COL870 Project
Unsupervised object detection using Slot Attention

Arnav Sudan (2019TT10954), Sanket Gandhi(2017TT10896)

November, 2022

1 Introduction

The ability to decompose complex natural scenes into meaningful object-centric abstractions lies at the core of human perception and reasoning. In the recent culmination of unsupervised object-centric learning, the Slot-Attention module has played an important role. In these small course project, we see possible improvements in modules to enhance the model accuracy. natural scenes. We target the task of generating the object masks given an image but totally unsupervised way.

2 Model

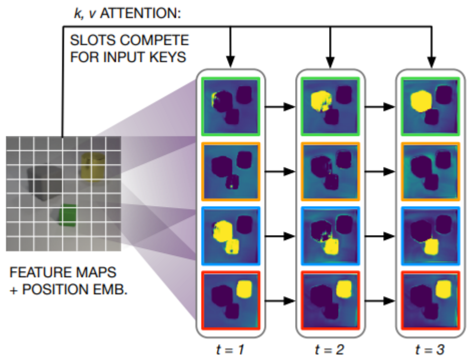
The model consists of the encoder, slot module, and decoder part. The pipeline is encoder-¿ slot module -¿ decoder in both the train and test methods.

2.1 Encoder

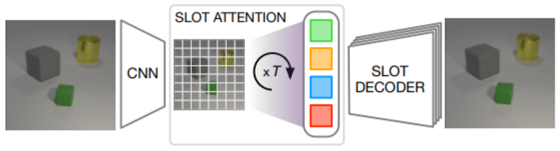
The encoder is a simple CNN with 5 layers. The output and input have the same dimension except for the channels. The channels in the input are three RGB channels and in the output, we have slot dimension output channels. We also apply position encoding to the input.

2.2 Slot Attention

The slot module is just implemented from paper with dimensions equal to 64. The variation has some change in the slot initiation method but the other parts remains same.



(a) Slot Attention module.



(b) Object discovery architecture.

2.3 Decoder

The encoder is a simple up CNN with 6 layers. The output have image size x image size x 4 shape and the input has 4 x 4 x slot dim or 8 x 8 x slot dim dimension depending on the dataset. The decoder generates one image and one mask for each slot.

3 Experiments

We tried the following ideas:

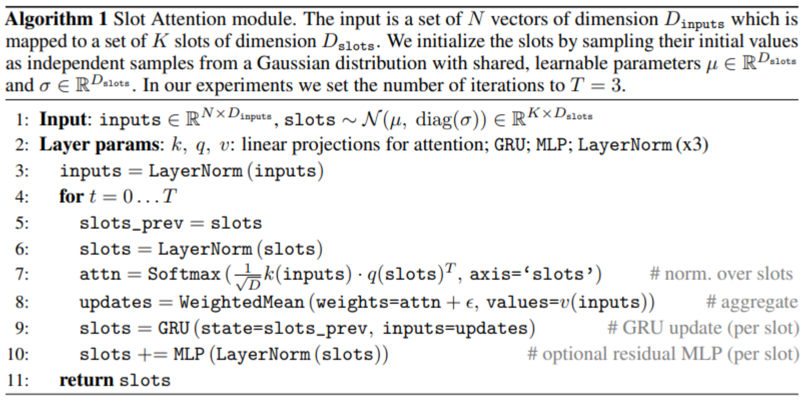
3.1 Auxilary Loss

: We added extra loss along with the reconstruction loss. The auxiliary loss was imposing that no two masks should have the same pixels on.

$$L_{aux}(X) = \sum_{i=0}^{T} \sum_{j=i+1}^{T} M_i(x) * M_j(x)$$

and the final loss becomes.

