Object-Centric representations with Slot Attention for Visual tasks

Daniil Kirilenko, FRC CSC RAS
Alexey Kovalev, AIRI
Aleksandr I Panov, AIRI

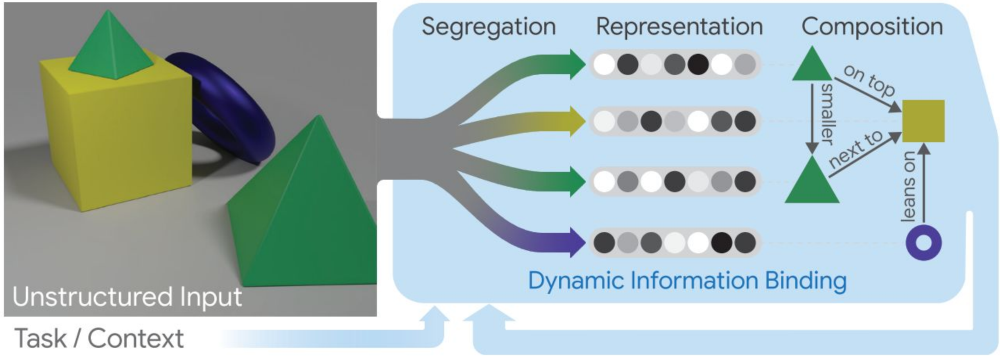
Moscow, 16.09.2022

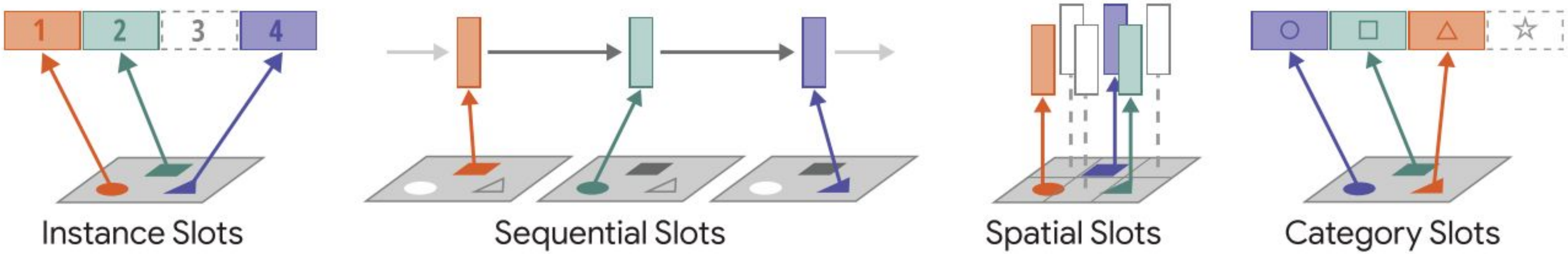
Outline

- Disentanglement and binding problem
- Slot-based object-centric architectures
  - Object discovery task (unsupervised)
  - Set property prediction task (supervised)
- Slot Attention and SLATE
  - Slot Attention
  - SA performance
  - Slot Attention Transformer
- Slot Attention as trainable clustering method
- GMM-based analogue
  - Gaussian Mixture Model
  - GMM-based object-centric model
- Discrete latent variables for slots representations
- Conclusion and future work

