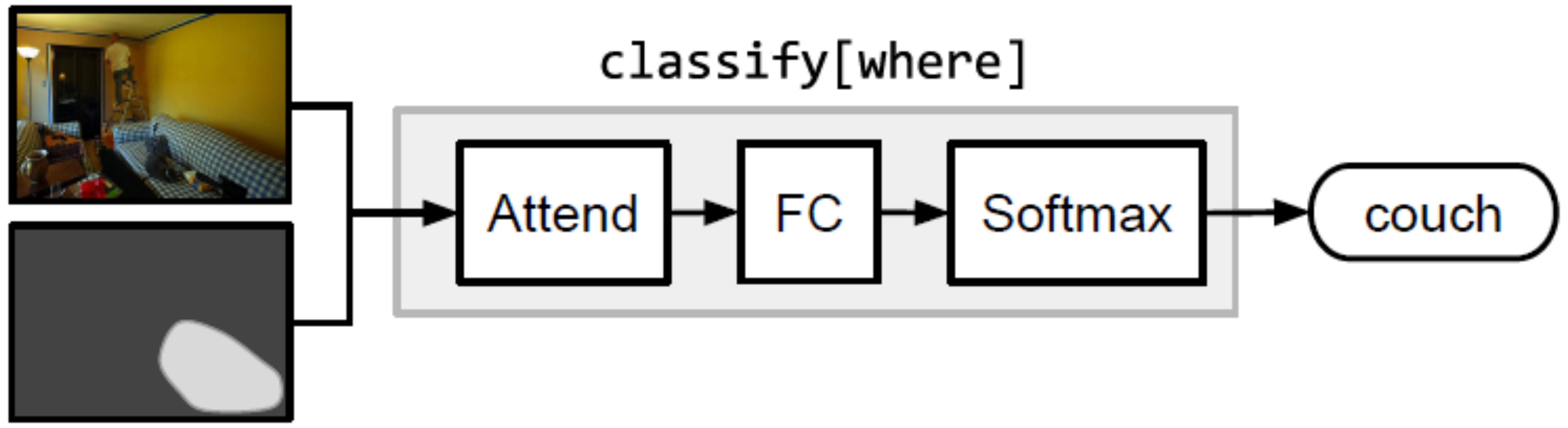
Shifts a attention based on logical relationship (e.g. above)

combine : $Attention \times Attention \rightarrow Attention$



Logical relations on attention (e.g., and/or)

classify : $Image \times Attention \rightarrow Label$



Given attention and image, generate a label

Learning To Reason: End-to-End Module Networks for VQA

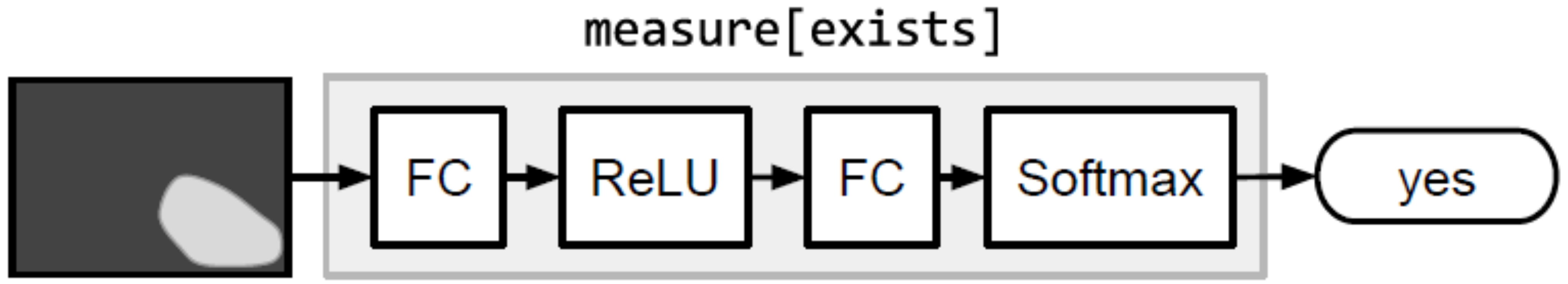
[Hu et al., 2017]

`attend` : $Image \rightarrow Attention$

Takes image and outputs attention map, conditions on the [label], i.e., find()

`re-attend` : $Attention \rightarrow Attention$

`measure` : $Attention \rightarrow Label$



relationship (e.g. above)

`combine` : $Attention \times Attention \rightarrow Attention$

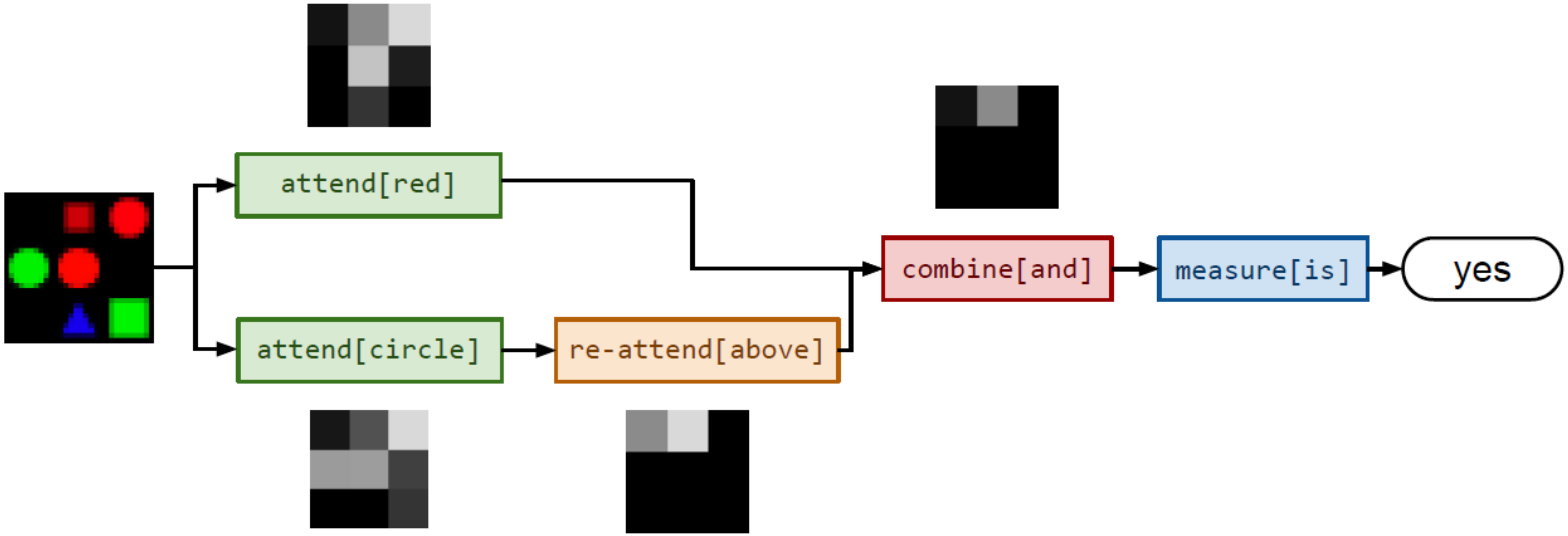
Logical relations on attention (e.g., and/or)

`classify` : $Image \times Attention \rightarrow Label$

Given attention and image, generate a label

Learning To Reason: End-to-End Module Networks for VQA

Is there a red shape above a circle?

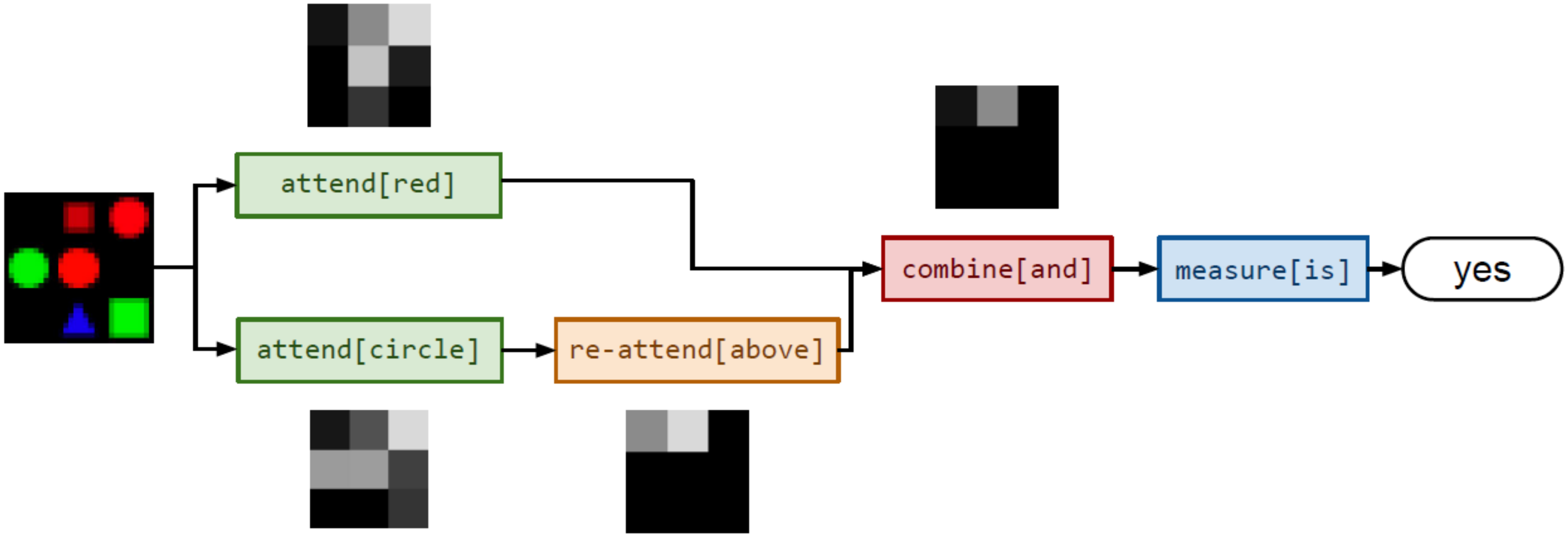


[Andreas et al., 2017]

Learning To Reason: End-to-End Module Networks for VQA

Is there a red shape above a circle?

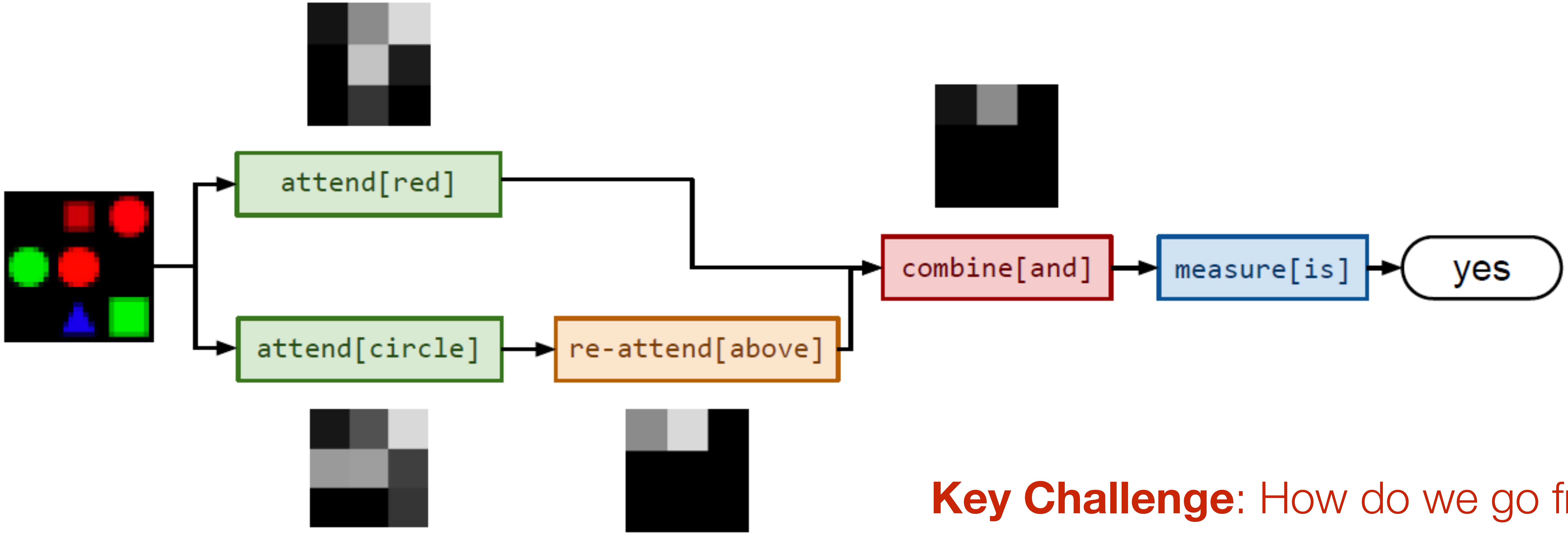
Note: Every sample = different computational graph (but that's OK)



[Andreas et al., 2017]

Learning To Reason: End-to-End Module Networks for VQA

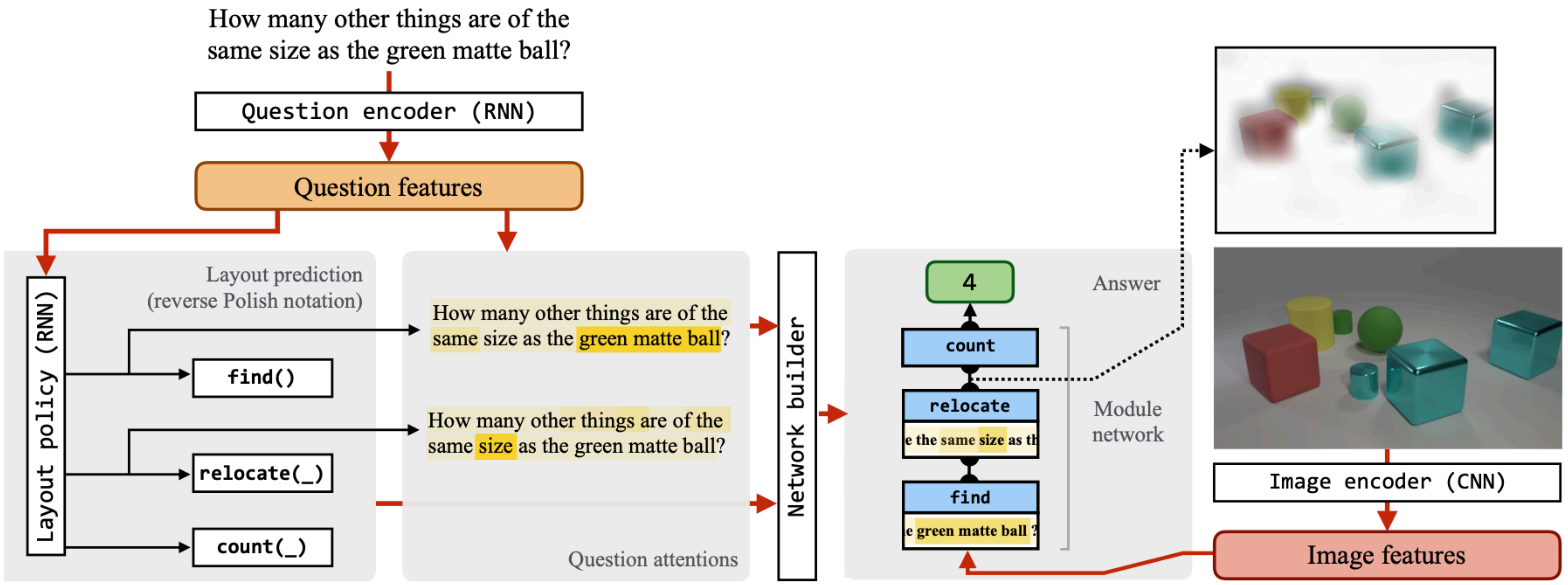
Is there a red shape above a circle?



Key Challenge: How do we go from question to module layout

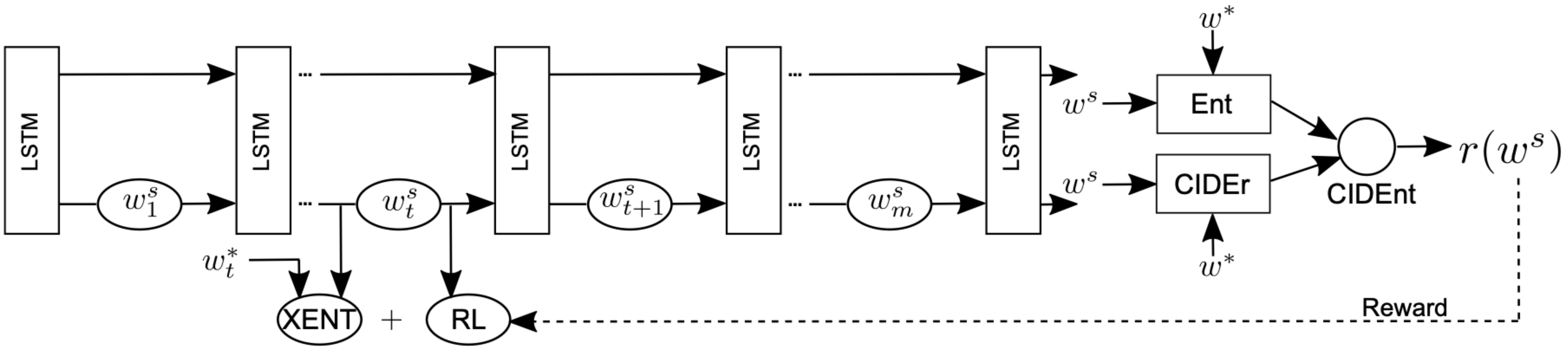
[Andreas et al., 2017]

Learning To Reason: End-to-End Module Networks for VQA



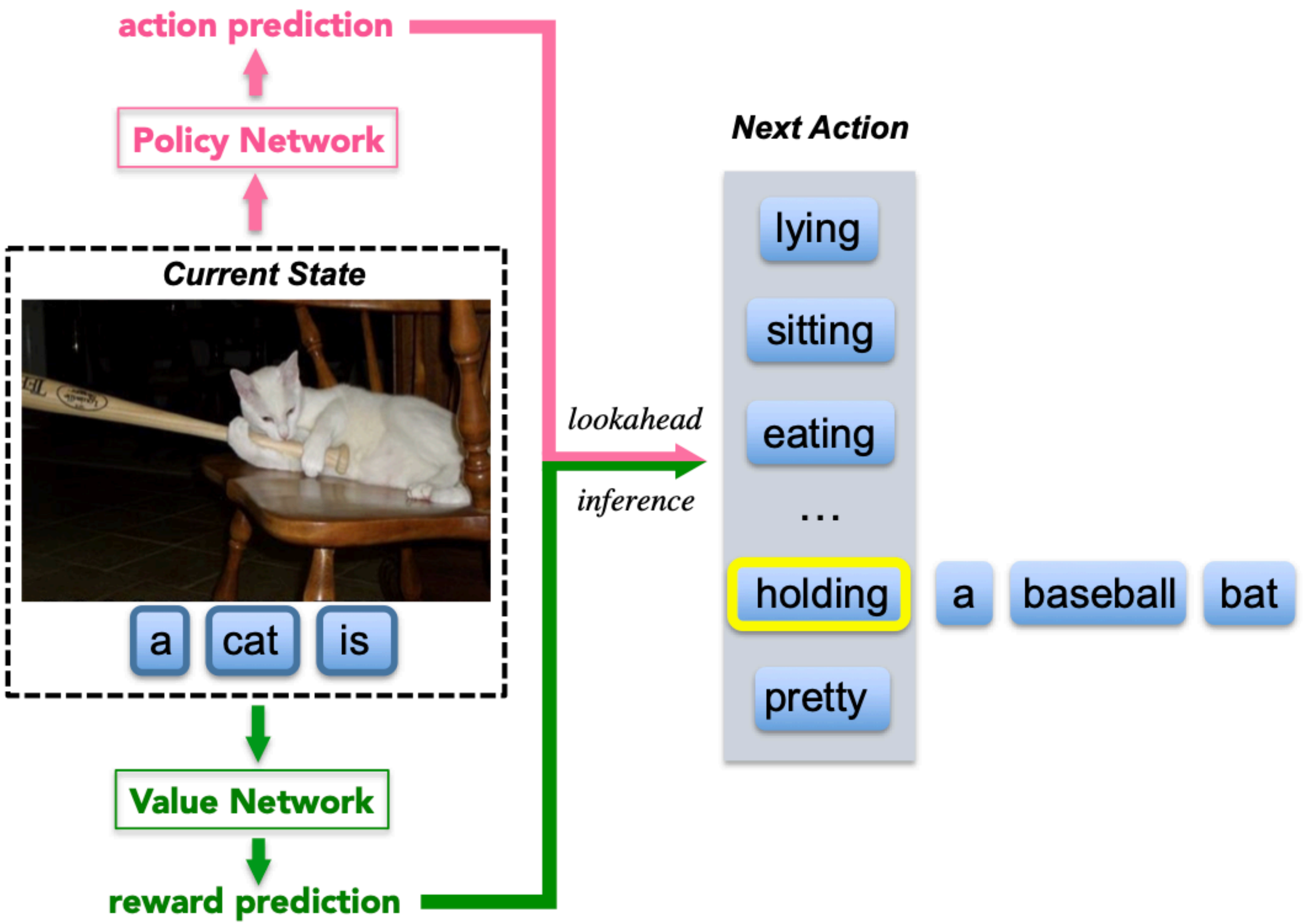
[Hu et al., 2017]

Deep **RL-based** Image Captioning /w REINFORCE



[Pasunuru and Bansal]

Deep **RL-based** Image Captioning



[Ren et al. 2017]

Summary

Policy gradients: very general but suffer from high variance so requires a lot of samples. ***Challenge***: sample-efficiency

Q-learning: does not always work but when it works, usually more sample-efficient. ***Challenge***: exploration

Guarantees:
— Policy Gradients: Converges to a local minima of $J(\theta)$, often good enough!
— Q-learning: Zero guarantees since you are approximating Bellman equation with a complicated function approximator

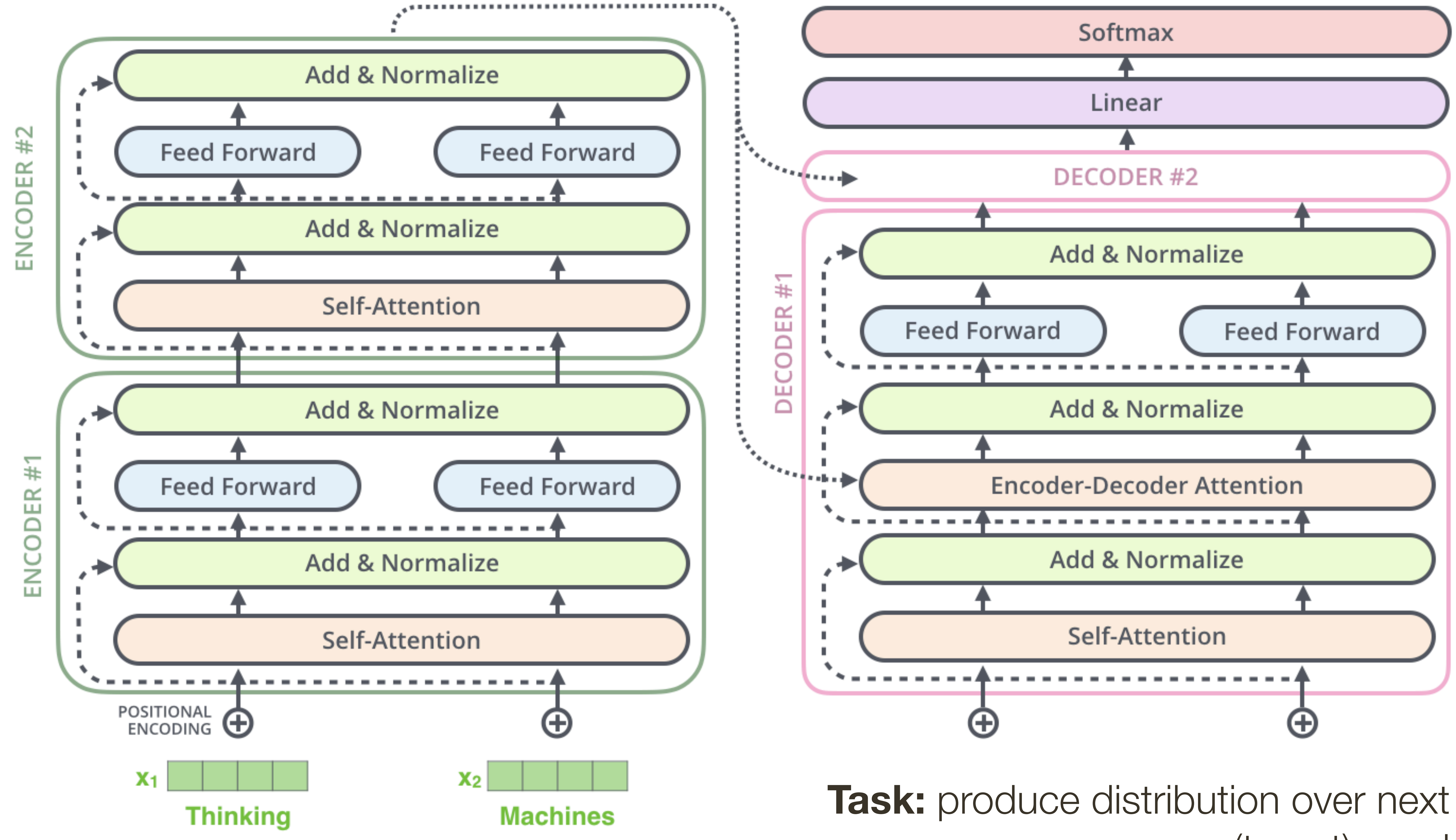
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 22: Large Scale Visio-Lingual Models (cont.)

