RT-2, where are you from?

Peizhen Li[1]

[1]Faculty of Science and Engineering
Macquarie University

Oct 31, 2023

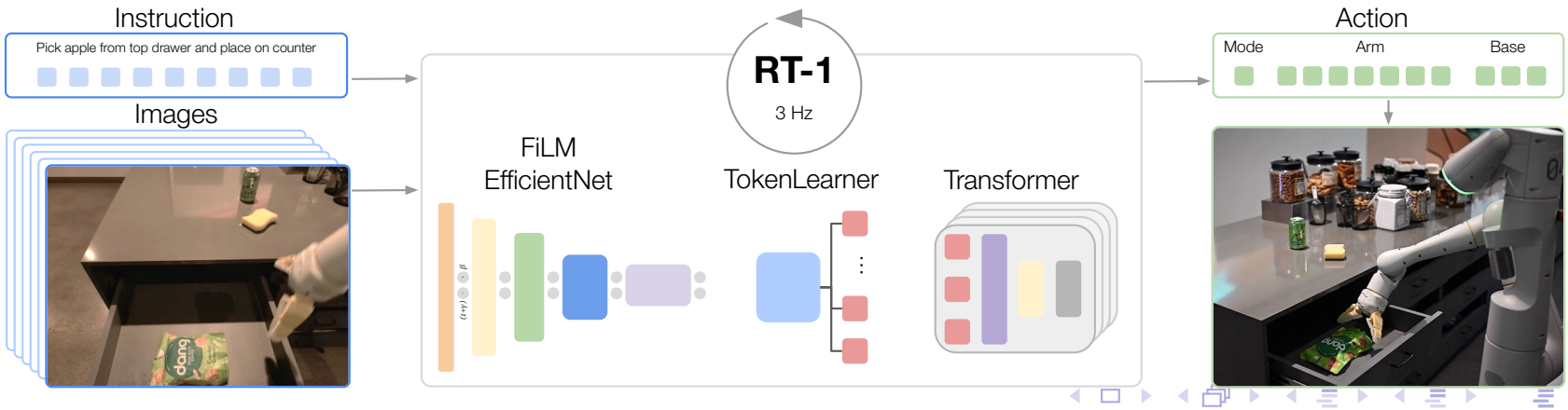
Outline

Key features of RT-1 and RT-2

- End-to-end robotic control
- Express the actions as text tokens
- RT-1: Large scale robot learning with data-absorbent model
- RT-2: Co-fine-tune vision-language models (vision-language-action)

End-to-end Explained

Figure: Output sequence of instructions vs. low-level actions



Robot Learning

- Learn robot policies to solve language-conditioned tasks

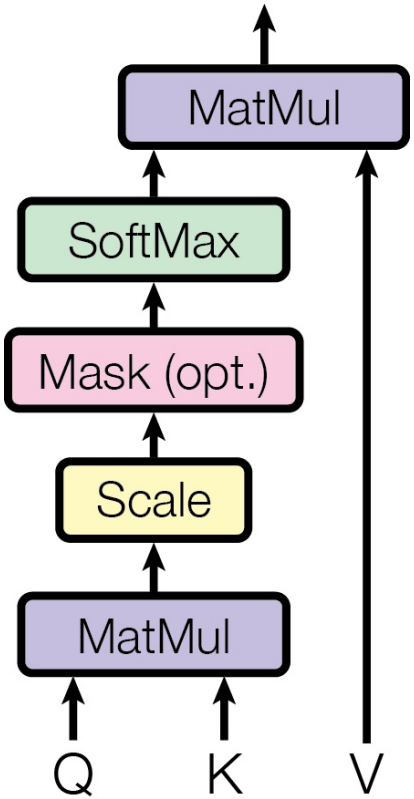
$$a_t \sim \pi(\cdot|i, \{x_j\}_{j=0}^{t}) \tag{1}$$

- Episode: full interaction $i, \{(x_j, a_j)\}_{j=0}^{T}$ from $t = 0$ to $t = T$
- Reward: $r \in \{0, 1\}$

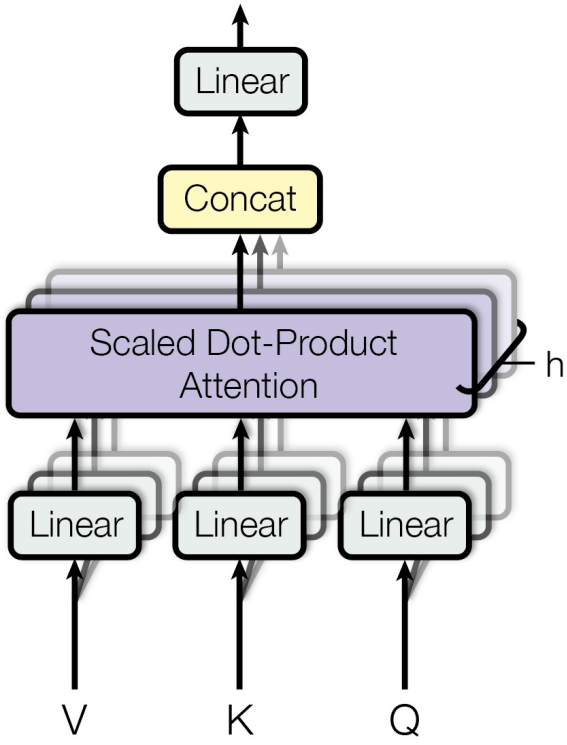
A more concrete view of Trajectory

```
class Trajectory(
    NamedTuple(
        'Trajectory',
        [
            ('step_type', types.SpecTensorOrArray),
            ('observation', types.NestedSpecTensorOrArray),
            ('action', types.NestedSpecTensorOrArray),
            ('policy_info', types.NestedSpecTensorOrArray),
            ('next_step_type', types.SpecTensorOrArray),
            ('reward', types.NestedSpecTensorOrArray),
            ('discount', types.SpecTensorOrArray),
        ],
    )
):
```