Disentanglement and binding problem

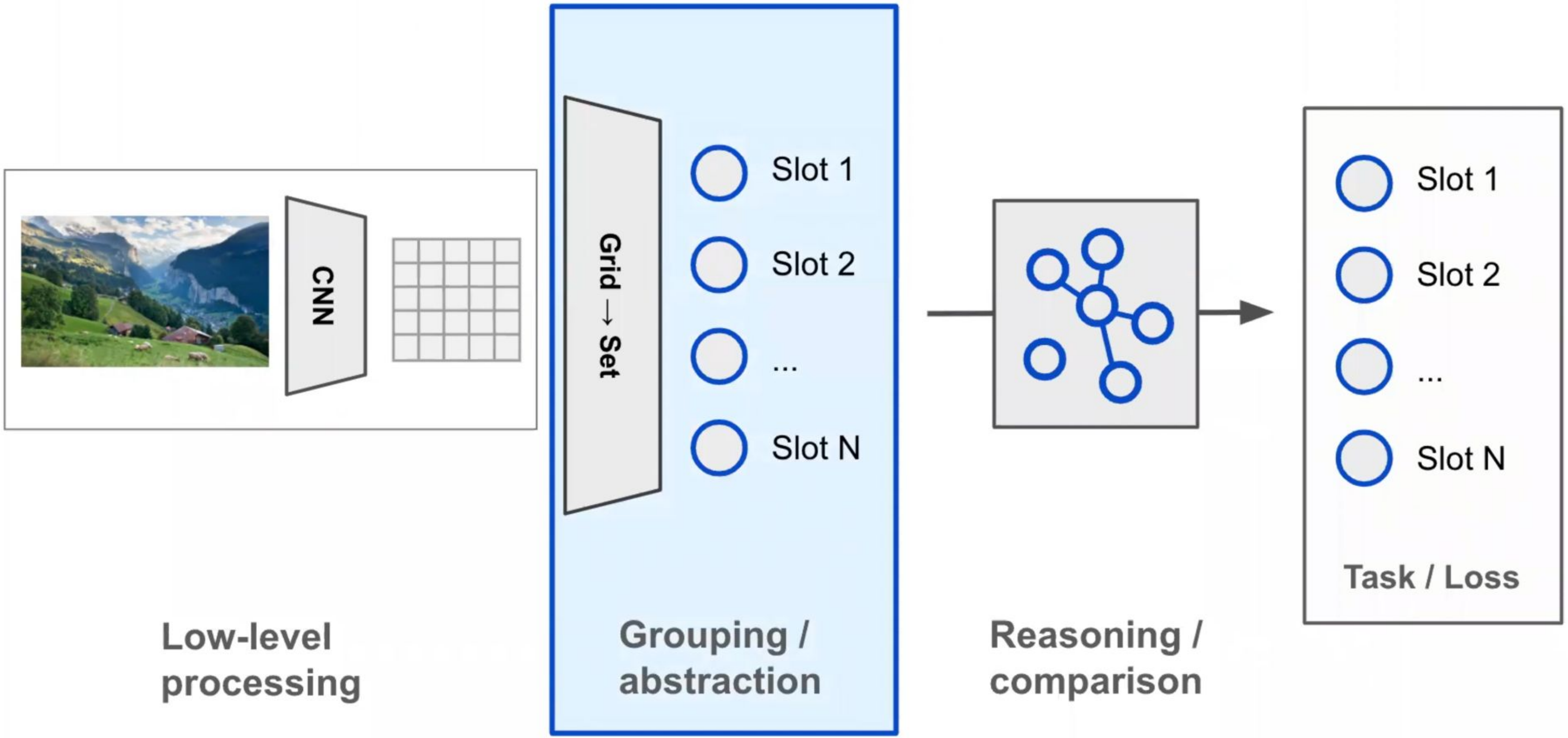
- Binding problems in ANN: *segregation, representation, and composition* subproblems
- We need dynamically binding neurally processed information
- Facilitating more symbolic information processing
- Slots paradigm: instances, sequences and categories





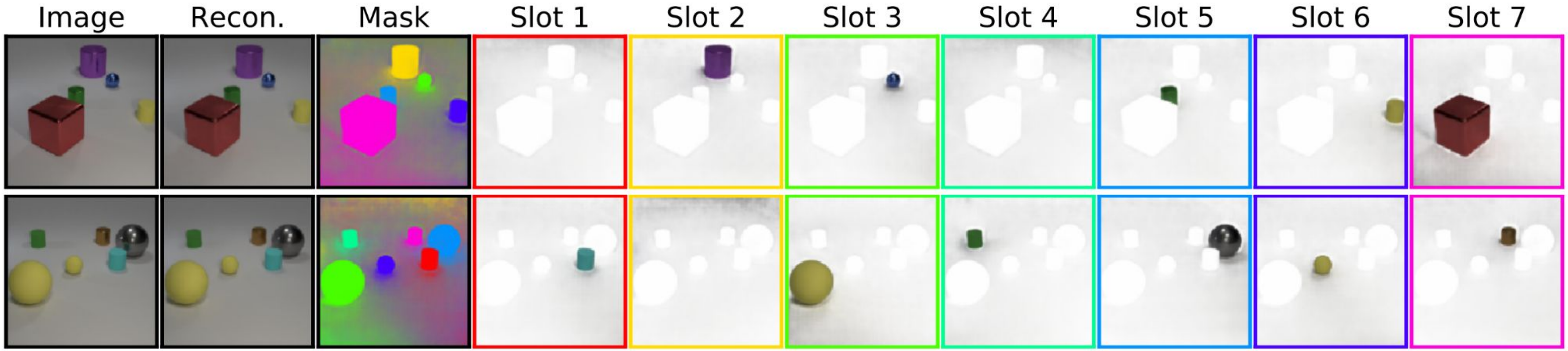
Greff K., van Steenkiste S., Schmidhuber J. On the Binding Problem in Artificial Neural Networks. 2020.

