# COL775: Assignment 2
# Object-Centric Learning With Slot Attention

Group members:
RISHIT SINGLA - 2021CS10547
S. NIKHIL TEJA - 2021CS10106

May 7, 2024

## Contents

# 1 Object-Centric Learning With Slot Attention

## 1.1 Model Link

The best model trained can be found here Click here

## 1.2 Encoder Architecture

The Encoder module is designed to encode input images into feature vectors.

1. Convolutional Neural Network (CNN) Backbone is constructed using custom ResNet blocks and is used to extract hierarchical features from the input images. consists of a series of convolutional layers followed by Rectified Linear Unit (ReLU) activation functions.

2. A positional embedding is added to the feature maps to provide spatial information about the location of each pixel within the image. Layer normalization is applied to the feature vectors to improve training stability and accelerate convergence.

3. The encoded feature vectors are further processed through an MLP consisting of fully connected layers. The MLP enables the model to perform non-linear transformations and capture complex patterns in the input data.

## 1.3 Slot-Attention Mechanism

### 1.3.1 Finalized Parameters

The best model is trained and obtained for the following parameters:

- **Encoder_input_dimension**: 64
- **Slot_channel_dimension**: 64
- **Projection_dimension**: 64
- **mlp_slot_dimension**: 128
- **number_of_slots**: 11
- **number_of _slot_iterations**: 3

### 1.3.2 Slot Attention Implementation

- **Initialization**: Layer normalization is applied to input and slot dimensions. Slots are initialized randomly using learnable parameters.
- **Projection**: Projection matrices (`project_key`, `project_value`, `project_query`) are used to project input, slots, and queries into the projected space.
- **Slot Update**:
  - The GRU cell updates the slots based on attention-weighted input values.
  - The attention mechanism computes attention weights by taking the dot product of input and query projections.

  - Softmax with temperature adjustment and epsilon for numerical stability are applied.
  - Weights are calculated to update slots, and updates are applied using the GRU cell.
  - Residual MLP is applied to the updated slots.

- **Forward Pass**: The forward pass iteratively updates the slots for a specified number of iterations (`num_iter`), refining their representations to capture object-centric features.

## 1.4 Custom Spatial-BroadCast Decoder

### 1.4.1 Finalized Parameters

The best model is trained and obtained for the following parameters:

- **Slot_channel_dimension**: 64
- **number_of_slots**: 11
- **images_resolution**: (128, 128)
- **Decoder_channel_dimension**: 64

### 1.4.2 Decoder Implementation

- **Overview**: The Spatial Broadcast Decoder is designed to decode slot vectors into 4-channel images, allowing for the reconstruction of object-centric representations in a spatially coherent manner.
- **Grid Generation**: The `create_grid` function generates a grid of size `resolution` with four channels, each representing a gradient in the range $[0, 1]$ for one of the four cardinal directions (up-down, left-right). This grid is constructed using NumPy and then converted into a PyTorch tensor.
- **CNN Decoder**: The decoder consists of a series of convolutional layers:
  1. **Input Transformation**: The input slot vectors are reshaped and tiled across the spatial dimensions of the image.
  2. **Spatial Grid Concatenation**: The spatial grid generated by `create_grid` is concatenated with the tiled slot vectors to incorporate spatial information.
  3. **Convolutional Layers**: The concatenated tensor is passed through convolutional layers, applying spatial transformations and feature extraction.
  4. **Output Splitting**: The output tensor is split into components representing the reconstructed image, individual channel values, and masks.

- **Forward Pass**: During the forward pass:
  1. The input slot vectors are processed to create a tiled representation across the spatial dimensions of the image.

2. The spatial grid is concatenated with the tiled slots to form the input to the CNN decoder.
3. The CNN decoder processes the input to produce the reconstructed image, individual channel values, masks indicating slot contributions, and the original slot vectors.

## 1.5 Training and Outputs

The training process involved optimizing the network parameters to minimize the reconstruction loss while maximizing the discriminate power of the slot representations. During training, the network was fed with input images, and the reconstructed images were compared with the ground-truth images to compute the loss using MSE loss.

