Grouped Discrete Representation for Object-Centric Learning

Rongzhen Zhao[1] ✉, Vivienne Wang[1], Juho Kannala[2,3], and Joni Pajarinen[1]

[1] Department of Electrical Engineering and Automation, Aalto University, Finland
{rongzhen.zhao, vivienne.wang, joni.pajarinen}@aalto.fi
[2] Department of Computer Science, Aalto University, Finland
juho.kannala@aalto.fi
[3] Center for Machine Vision and Signal Analysis, University of Oulu, Finland

Abstract. Object-Centric Learning (OCL) aims to discover objects in images or videos by reconstructing the input. Representative methods achieve this by reconstructing the input as its Variational Autoencoder (VAE) discrete representations, which suppress (super-)pixel noise and enhance object separability. However, these methods treat features as indivisible units, overlooking their compositional attributes, and discretize features via scalar code indexes, losing attribute-level similarities and differences. We propose Grouped Discrete Representation (GDR) for OCL. For better generalization, features are decomposed into combinatorial attributes by organized channel grouping. For better convergence, features are quantized into discrete representations via tuple code indexes. Experiments demonstrate that GDR consistently improves both mainstream and state-of-the-art OCL methods across various datasets. Visualizations further highlight GDR's superior object separability and interpretability. The source code is available on https://github.com/Genera1Z/GroupedDiscreteRepresentation.

Keywords: Object-Centric Learning · Variational Autoencoder · Discrete Representation · Channel Grouping.

1 Introduction

Under self or weak supervision, Object-Centric Learning (OCL) [10, 5] represents dense image or video pixels as sparse object feature vectors, known as *slots*. These slots can be used for *set prediction* while their corresponding attention maps for *object discovery* [20]. OCL is bio-plausible, as humans perceive visual scenes as objects for visual cognition, like understanding, reasoning, planning, and decision-making [2, 7, 22]. OCL is versatile, as object-level representations of images or videos fit to tasks involving different modalities [34, 31].

The training signal comes from reconstructing the input. Directly reconstructing input pixels [20, 10] struggles with complex-textured objects. Mixture-based OCL methods [16, 9] reconstruct more object-separable modalities, like optical flow and depth maps. Foundation-based methods [24, 35] use the input's foundation model features as the target. Transformer-based [25, 27] and Diffusion-based

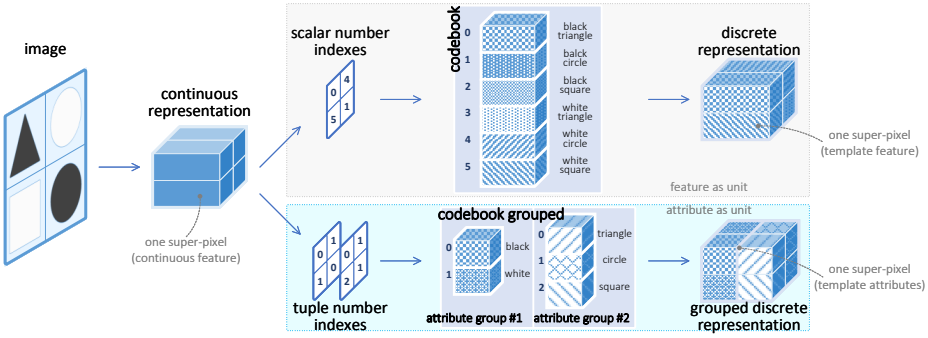

Fig. 1: Non-grouped vs grouped discrete representation. (*upper*) Existing methods treat features as units, selecting template features from a codebook by scalar indexes to discretize superpixels. (*lower*) We treat attributes as units, selecting template attributes from a grouped codebook by tuple indexes.

methods [32, 14] reconstruct the input's Variational Autoencoder (VAE) intermediate representation. With a limited number of shared template features, i.e., codes in a codebook, continuous-valued superpixels in VAE representations are discretized [12, 28]. This suppresses (super-)pixel noise and enhances object separability. Empirically, improved object separability in the reconstruction target offers OCL more effective training guidance.

However, these methods treat features as atomic units and entangle their composing attributes together, thus limiting model generalization. Moreover, the corresponding scalar code indexes fail to capture superpixels' attribute-level similarities and differences, thus hindering model convergence.

As illustrated in Fig. 1, consider a dataset characterized by two attribute groups: color (black, white) and shape (triangle, square, circle). An image in it contains four objects, each downsampled to a superpixel in the feature map. To select template features from a feature-level codebook, six scalar code indexes are needed, where digits 0-5 refer to black-triangle, black-circle, black-square, etc. Each code is reused with probability $\frac{1}{6}$. The feature map can thus be discretized as $\left[\begin{smallmatrix}0 & 4\\ 5 & 1\end{smallmatrix}\right]$. But if decomposed, superpixels can be discretized as combinations of template attributes from two attribute groups, i.e., $\left[\begin{smallmatrix}0,0 & 1,1\\ 1,2 & 0,1\end{smallmatrix}\right]$. The first and second numbers in these index tuples indicate whether superpixels' attributes are the same or different, facilitating model convergence. These codes are reused with higher probabilities $\frac{1}{2}$ and $\frac{1}{3}$ respectively, benefiting model generalization.

