COL775: Slot Attention

Harshit Goyal
2021MT10143

May 8, 2024

1 Part 1

2 Image Preprocessing

2.1 Image

- The CLEVRTex dataset has `RGB` 320×240 pixel images with an `Alpha channe`. We transform the images, as done in the **Slot Attention** paper [1], below described.
- We use `PIL.Image` to convert them to `RGB`. Then take the `Center Crop` of size 192×192.
- This image is then resized using `torchvision.transforms.Resize` with `interpolation` method `InterpolationMode.BILINEAR`
- The images are normalized using mean 0.5 and standard deviation 0.5 for each channel, independently.

2.2 Masks

- The processing is same as that mentioned above, except here, the`Resize` operation uses `InterpolationMode.NEAREST`.
- For the `ARI` (Adjusted Rand Index) score calculation, we'll need to `cluster` the image `pixels` (128×128) into different objects in the picture (and the background), for this the `masks` are used to produce the `ground truth`.
- `InterpolationMode.NEAREST` preserves the number of distinct `RGB` values in the input, hence it is used.

2.3 About the data

- In the given data, each image has atmost 10 objects (foreground) + 1 background. This data will be called `CLEVRTex10`.
- To experiment with more training schemes, encoder architectures and `slot` initialization method, we first filtered `CLEVRTex11` to `CLEVRTex6` i.e. having atmost 6 objects (foreground) + 1 background.
- The filtering described above is dome by counting the number of distinct `RGB` values in in the 320×240 pixels in the `train masks`.

Table 1: Dataset Sizes

Dataset	Train size	Val Size
`CLEVRTex11`	40000	10000
`CLEVRTex6`	21361	5319

3 Architecture

3.1 Overall Architecture

- The overall architecture (D_{inputs} , D_{slots} , MLP architectures) is inspired by the `SAVi` model in the **SlotFormer** paper [3].

$$N = 128 * 128$$
$$D_{inputs} = 128$$
$$D_{slots} = 128$$

- K will be the 7 for `CLEVRTex6` and 11 for `CLEVRTex11`.
- The encoder similar to `ResNet18` and described later.
- `Slots` are initialized as K learnable vectors of size D_{slots}. The problem with this, instead of learning the `mean` and `variance` of a Gaussian is that, we can't change the number of `slots` during inference time.

3.2 Encoder Architecture

- Authors in the **Slot Attention** paper [1] have used a 4 layer deep `CNN` encoder, however data in this paper, was `CLEVR6` which is much simpler than `CLEVRTex6`. We tried this encoding scheme but found that the below described `ResNet18` architecture does much better.
- `kernel_size = 5` for all `Convolution layers` and `stride = 1`. The input image of 128×128 has the same `height` and `width` throughout the `CNN` encoder, we only increase the number of `channels`.
- A single `Convolution` layers increases the number of `channels` from 3 to 16, then over the 6 blocks, the number of `channels` are increased to 32 and then to 64. The feedforward, then increases it to 128 before it's passed into the `SlotAttention` module.

4 Optimizer and Learning Rate Scheduling

We use Adam Optimizer with the defauls settings given in `torch.optim.Adam` documentation here.

```
betas = (0.9 , 0.999)
eps = 1e-08
weight_decay = 0
```

For `scheduling` the `learning rate`, we use `torch.optim.lr_scheduler.CosineAnnealingLR` with settings,

For the first 80 `epochs`, we start with `learning rate` 10^{-3}, ending to 5×10^{-4} and use,

```
T_max = 160
eta_min = 0
last_epoch = -1
```

For the next 180 `epochs`, we start with `learning rate` 3×10^{-4} and use,

```
T_max = 160
eta_min = 0
last_epoch = -1
```

The jump in the `Learning Rate` in Figure 2 is hence explained.