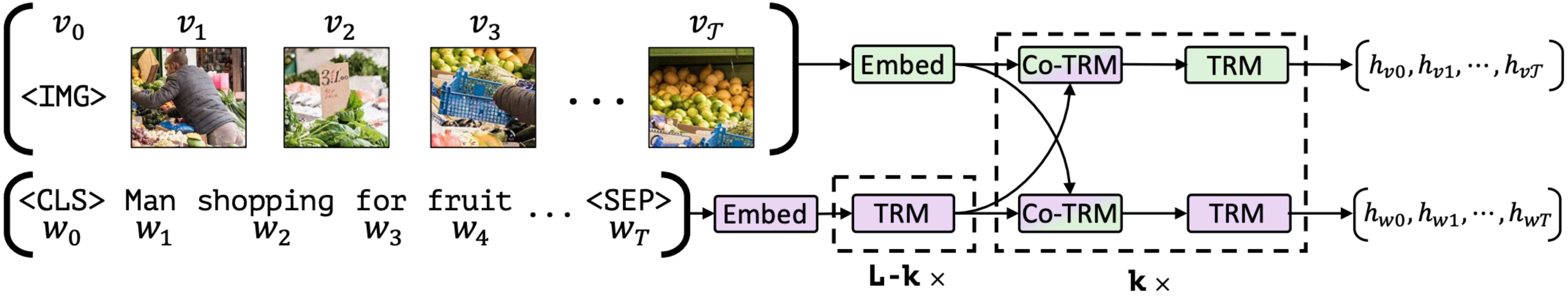
Review: Transformers

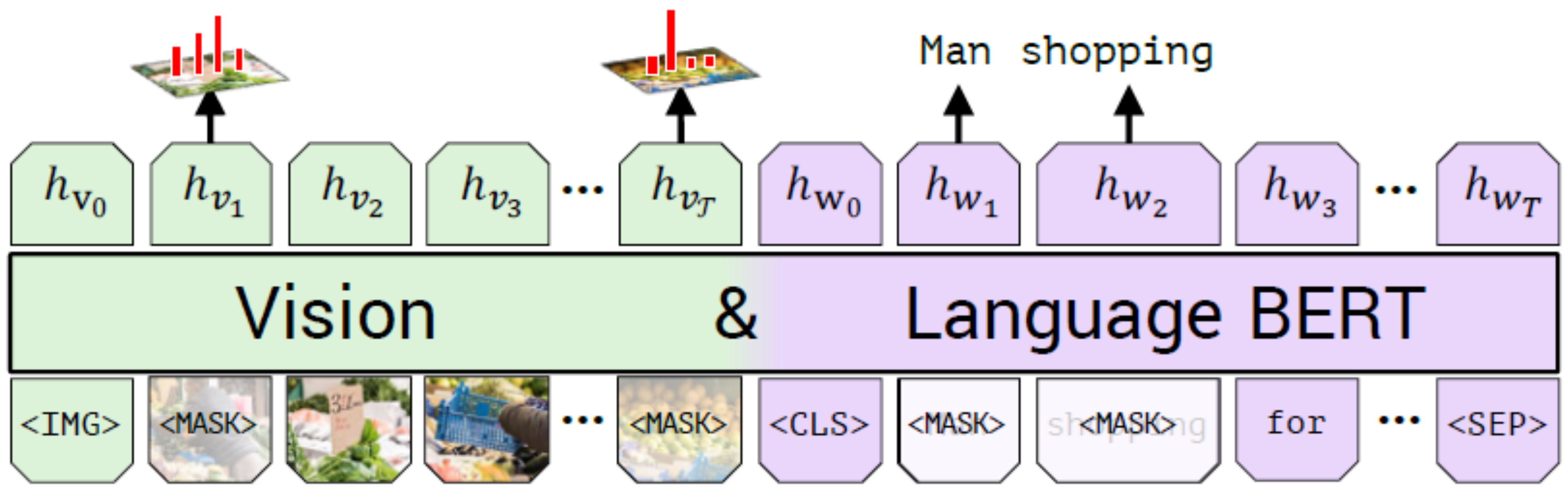
Task: produce contextualized representation of each (source) words



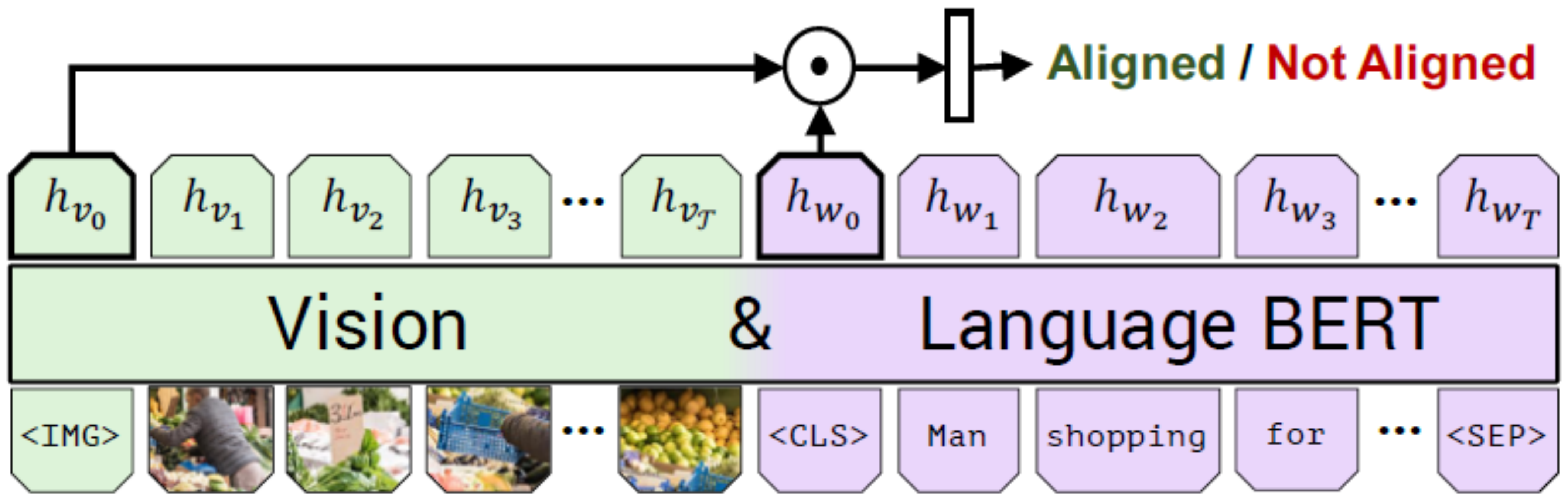
Task: produce distribution over next (target) word

Visual BERT (ViLBERT)





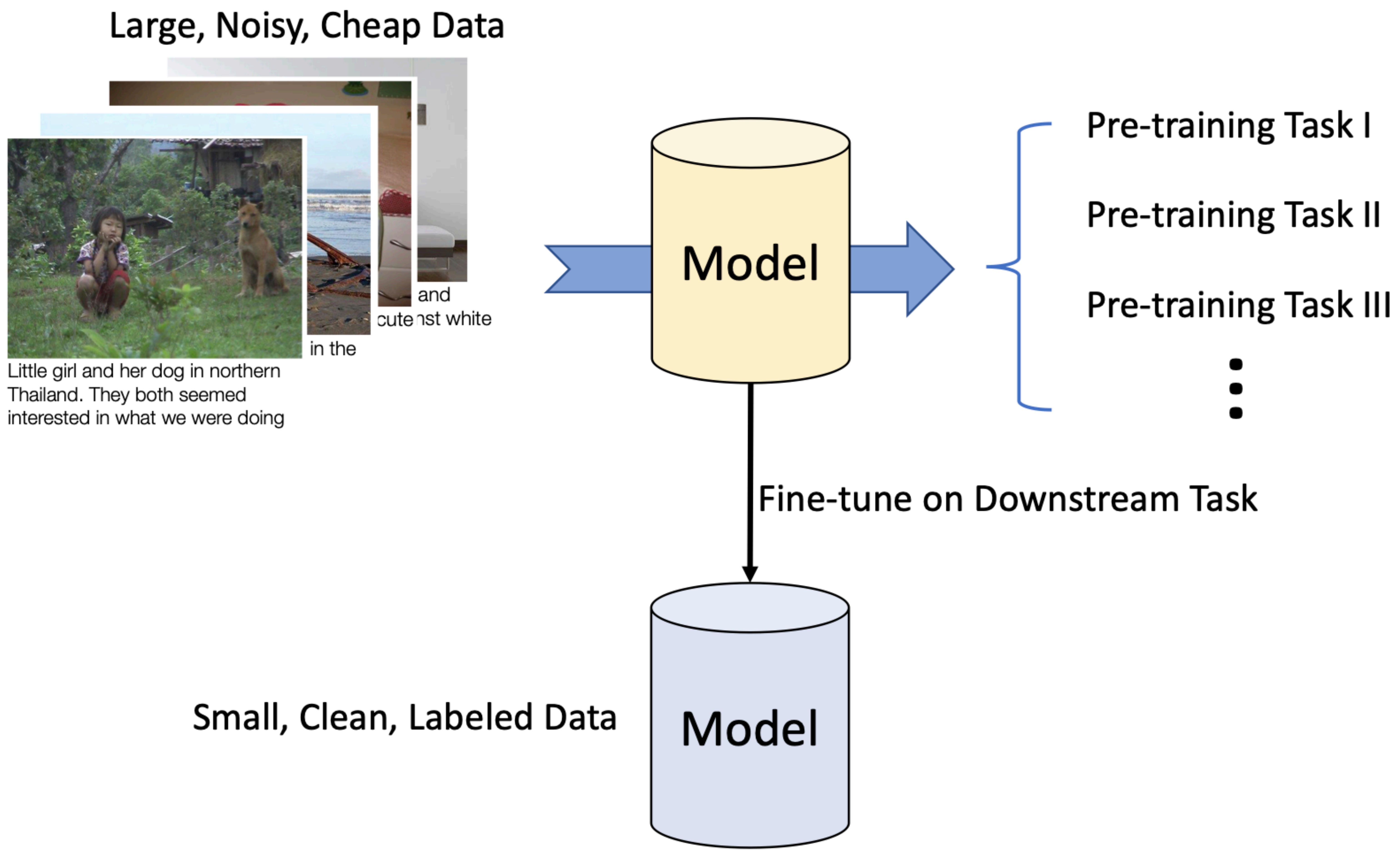
(a) Masked multi-modal learning



(b) Multi-modal alignment prediction

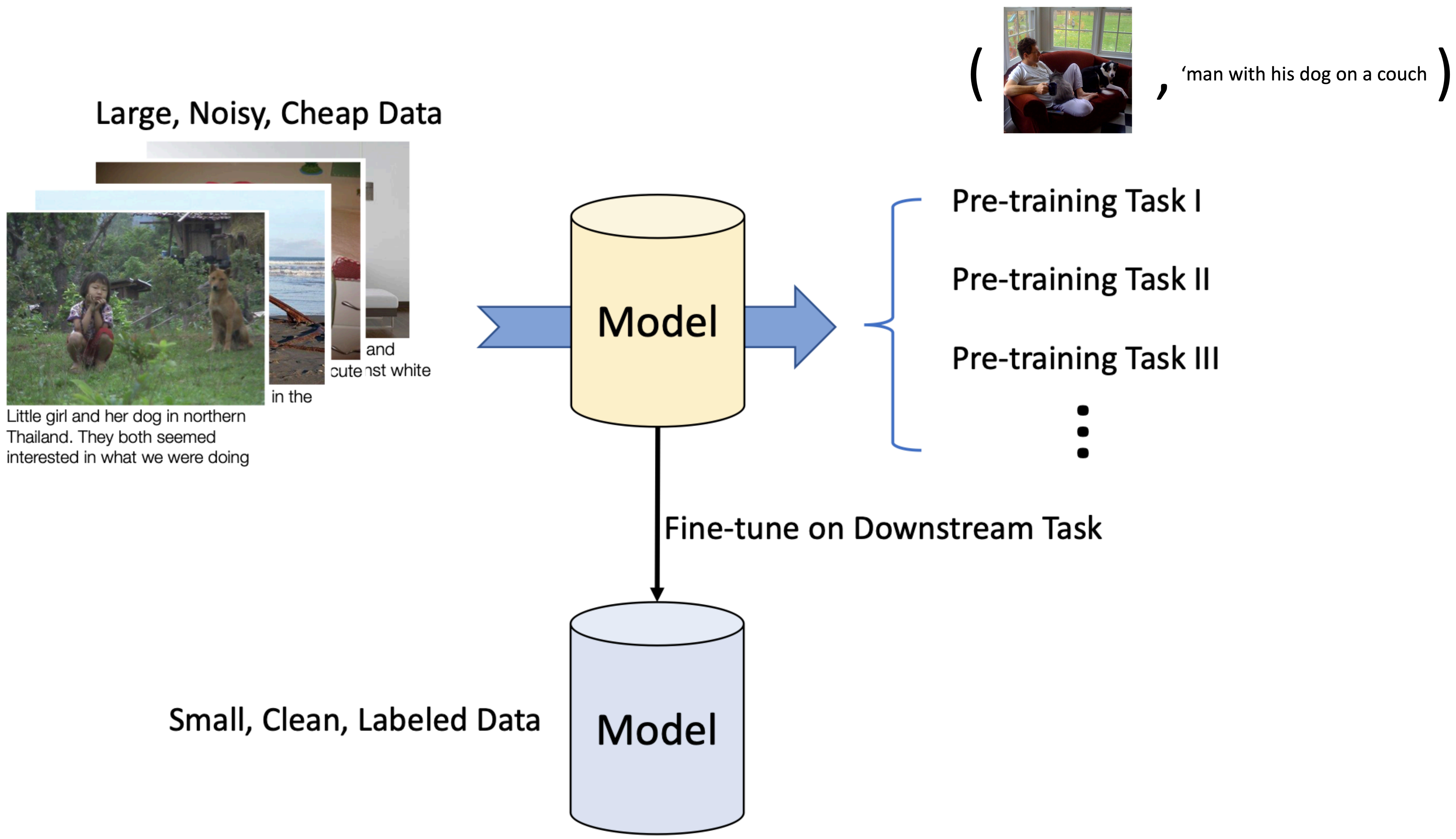
[Lu et al., 2019]

Pre-training and Foundational Models



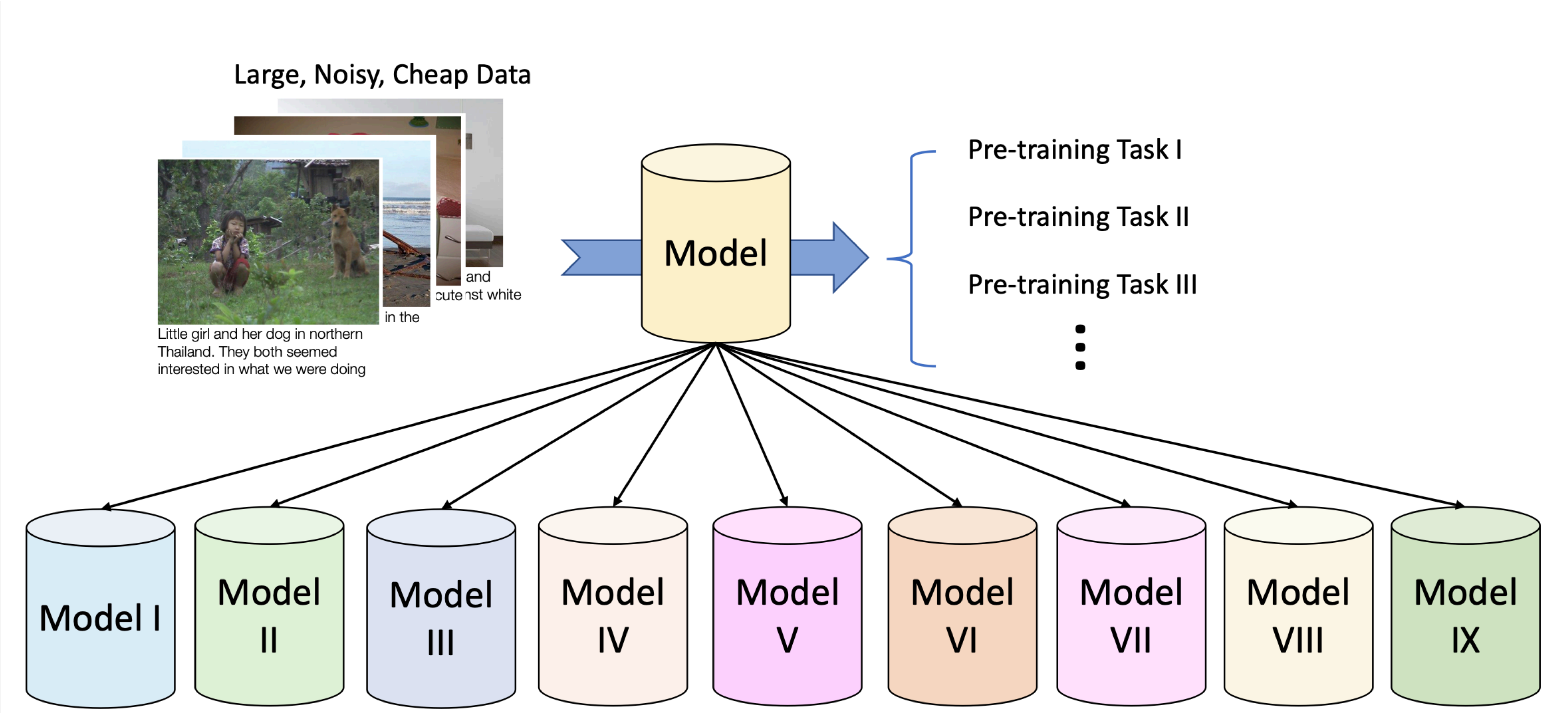
Slide from **Zhe Gan**

Pre-training and Foundational Models



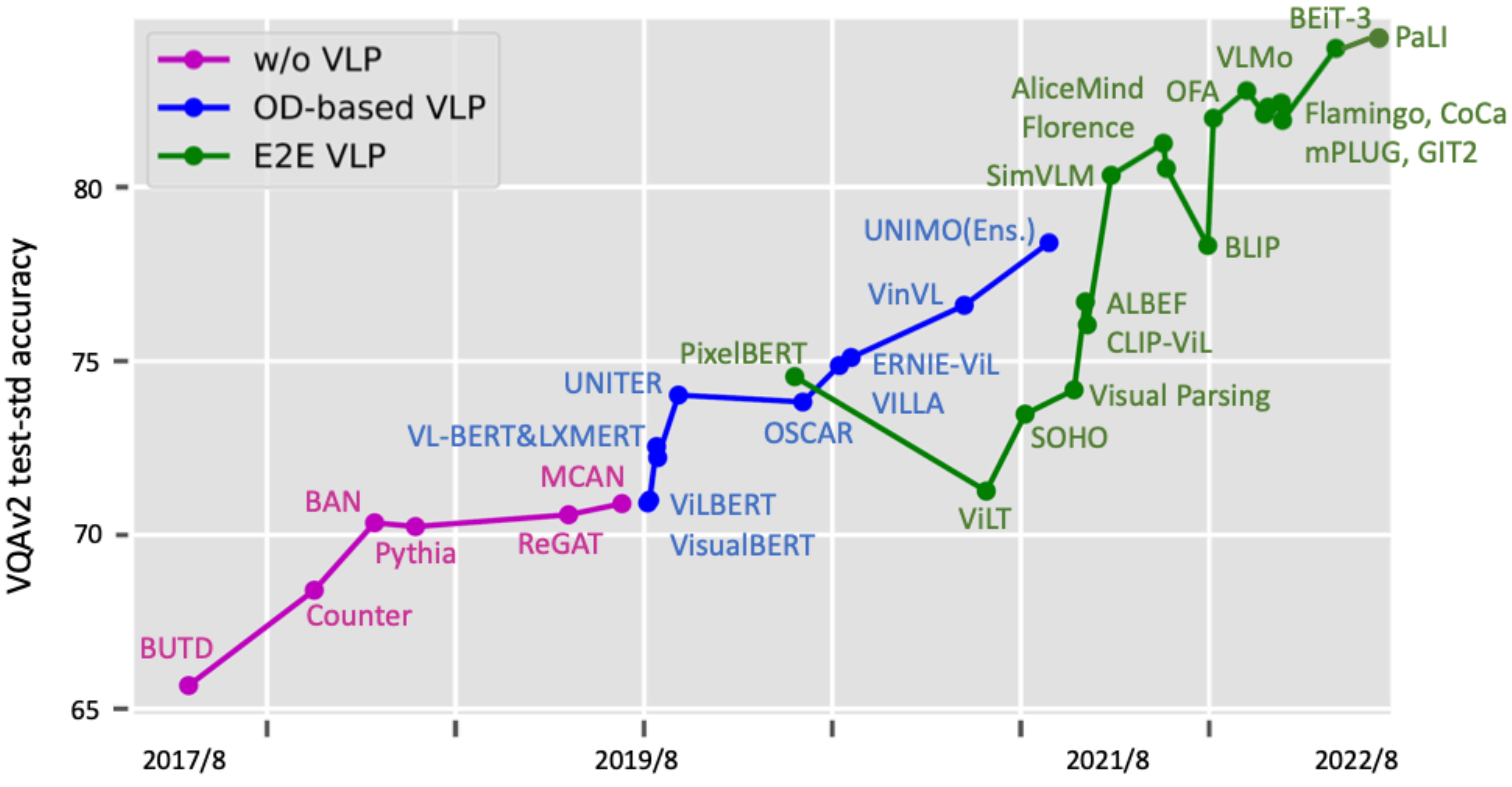
Slide from **Zhe Gan**

Pre-training and **Foundational Models**



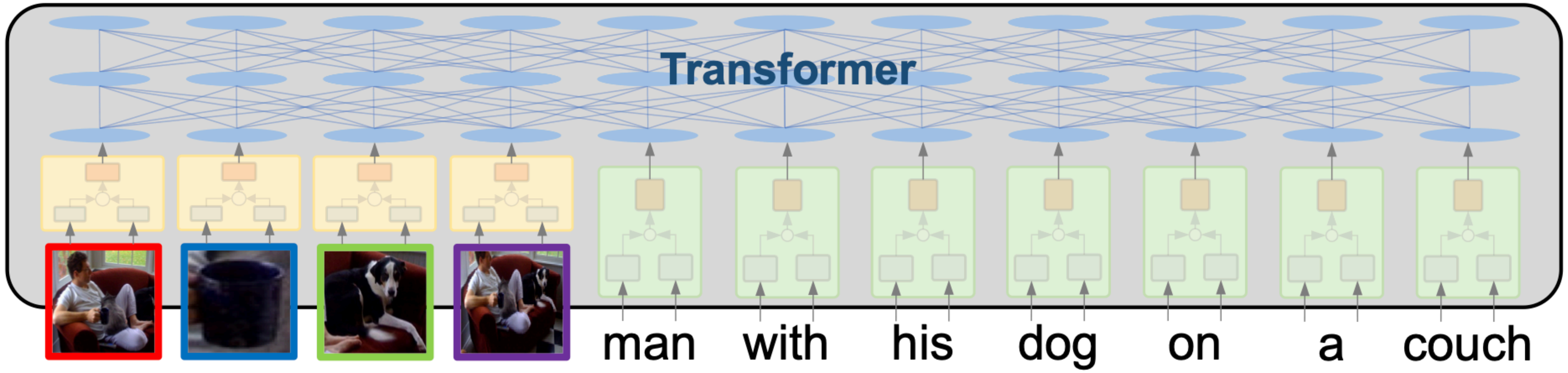
Slide from **Zhe Gan**

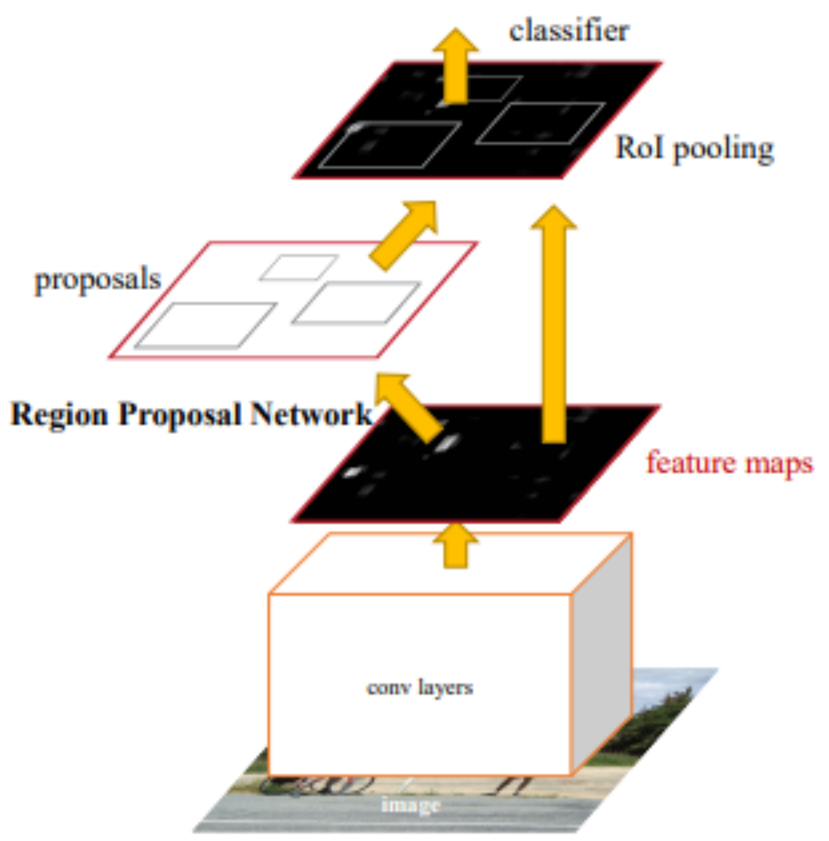
Recent History of **Visio-Lingual Models**