Transformers



(a) scaled dot-product attention: $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$



(b) multi-head attention: $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

Imitation Learning

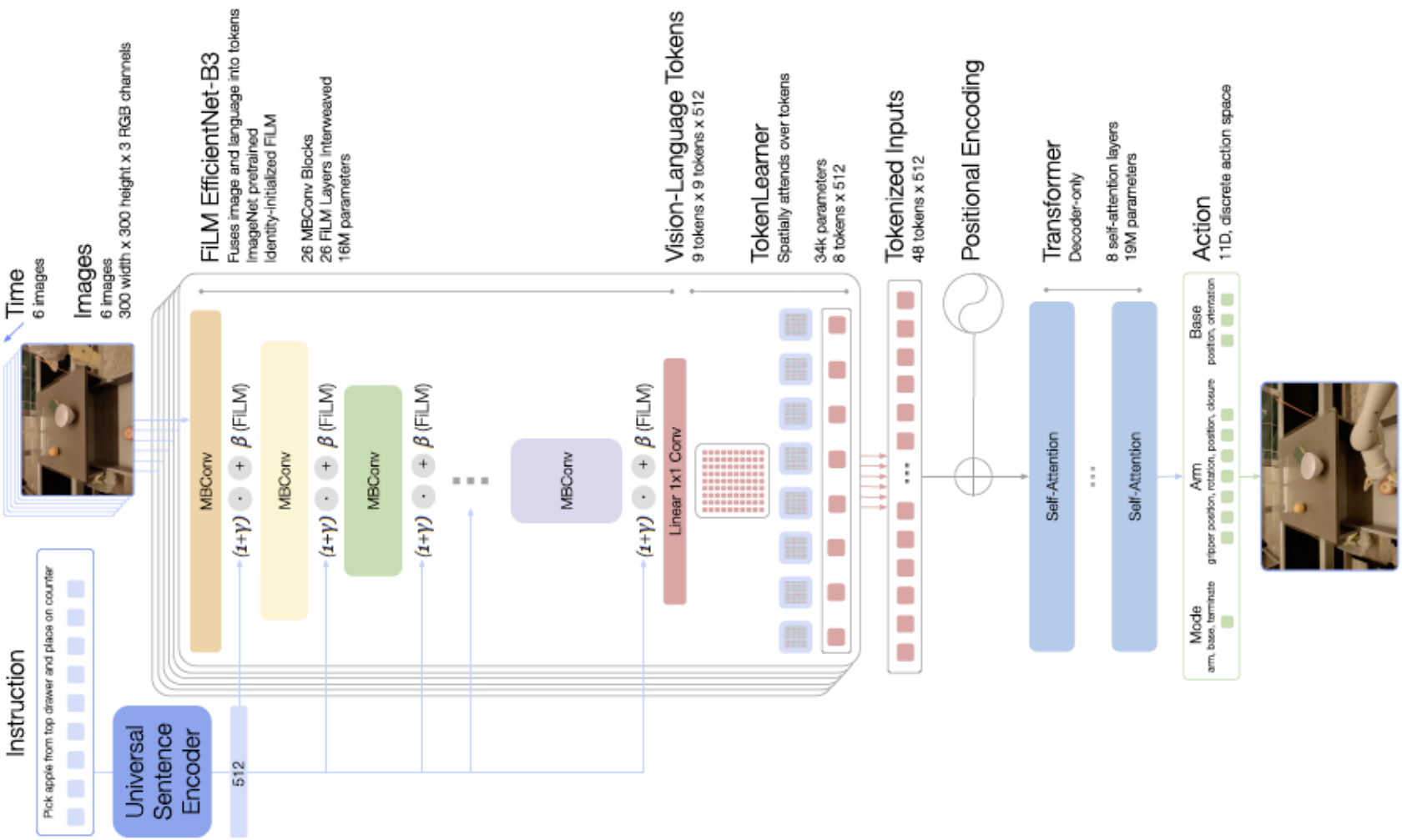
- Given a dataset of episodes:

$$\mathcal{D} = \{(i^{(n)}, \{(x_t^{(n)}, a_t^{(n)})\}_{t=0}^{T})\}_{n=0}^{N} \tag{2}$$

- Learn π using behavioral cloning: directly learn the expert policy

$$\min \sum_{i} \sum_{\{(x_t, a_t)\}_{t=0}^{T}} -\log \pi(a_t | i, \{x_j\}_{j=0}^{t}) \tag{3}$$

RT-1 architecture



Let's Break It Down

- Input: images, language instruction
- Output: de-tokenized actions

```
observation = {
    'image': tf.constant(tf.squeeze(obs_image).numpy(), shape=image_shape),
    'natural_language_embedding': tf.constant(tf.squeeze(embeded).numpy(),shape=emb_shape),
}
time_step = ts.restart(observation, batch_size=1)
init_state = eval_policy.get_initial_state(batch_size=1)
policy_step = eval_policy.action(time_step, init_state)
print(policy_step.action)
```