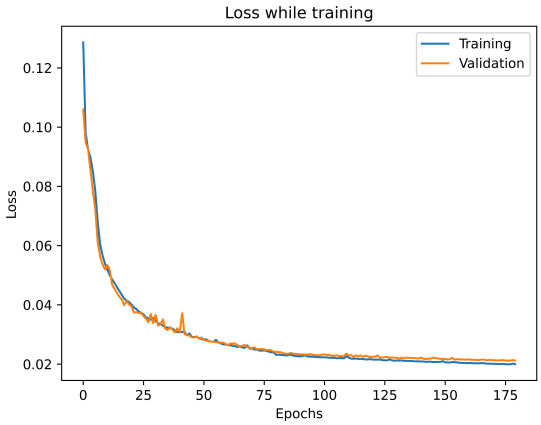

Figure 1: Loss curves

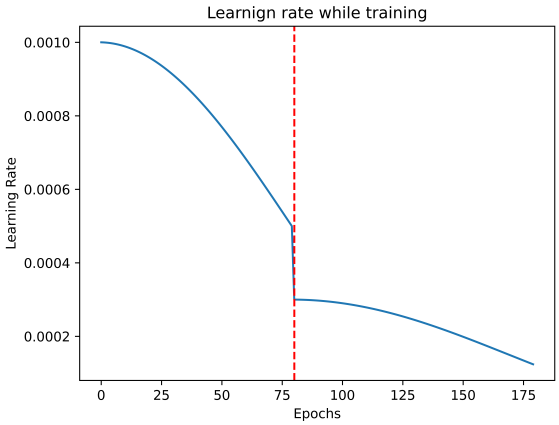

Figure 2: Learning Rate

Figure 3: Slot Attention model

5 Metrics

Metrics are shown in Table 2. The maximum ARI on CLEVRTex6 (and CLEVRTex10 too) was 93.898%. The Top 5 outputs, (by ARI score) are shown in Figure 4.

Dataset	ARI (%)	FID
CLEVRTex6	56.269	143.379
CLEVRTex11	55.447	161.366

Table 2: Metrics on Slot Attention model

6 Slot Composition

1. We show here 2 examples of mixing images, by using some slots from one image and some from other.

2. The first 2 images ($Image_1$ and $Image_2$ respectively) in Figure 5 and Figure 6 are the images used for `mixing` and the rightmost on is produced by mixing `slots`, from the `encodings` of first 2 and then decoding them.

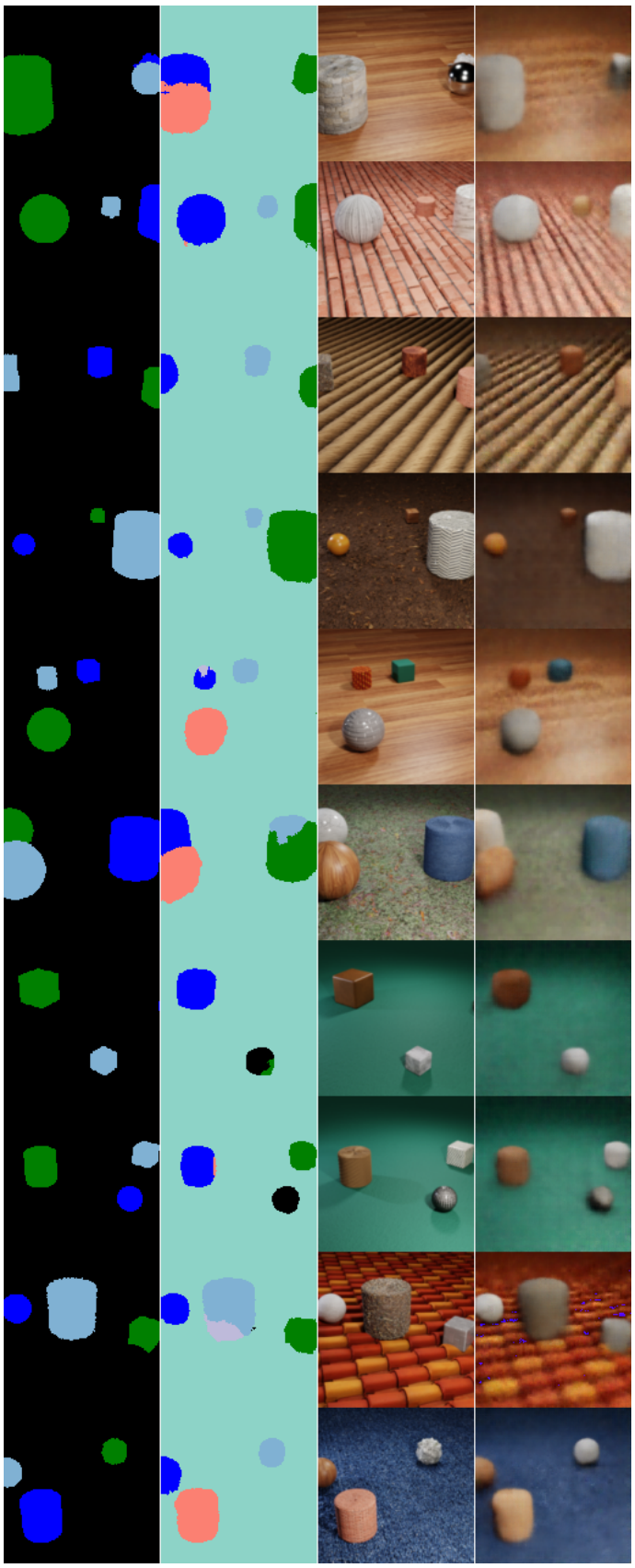

Figure 4: True Mask | Predicted Mask | True Image | Reconstructed Image

3. For reference, Figure 7 and Figure 8 shows respectively the masks generated from the 7 slots for $Image_1$ and $Image_2$

4. Figure 9 shows other generated images after using K-Means clustering to cluster slots and then randomly picking one slot from each cluster to generate the image.