Slot-based object-centric architectures



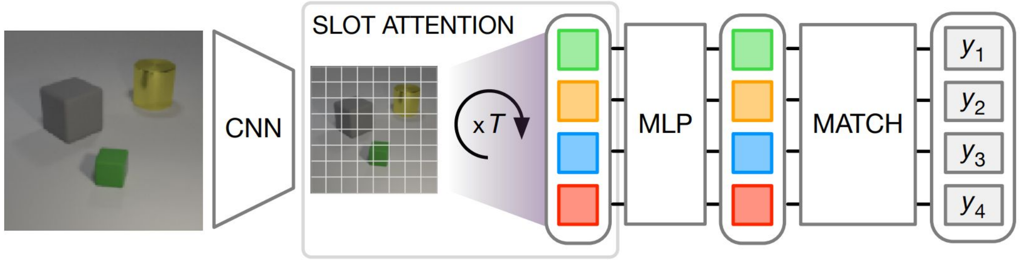
Object discovery task

- For given image x get a set of latent variables $\{z_i\}$
- Decode every z into image space $p_\theta(\hat{x}_i|z_i)$
  and get the corresponding mask $p_\theta(m_i|z_i)$
- Get the final reconstruction: $\hat{x} = \sum_i m_i p(\hat{x}_i|z_i)$



Set property prediction task

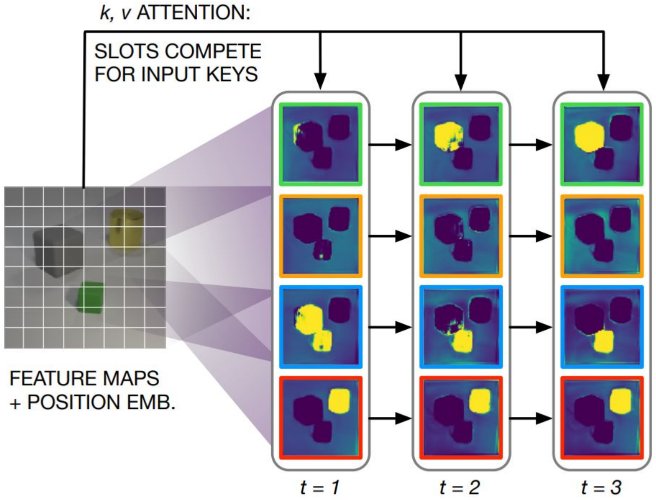
- An input image and an unordered set of prediction targets are given
- The key challenge in predicting sets is that there are K! possible equivalent representations for a set of K elements (motivation for permutation invariance)
- This inductive bias needs to be explicitly modeled in the architecture



(c) Set prediction architecture.

Slot Attention

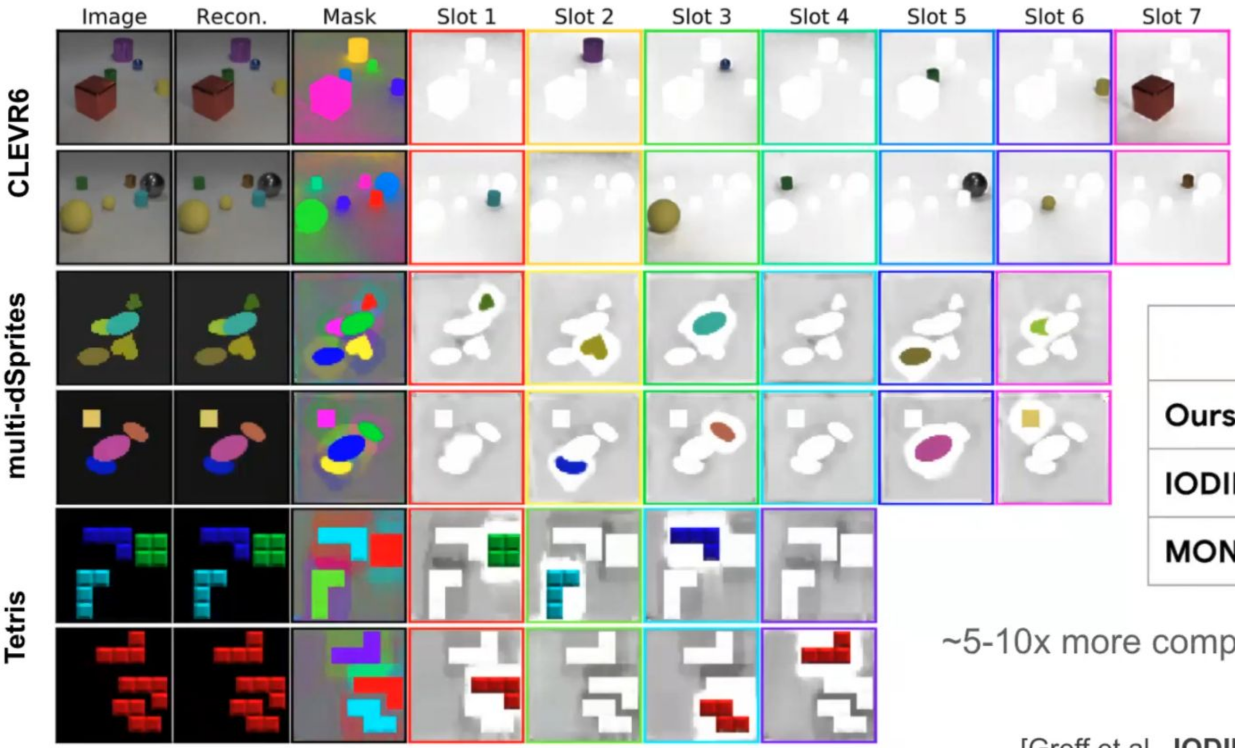
```
# inputs: cnn feature maps + position embedding
slots ~ normal(mean, std)
for t in range(num_steps):
      scores = dot(k(inputs), q(slots))
      weights = softmax(scores, dim='slots')
      updates = weighted_mean(weights, v(inputs))
      slots = GRU(slots, updates)
      slots = slots + MLP(norm(slots))
```



(a) Slot Attention module.