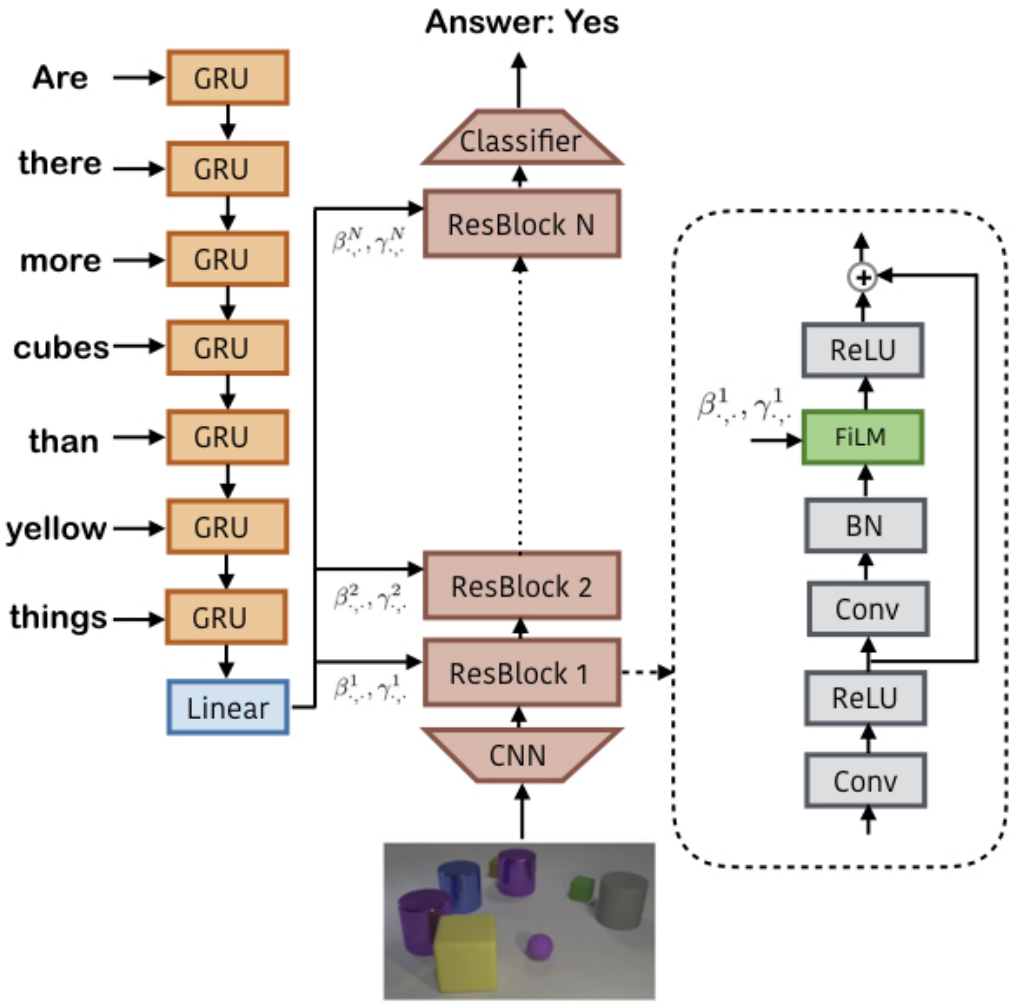
```
{'base_displacement_vector': BoundedArraySpec(shape=(2,), dtype=dtype('float32'), name='base_displacement_vector',
                                              minimum=-1.0, maximum=1.0),
'base_displacement_vertical_rotation': BoundedArraySpec(shape=(1,), dtype=dtype('float32'), name='base_displacement_vertical_rotation',
                                                        minimum=-3.1415927410125732, maximum=3.1415927410125732),
'gripper_closedness_action': BoundedArraySpec(shape=(1,), dtype=dtype('float32'), name='gripper_closedness_action',
                                              minimum=-1.0, maximum=1.0),
 'world_vector': BoundedArraySpec(shape=(3,), dtype=dtype('float32'), name='world_vector', minimum=-1.0, maximum=1.0),

 'terminate_episode': BoundedArraySpec(shape=(3,), dtype=dtype('int32'), name='terminate_episode', minimum=0, maximum=1),  # mode switch

 'rotation_delta': BoundedArraySpec(shape=(3,), dtype=dtype('float32'), name='rotation_delta',
                                    minimum=-1.5707963705062866, maximum=1.5707963705062866)}
```

Image and Language Fusion

- Condition the image tokenizer on the language instruction



FiLM: Feature-wise Linear Modulation

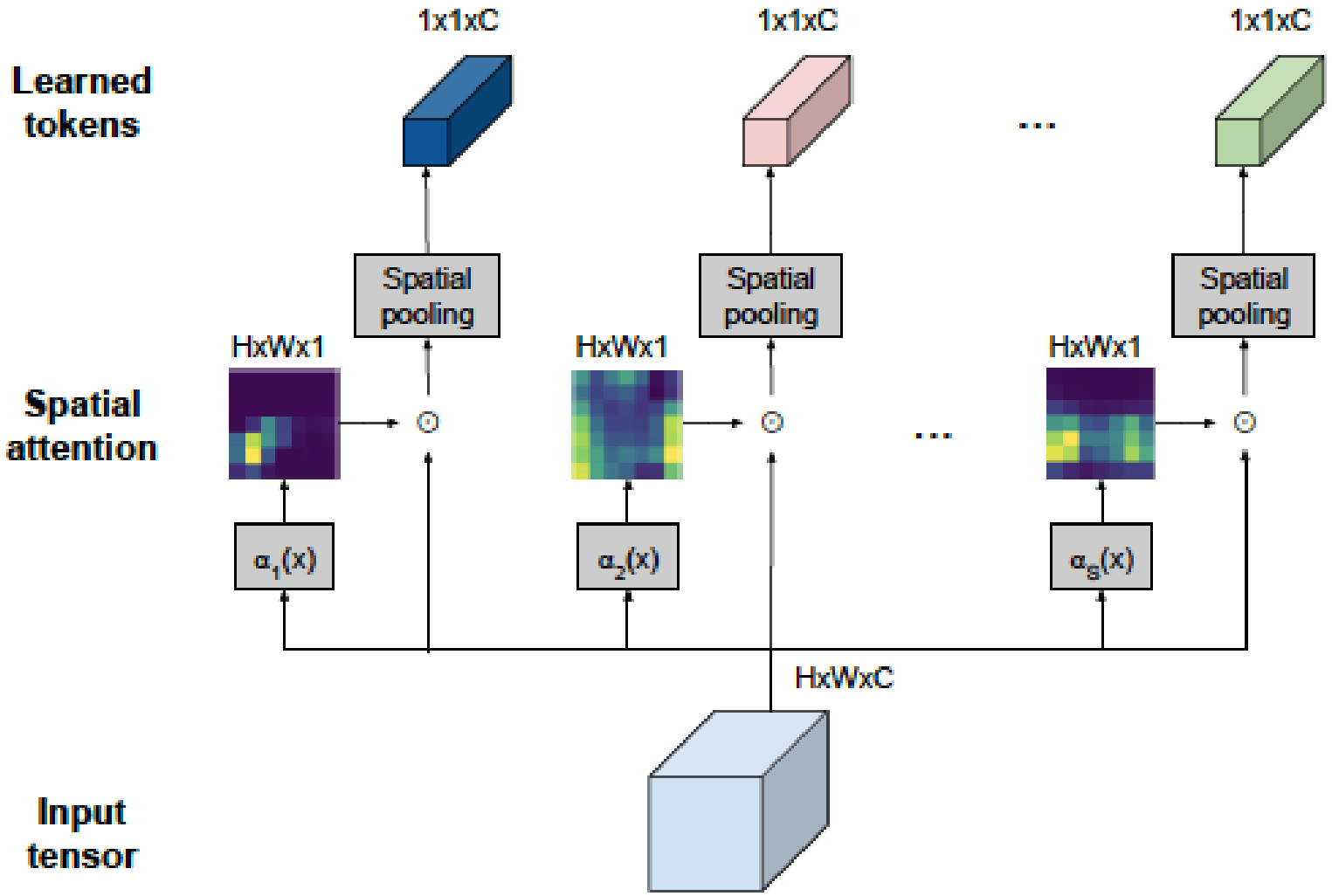
$$\gamma_{i,c} = f_c(x_i) \quad \beta_{i,c} = h_c(x_i) \tag{4}$$