Slide from **Zhe Gan**

UNITER: UNiversal Image-TExt Representation Learning



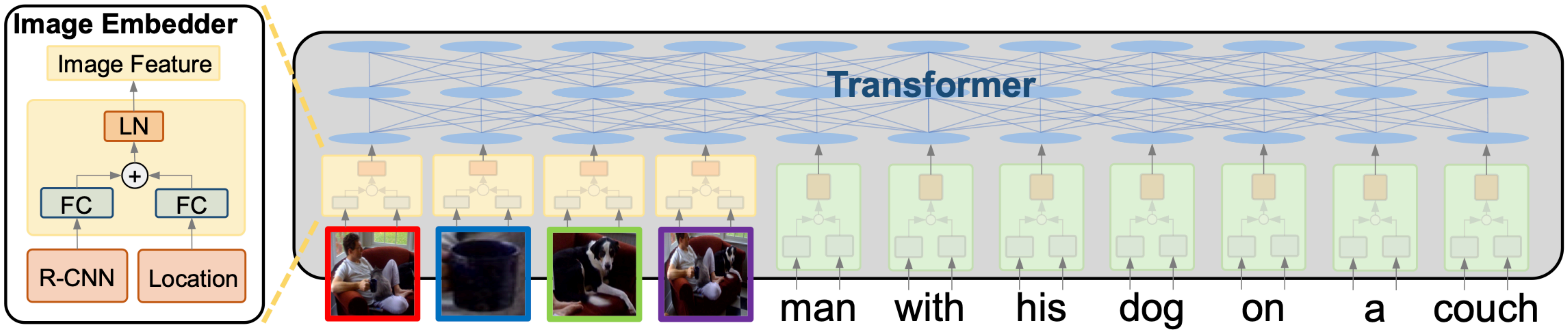


Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

UNITER: UNiversal Image-TExt Representation Learning

300 dim

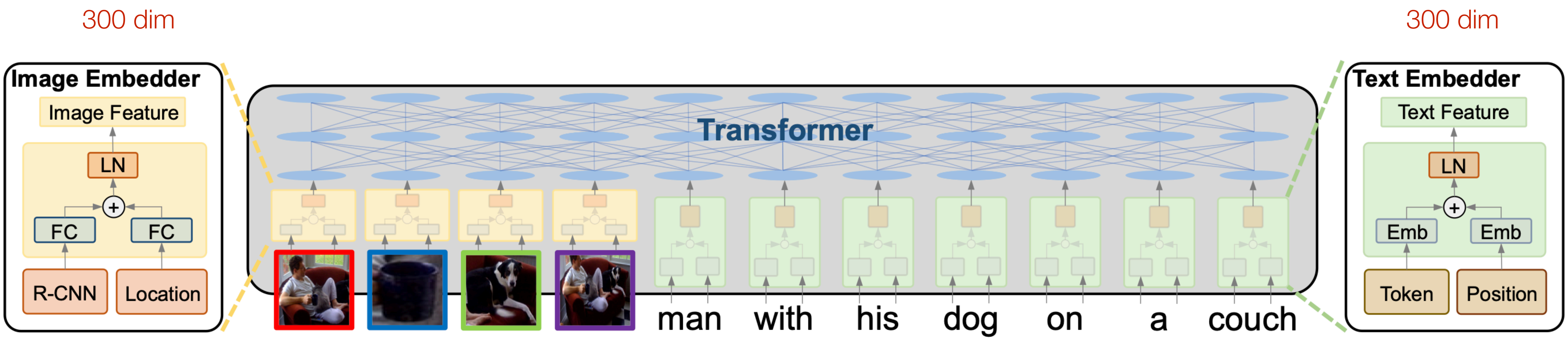


4096 dim   7 dim = [x1, y1, x2, y2, w, h, w ∗ h]

Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

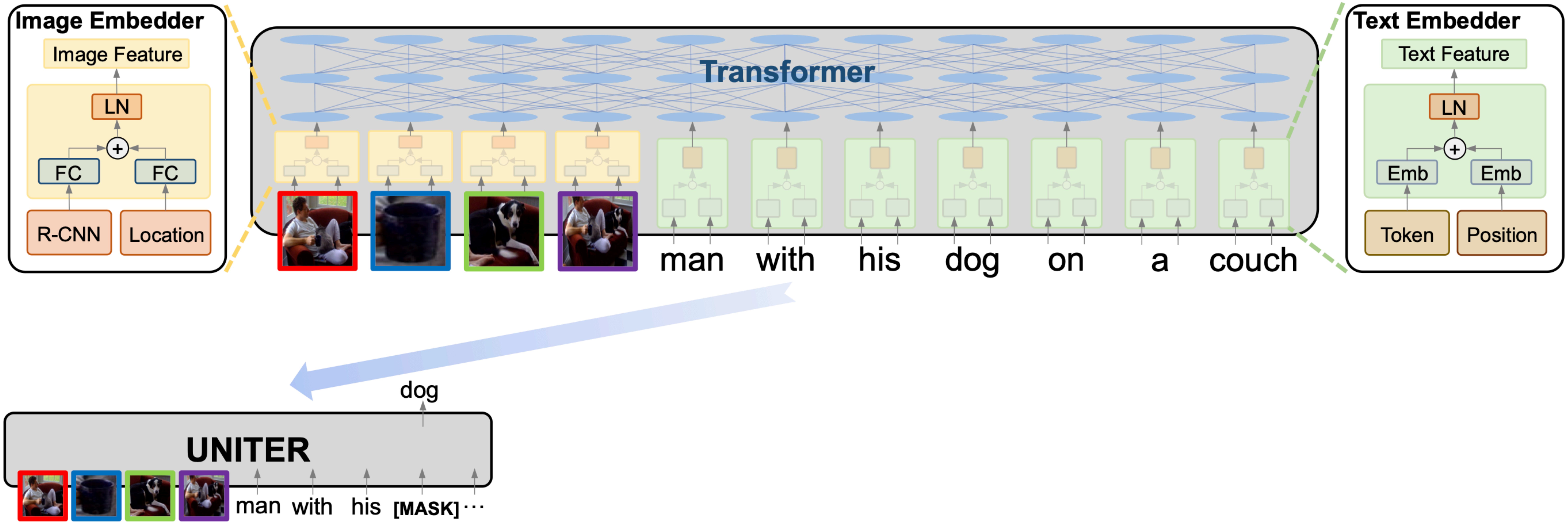
UNITER: UNiversal Image-TExt Representation Learning



Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

UNITER: UNiversal Image-TExt Representation Learning

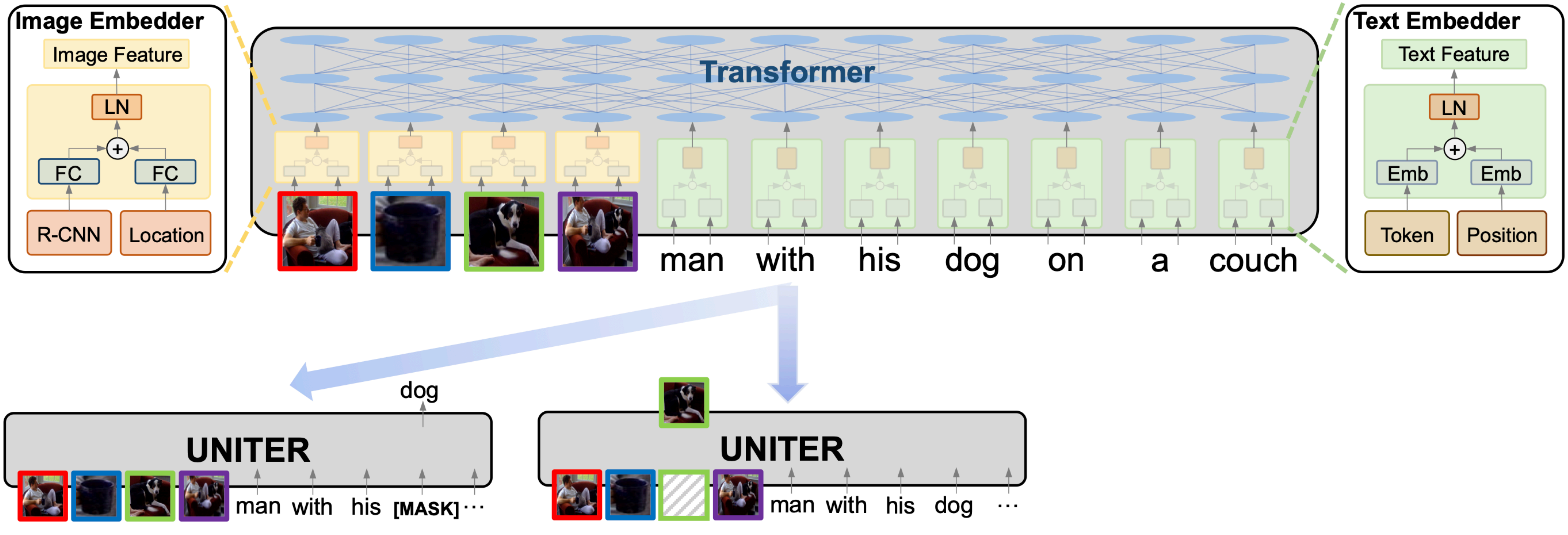


Masked Language Modeling (MLM)

Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

UNITER: UNiversal Image-TExt Representation Learning



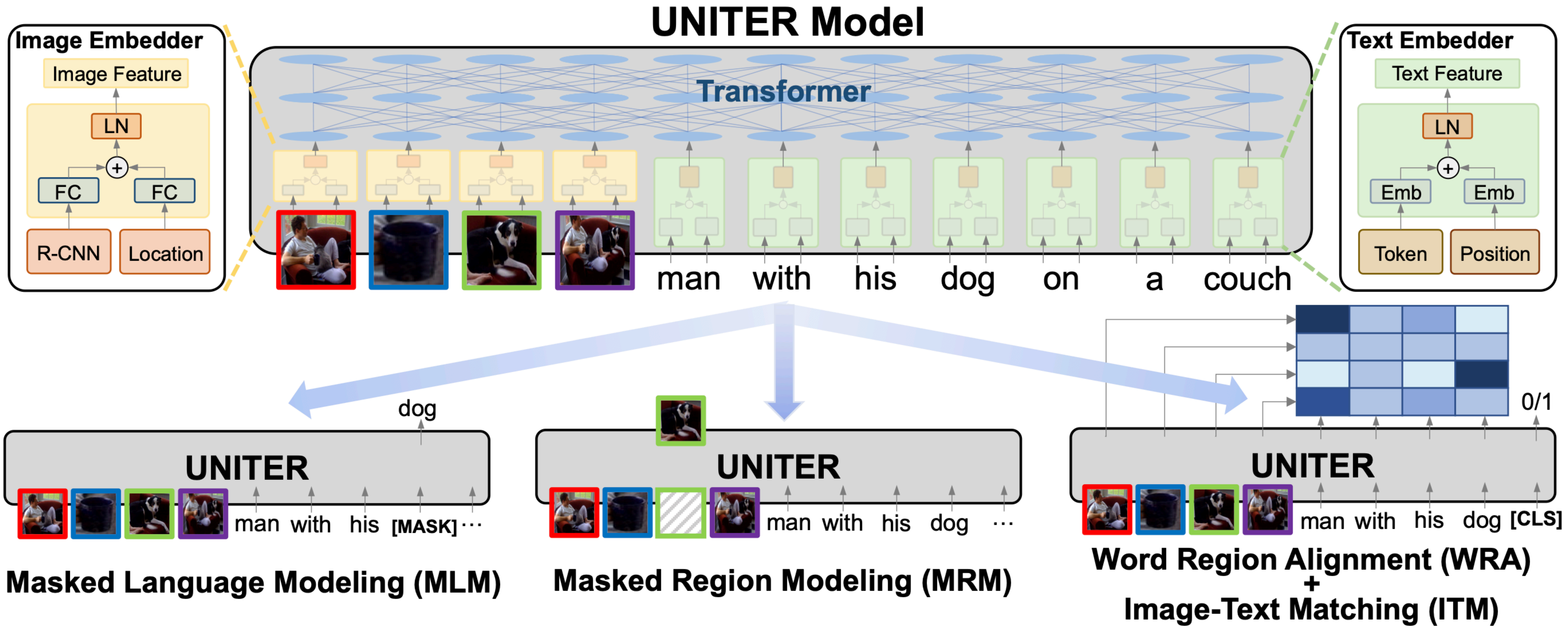
Masked Language Modeling (MLM)

Masked Region Modeling (MRM)

Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

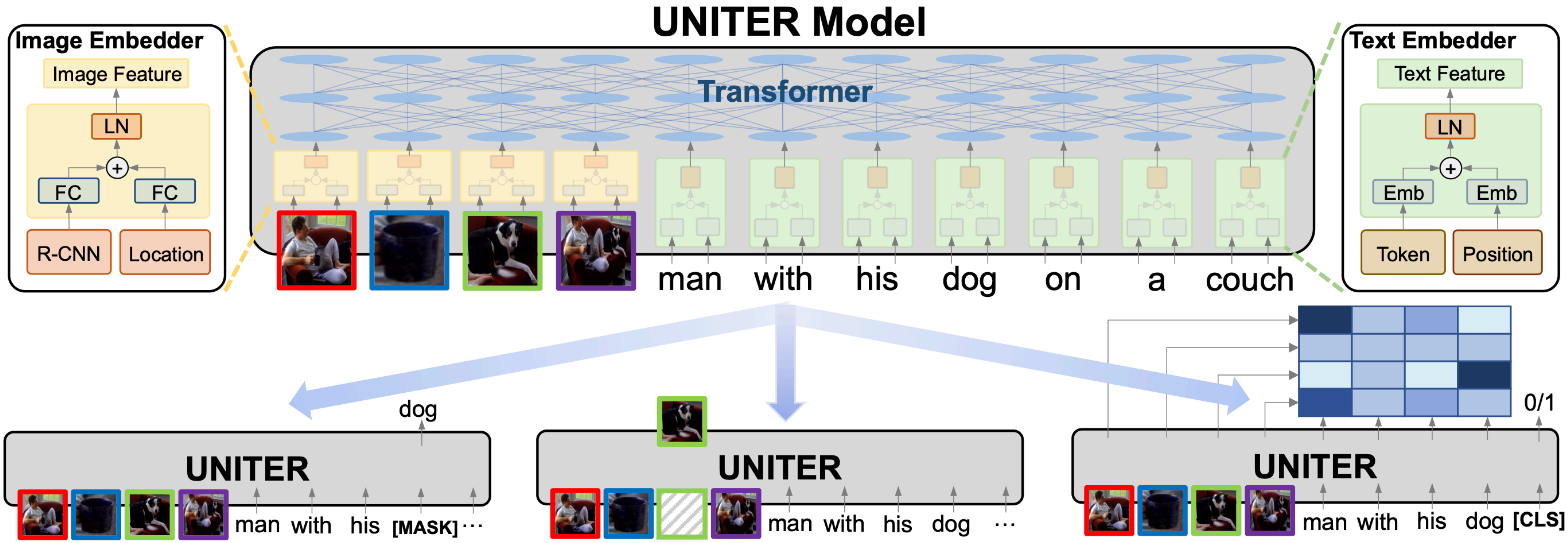
UNITER: UNiversal Image-TExt Representation Learning



Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

UNITER: UNiversal Image-TExt Representation Learning



Masked Language Modeling (MLM)

Masked Region Modeling (MRM)

— Masked Region Feature Regression (MRFR)

— Masked Region Classification (MRC)

— Masked Region Classification with KL-Divergence (MRC-kl)

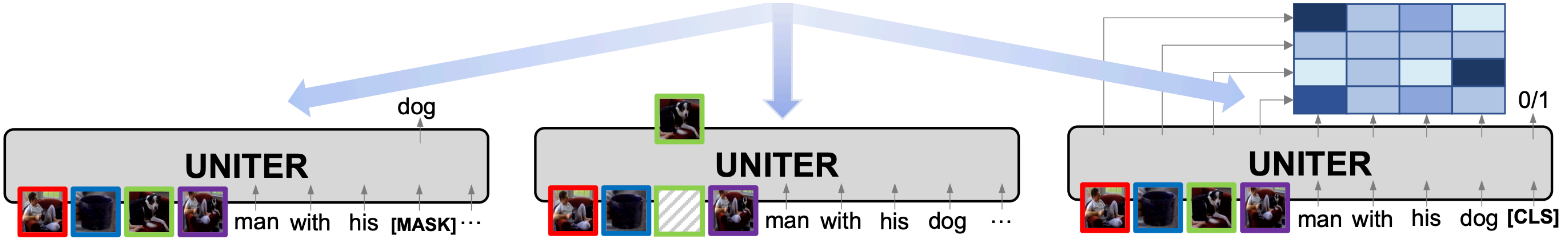
Word Region Alignment (WRA) + Image-Text Matching (ITM)

Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

UNITER: UNiversal Image-TExt Representation Learning

Pre-training Tasks	Meta-Sum	VQA	IR (Flickr)	TR (Flickr)	NLVR^2	Ref-COCO+
		test-dev	val	val	dev	val^d
MLM + ITM + MRC	393.97	71.46	81.39	91.45	76.18	73.49
MLM + ITM + MRFR	396.24	71.73	81.76	92.31	76.21	74.23
MLM + ITM + MRC-kl	397.09	71.63	82.10	92.57	76.28	74.51
MLM + ITM + MRC-kl + MRFR	399.97	71.92	83.73	92.87	76.93	74.52



Masked Language Modeling (MLM)

Masked Region Modeling (MRM)

— Masked Region Feature Regression (MRFR)

— Masked Region Classification (MRC)

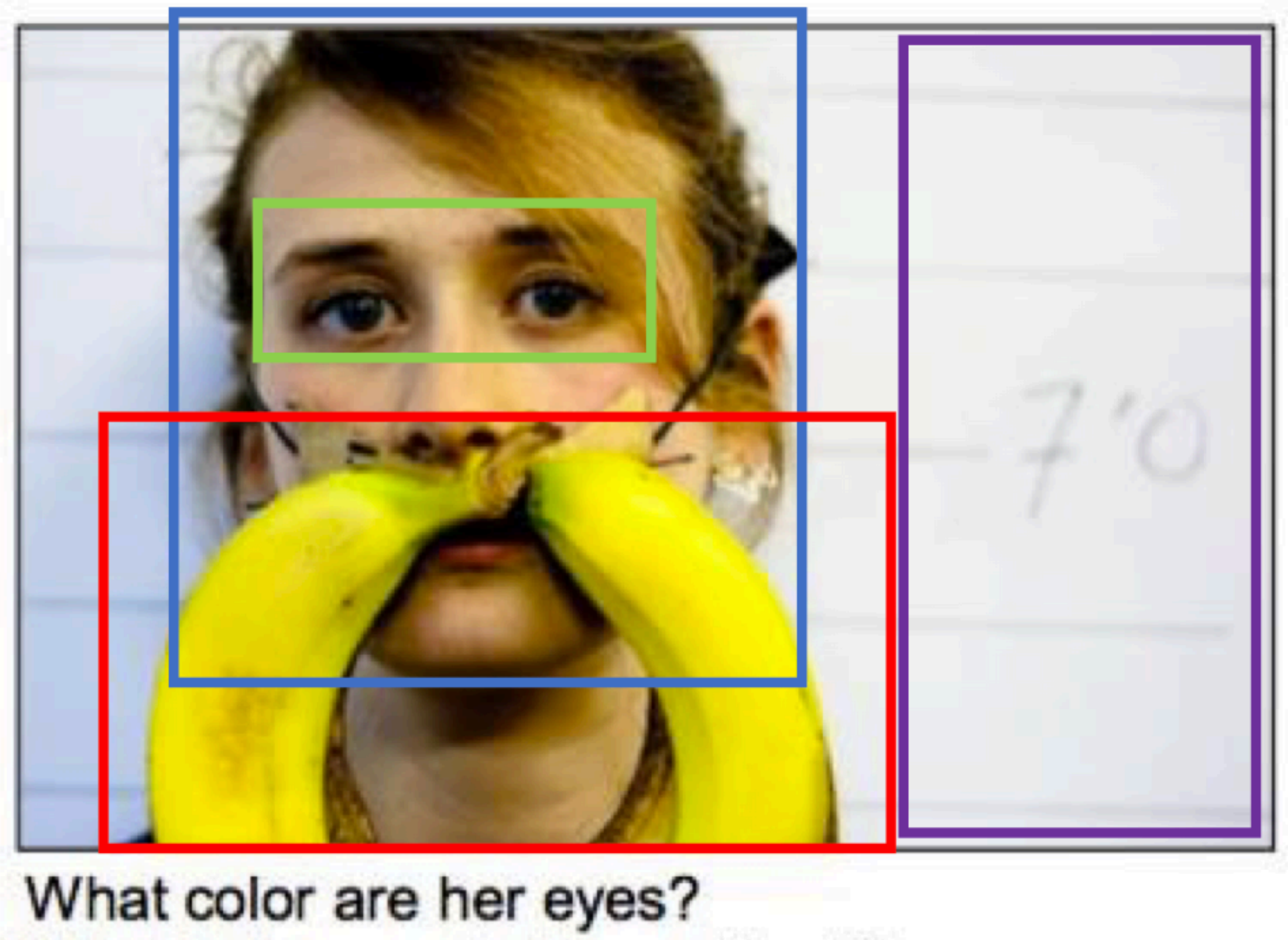
— Masked Region Classification with KL-Divergence (MRC-kl)

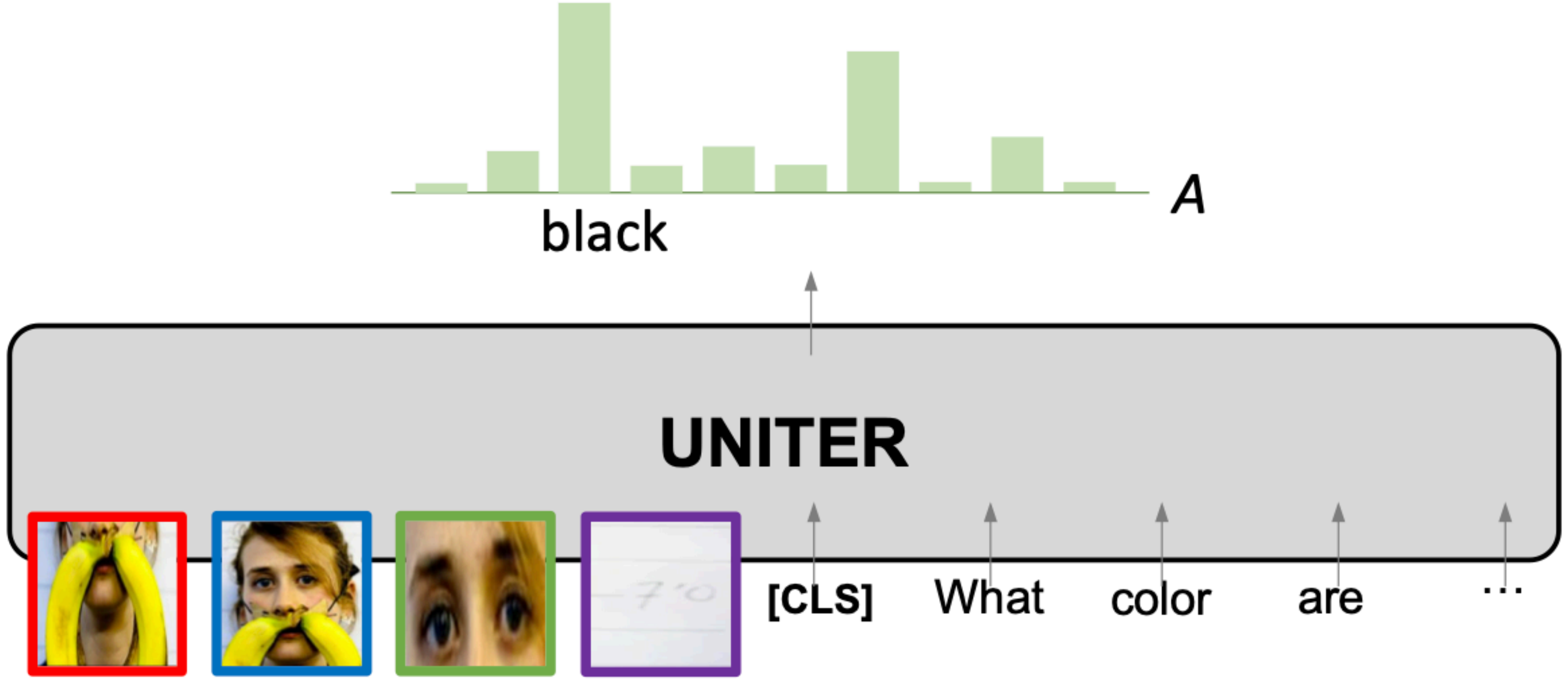
Word Region Alignment (WRA) + Image-Text Matching (ITM)

Slide from **Zhe Gan**

[Chen et al. ECCV 2020]

Downstream Task 1: **Visual Question Answering**





Slide from **Zhe Gan**

[Antol et al., ICCV 2015]

Downstream Task 2: **Visual Entailment**

+

- *Two woman are holding packages.*
- *The sisters are hugging goodbye while holding to go packages after just eating lunch.*
- *The men are fighting outside a deli.*