Francesco Locatello et al. Object-Centric Learning with Slot Attention. NeurIPS 2020

Object discovery performance



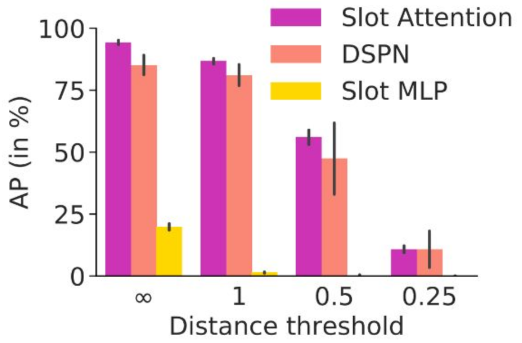
Segmentation results (ARI scores in %):

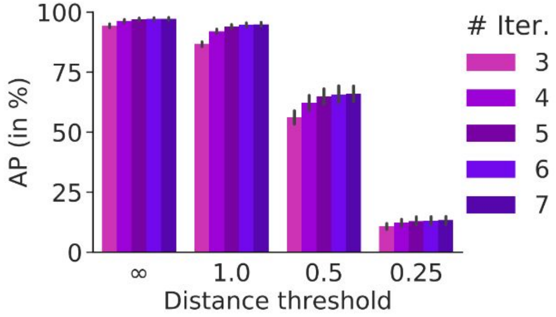
	CLEVR6	m-dSprites	Tetris
Ours	**98.8**	**91.3**	**99.5**
IODINE	**98.8**	76.7	**99.2**
MONet	96.2	**90.4**	---

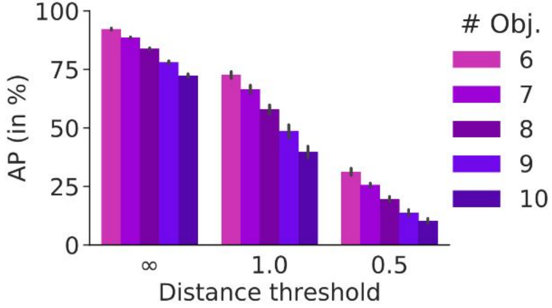
~5-10x more compute/memory efficient than IODINE

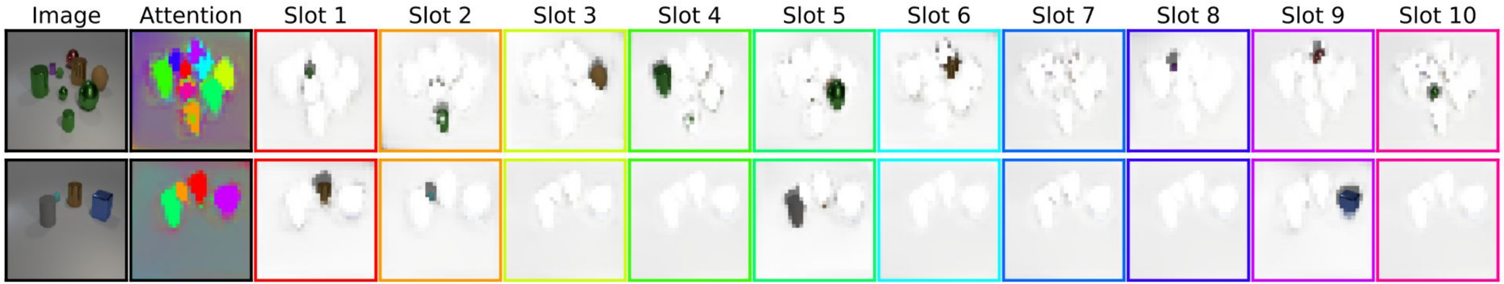
[Greff et al., **IODINE** (2019), Burgess et al., **MONet** (2019)]