$$FiLM(\mathbf{F}_{i,c}|\gamma_{i,c}, \beta_{i,c}) = \gamma_{i,c}\mathbf{F}_{i,c} + \beta_{i,c} \tag{5}$$

```
def call(self, conv_filters: tf.Tensor, conditioning: tf.Tensor):
  tf.debugging.assert_rank(conditioning, 2)
  projected_cond_add = self._projection_add(conditioning)  # beta
  projected_cond_mult = self._projection_mult(conditioning)  # gamma

  # Original FiLM paper argues that 1 + gamma centers the initialization at
  # identity transform.
  result = (1 + projected_cond_mult) * conv_filters + projected_cond_add
  return result
```

TokenLearner



TokenLearner

$$Z = [z_i]_{i=1}^{S}$$
$$z_i = \rho(X \odot \gamma(\alpha_i(X))) \tag{6}$$

```
def call(self, inputs: tf.Tensor, training: bool = False) -> tf.Tensor:
  if len(inputs.shape) == 4:
    bs, h, w, c = inputs.shape
    inputs = tf.reshape(inputs, shape: [bs, h * w, c])
  selected = self.layernorm(inputs)
  selected = self.mlp(
      *args: selected, is_training=training)  # Shape: [bs, h*w, n_token].
  selected = tf.transpose(selected, perm: [0, 2, 1])  # Shape: [bs, n_token, h*w].
  selected = tf.nn.softmax(selected, axis=-1)
  # element-wise operation
  feat = tf.einsum( equation: "...si,...id->...sd", *inputs: selected, inputs)
  return feat  # Shape: [bs, n_token, c]
```

Intuitions Behind RT-1 Model Design

Transformer-based model:

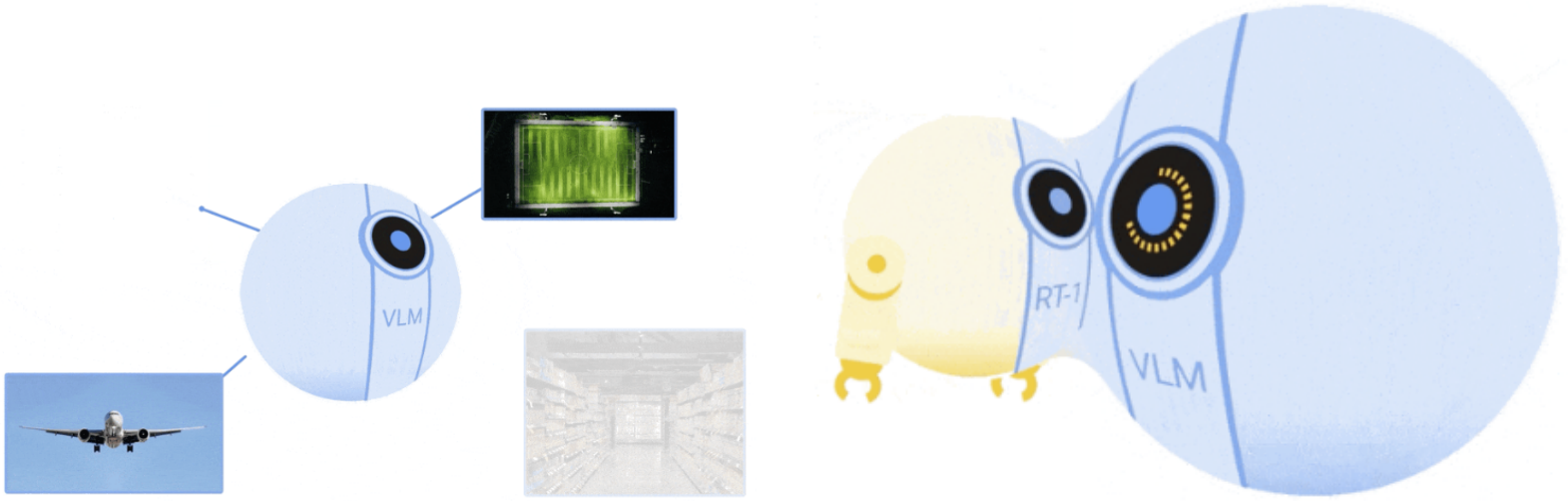
- FiLM-conditioned EfficientNet
- TokenLearner
- Transformer