=

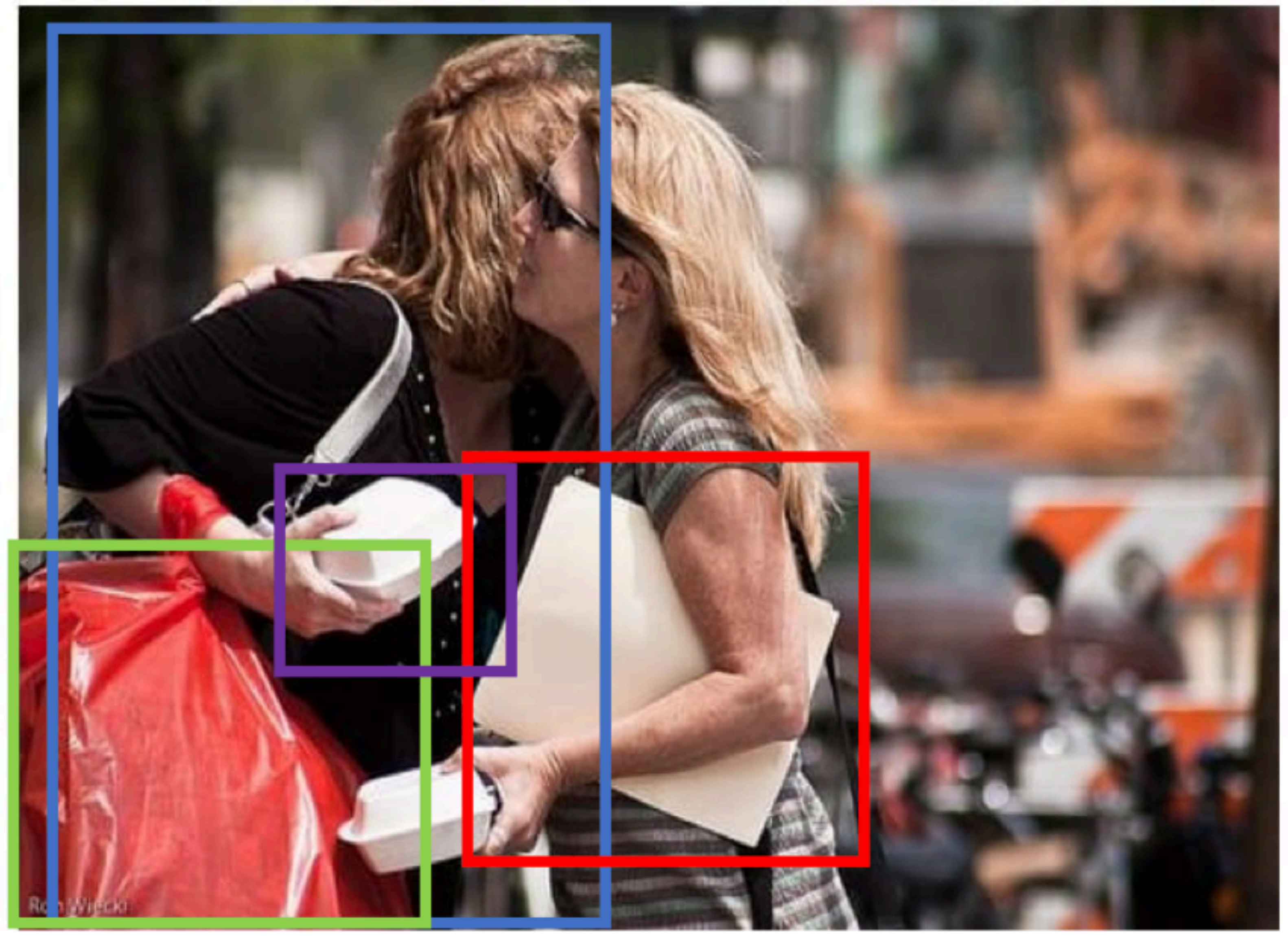
- *Entailment*
- *Neutral*
- *Contradiction*

Premise *Hypothesis* *Answer*

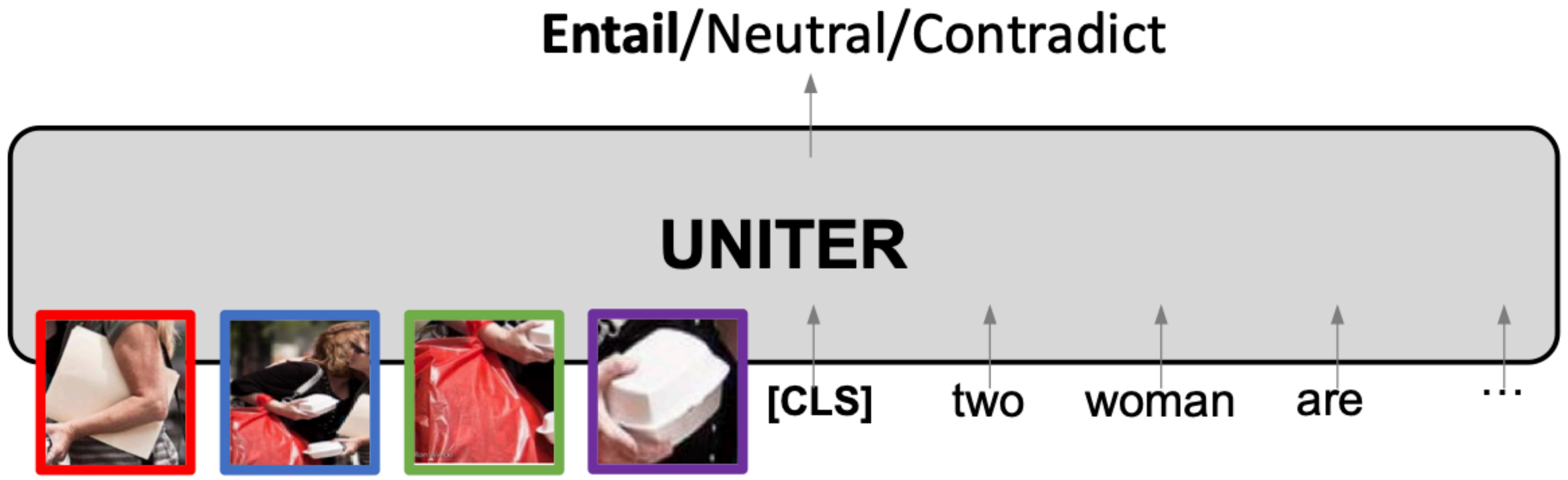
Slide from **Zhe Gan**

[Xie et al., 2019]

Downstream Task 2: **Visual Entailment**



Two woman are holding packages.



Slide from **Zhe Gan**

Downstream Task 3: Natural Language for **Visual Reasoning**

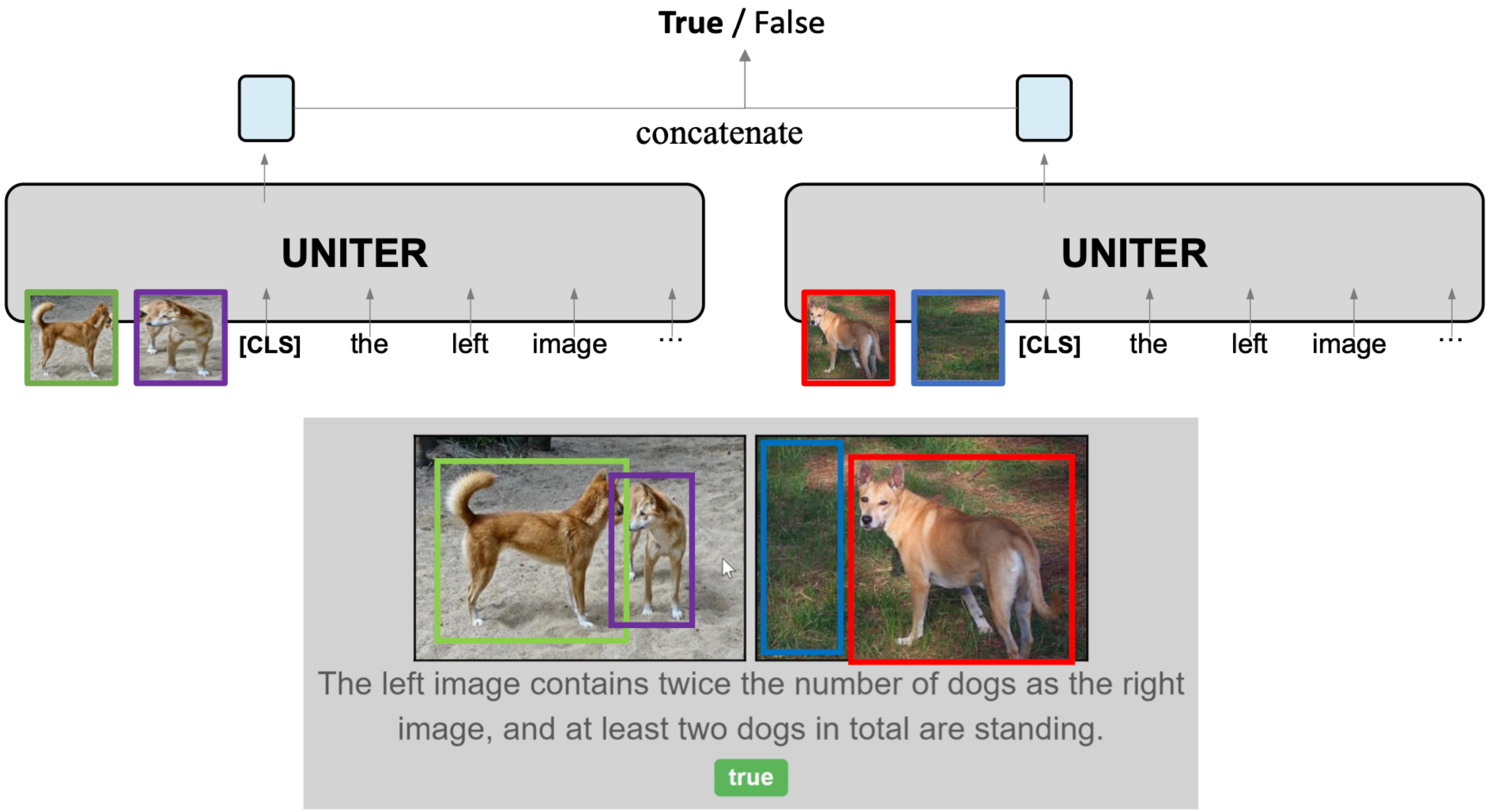
The left image contains twice the number of dogs as the right image, and at least two dogs in total are standing.

true

One image shows exactly two brown acorns in back-to-back caps on green foliage.

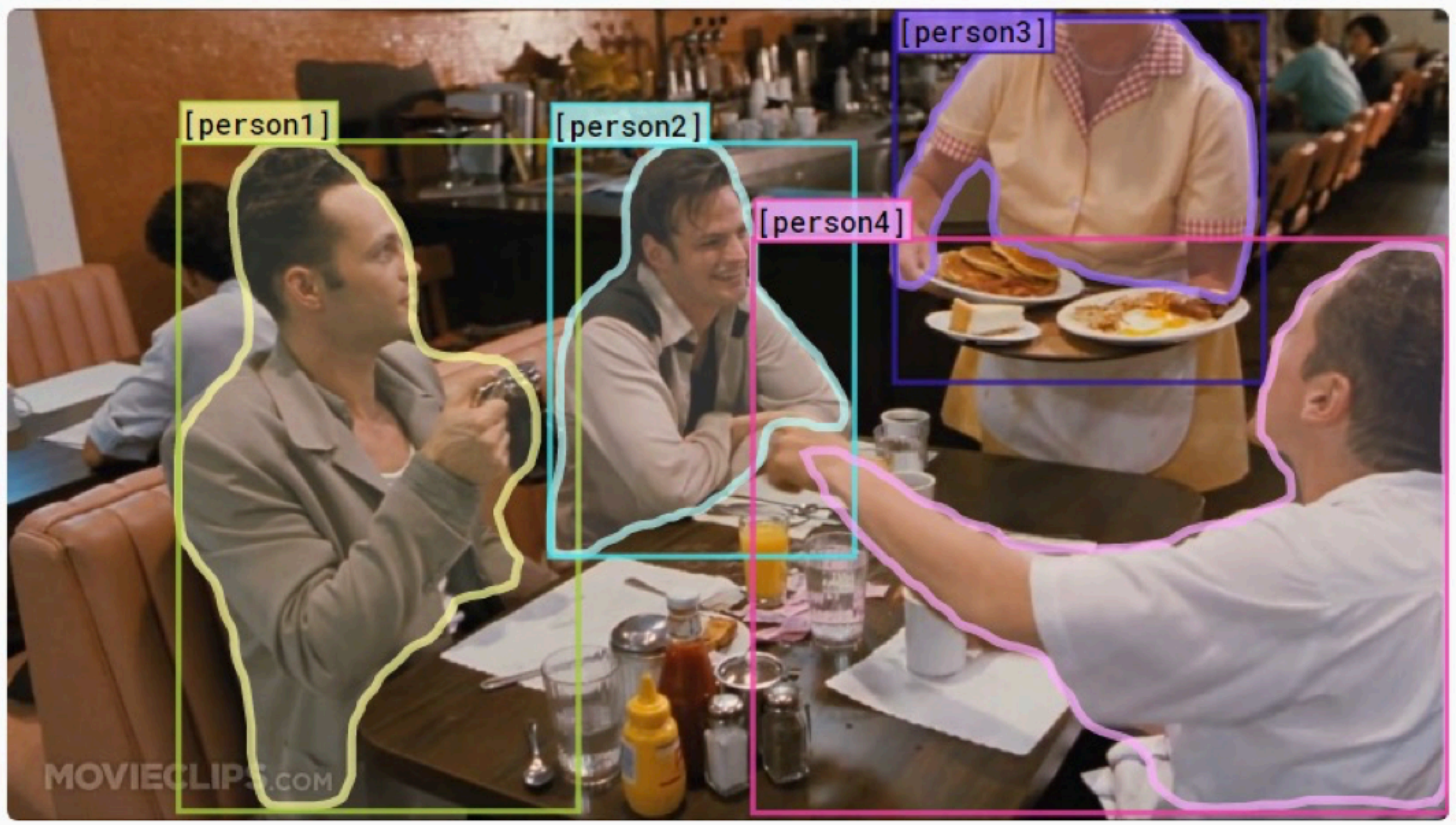
false

Downstream Task 3: Natural Language for **Visual Reasoning**



Slide from **Zhe Gan**

Downstream Task 4: **Visual Commonsense Reasoning**



Why is [person4] pointing at [person1]?

- **a) He is telling [person3] that [person1] ordered the pancakes.**
- b) He just told a joke.
- c) He is feeling accusatory towards [person1].
- d) He is giving [person1] directions.

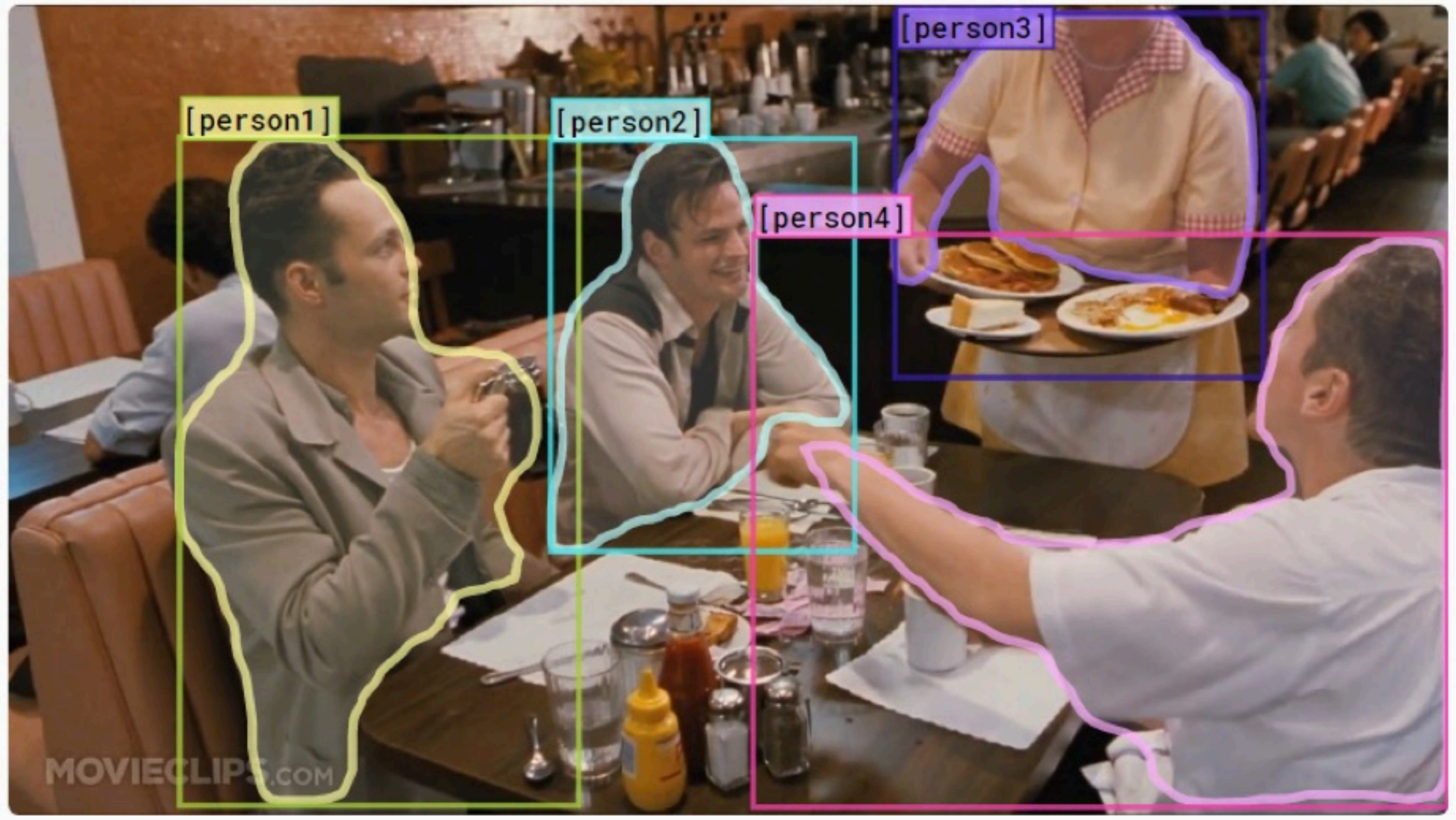
I choose (a) because:

- a) [person1] has the pancakes in front of him.
- b) [person4] is taking everyone's order and asked for clarification.
- c) [person3] is looking at the pancakes and both she and [person2] are smiling slightly.
- **d) [person3] is delivering food to the table, and she might not know whose order is whose.**

Slide from **Zhe Gan**

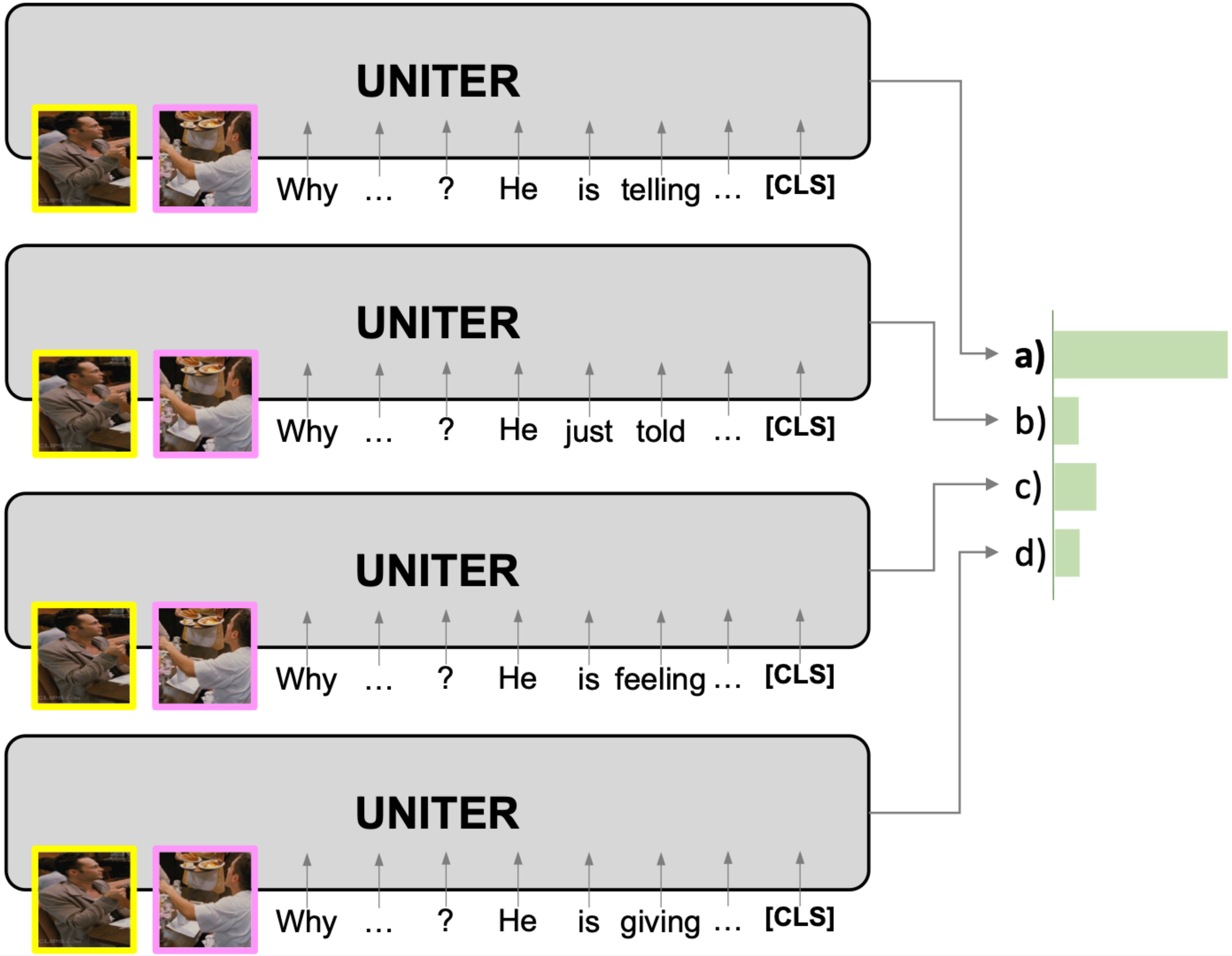
[Zellers et al., CVPR 2019]

Downstream Task 4: **Visual Commonsense Reasoning**



Why is [person4] pointing at [person1]?

a) **He is telling [person3] that [person1] ordered the pancakes.**
b) He just told a joke.
c) He is feeling accusatory towards [person1].
d) He is giving [person1] directions.



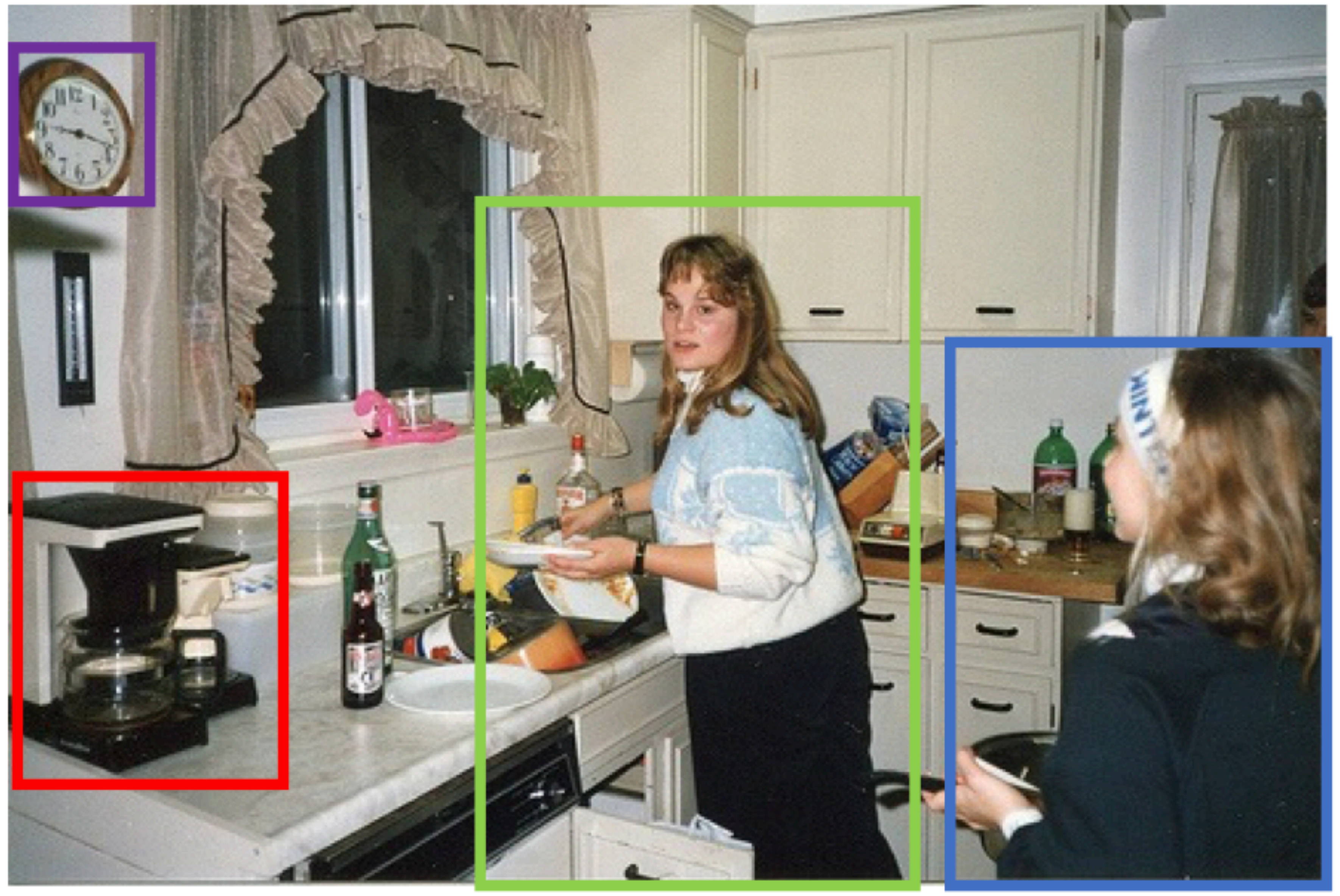
Slide from **Zhe Gan**

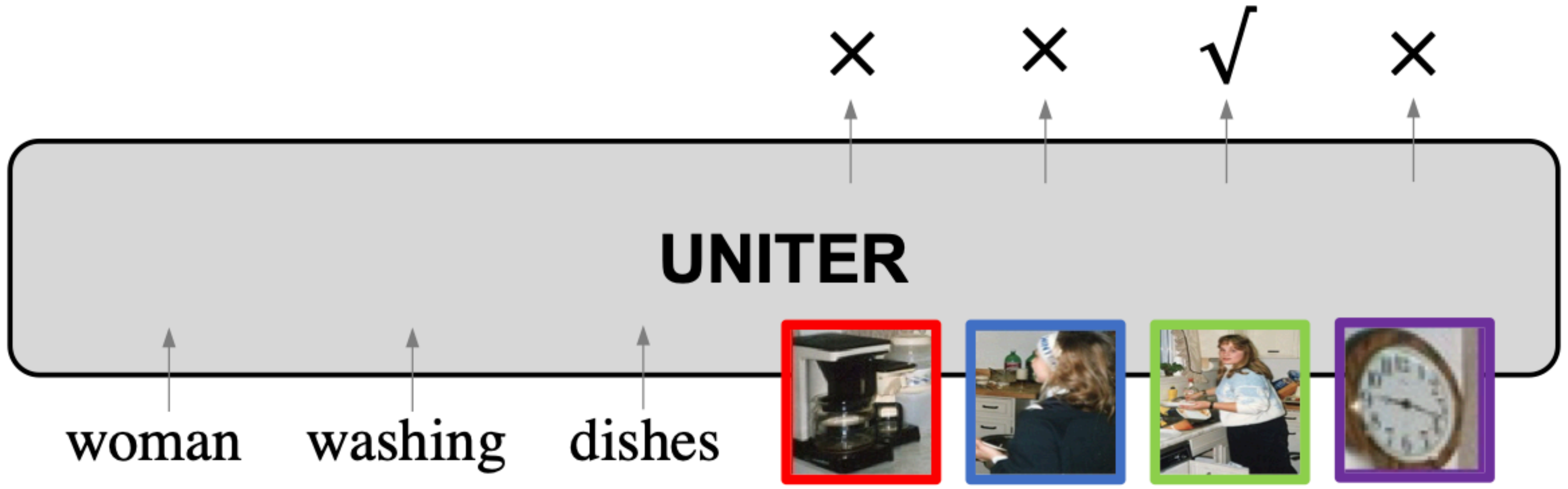
Downstream Task 5: **Referring Expression Comprehension (Grounding)**