$$Loss_{final}(X, X_p) = MSE(X, X_p) + L_{aux}(X)$$



Algorithm 1 Slot Attention module. The input is a set of N vectors of dimension $D_{\texttt{inputs}}$ which is mapped to a set of K slots of dimension $D_{\texttt{slots}}$. We initialize the slots by sampling their initial values as independent samples from a Gaussian distribution with shared, learnable parameters $\mu \in \mathbb{R}^{D_{\texttt{slots}}}$ and $\sigma \in \mathbb{R}^{D_{\texttt{slots}}}$. In our experiments we set the number of iterations to $T = 3$.

```
1:  Input: inputs $\in \mathbb{R}^{N \times D_{\texttt{inputs}}}$, slots $\sim \mathcal{N}(\mu,\ \mathrm{diag}(\sigma)) \in \mathbb{R}^{K \times D_{\texttt{slots}}}$
2:  Layer params: $k,\ q,\ v$: linear projections for attention; GRU; MLP; LayerNorm (x3)
3:    inputs = LayerNorm (inputs)
4:    for $t = 0 \ldots T$
5:      slots_prev = slots
6:      slots = LayerNorm (slots)
7:      attn = Softmax ($\frac{1}{\sqrt{D}} k$(inputs) $\cdot\, q$(slots)$^T$, axis='slots')      # norm. over slots
8:      updates = WeightedMean (weights=attn + $\epsilon$, values=$v$(inputs))      # aggregate
9:      slots = GRU (state=slots_prev, inputs=updates)      # GRU update (per slot)
10:     slots += MLP (LayerNorm (slots))      # optional residual MLP (per slot)
11:   return slots
```

3.2 Slot inititation

: We also try to initiate slots with the function of input. So we used the input X and generated the slot invitations as

$$\mu = f_1(X)$$

$$\sigma = f_2(X)$$

$$slots \sim N(\mu, \sigma)$$

Both f_1 and f_2 are neural networks over the last CNN output of the encoder. They both share one CNN layer with each other.

3.3 Vraiable number of Slots

: We also try to make a model which could infer the number of slots in the image. We tried to use RNN over the encoder output and then roll out RNN till it gave less than 0.5 probability of more slots. We used $rollout\ len + 1$ total slot. But we were not able to make a decoder in that case and were stuck in the code.

3.4 Datasets

We used two datasets for training:
1. Clever with mask (15K samples)
2. Multi object room (20K samples)
3. Synthetic Jumping balls

3.5 Training

We trained with Adam optimizer with a learning rate of 0.0005 and batch size 32. The whole model was trained on GPU.

3.6 Results

We didn't get good results for dataset 1 as the background was static and model was not able to capture the foreground.

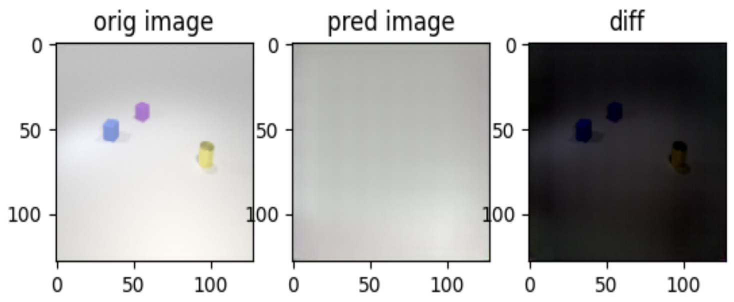


Figure 1: Clever simple

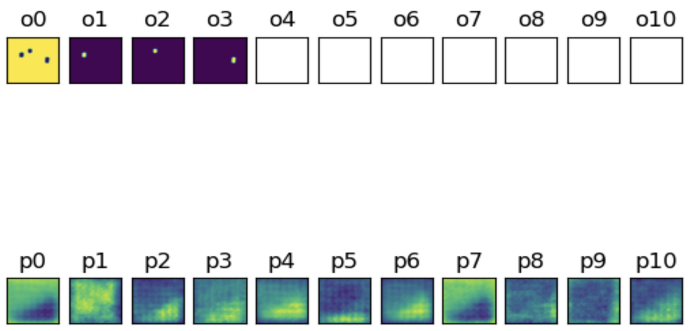


Figure 2: Masks of Clever simple

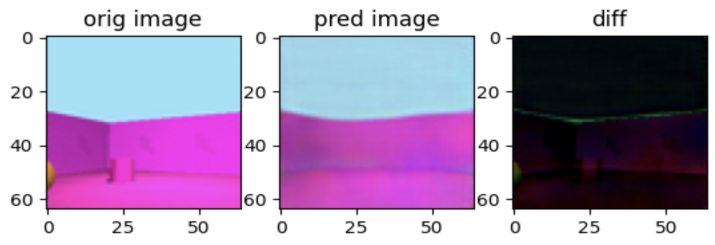


Figure 3: Object room (auxiliary loss)