Intuitions:

- Effective robotic multi-task learning
- Efficient real-time robotic control

From RT-1 to RT-2

- RT-2 learns from both web data and robotics data



Transfer Web Knowledge to Robotic Control

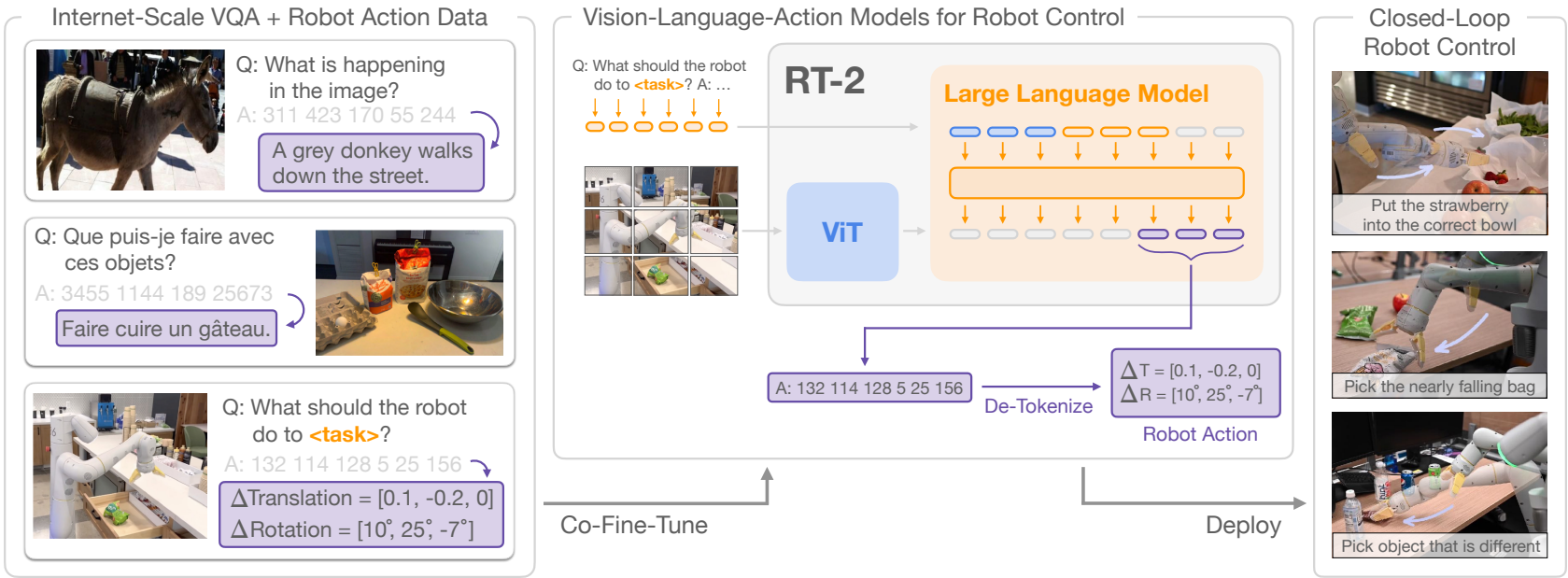


Figure: Present robot actions as another language. The text tokens are de-tokenized into robot actions during inference.

Vision-Language-Action Models

- Learn to map robot observations to actions
- Express robotic actions as text tokens
- Co-fine-tune vision-language models on both robotic trajectory data and web data

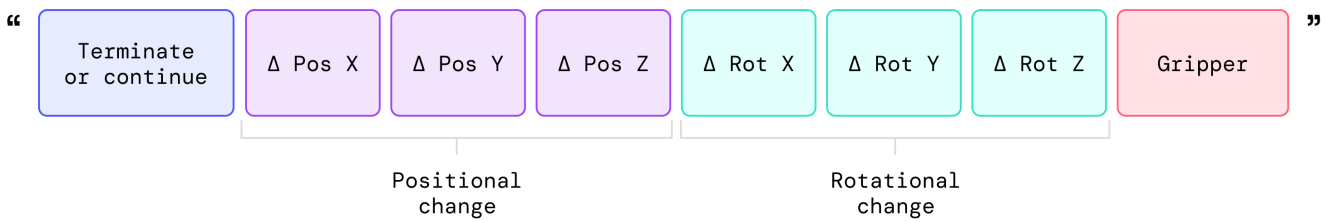
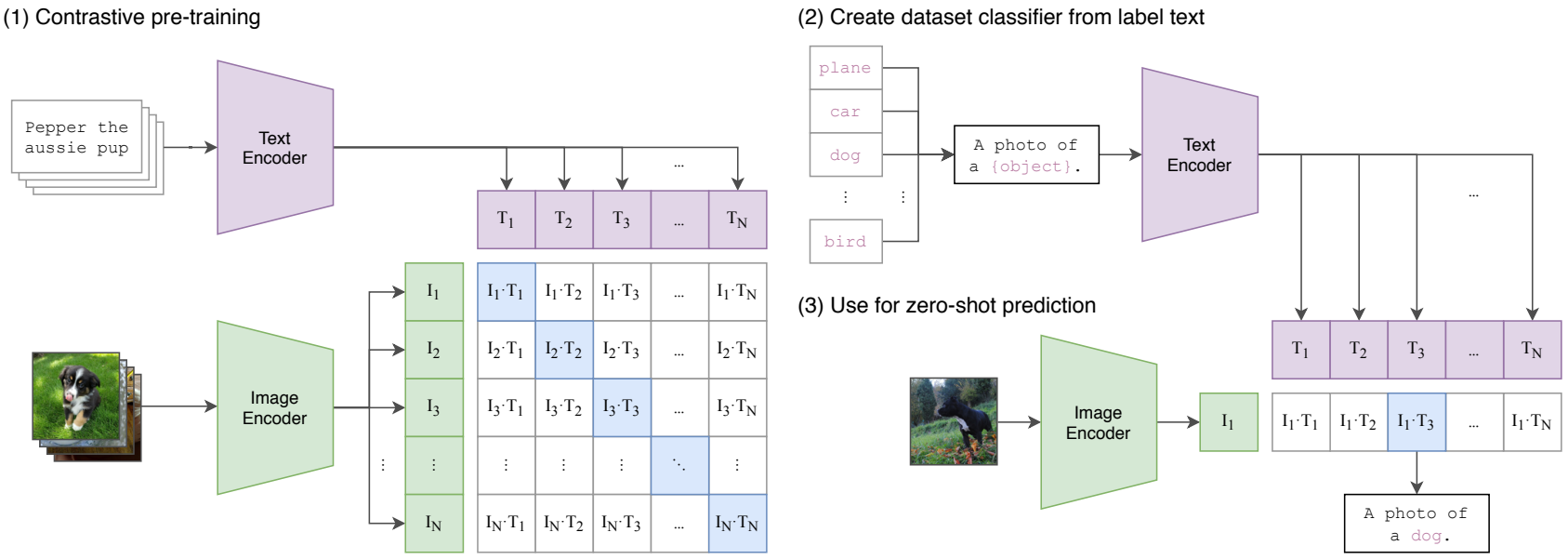