woman washing dishes

Slide from **Zhe Gan**

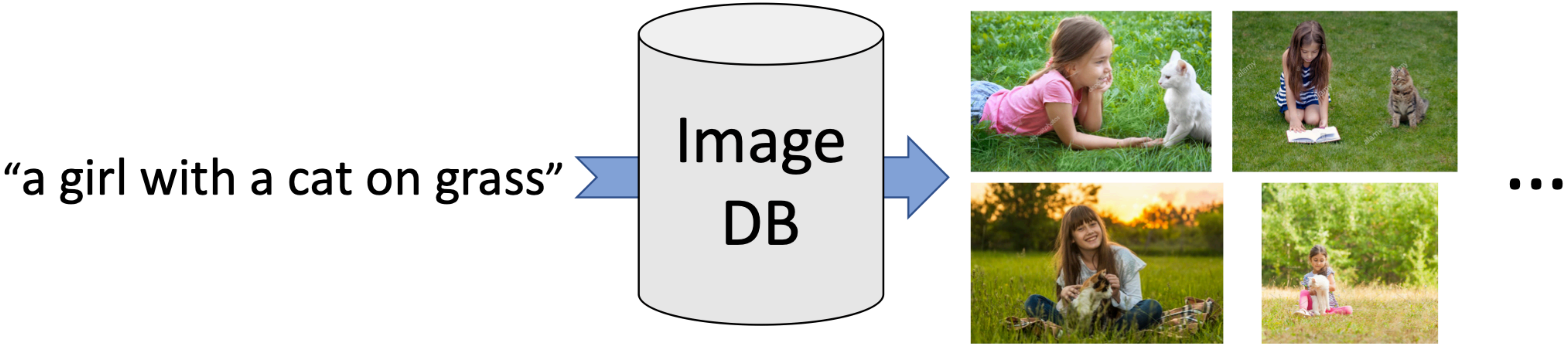
Downstream Task 5: **Referring Expression Comprehension (Grounding)**





Slide from **Zhe Gan**

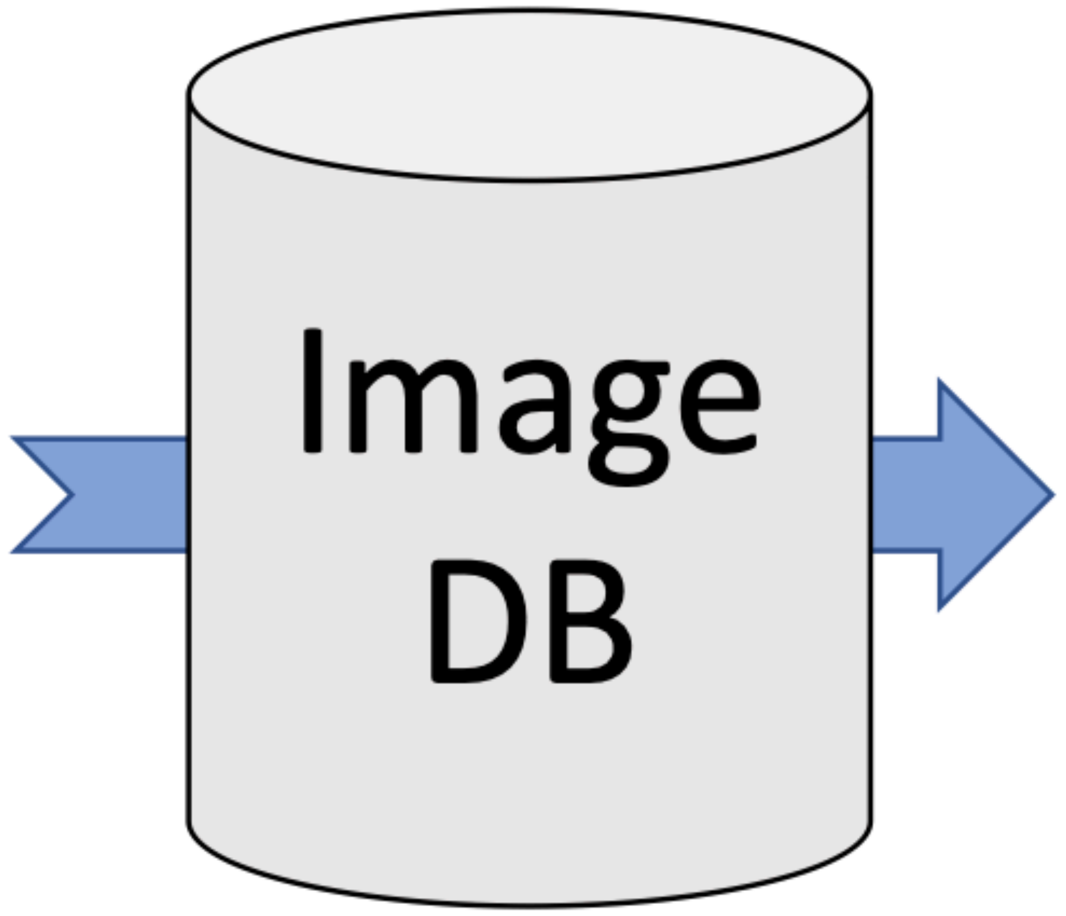
Downstream Task 6: **Image-Text Retrieval**



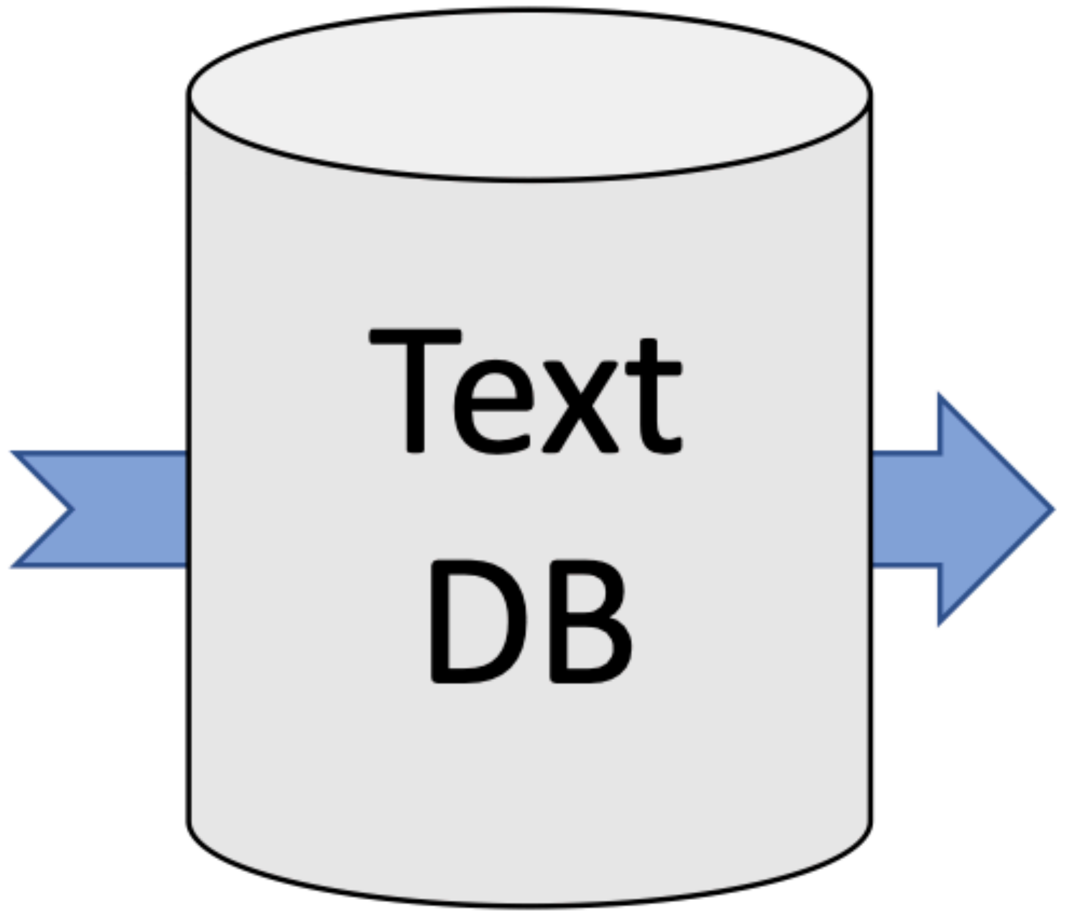
Slide from **Zhe Gan**

Downstream Task 6: **Image-Text Retrieval**

“a girl with a cat on grass”



...

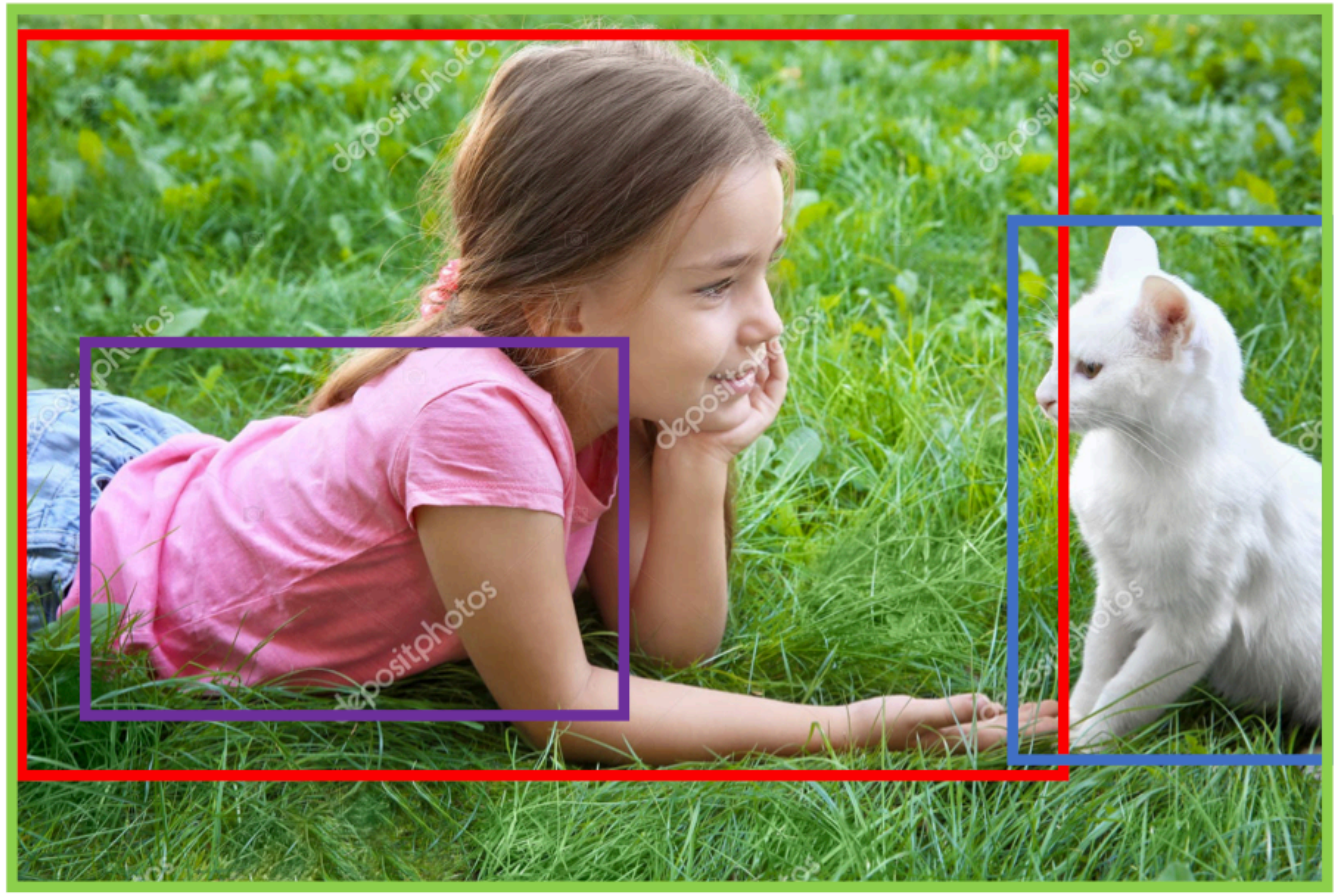


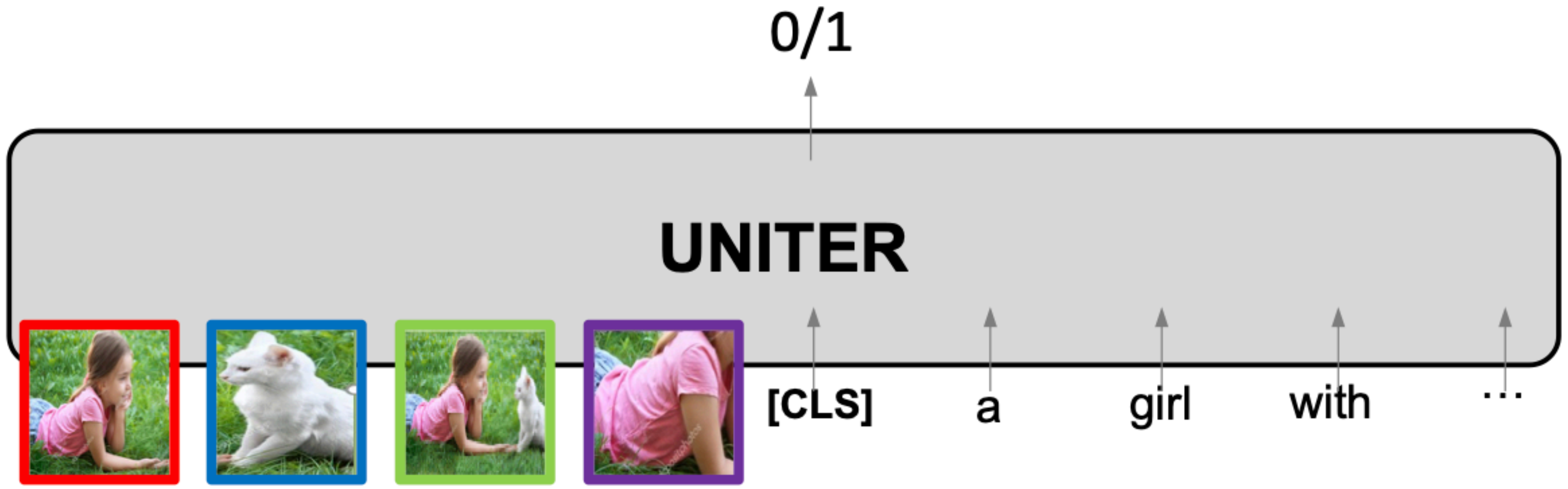
“four people with ski poles in their hands in the snow”
“four skiers hold on to their poles in a snowy forest”
“a group of young men riding skis”
“skiers pose for a picture while outside in the woods”
“a group of people cross country skiing in the woods”

⋮

Slide from **Zhe Gan**

Downstream Task 6: **Image-Text Retrieval**





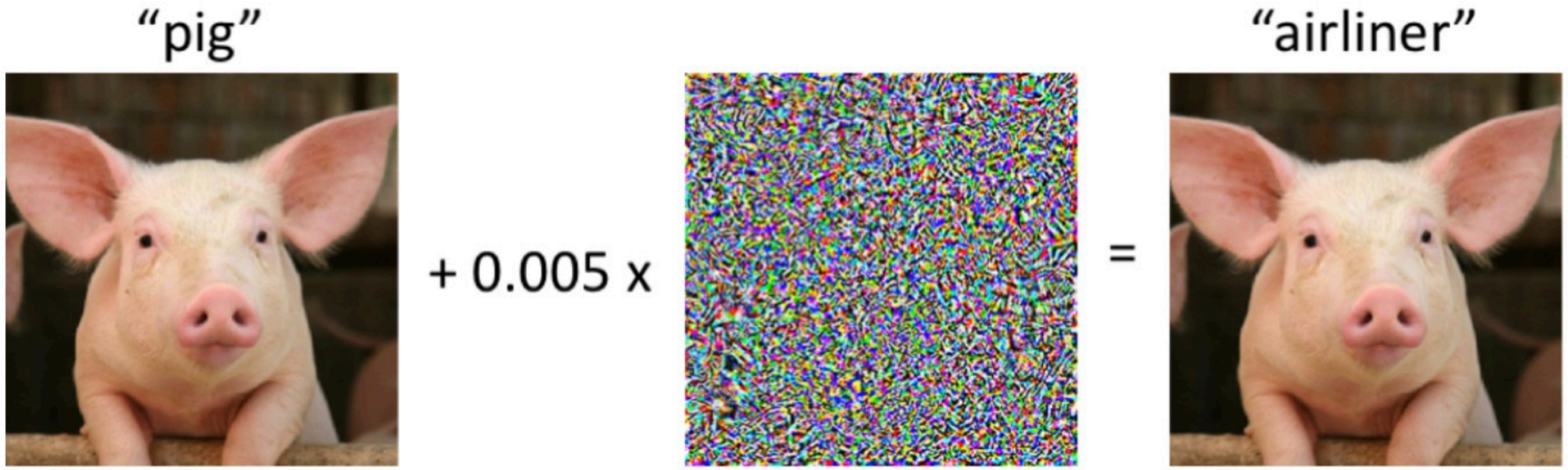
Slide from **Zhe Gan**

VILLA: Vision-and-Language Large-scale Adversarial Training

Preliminary: Adversarial Attacks

- Neural Networks are prone to label-preserving adversarial examples

Computer Vision:



Natural Language Processing:

Original: What is the oncorhynchus also called? **A:** chum salmon	**Original:** How long is the Rhine? **A:** 1,230 km
Changed: What's the oncorhynchus also called? **A:** keta	**Changed:** How long is the Rhine?? **A:** more than 1,050,000
(b) Example for (*WP is*→*WP's*)	(c) Example for (*?*→*??*)

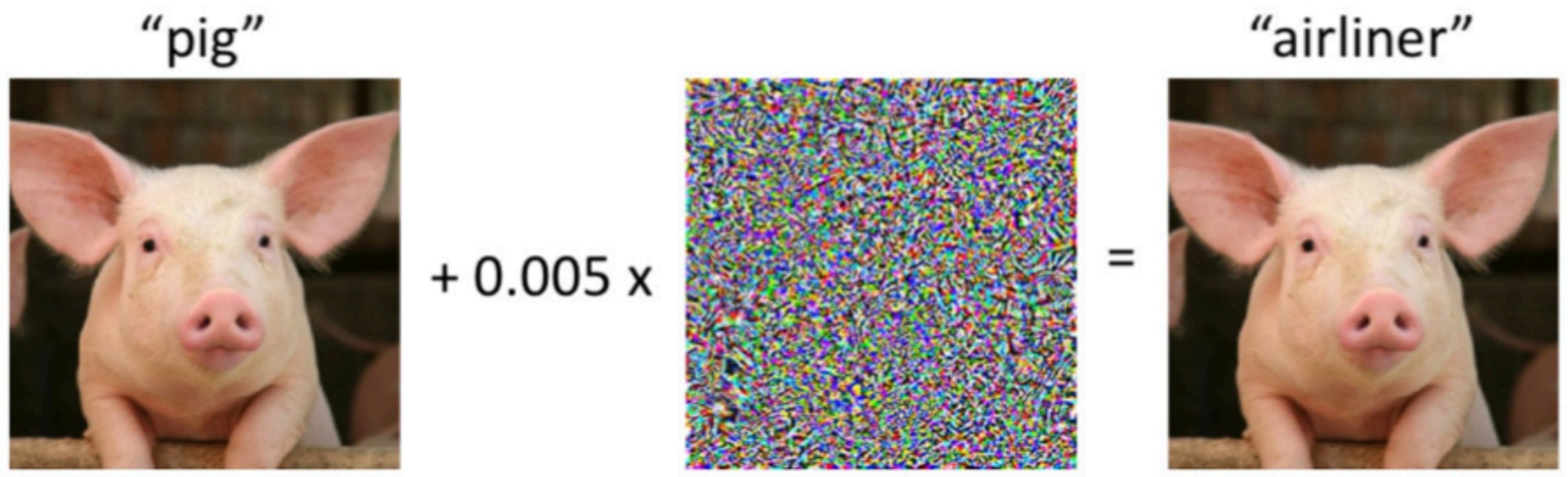
[1] Explaining and harnessing adversarial examples. *arXiv:1412.6572*
[2] Semantically equivalent adversarial rules for debugging nlp models. *ACL (2018)*

Slide from **Zhe Gan**

Preliminary: Adversarial Training

- A min-max game to harness adversarial examples

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}} \left[\max_{\delta \in S} \mathcal{L}(x+\delta, y; \theta) \right]$$



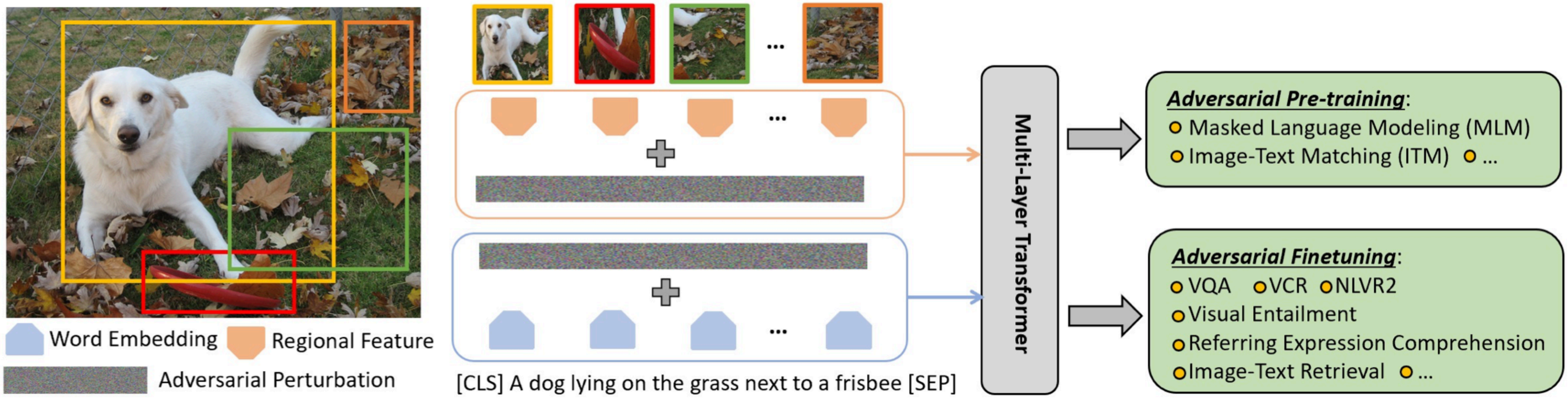
- Use adversarial examples as additional training samples
  - On one hand, we try to find perturbations that maximize the empirical risk
  - On the other hand, the model tries to make correct predictions on adversarial examples
- *What doesn't kill you makes you stronger!*

Explaining and harnessing adversarial examples. *arXiv:1412.6572*

Slide from **Zhe Gan**

VILLA: Vision-and-Language Large-scale Adversarial Training

- Ingredient #1: Adversarial pre-training + finetuning
- Ingredient #2: Perturbations in the embedding space
- Ingredient #3: Enhanced adversarial training algorithm



Slide from **Zhe Gan**

VILLA: Vision-and-Language Large-scale Adversarial Training

- Training objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt}, \boldsymbol{y}) \sim \mathcal{D}} \Big[\mathcal{L}_{std}(\boldsymbol{\theta}) + \mathcal{R}_{at}(\boldsymbol{\theta}) + \alpha \cdot \mathcal{R}_{kl}(\boldsymbol{\theta}) \Big]$$

- Cross-entropy loss on clean data:

$$\mathcal{L}_{std}(\boldsymbol{\theta}) = L(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt}), \boldsymbol{y})$$