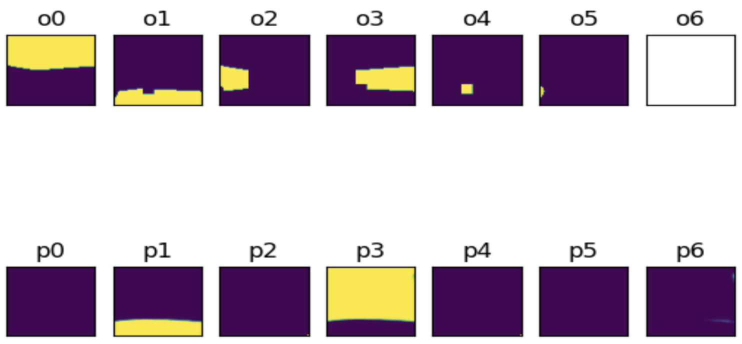


Figure 4: Object room (auxiliary loss)

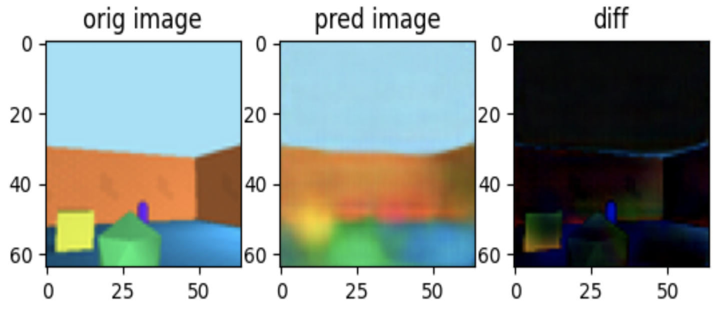


Figure 5: Object room (simple loss)

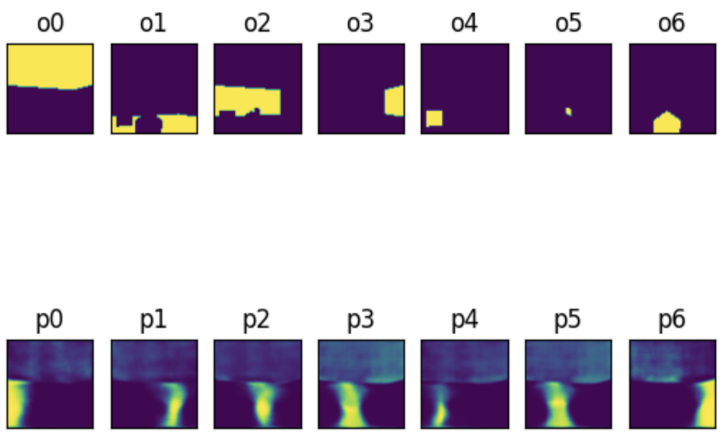


Figure 6: Object room (simple loss)

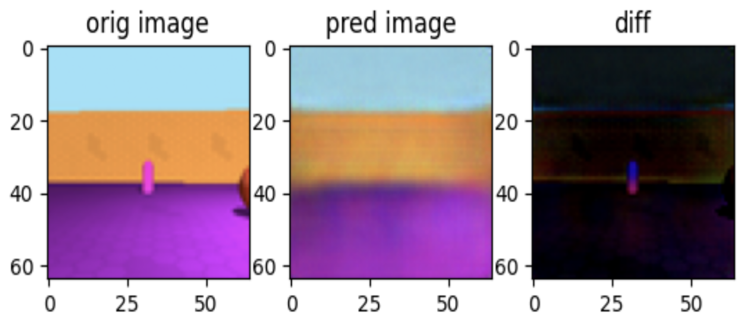


Figure 7: Object room (conditional init)