Figure: Robot action token numbers: “1 128 91 241 5 101 127 217”

CLIP: Contrastive Language Image Pre-training



Contrastive Language Image Pre-training

```
# 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
```

PaLI: Pathways Language and Image Model

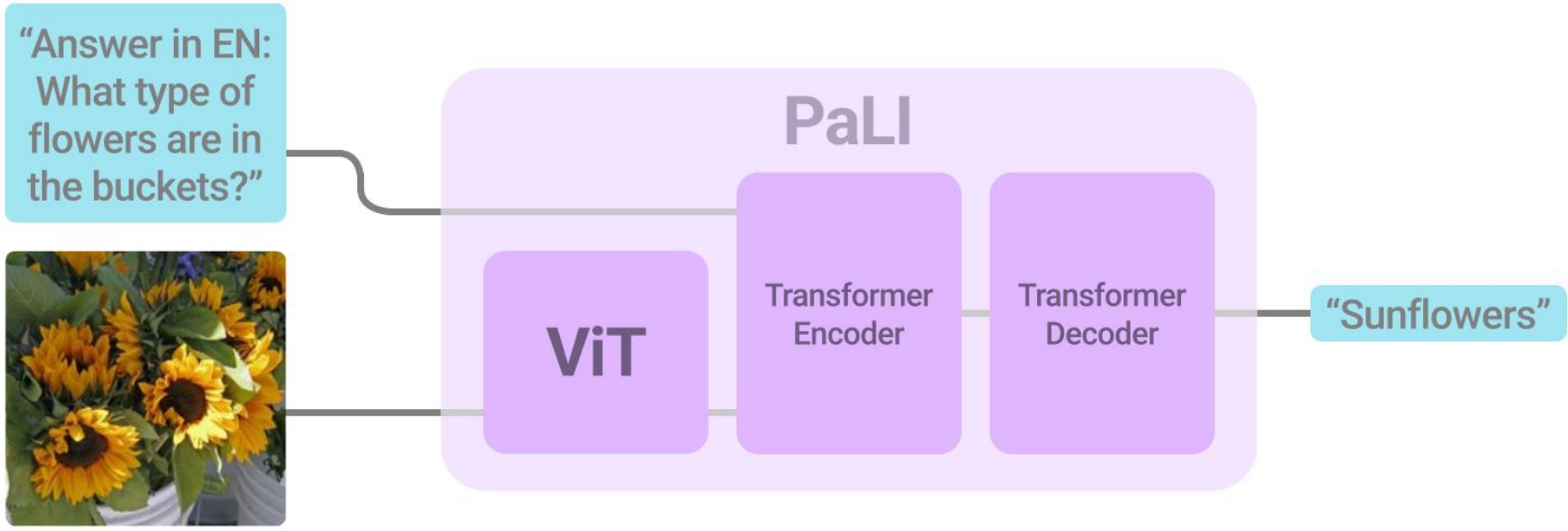


Figure: A simple and scalable architecture.

Pathways Explained

"A next-generation AI architecture"

TODAY'S AI MODELS	PATHSWAYS
Typically trained to do only one thing	Train a single model to do thousands or millions of things
Mostly focus on one sense	Enable multiple senses
Dense and inefficient	Sparse and efficient