( , A [MASK] lying on the grass next to a frisbee) ⇒

Probability vector

Ground-truth label

Slide from **Zhe Gan**

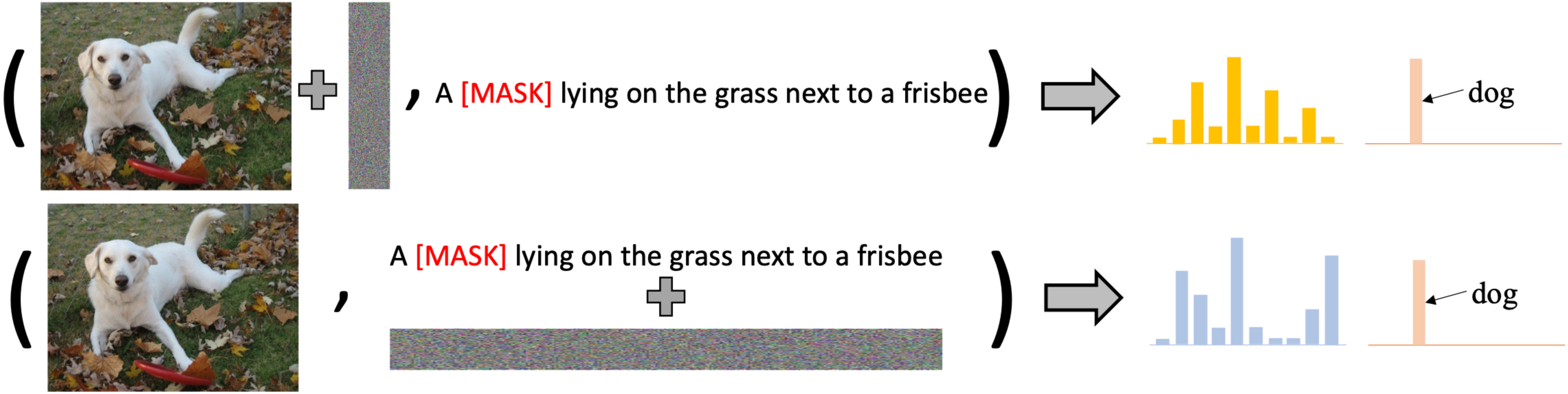
VILLA: Vision-and-Language Large-scale Adversarial Training

- Training objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt}, \boldsymbol{y})\sim \mathcal{D}} \Big[\mathcal{L}_{std}(\boldsymbol{\theta}) + \mathcal{R}_{at}(\boldsymbol{\theta}) + \alpha \cdot \mathcal{R}_{kl}(\boldsymbol{\theta})\Big]$$

- Cross-entropy loss on adversarial embeddings:

$$\mathcal{R}_{at}(\boldsymbol{\theta}) = \max_{||\boldsymbol{\delta}_{img}|| \le \epsilon} L(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{img} + \boldsymbol{\delta}_{img}, \boldsymbol{x}_{txt}), \boldsymbol{y}) + \max_{||\boldsymbol{\delta}_{txt}|| \le \epsilon} L(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt} + \boldsymbol{\delta}_{txt}), \boldsymbol{y})$$



Slide from **Zhe Gan**

VILLA: Vision-and-Language Large-scale Adversarial Training

- Training objective:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt}, \boldsymbol{y}) \sim \mathcal{D}} \Big[\mathcal{L}_{std}(\boldsymbol{\theta}) + \mathcal{R}_{at}(\boldsymbol{\theta}) + \alpha \cdot \mathcal{R}_{kl}(\boldsymbol{\theta}) \Big]$$

- KL-divergence loss for fine-grained adversarial regularization

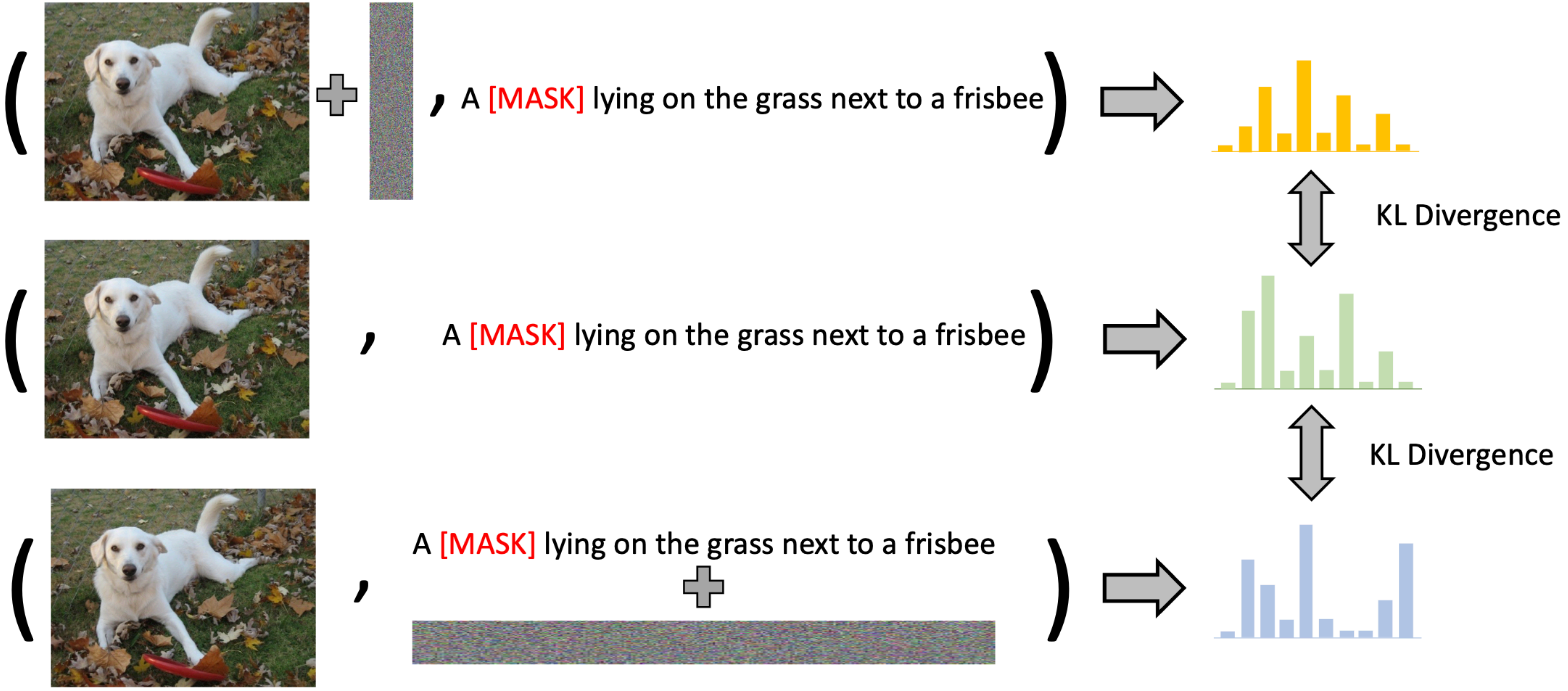
$$\begin{aligned} \mathcal{R}_{kl}(\boldsymbol{\theta}) = & \max_{||\boldsymbol{\delta}_{img}|| \leq \epsilon} L_{kl}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{img} + \boldsymbol{\delta}_{img}, \boldsymbol{x}_{txt}), f_{\boldsymbol{\theta}}(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt})) \\ & + \max_{||\boldsymbol{\delta}_{txt}|| \leq \epsilon} L_{kl}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt} + \boldsymbol{\delta}_{txt}), f_{\boldsymbol{\theta}}(\boldsymbol{x}_{img}, \boldsymbol{x}_{txt})) \, , \end{aligned}$$

where $L_{kl}(p, q) = \mathrm{KL}(p||q) + \mathrm{KL}(q||p)$.

- Not only label-preserving, but the confidence level of the prediction between clean data and adversarial examples should also be close

Slide from **Zhe Gan**

VILLA: Vision-and-Language Large-scale Adversarial Training



Slide from **Zhe Gan**

VILLA: Vision-and-Language Large-scale Adversarial Training

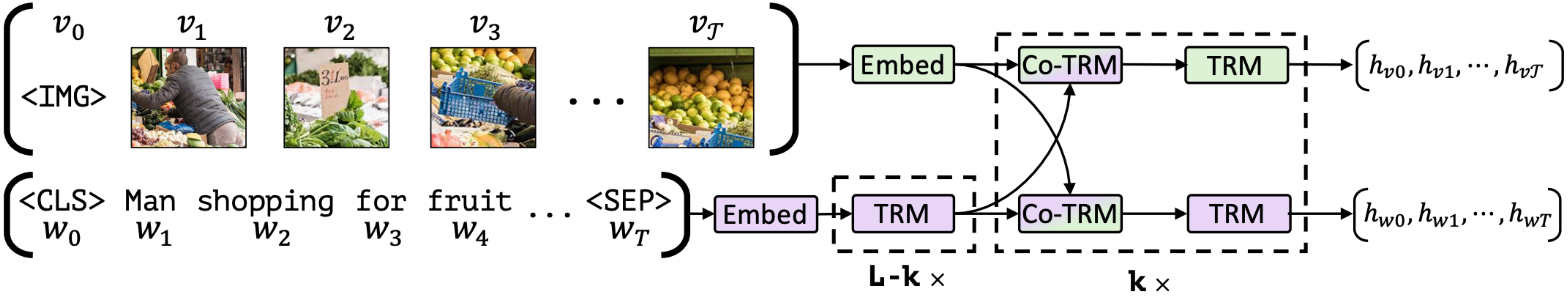
- Established new state of the art on all the tasks considered
- Gain: +0.85 on VQA, +2.9 on VCR, +1.49 on NLVR2, +0.64 on SNLI-VE

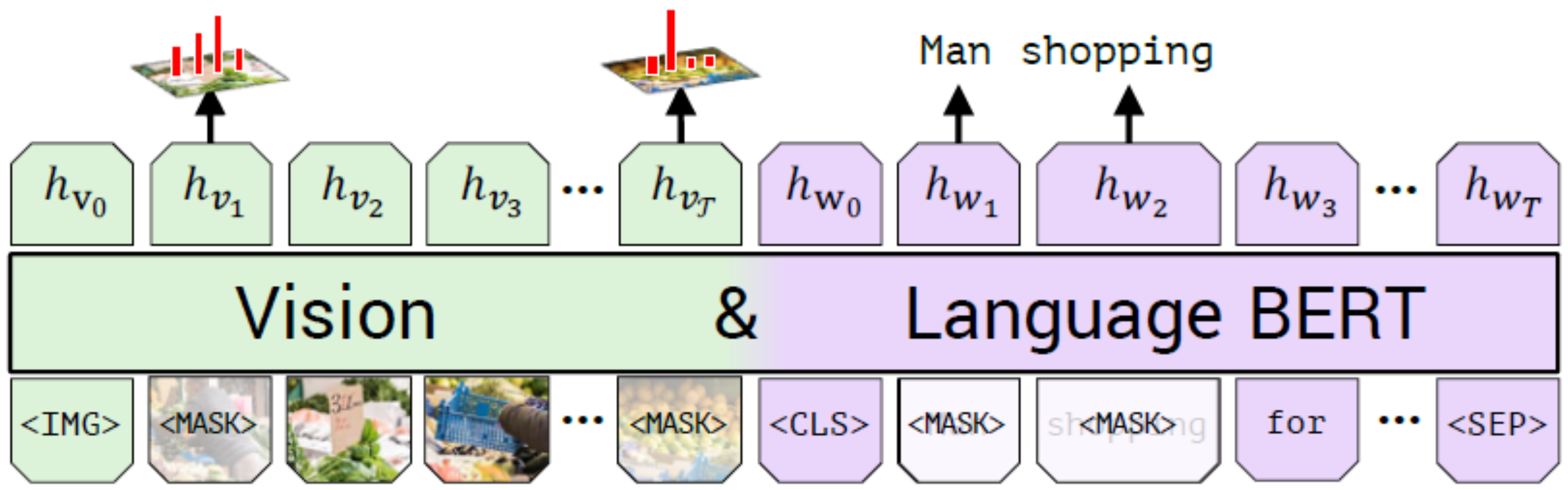
Method	VQA		VCR			NLVR^2		SNLI-VE	
	test-dev	test-std	Q→A	QA→R	Q→AR	dev	test-P	val	test
ViLBERT	70.55	70.92	72.42 (73.3)	74.47 (74.6)	54.04 (54.8)	-	-	-	-
VisualBERT	70.80	71.00	70.8 (71.6)	73.2 (73.2)	52.2 (52.4)	67.4	67.0	-	-
LXMERT	72.42	72.54	-	-	-	74.90	74.50	-	-
Unicoder-VL	-	-	72.6 (73.4)	74.5 (74.4)	54.4 (54.9)	-	-	-	-
12-in-1	73.15	-	-	-	-	-	78.87	-	76.95
VL-BERT$_{\text{BASE}}$	71.16	-	73.8 (-)	74.4 (-)	55.2 (-)	-	-	-	-
Oscar$_{\text{BASE}}$	73.16	73.44	-	-	-	78.07	78.36	-	-
UNITER$_{\text{BASE}}$	72.70	72.91	74.56 (75.0)	77.03 (77.2)	57.76 (58.2)	77.18	77.85	78.59	78.28
VILLA$_{\text{BASE}}$	**73.59**	**73.67**	**75.54 (76.4)**	**78.78 (79.1)**	**59.75 (60.6)**	**78.39**	**79.30**	**79.47**	**79.03**
VL-BERT$_{\text{LARGE}}$	71.79	72.22	75.5 (75.8)	77.9 (78.4)	58.9 (59.7)	-	-	-	-
Oscar$_{\text{LARGE}}$	73.61	73.82	-	-	-	79.12	80.37	-	-
UNITER$_{\text{LARGE}}$	73.82	74.02	77.22 (77.3)	80.49 (80.8)	62.59 (62.8)	79.12	79.98	79.39	79.38
VILLA$_{\text{LARGE}}$	**74.69**	**74.87**	**78.45 (78.9)**	**82.57 (82.8)**	**65.18 (65.7)**	**79.76**	**81.47**	**80.18**	**80.02**

(a) Results on VQA, VCR, NLVR^2, and SNLI-VE.

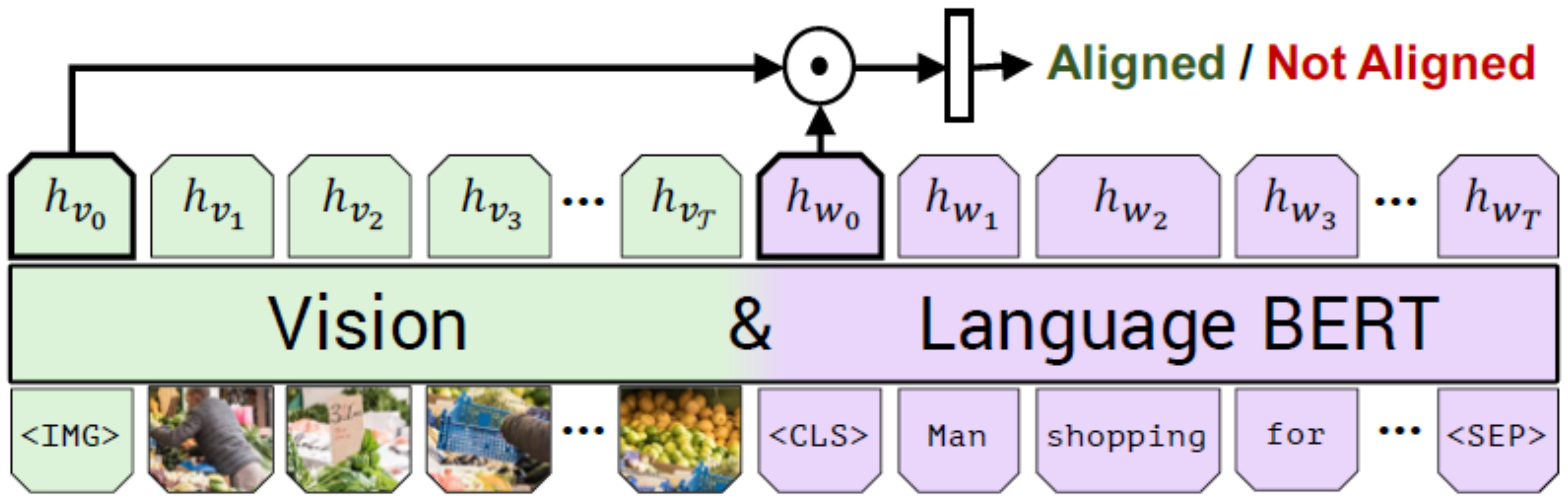
Slide from **Zhe Gan**

Visual BERT (ViLBERT)





(a) Masked multi-modal learning



(b) Multi-modal alignment prediction

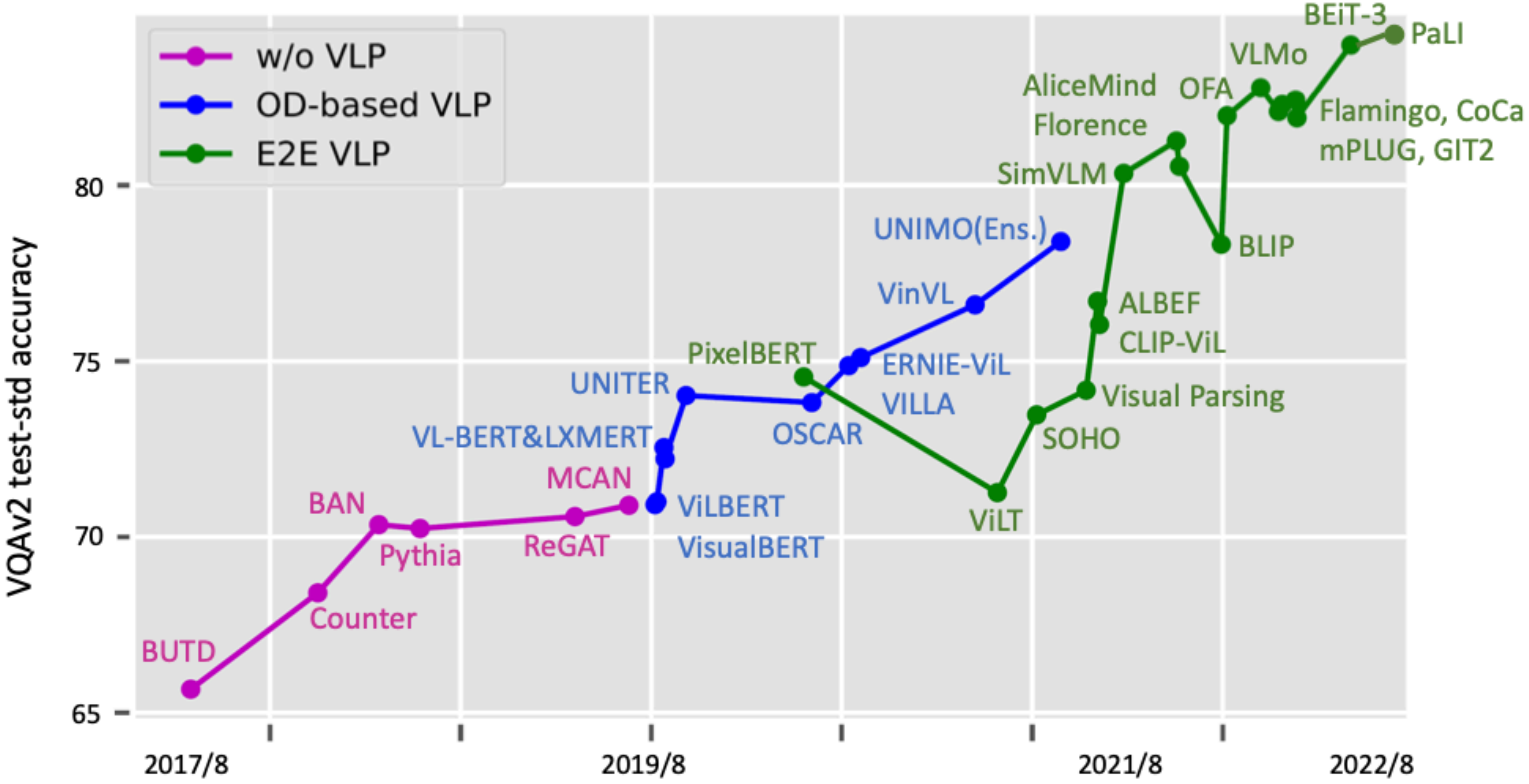
[Lu et al., 2019]

12-in-1: Multi-task Vision and Language Representation

		Clean	Vocab-based VQA (G1)			Image Retrieval (G2)		Referring Expression (G3)					Verification (G4)		# params (# models)	All Tasks Average
			VQAv2	GQA	VG QA	COCO	Flickr30k	COCO	COCO+	COCOg	V7W	GW	$NLVR^2$	SNLI-VE		
			test-dev	test-dev	val	test(R1)	test(R1)	test	test	test	test	test	testP	test		
1	Single-Task (ST)		71.82	58.19	34.38	65.28	61.14	78.63	71.11	72.24	80.51	62.81	74.25	76.72	3B (12)	67.25
2	Single-Task (ST)	✓	71.24	59.09	34.10	64.80	61.46	78.17	69.47	72.21	80.51	62.53	74.25	76.53	3B (12)	67.03
3	Group-Tasks (GT)	✓	72.03	59.60	36.18	65.06	66.00	80.23	72.79	75.30	81.54	64.78	74.62	76.52	1B (4)	68.72
4	All-Tasks (AT)	✓	72.57	60.12	36.36	63.70	63.52	80.58	73.25	75.96	82.75	65.04	78.44	76.78	**270M (1)**	69.08
5	$\text{All-Tasks}_{\text{w/o G4}}$	✓	72.68	62.09	36.74	64.88	64.62	80.76	73.60	75.80	83.03	65.41	-	-	266M (1)	-
6	$\text{GT} \xrightarrow{\text{finetune}} \text{ST}$	✓	72.61	59.96	35.81	66.26	66.98	79.94	72.12	75.18	81.57	64.56	74.47	76.34	3B (12)	68.81
7	$\text{AT} \xrightarrow{\text{finetune}} \text{ST}$	✓	72.92	60.48	36.56	65.46	65.14	80.86	73.45	76.00	83.01	65.15	**78.87**	76.73	3B (12)	69.55
8	$\text{AT} \xrightarrow{\text{finetune}} \text{ST}$		**73.15**	**60.65**	**36.64**	**68.00**	**67.90**	**81.20**	**74.22**	**76.35**	**83.35**	**65.69**	**78.87**	**76.95**	3B (12)	**70.24**

[Lu et al., 2020]

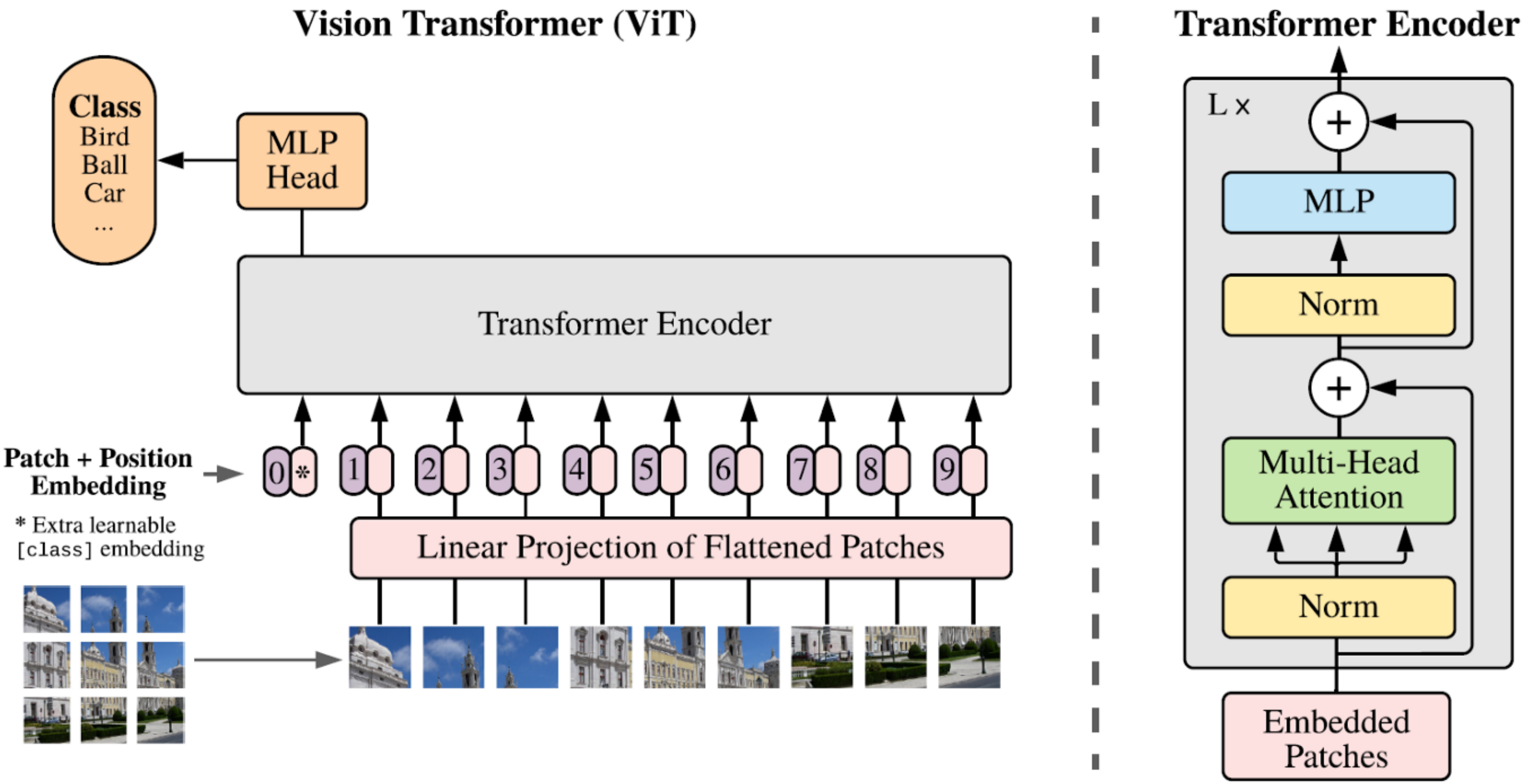
Recent History of **Visio-Lingual Models**