### 1.5.1 Training Loss Curves

The training loss curves depict the variation of the reconstruction loss and other relevant metrics (if any) over the training epochs

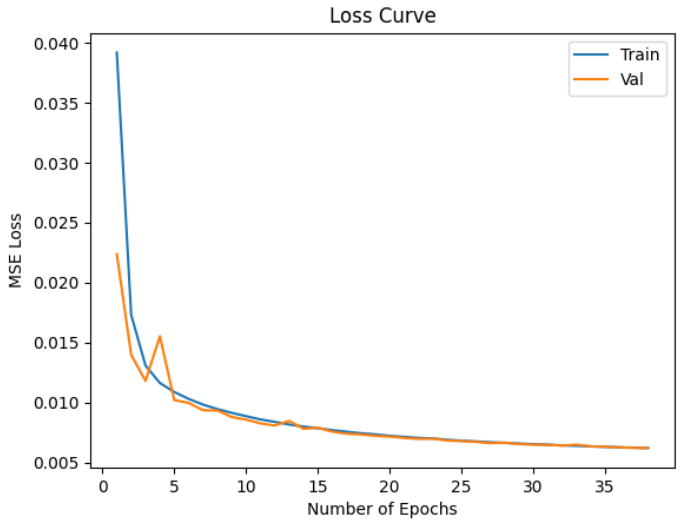


Figure 1: Plot of the effect of loss over the number of epochs for the train and val

### 1.5.2 Adjusted Rand Index (ARI) Score

1. The original and predicted masks are flattened and reshaped to match the format required for ARI calculation.
2. One-hot encoding is applied to both the original and predicted masks to convert them into binary matrices, where each row corresponds to a pixel and each column represents a possible category or slot.
3. ARI is calculated based on the overlap between objects in the original and predicted masks.The number of objects in common between the original and predicted masks is computed using the overlap of their one-hot encoded representations.

Full `ARI score` of our model is **0.681**.

```
Batch: 5000     ARI score: 0.7471817243278026  Time: 24770.945764541626
```

Figure 2: Mean frontier ARI score calculated for 10000 images.

### 1.5.3 Visualizing Generated Outputs

1. The visualize method iterates through the training data loader without masks, generating and saving visualizations of reconstructed images, original images, individual slot images, and thresholded slot images.

2. It utilizes the trained model to reconstruct images and extract masks, and then saves the visualizations for analysis and evaluation purposes. This method helps in assessing the quality of image reconstructions and the effectiveness of slot-based attention in segmenting objects.

3. It facilitates visual inspection of individual slots and their contributions to the overall image reconstruction process, aiding in understanding the learned representations.

(a) Original Image (b) Regen Image

Figure 3: Comparison of Original and Regenerated Images

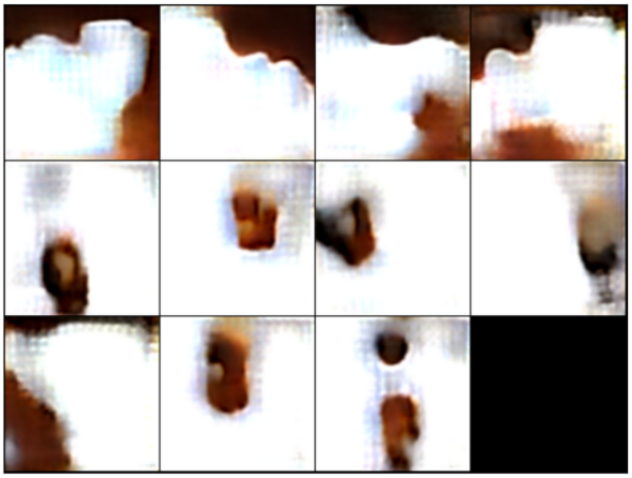

Figure 4: Plot of the images generated for each slot.

Figure 5: Plot of the images generated by using mask as threshold for each slot.

#### 1.5.4 Threshold visual representation

1. The raw images represents the raw output of the model for a specific slot. It shows the image content associated with the corresponding slot without any modification.

2. Contrastly in the Threshold images, the pixel values of the slot image are modified based on a thresholding operation applied to the associated mask. Only the pixels within the mask region corresponding to the slot are retained, while the rest are set to zero. This thresholding process highlights the areas in the image that the model associates with the specific slot, facilitating a **clearer visualization** of the segmented regions.

#### 1.5.5 Attention Map Visualization

We have visualized the attention maps for each iteration in the slot attention algorithm. It is clear from these attention maps that after every iteration the slots attend the object more accurately and for each object there is a slot attending to it. Rest of the slots attend to background.