Set prediction performance

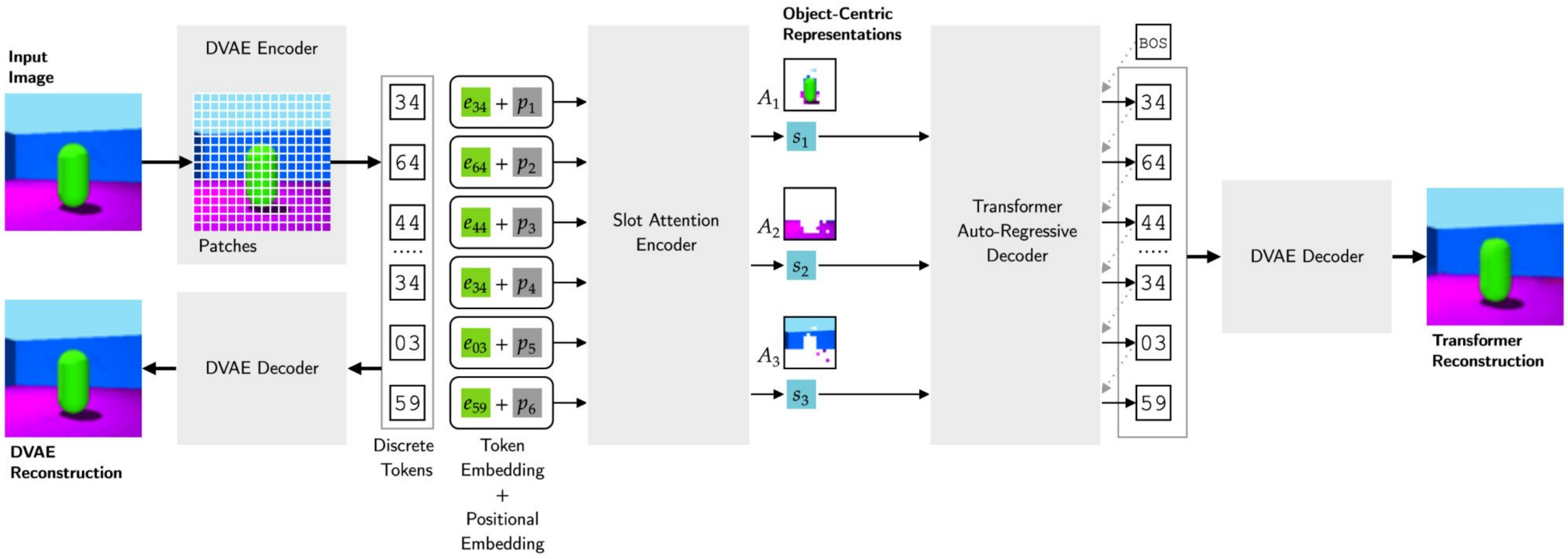




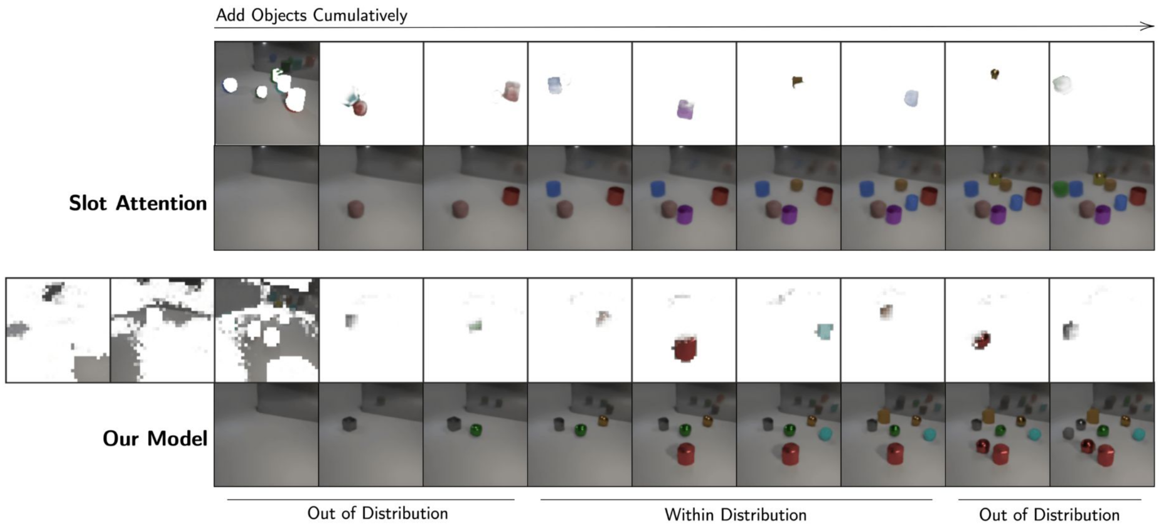


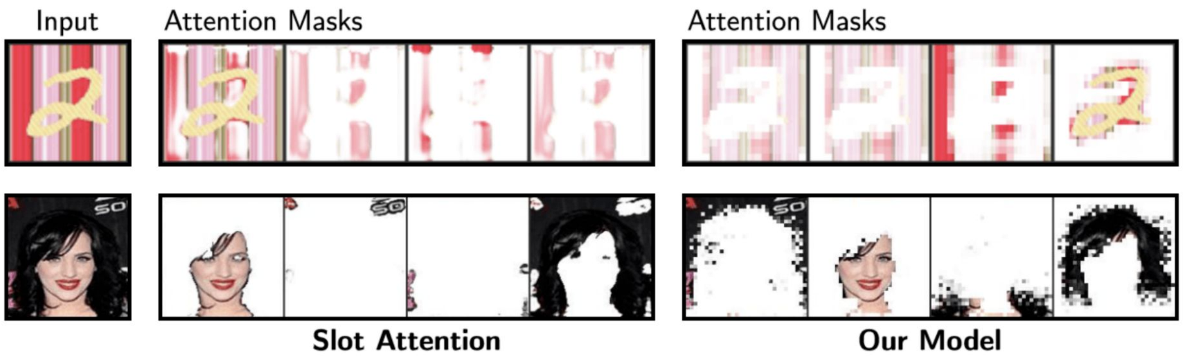


SLot Attention TransformEr (SLATE)



Gautam Singh et al. Illiterate DALL-E learns to compose. ICLR 2022


Add Objects Cumulatively
Slot Attention
Our Model
Out of Distribution
Within Distribution
Out of Distribution


Input
Attention Masks
Attention Masks
Slot Attention
Our Model

Slot Attention vs Clustering

Slot Attention Pseudocode

```
# inputs: cnn feature maps + position embedding
slots ~ normal(mean, std)
for t in range(num_steps):
      scores = dot(k(inputs), q(slots))
      weights = softmax(scores, dim='slots')
      updates = weighted_mean(weights, v(inputs))
      slots = GRU(slots, updates)
      slots = slots + MLP(norm(slots))
```

Soft k-Means Pseudocode

```
slots ~ normal(mean, std)
for t in range(num_steps):
      scores = -euclidian_dist(inputs, slots)
      weights = softmax(scores, dim='slots')
      updates = weighted_mean(weights, inputs)
      slots = updates
```

Idea: apply GMM-like approach

Gaussian Mixture Model (GMM)

Hypothesis:

$$x_i \sim p(x) = \sum_{j=1}^{K} \pi_j N(\mu_j, \sigma_j), \quad \sum_{j=1}^{K} \pi_j = 1$$

Initialization:

$$\mu_j,\ \sigma_j \sim p_\theta(\mu, \sigma), \quad \pi_j = \frac{1}{K}$$

Expectation step:

$$p(c = j|x_i) = \frac{p(x_i|c = j)p(c = j)}{\sum_c p(x_i|c)p(c)} = \frac{\pi_j N(x_i|\mu_j, \sigma_j)}{\sum_c \pi_c N(x_i|\mu_c, \sigma_c)}$$

Maximization step:

$$\mu_j = \sum_i p(c = j|x_i) x_i$$

$$\sigma_j^2 = \sum_i p(c = j|x_i)(x_i - \mu_j)(x_i - \mu_j)^T$$

Pseudocode

```
# params initialization for K Gaussians
mu ~ normal(mean_mu, std_mu)
logsigma ~ normal(mean_logs, std_logs)
pi = 1 / K
for t in range(num_steps):
    # E step, p(c=j|x) estimation
    scores = -0.5 * euclidian_dist(inputs, mu)
    scores /= exp(2 * logsigma)
    probs = exp(scores) * pi
    probs /= probs.sum(dim='slots')
    # M step for gaussians centers
    weights = probs / probs.sum(dim='x_i')
    mu_updates = weighted_sum(weights, inputs)
    # NN update for gaussians centers
    mu = GRU(mu, mu_updates)
    mu = mu + MLP(norm(mu))
    # M step for remaining params
    logsigma = 0.5 * log(weighted_sum(weights, (inputs - mu)**2))
    pi = probs.sum(dim='x_i') / N
slots = concat([mu, logsigma])
```