PaLM-E

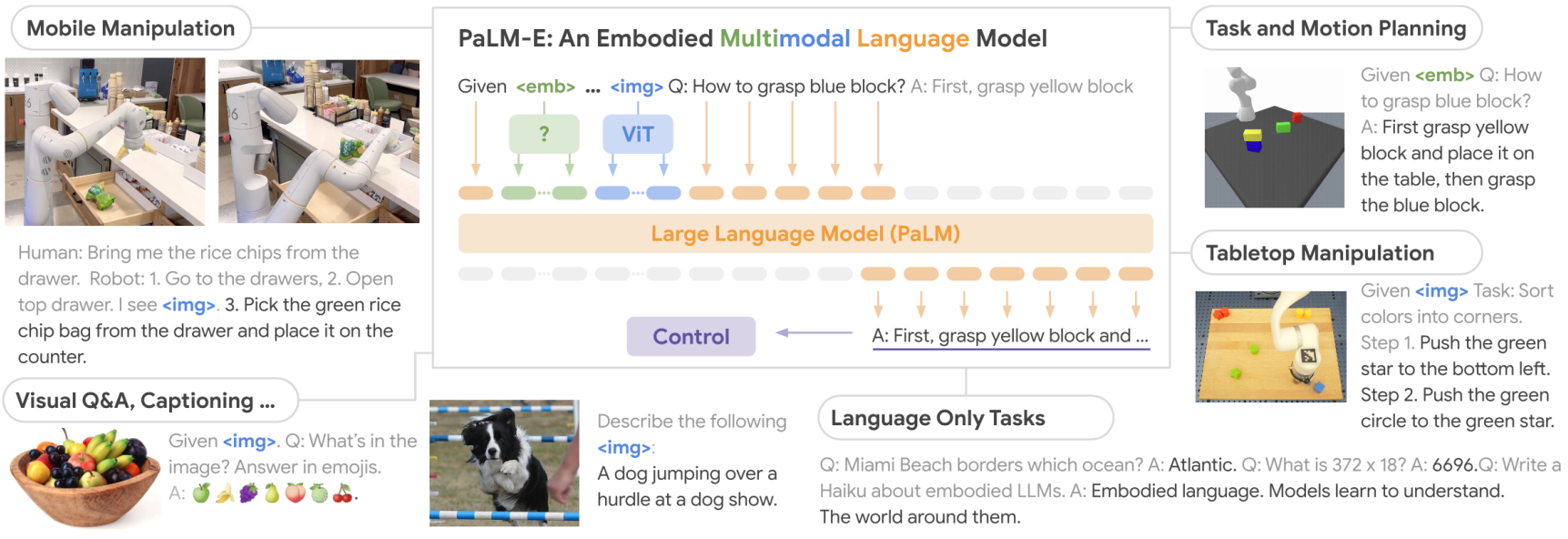
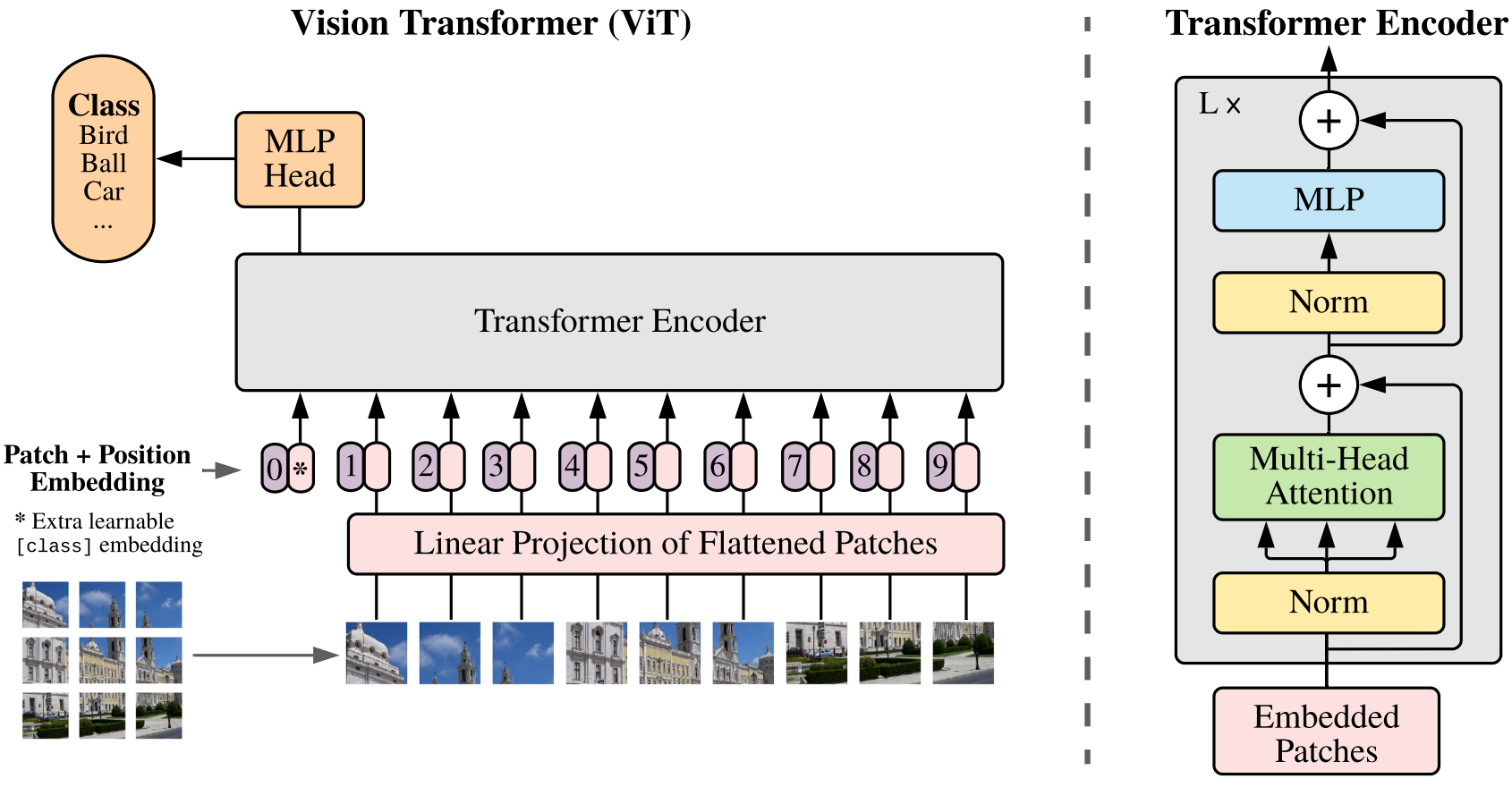


Figure: A single general-purpose multimodal language model.