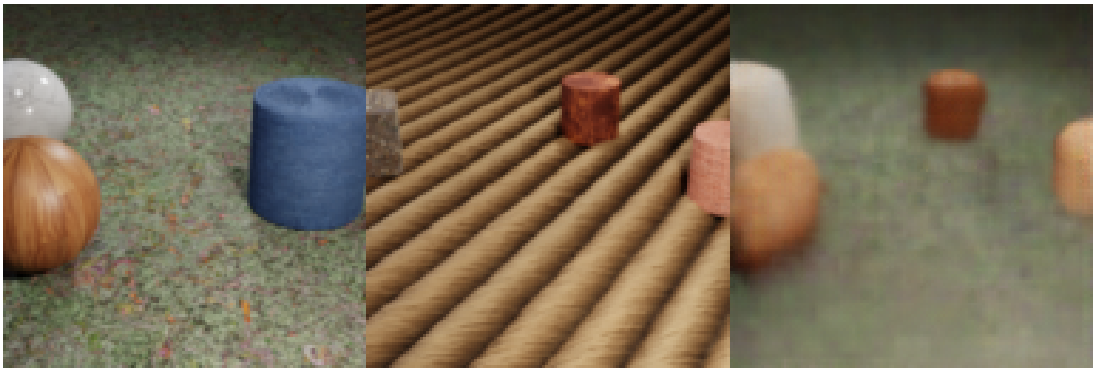

Figure 5: Slot mixing example 1

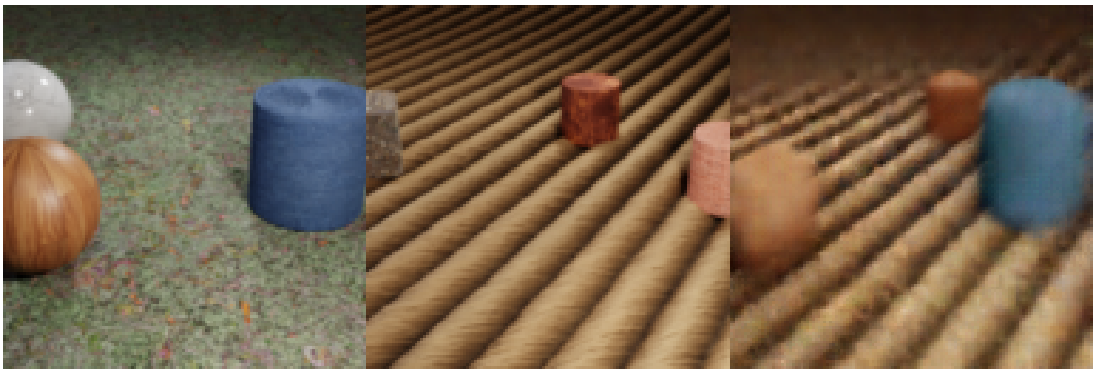

Figure 6: Slot mixing example 2

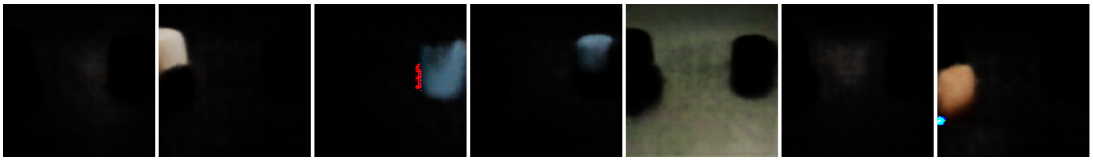

Figure 7: Slot-wise images for $Image_1$

Figure 8: Slot-wise images for $Image_2$

References

[1] Francesco Locatello **andothers**. "Object-Centric Learning with Slot Attention". **in***CoRR*: abs/2006.15055 (2020). arXiv: 2006.15055. URL: https://arxiv.org/abs/2006.15055.

Figure 9: Generated images by randomly picking Slots after K-means clustering

[2] Ziyi Wu **andothers**. *SlotDiffusion: Object-Centric Generative Modeling with Diffusion Models*. 2023. arXiv: 2305.11281 [cs.CV].

[3] Ziyi Wu **andothers**. *SlotFormer: Unsupervised Visual Dynamics Simulation with Object-Centric Models*. 2023. arXiv: 2210.05861 [cs.CV].