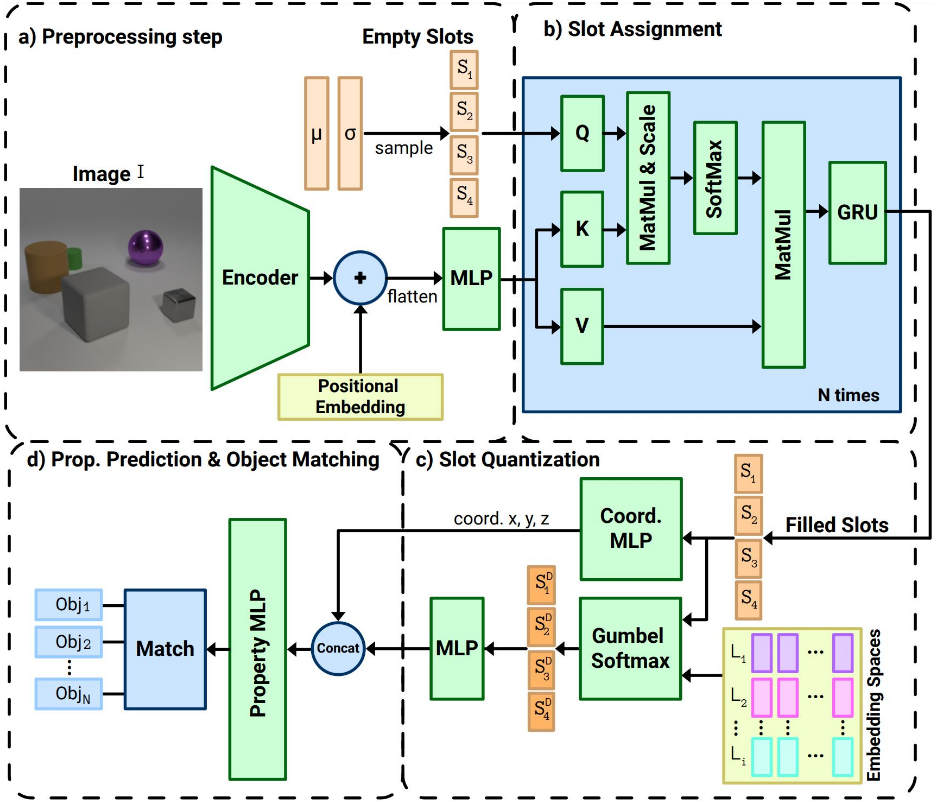
- E step is performed in exactly the same way as in the original algorithm
- M step differs from the original in the use of neural nets, which serve as the bridge between the internal and external models
- Concatenated Gaussian parameters are a more expressive representation of a cluster than centroid coordinates

GMM-based model performance (set prediction)

	AP(-1)	AP(1)	AP(0.5)	AP(0.25)	AP(0.125)	AP(0.0625)
Slot Attention	94.3	86.7	56.0	10.8	0.9	-
iDSPN (SOTA)	98.8	98.5	98.2	95.8	76.9	32.3
GMM-based	**99.4**	**99.4**	**99.2**	**98.8**	**92.8**	**47.7**

image2image experiments in progress

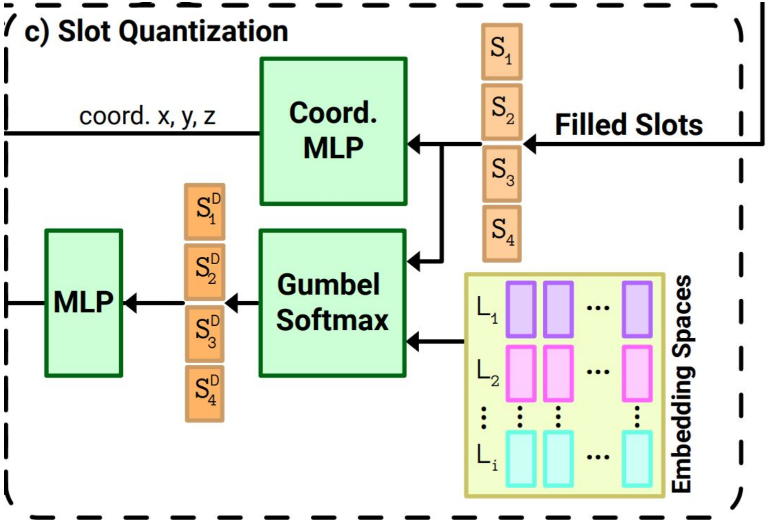
VQ-SA



Goal: to model slots with discrete latent variables with minimal performance reduction

Ideally, we would like to have representations such that each latent variable corresponds to only one property, and each value of that variable corresponds to only one value of that property.