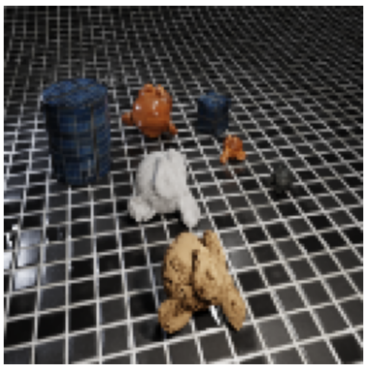

Figure 6: Original Image

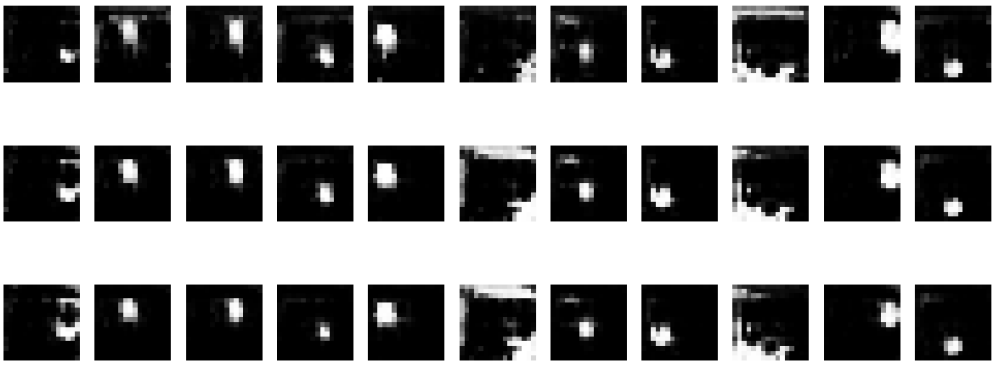

Figure 7: Attention Map, each row corresponds to each iteration

## 1.6 Compositional Generation

### 1.6.1 Generated Outputs Using K-Means

The generated outputs using K-means clustering showcase the network's ability to capture compositional attributes and generate images with diverse object configurations. We sample new slots from each of the cluster to generate an image with compositional attributes as below:

Figure 8: Plot of the images generated by sampling slots from clusters.

Figure 9: Plot of the images generated by sampling slots from clusters.

### 1.6.2 Clean-FID Metric Score

The clean-FID metric score quantitatively evaluates the similarity between the generated images and the ground-truth images using the Frechet Inception Distance (FID) metric. A lower clean-FID score indicates higher similarity and better quality of the generated images compared to the ground-truth images. Our score is 235.67.

```
[5]: score = fid.compute_fid(fdir1, fdir2)

compute FID between two folders
/opt/conda/lib/python3.10/site-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 12 worker processes in total. Our suggested max number of worker in current system is 4, which is smaller than what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
Found 10000 images in the folder /kaggle/input/clevertex/dataset/images/val
FID val : 100%|██████████| 313/313 [00:54<00:00,  5.70it/s]
Found 10000 images in the folder /kaggle/input/results-new/results/slot_attention/gen_img
FID gen_img : 100%|██████████| 313/313 [00:40<00:00,  7.75it/s]

[6]: print(score)

235.6785515980494
```

Figure 10: clean_fid score calculated by generated images with ground truth

# 2 Slot Learning using Diffusion based Decoder

## 2.1 Model Link

The best model trained can be found here Click here

## 2.2 MultiHead Attention and Transformer block

### 2.2.1 Implementation

- **ResBlockUNET**:
    1. This class is a custom implementation of a residual block for a UNET architecture. It includes several modes ('down', 'up', 'same') which correspond to different stages of the UNET (downsampling, upsampling, and bottleneck).
    2. **Down Mode**: This mode is used for the downsampling part of the UNET where the spatial dimensions of the input are reduced while increasing the depth (number of channels). This is achieved using an average pooling layer.
    3. **Up Mode**: This mode is used for the upsampling part of the UNET where the spatial dimensions of the input are increased while reducing the depth. This is achieved using an interpolation function.
    4. **Same mode**: This mode is used for the bottleneck part of the UNET where the spatial dimensions of the input are kept the same. This is achieved using a convolutional layer with 'same' padding.
- **Transformer Block**:
    1. It initializes the layers used in the transformer block. This includes a series of self-attention and cross-attention layers, as well as a feed-forward network.
    2. The **forward** method applies these layers to the input and returns the result. The input is first passed through a normalization layer and a convolutional layer. Then, for each repetition, the input is passed through a self-attention layer, a cross-attention layer, and a feed-forward network. The output of each layer is added to the input of that layer (residual connection). Finally, the result is passed through another convolutional layer.