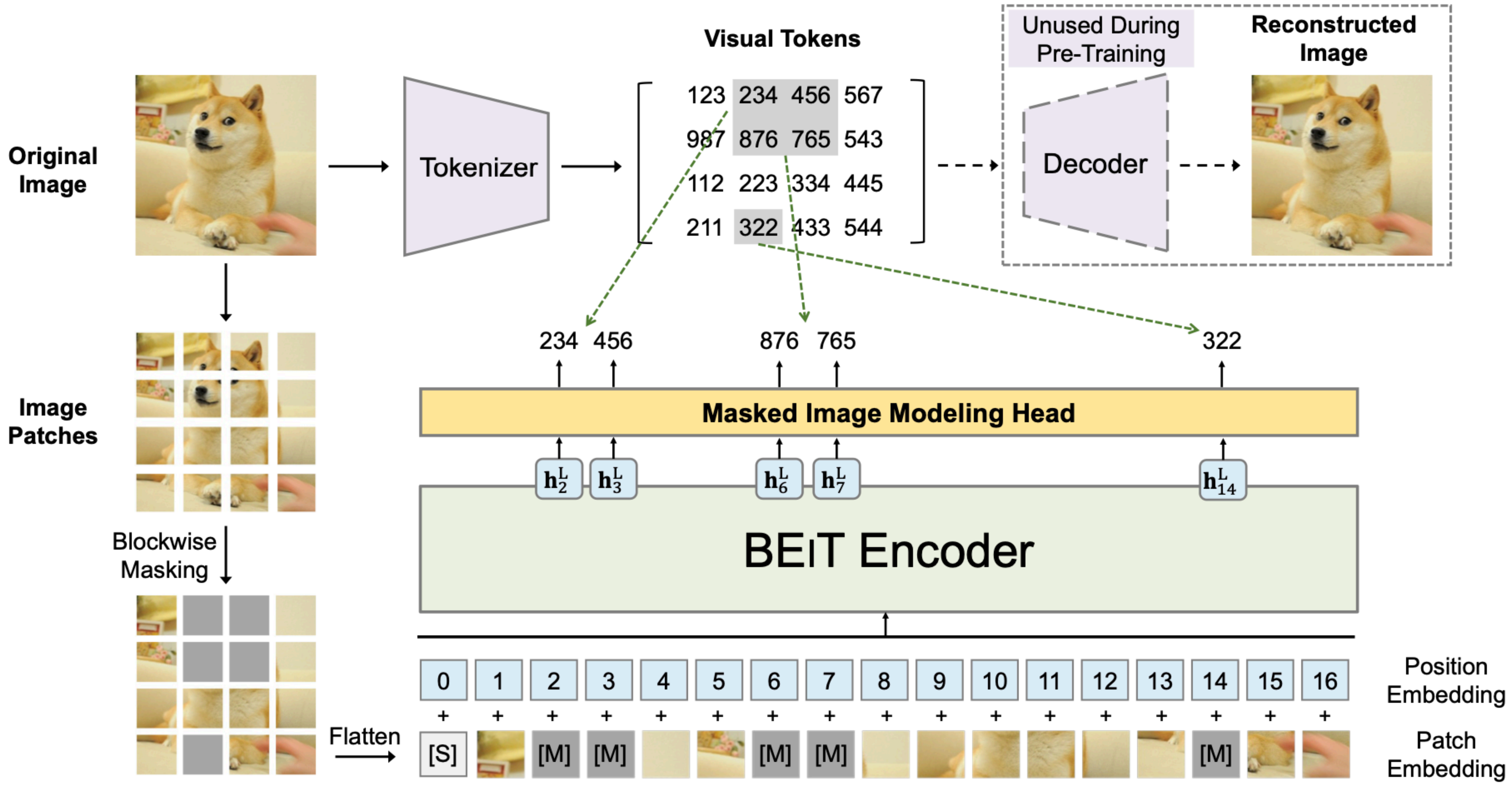
Slide from **Zhe Gan**

Vision Transformer

[Dosovitskiy et al., 2020]

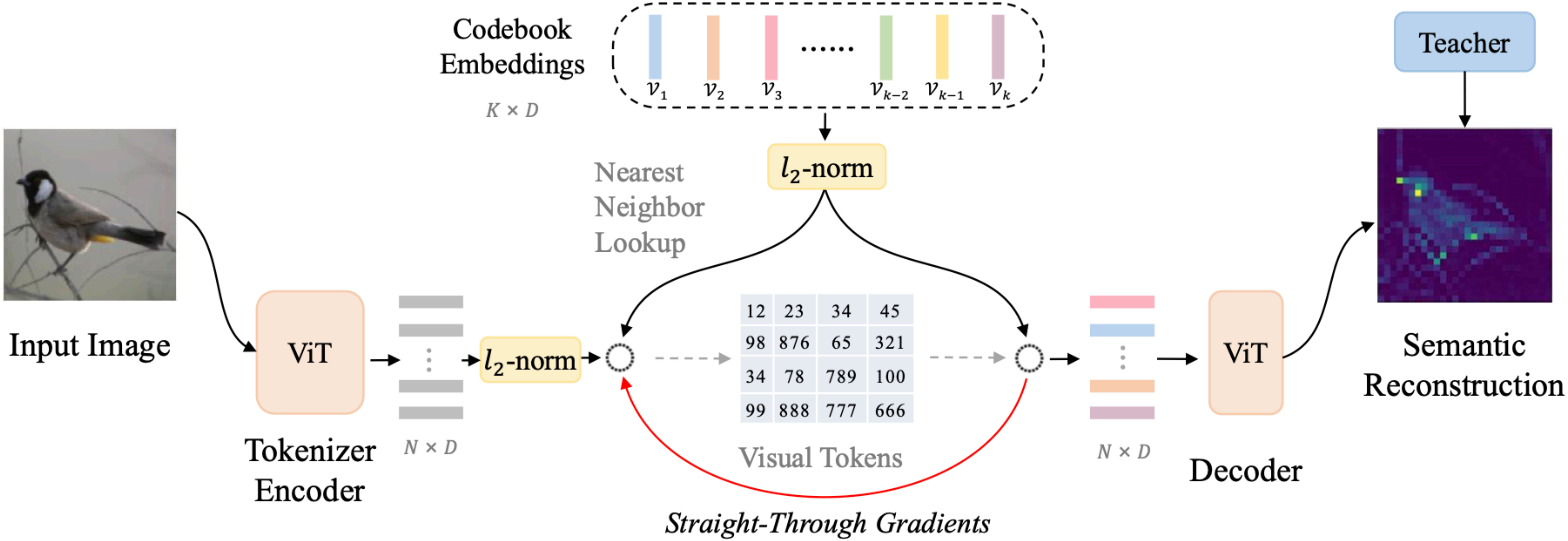


BEiT: BERT Pre-Training of Image Transformers



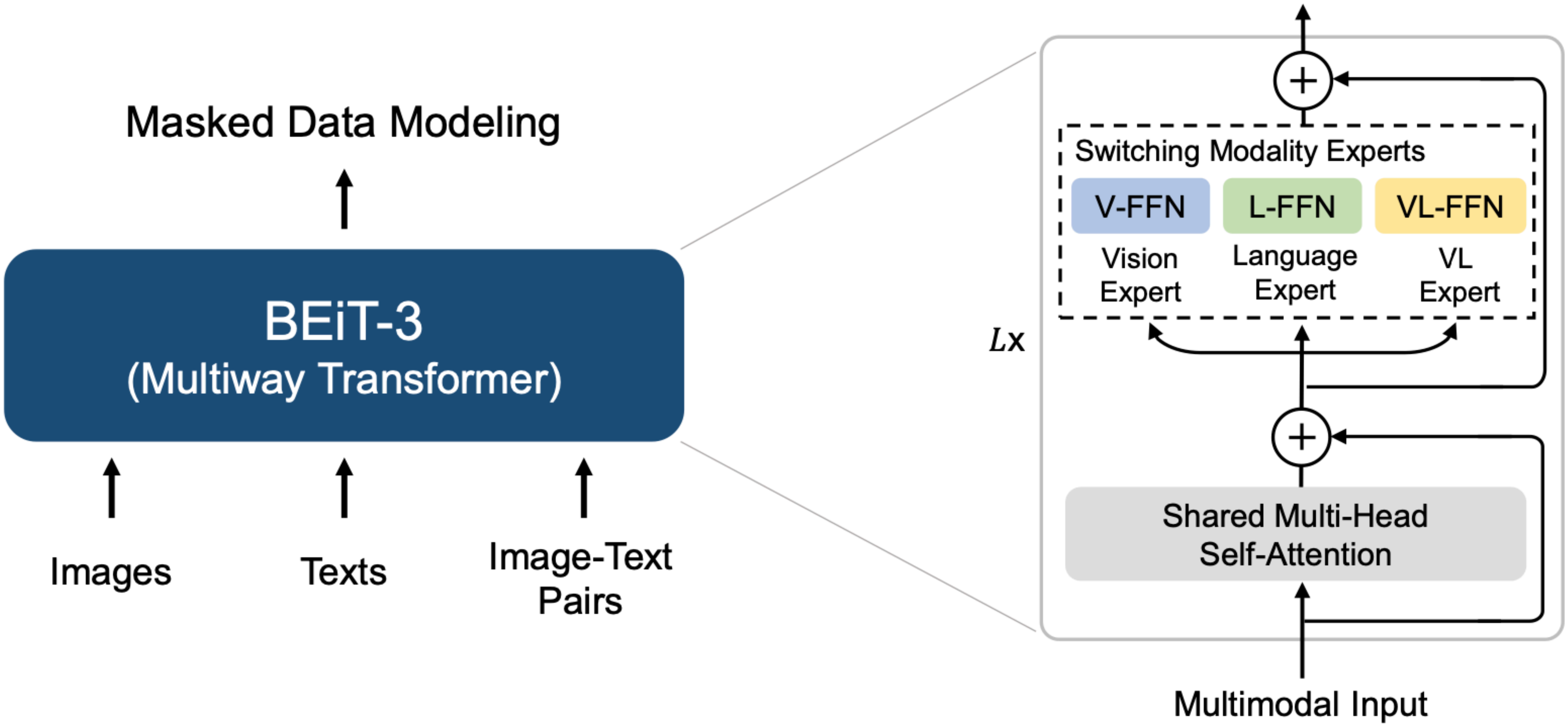
[Bao et al., 2022]

BEiT-v2



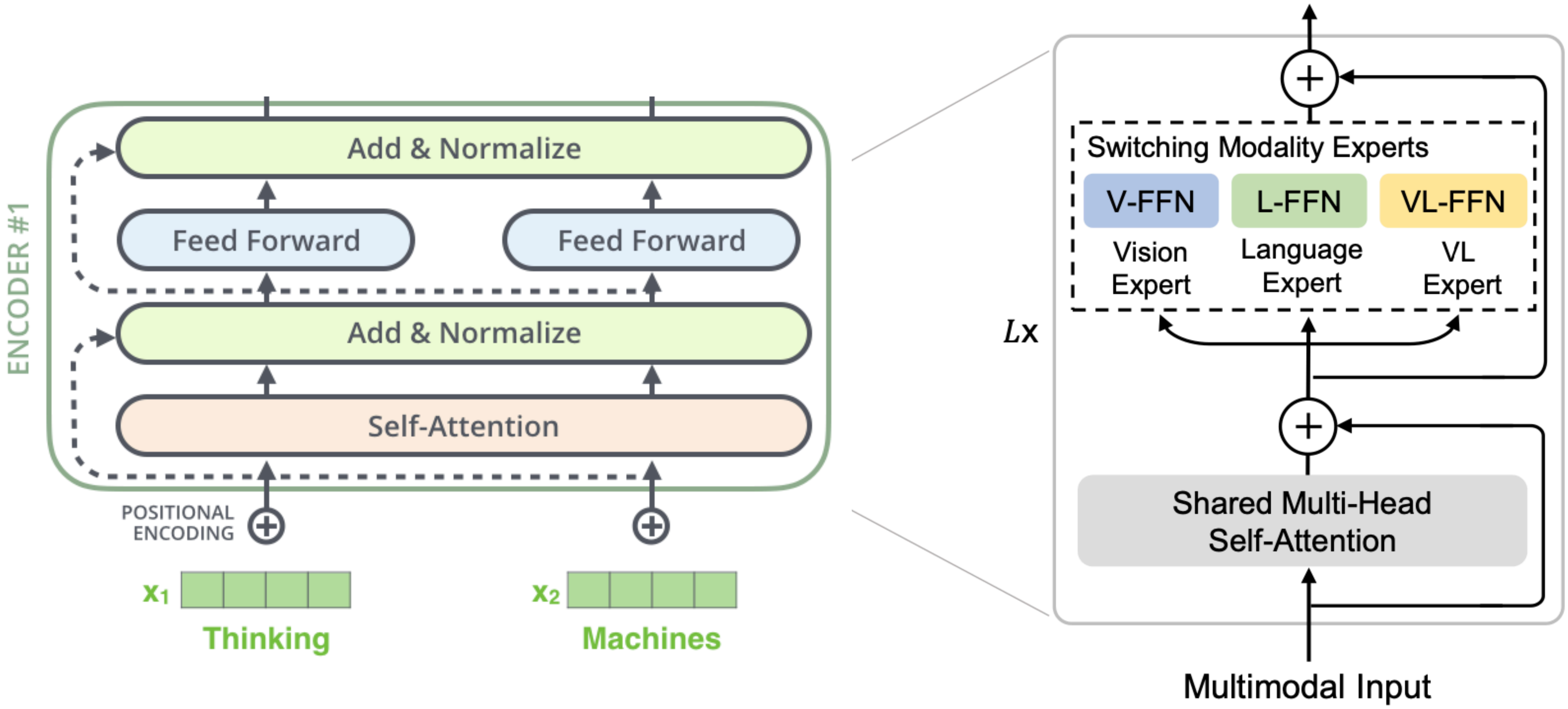
[Peng et al., 2022]

BEiT-v3: Image as a Foreign Language



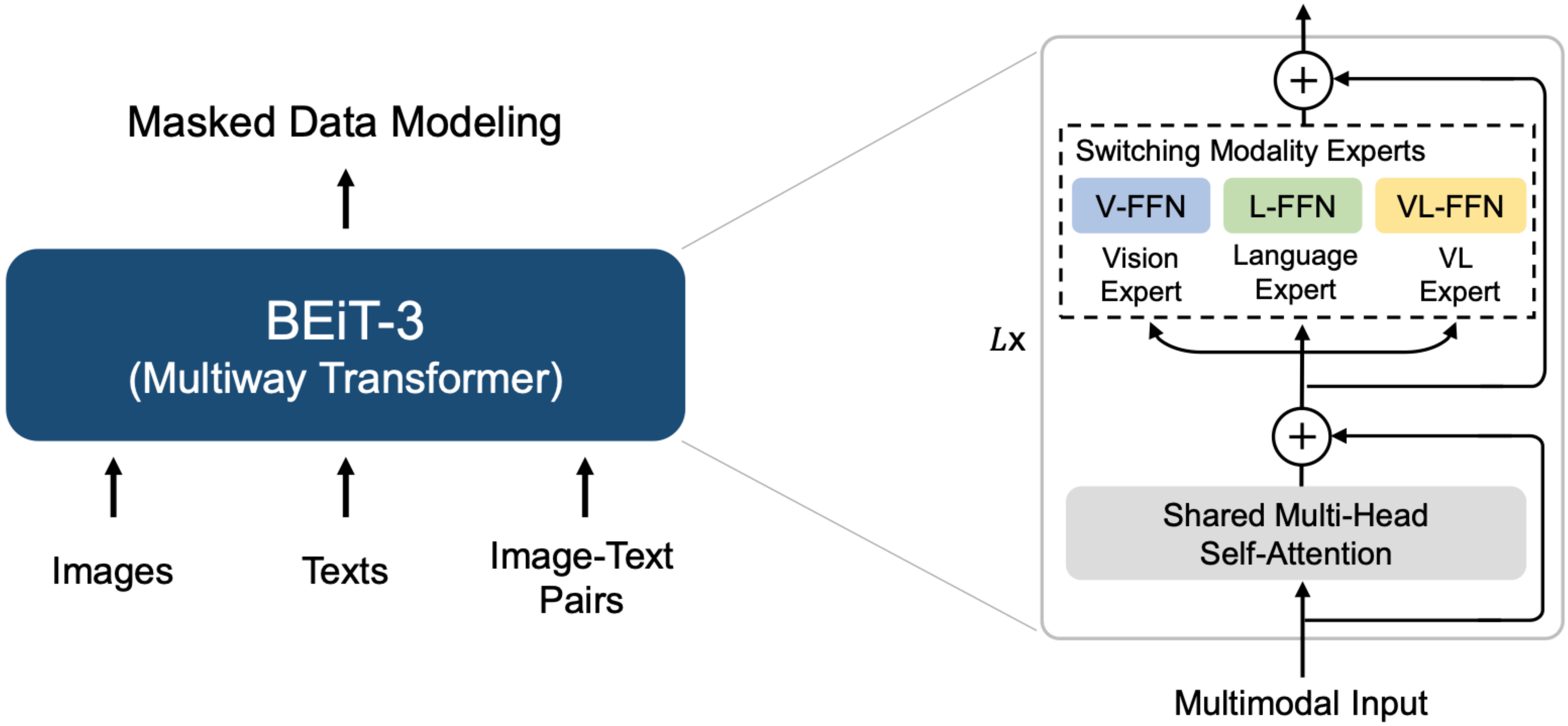
[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language



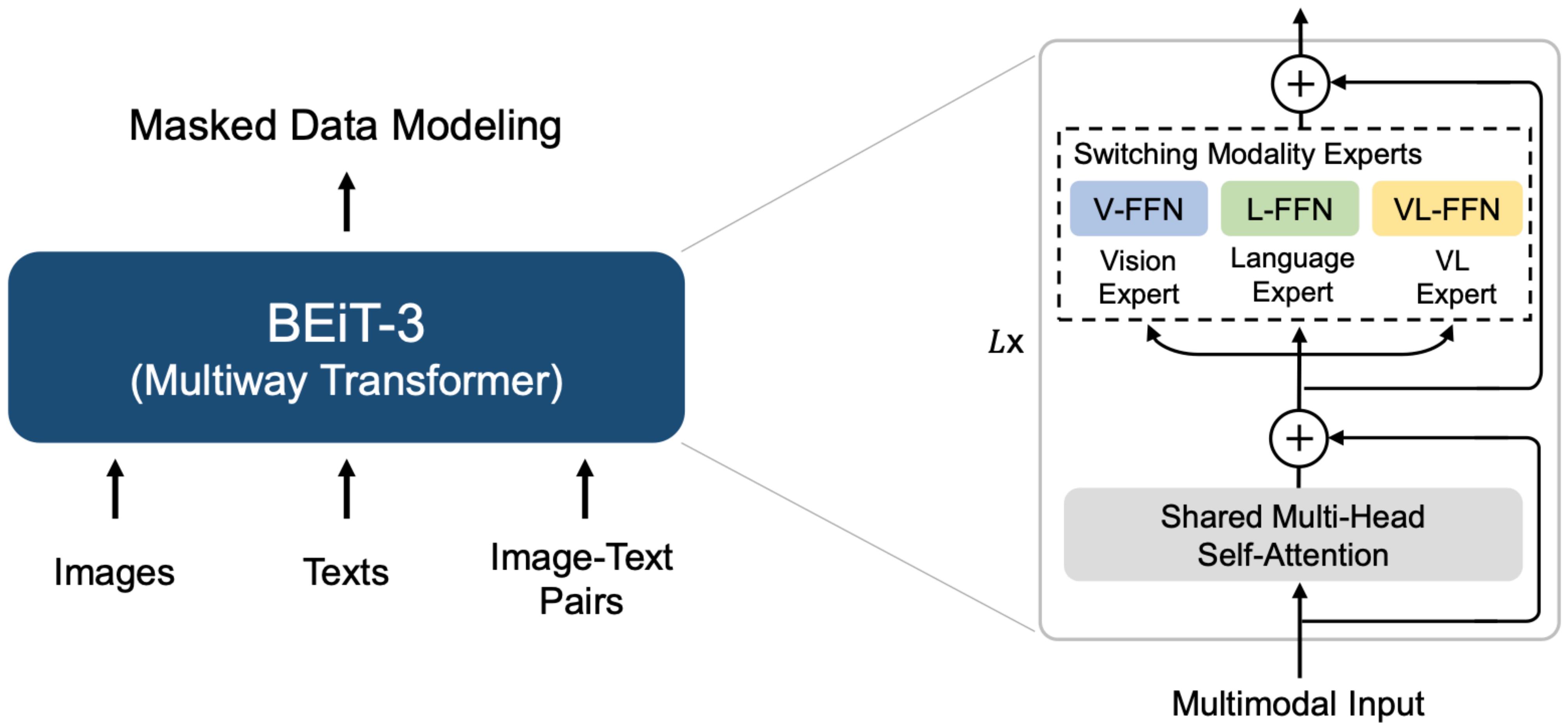
[Wang et al., 2022]

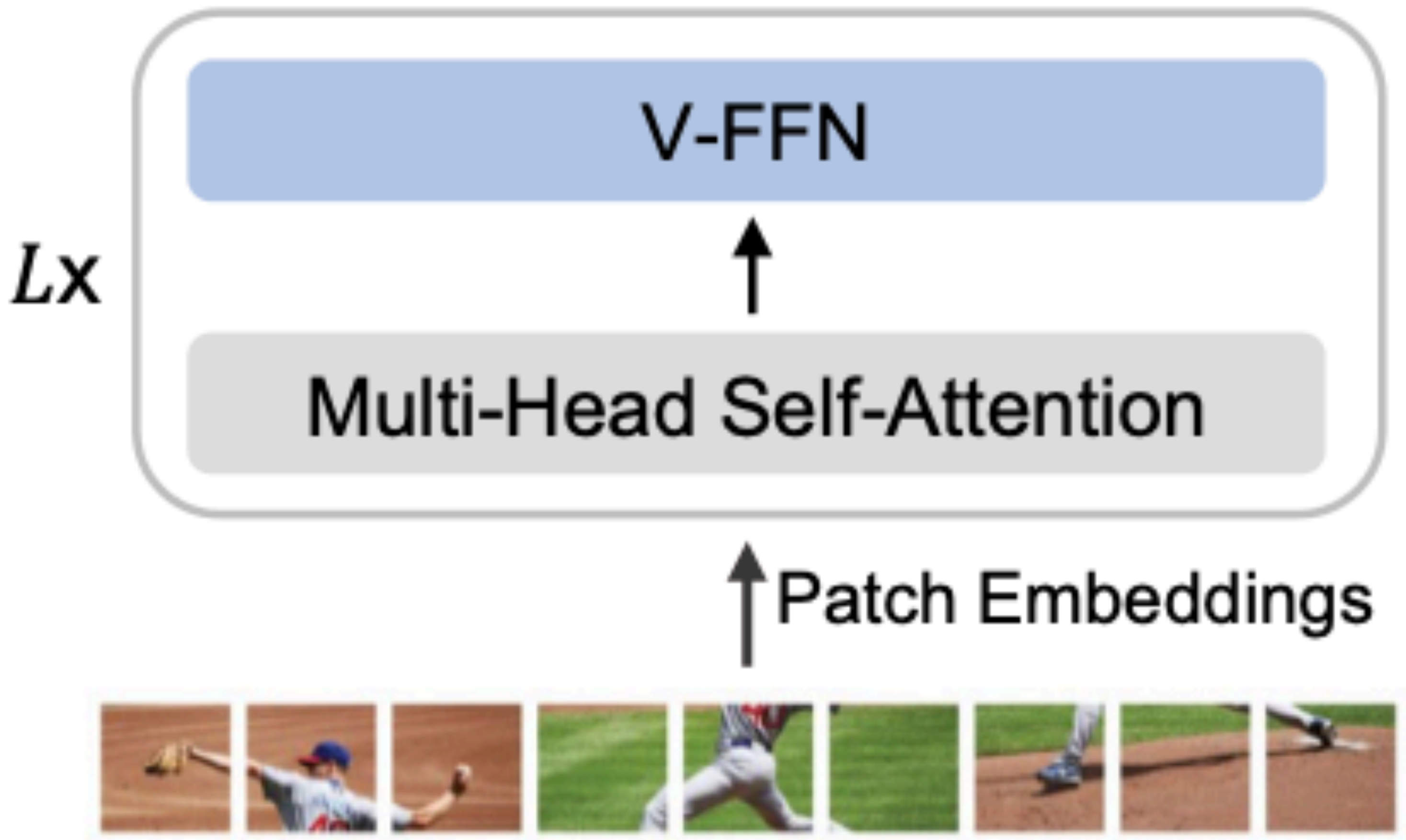
BEiT-v3: Image as a Foreign Language



[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language





(a) Vision Encoder

Masked Image Modeling

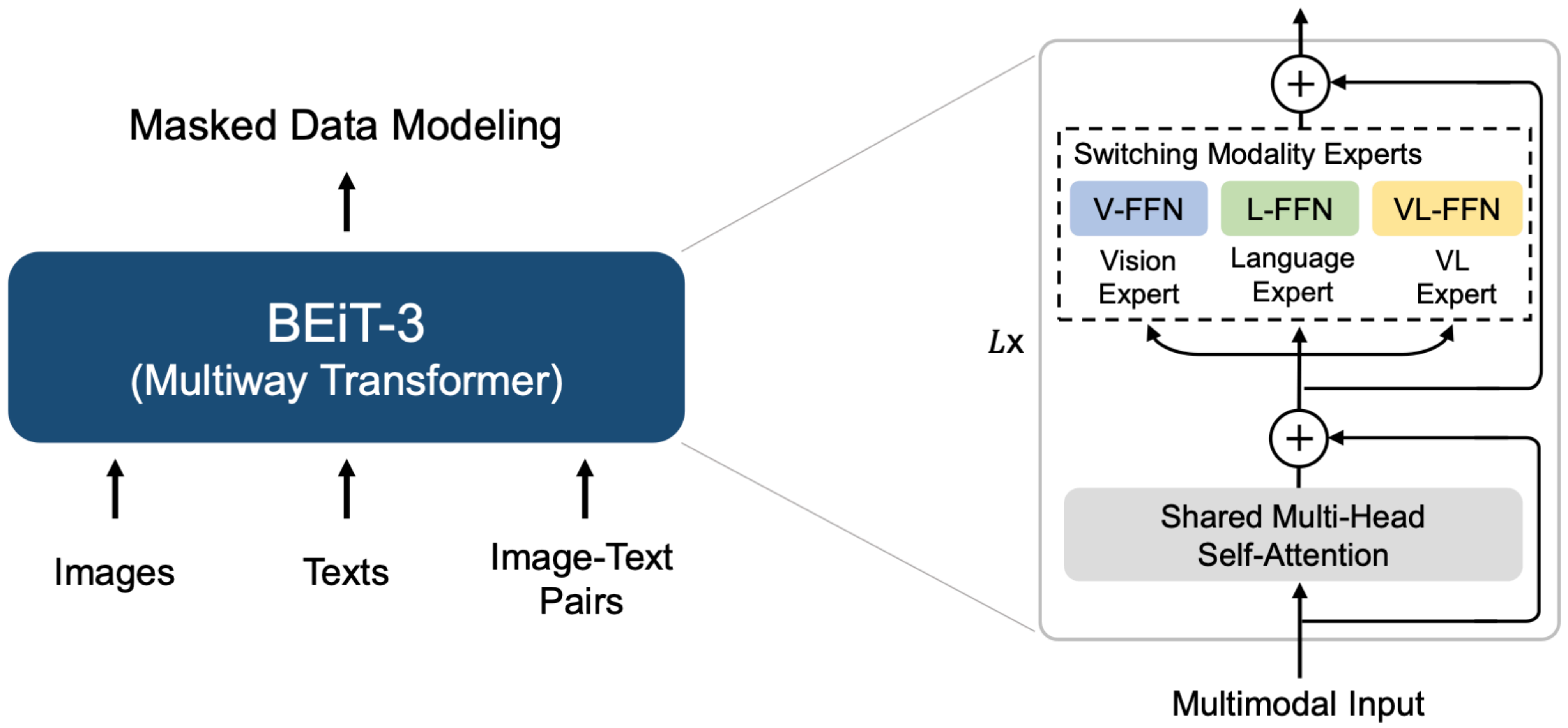
Image Classification (IN1K)