Vision Transformer



Vision Transformer

$$z_0 = [\mathrm{x}_{class}; \mathrm{x}_p^1\mathbf{E}; \mathrm{x}_p^2\mathbf{E}; \cdots; \mathrm{x}_p^N\mathbf{E}] + \mathbf{E}_{pos} \quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{7}$$

$$z_l^{'} = MSA(LN(z_{l-1})) + z_{l-1}, \quad l = 1 \cdots L \tag{8}$$

$$z_l = MLP(LN(z_l^{'})) + z_l^{'} \tag{9}$$

$$y = LN(z_L^0) \tag{10}$$

Vision Transformer

```
# Attention block.
assert inputs.ndim == 3, f'Expected (batch, seq, hidden) got {inputs.shape}'
x = nn.LayerNorm(dtype=self.dtype)(inputs)
x = nn.MultiHeadDotProductAttention(
    dtype=self.dtype,
    kernel_init=nn.initializers.xavier_uniform(),
    broadcast_dropout=False,
    deterministic=deterministic,
    dropout_rate=self.attention_dropout_rate,
    num_heads=self.num_heads)(
        x, x)
x = nn.Dropout(rate=self.dropout_rate)(x, deterministic=deterministic)
x = x + inputs

# MLP block.
y = nn.LayerNorm(dtype=self.dtype)(x)
y = MlpBlock(
    mlp_dim=self.mlp_dim, dtype=self.dtype, dropout_rate=self.dropout_rate)(
        y, deterministic=deterministic)

return x + y
```

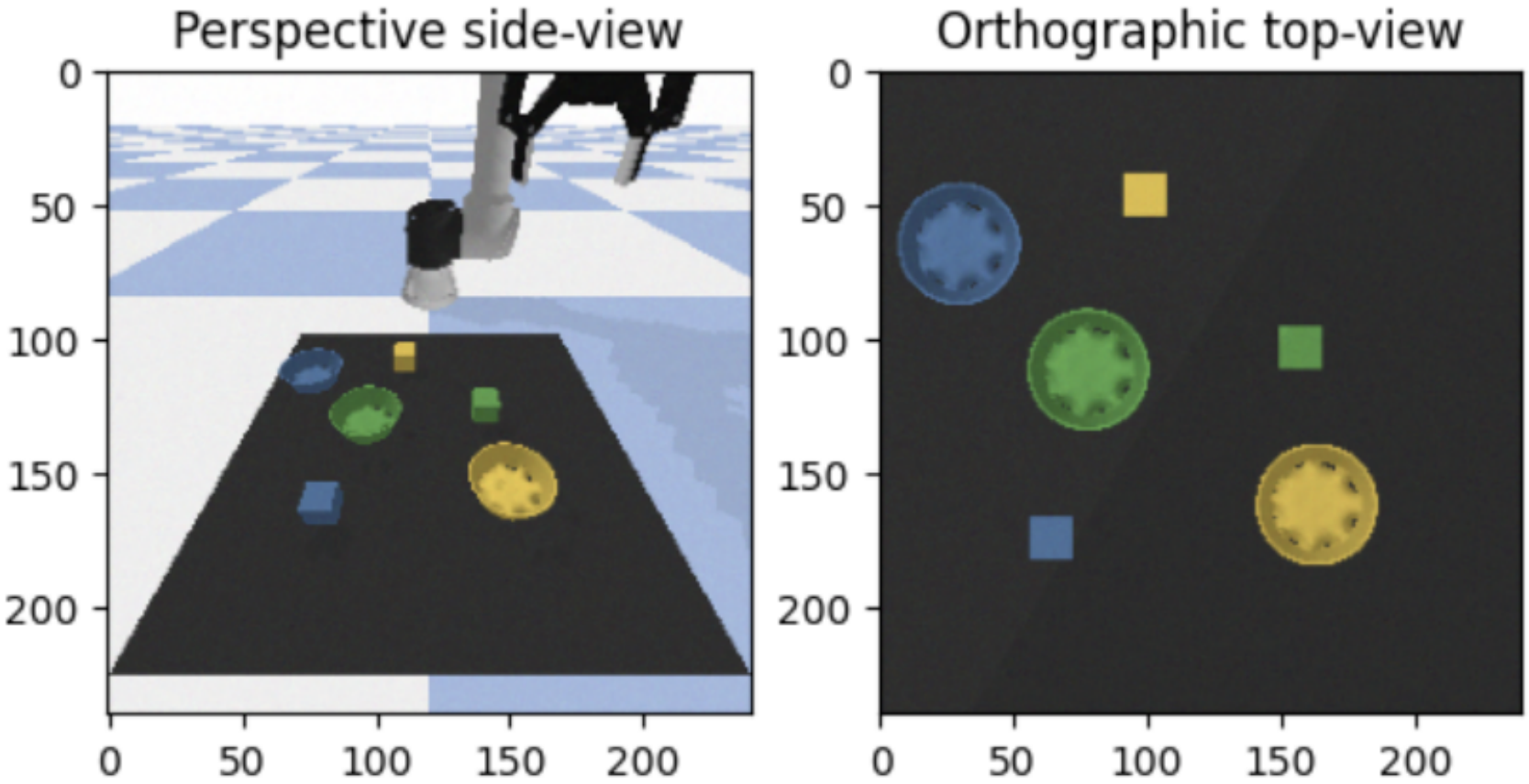
Reflection

- If few equations, then conduct extensive experiments
- Multi-modal fusion happens in the embedding space
- Image and text inputs are processed in similar way

Multimodal Learning

- Explore inter-modal and intra-modal relations from a statistical perspective
- Aim for task-agnostic multimodal representations
- Robust and efficient enough for a robotic system

Simulation Environment



Thank you very much!
Q&A