## 2.3 Latent-Diffusion Decoder Mechanism

### 2.3.1 Finalized Parameters

The best model is trained and obtained for the following parameters:

- **dim_time**: 128
- **diffusion_time**: 1000
- **num_channel**: 64
- **num_repeat**: 6
- **dim_slot**: 64
- **dim_head**: 32

### 2.3.2 Implementation

- **UNET**:
    1. The UNET model used in this class is a U-Net architecture with residual blocks (**ResBlockUNET**) and transformer blocks (**TransformerBlock**).
    2. The U-Net architecture is a type of convolutional neural network that is commonly used for semantic segmentation tasks. It consists of a series of convolutional and max pooling layers to downsample the input, followed by a series of upsampling and convolutional layers to produce the output. There are also several shortcut connections between the downsampling and upsampling blocks.
    3. The residual blocks and transformer blocks are used to capture long-range dependencies in the data.
- **Forward Pass**:
    1. The **forward** method takes in the input data and several optional flags. It first encodes the input data and applies the slot attention mechanism to the encoded data.
    2. If the **attn** flag is set, it returns the attention map. If the **slot_only** flag is set, it returns the slots. If the **generate** flag is set, it generates an image from the slots and returns it.
    3. Otherwise, it encodes the input data using the VAE, adds noise to the encoded data, and decodes the noised data using the diffusion decoder. It then returns the noise and the decoded noise.
- The **var_schedule** method generates a schedule for the variance of the noise in the diffusion process. It first creates an array of beta values, then calculates the corresponding alpha and alpha bar values. The beta, alpha, and alpha bar values are then stacked and returned.
- The **time_embedding** method generates a time embedding for each time step in the diffusion process. It first creates an array of evenly spaced values, then uses a sinusoidal positional encoding scheme to generate the time embedding. The resulting time embedding is returned.
- The **generate** method generates an image from the slots. It first initializes a random tensor, then iteratively updates this tensor using the diffusion decoder and the variance schedule. Finally, it decodes the tensor using the VAE and returns the result.

## 2.4 Training and Outputs

The training process involved optimizing the network parameters to minimize the reconstruction loss while maximizing the discriminate power of the slot representations. This function is responsible for training the model over a specified number of epochs. It uses the Adam optimizer and a learning rate scheduler, and computes the loss using the Mean Squared Error (MSE) loss function.

### 2.4.1 Training Loss Curves

The training loss curves depict the variation of the reconstruction loss and other relevant metrics (if any) over the training epochs

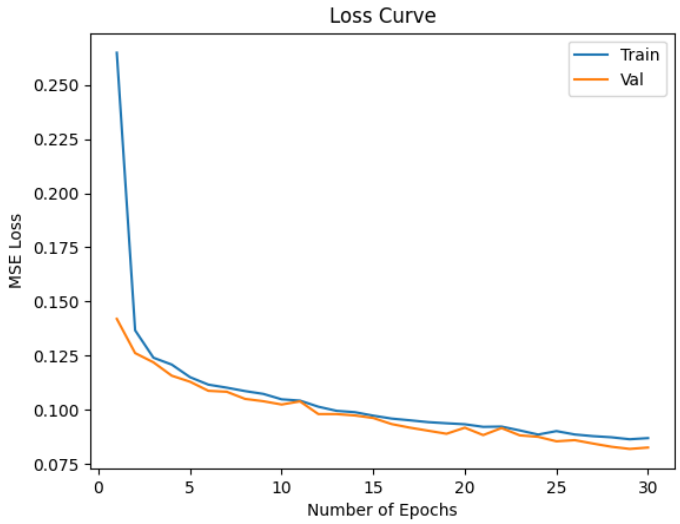


Figure 11: Plot of the effect of loss over the number of epochs for the train and val

### 2.4.2 Adjusted Rand Index (ARI) Score

1. The original and predicted masks are flattened and reshaped to match the format required for ARI calculation.

2. One-hot encoding is applied to both the original and predicted masks to convert them into binary matrices, where each row corresponds to a pixel and each column represents a possible category or slot.