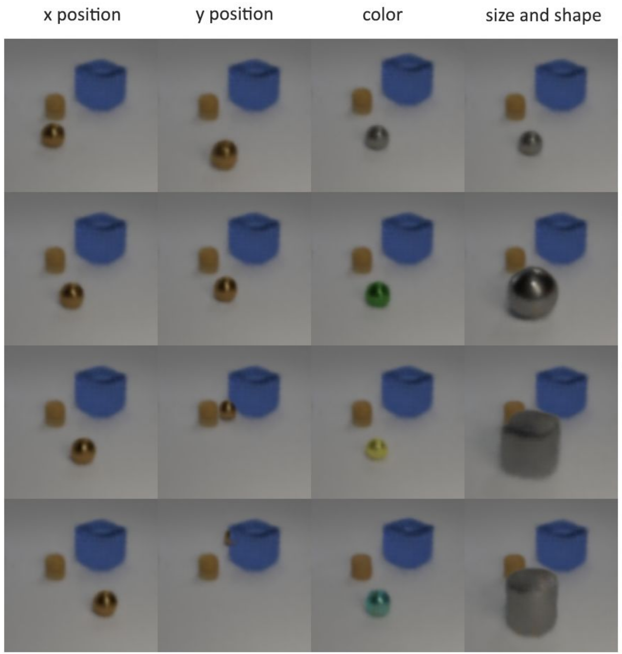
Slot quantization

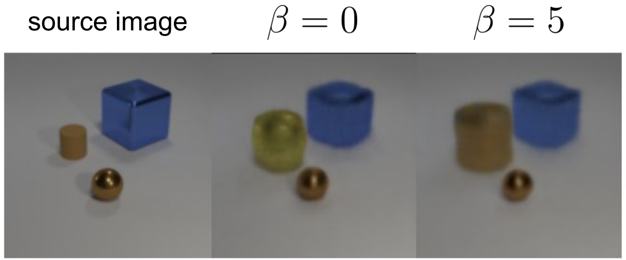


$$q(z|x) = \prod_i q(z_i|x)$$

- 4 latent variables
- Distributions for each latent variable are calculated independently using bilinear forms
- Prior distributions are uniform
- Gumbel Softmax temperature is decreasing during training

Controllable object editing





$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta \; D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

Set property prediction performance

Model	AP_{∞} (%)	AP_1 (%)	$AP_{0.5}$ (%)	$AP_{0.25}$ (%)	$AP_{0.125}$ (%)
Slot MLP	19.8 ± 1.6	1.4 ± 0.3	0.3 ± 0.2	0.0 ± 0.0	0.0 ± 0.0
DSPN T=30	85.2 ± 4.8	81.1 ± 5.2	47.4 ± 17.6	10.8 ± 9.0	0.6 ± 0.7
DSPN T=10	72.8 ± 2.3	59.2 ± 2.8	39.0 ± 4.4	12.4 ± 2.5	1.3 ± 0.4
Slot Attention	94.3 ± 1.1	86.7 ± 1.4	56.0 ± 3.6	10.8 ± 1.7	0.9 ± 0.2
VQ-SA (ours)	$\mathbf{96.1 \pm 0.4}$	$\mathbf{91.2 \pm 0.5}$	$\mathbf{71.8 \pm 2.3}$	$\mathbf{22.2 \pm 2.1}$	$\mathbf{2.4 \pm 0.2}$
iDSPN	98.8 ± 0.5	98.5 ± 0.6	98.2 ± 0.6	95.8 ± 0.7	76.9 ± 2.5

Disentanglement

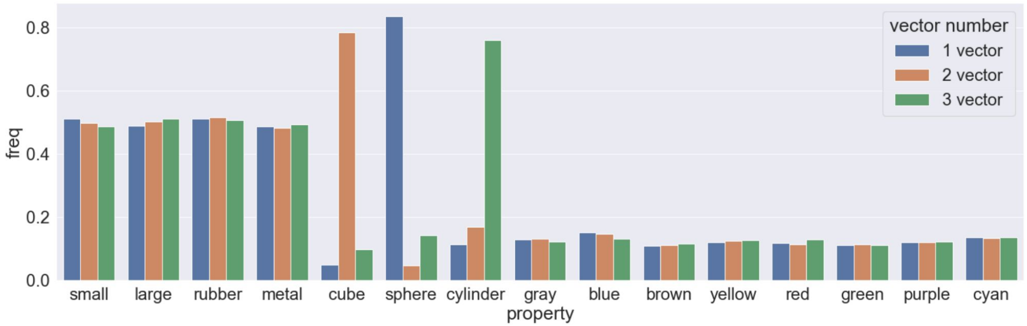
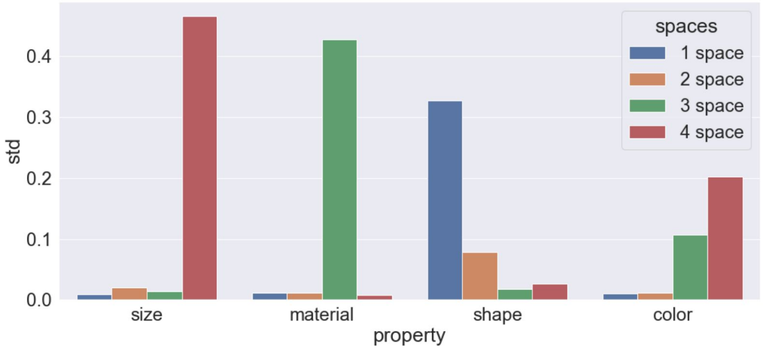


Figure 3: Example of $p(prop_k = value_m | e^i_j)$ for embeddings from the first space. The probability is calculated as the frequency of objects with $value_m$ of property $prop_k$ for which the vector e^i_j was sampled.



Conclusion

- GMM-like approach performs much better than Slot Attention (k-Means-like approach)
- It is possible (for synthetic data) to model slot representations with discrete latent variables, in such an architecture multilevel disentanglement is achieved

Future work

- Scale to more complex real-world and texture-rich data
- Representative measure of expressiveness, effectiveness, disentanglement of object-centric representations
- Application of object-centric architectures for RL world models in hierarchical envs (Crafter, NetHack etc.)

Thanks for attention!