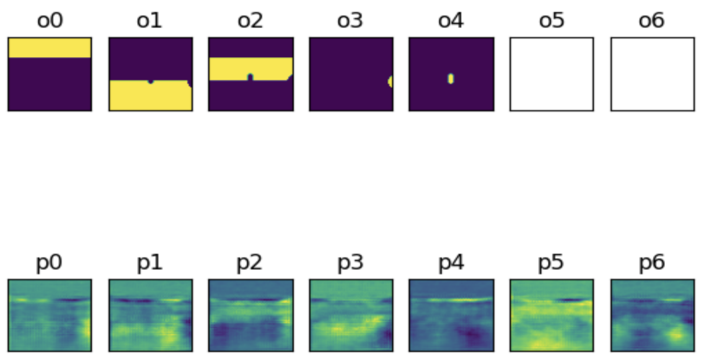


Figure 8: Object room (conditional init)

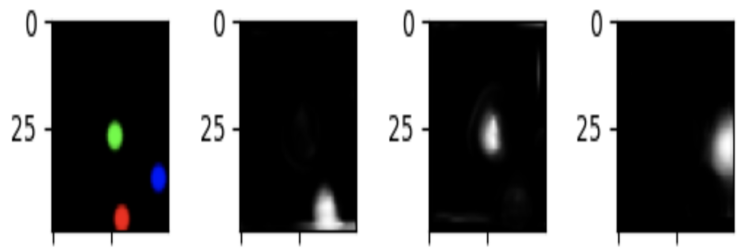


Figure 9: Synthetic data(64 x 64)

Training losses(object room)

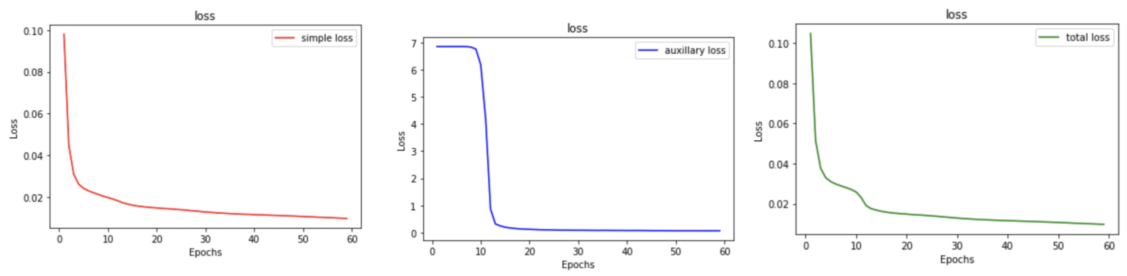


Figure 10: Training Loss

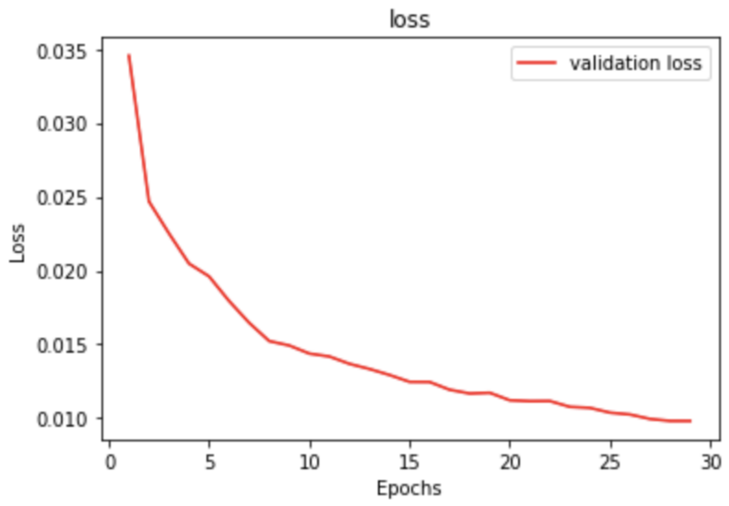


Figure 11: validation Loss