Full `ARI score` of our model is **0.221**.

Frontier `ARI score` of our model is **0.684**.

### 2.4.3 Visualizing Generated Outputs

The visualize method iterates through the training data loader without masks, generating and saving visualizations of reconstructed images and original images.

(a) Original Image (b) Regen Image

(c) Original Image (d) Regen Image

Figure 12: Comparison of Original and Regenerated Images

### 2.4.4 Attention Map Visualization

We have visualized the attention maps for each iteration in the slot attention algorithm. It is clear from these attention maps that after every iteration the slots attend the object more accurately and for each object there is a slot attending to it. Rest of the slots attend to background.

Figure 13: Original Image

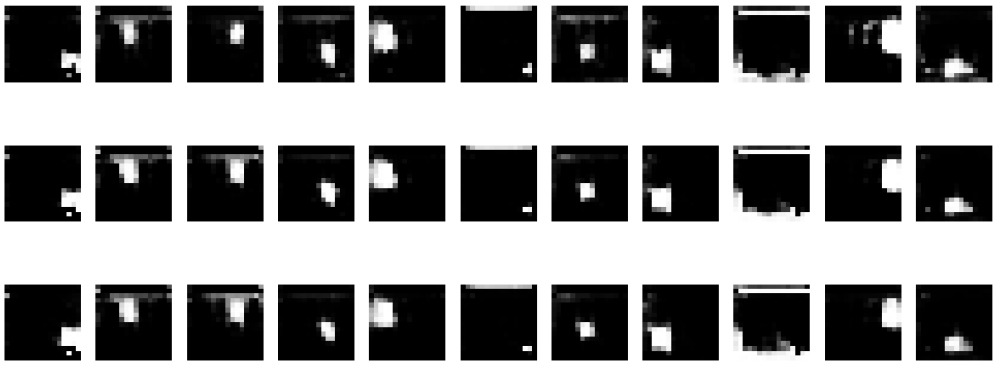

Figure 14: Attention Map, each row corresponds to each iteration

## 2.5 Compositional Generation

### 2.5.1 Generated Outputs Using K-Means

The generated outputs using K-means clustering showcase the network's ability to capture compositional attributes and generate images with diverse object configurations. We sample new slots from each of the cluster to generate an image with compositional attributes as below:

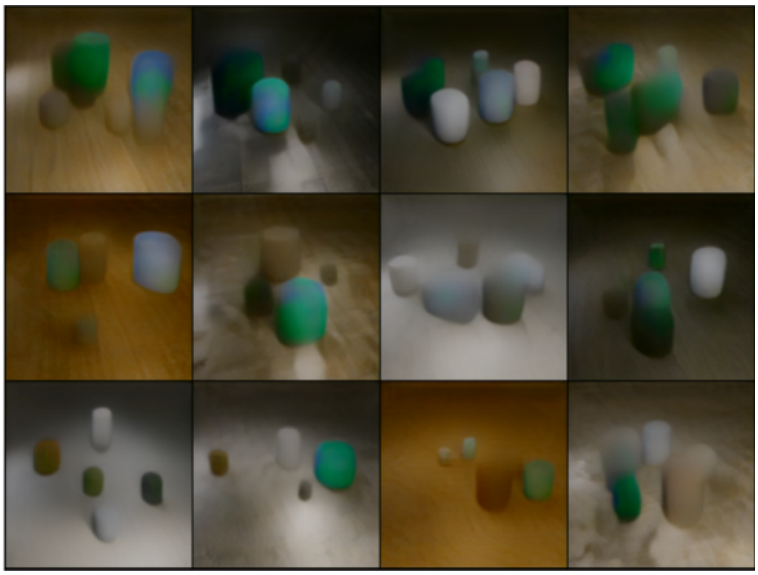

Figure 15: Plot of the images generated by sampling slots from clusters.

Figure 16: Plot of the images generated by sampling slots from clusters.

### 2.5.2 Clean-FID Metric Score

The clean-FID metric score quantitatively evaluates the similarity between the generated images and the ground-truth images using the Frechet Inception Distance (FID) metric. A lower clean-FID score indicates higher similarity and better quality of the generated images compared to the ground-truth images. Our score is 179.89.

```
Fid score is: 179.8988703513272
```

Figure 17: clean_fid score calculated by generated images with ground truth