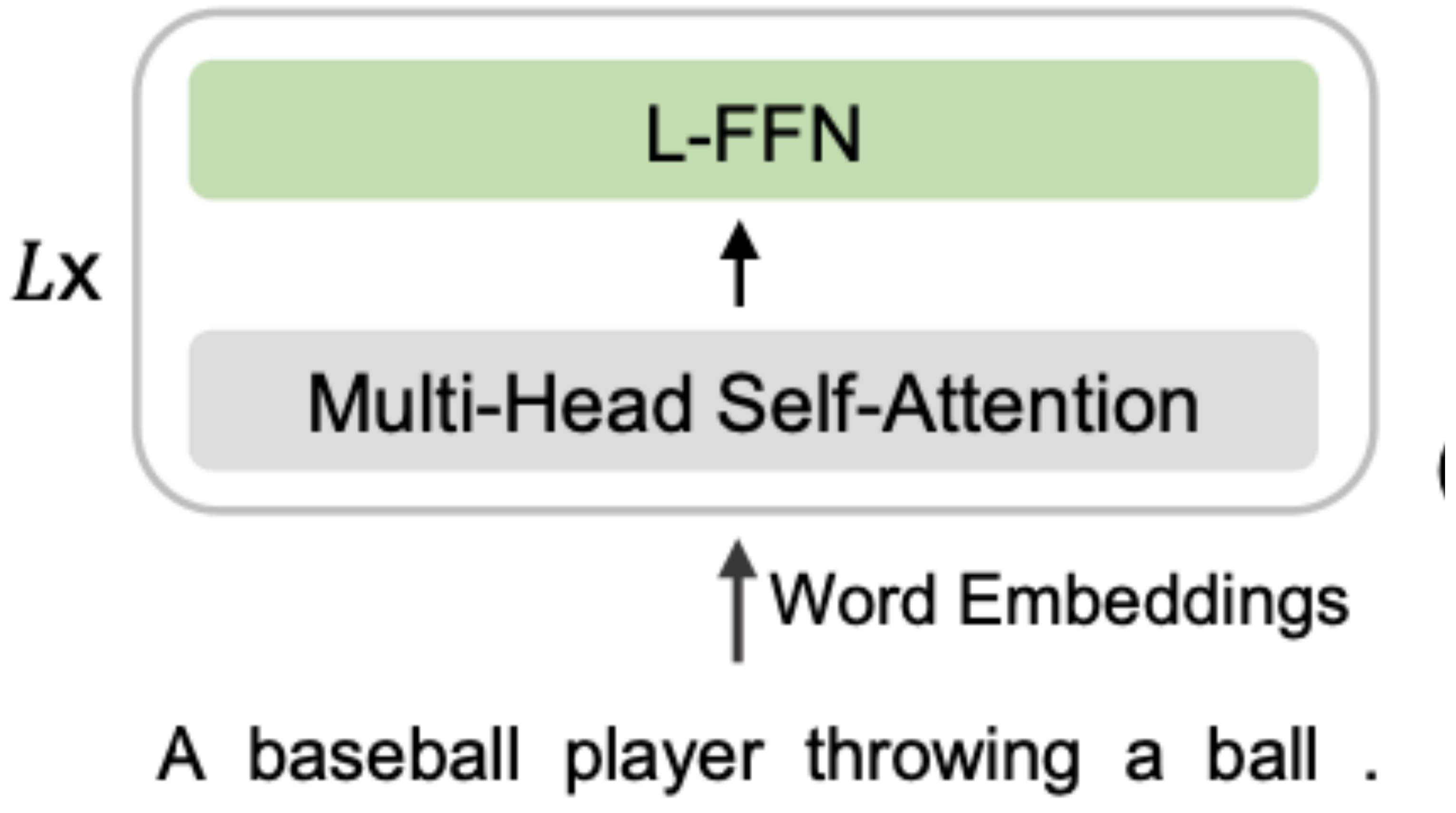
Semantic Segmentation (ADE20K)

Object Detection (COCO)

[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language



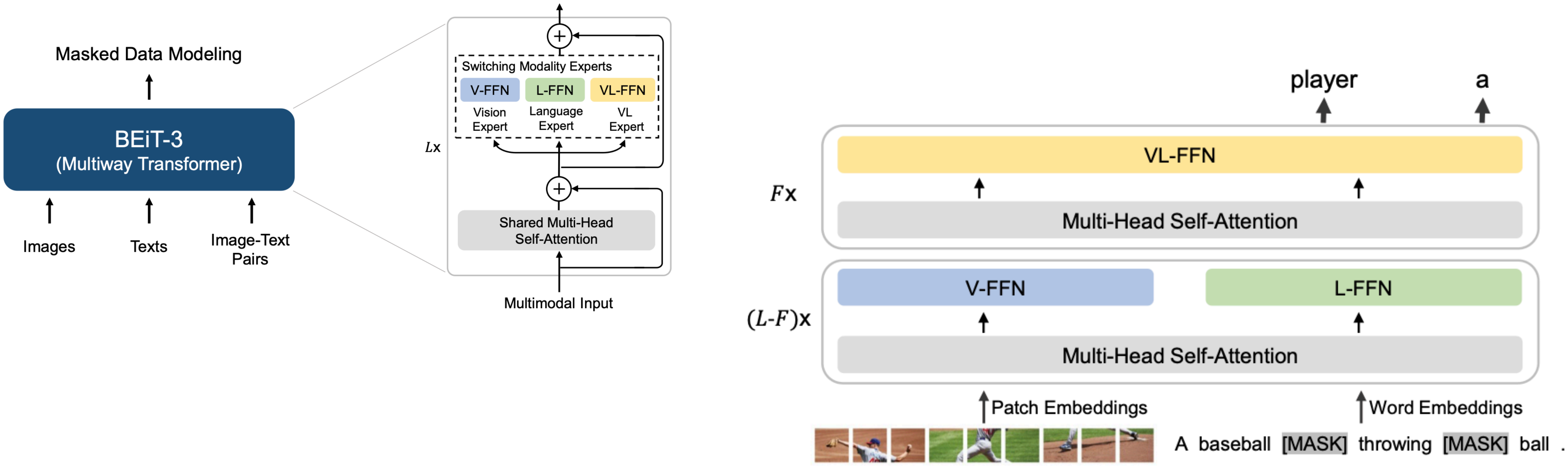


(b) Language Encoder

Masked Language Modeling

[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language



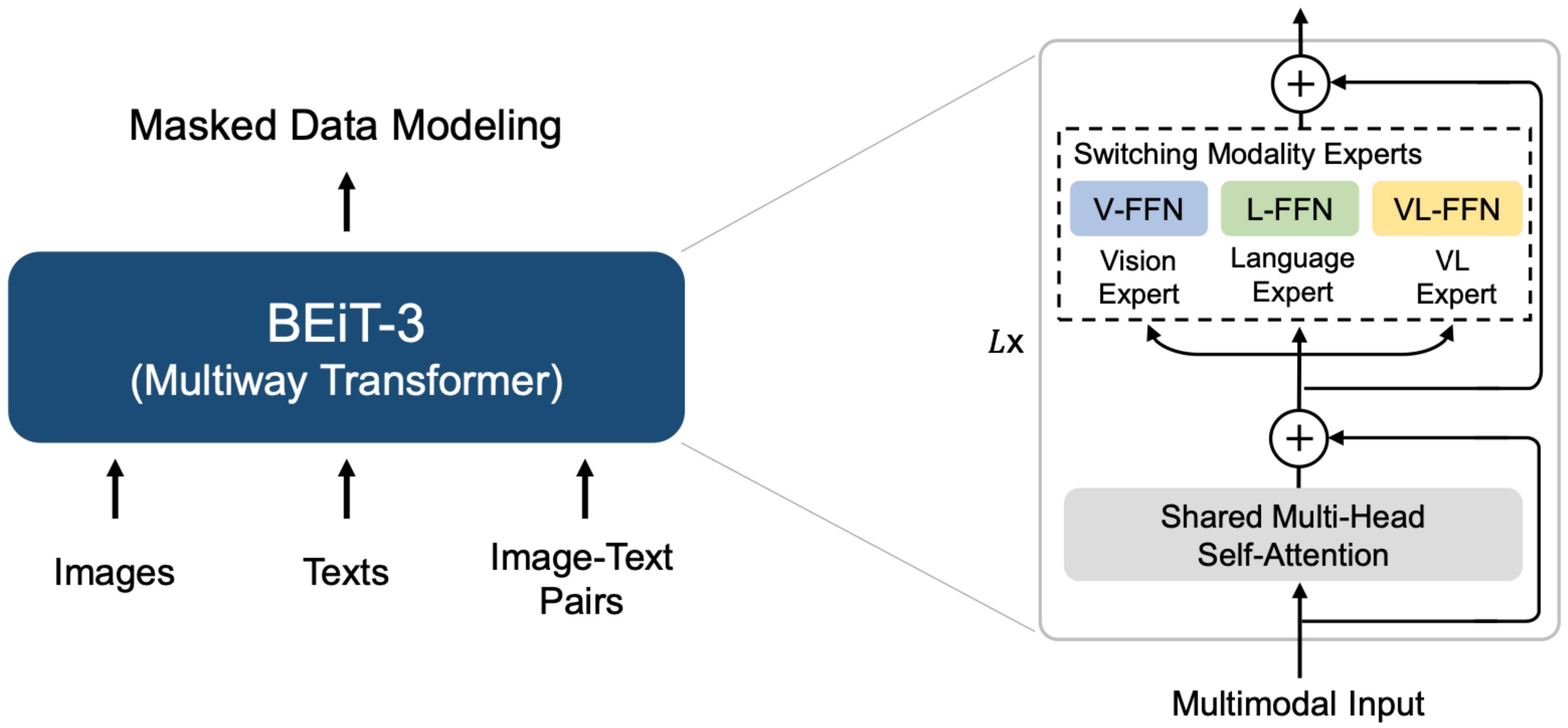
(c) Fusion Encoder

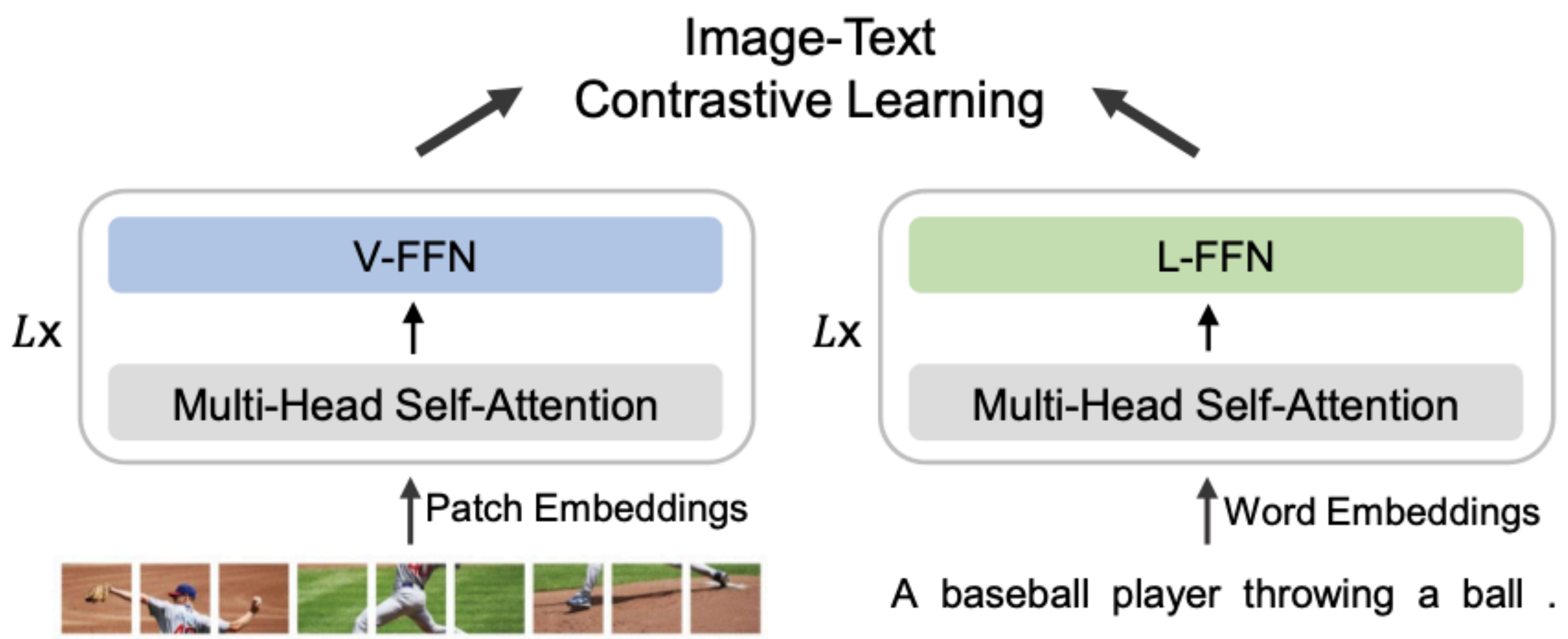
Masked Vision-Language Modeling

Vision-Language Tasks (VQA, NLVR2)

[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language



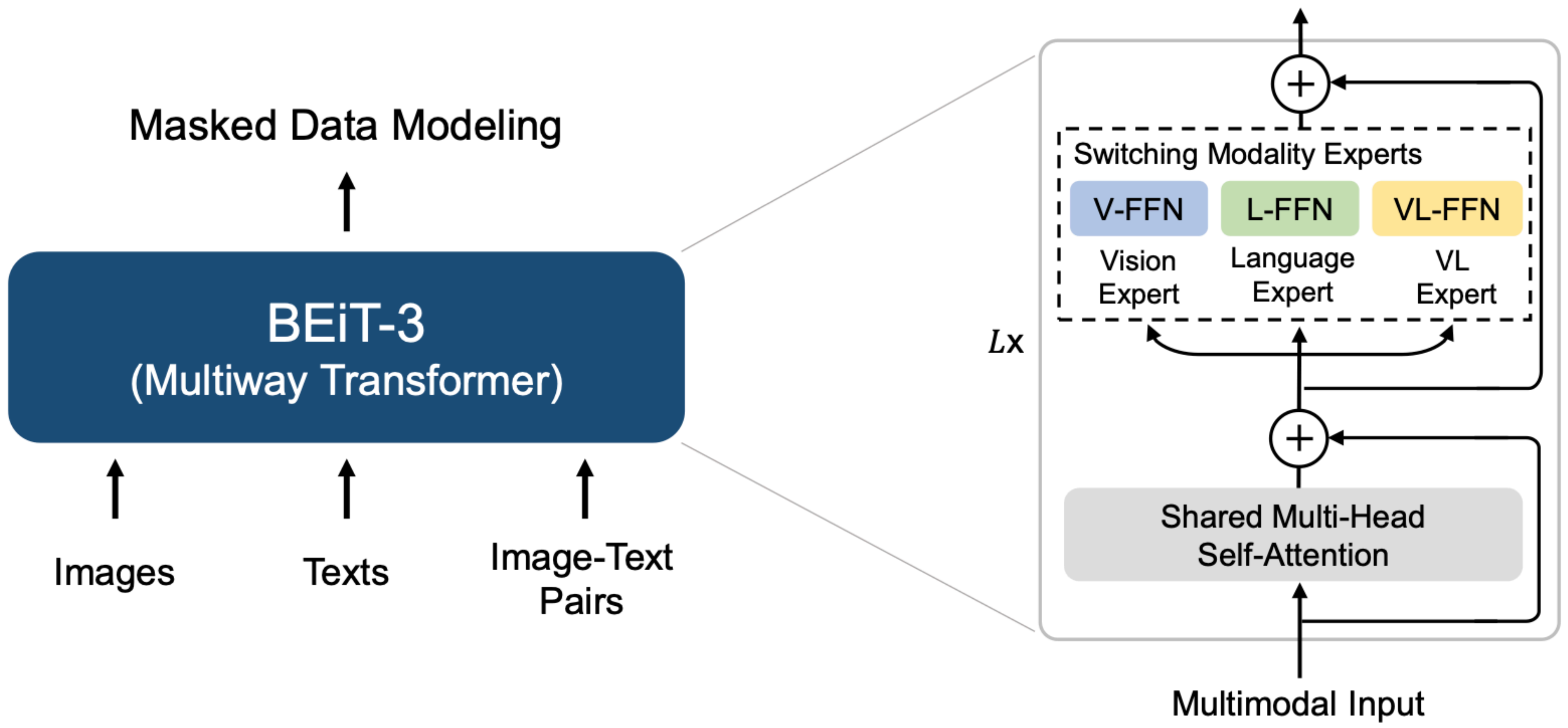


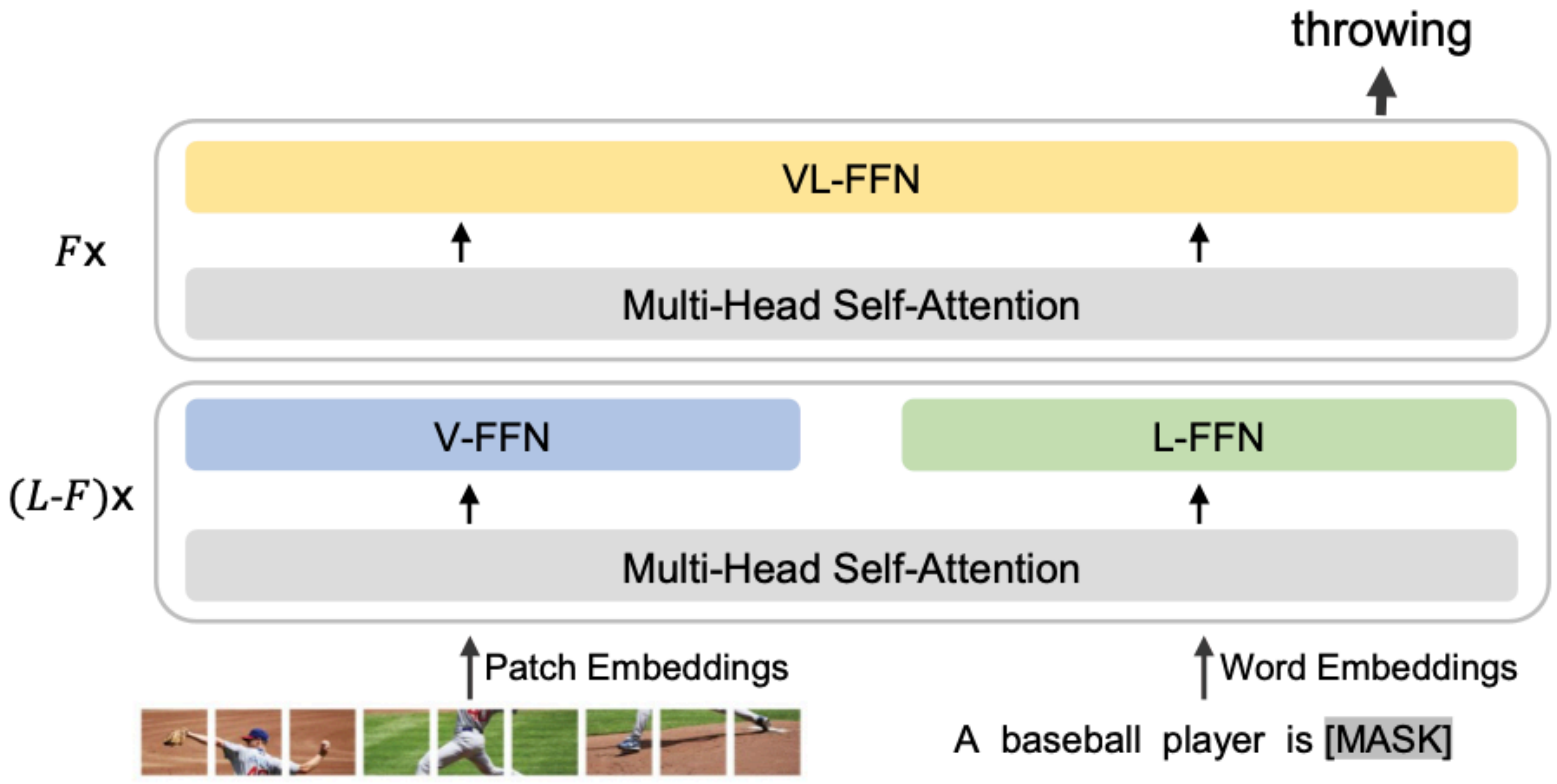
(d) Dual Encoder

Image-Text Retrieval (Flickr30k, COCO)

[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language





(e) Image-to-Text Generation

Image Captioning (COCO)

[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language

Model	MSCOCO (5K test set) Image → Text			MSCOCO (5K test set) Text → Image			Flickr30K (1K test set) Image → Text			Flickr30K (1K test set) Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
Fusion-encoder models												
UNITER [CLY+20]	65.7	88.6	93.8	52.9	79.9	88.0	87.3	98.0	99.2	75.6	94.1	96.8
VILLA [GCL+20]	-	-	-	-	-	-	87.9	97.5	98.8	76.3	94.2	96.8
Oscar [LYL+20]	73.5	92.2	96.0	57.5	82.8	89.8	-	-	-	-	-	-
VinVL [ZLH+21]	75.4	92.9	96.2	58.8	83.5	90.3	-	-	-	-	-	-
Dual encoder + Fusion encoder reranking												
ALBEF [LSG+21]	77.6	94.3	97.2	60.7	84.3	90.5	95.9	99.8	**100.0**	85.6	97.5	98.9
BLIP [LLXH22]	82.4	95.4	97.9	65.1	86.3	91.8	97.4	99.8	99.9	87.6	97.7	99.0
Dual-encoder models												
ALIGN [JYX+21]	77.0	93.5	96.9	59.9	83.3	89.8	95.3	99.8	**100.0**	84.9	97.4	98.6
FILIP [YHH+21]	78.9	94.4	97.4	61.2	84.3	90.6	96.6	**100.0**	**100.0**	87.1	97.7	99.1
Florence [YCC+21]	81.8	95.2	-	63.2	85.7	-	97.2	99.9	-	87.9	98.1	-
BEIT-3	**84.8**	**96.5**	**98.3**	**67.2**	**87.7**	**92.8**	**98.0**	**100.0**	**100.0**	**90.3**	**98.7**	**99.5**

[Wang et al., 2022]

BEiT-v3: Image as a Foreign Language

Model	**Extra OD Data**	**Maximum Image Size**	**COCO test-dev** AP^{box}	AP^{mask}
ViT-Adapter [CDW+22]	-	1600	60.1	52.1
DyHead [DCX+21]	ImageNet-Pseudo Labels	2000	60.6	-
Soft Teacher [XZH+21]	Object365	-	61.3	53.0
GLIP [LZZ+21]	FourODs	-	61.5	-
GLIPv2 [ZZH+22]	FourODs	-	62.4	-
Florence [YCC+21]	FLOD-9M	2500	62.4	-
SwinV2-G [LHL+21]	Object365	1536	63.1	54.4
Mask DINO [LZX+22]	Object365	1280	-	54.7
DINO [ZLL+22]	Object365	2000	63.3	-
BEIT-3	Object365	1280	**63.7**	**54.8**

[Wang et al., 2022]