Our main contributions are as follows: (*i*) We propose *Grouped Discrete Representation* (GDR) for VAE discrete representation to guide OCL training better; (*ii*) GDR is compatible with mainstream OCL methods and boosts both their convergence and generalization; (*iii*) GDR captures attribute-level similarities and differences, also enhances object separability in VAE representations.

2 Related Work

Object-Centric Learning (OCL). Mainstream OCL obtains supervision from reconstruction using slots aggregated by SlotAttention [20, 1] from the input's dense superpixels. SLATE [25] and STEVE [27], which are Transformer-based, generate input tokens from slots via a Transformer decoder [29], guided by dVAE [12] discrete representations. SlotDiffusion [32] and LSD [14], which are Diffusion-based, recover input noise from slots via a Diffusion model [23], guided by VQ-VAE [28] discrete representations. DINOSAUR [24] and VideoSAUR [35], which are foundation model-based, reconstruct input features from slots via a spatial broadcast decoder [30], guided by well-pretrained features of the foundation model DINO [6, 21]. We focus on the VAE part of OCL.

Variational Autoencoder (VAE). Discrete representations of VAE have been shown to guide OCL better than direct input pixels as reconstruction targets. Transformer-based OCL methods [25, 27] utilize dVAE [12] to discretize encoder representations by selecting template features from a codebook via Gumbel sampling [13]. Diffusion-based OCL methods [14, 32] employ VQ-VAE [28] to achieve discretization by replacing features with their closest codebook codes. Similar to our idea, both [4] and [19] seek to decompose features into attributes, but their monolithic VAE representation is incompatible with OCL. Other VAE variants also offer techniques, like grouping [33], residual [3] and clustering [18], to enhance VAE representations. We borrow some for the OCL setting.

Channel Grouping. Splitting features along the channel dimension and transforming them separately is often used to diversify representations [17, 8, 11, 37, 36]. These solutions mainly perform grouping directly on feature maps [17] or on learnable parameters [36]. To the best of our knowledge, only one work has explored this idea in the OCL setting. SysBinder [26] groups slots along the channel dimension in the slot attention [20] to aggregate different attributes of objects, yielding better interpretability in object representation but limited performance gains. We group VAE intermediate representations along channels, yielding grouped discrete representations to guide OCL training better.

3 Proposed Method

We propose Grouped Discrete Representation (GDR), applicable to mainstream OCL methods, either Transformer-based [25, 27, 15, 35] or Diffusion-based [32, 14]. Simply modifying their VAE, our GDR improves them by providing reconstruction targets, or *guidance*, with better object separability.

Notations: As shown in Fig. 2, image or video frame $\boldsymbol{I}$, continuous representation $\boldsymbol{Z}$, discrete representation $\boldsymbol{X}$, and noise $\boldsymbol{N}$ are tensors in shape (height, width, channel); queries $\boldsymbol{Q}$ and slots $\boldsymbol{S}$ are tensors in shape (number-of-slots, channel); segmentation $\boldsymbol{M}$ is a tensor in shape (height, width).

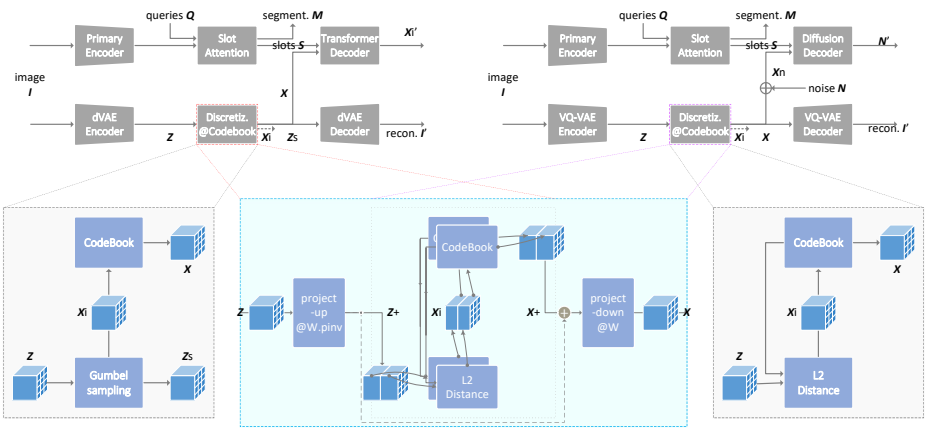

Fig. 2: Our GDR is applicable to mainstream OCL. First row: architectures of Transformer-based (*left*) and Diffusion-based (*right*) methods. Second row: non-grouped representation discretization in dVAE (*left*), non-grouped discretization in VQ-VAE (*right*), and grouped discretization (*center*) of our method.

3.1 Preliminary: Discrete Representation

Both Transformer-based and Diffusion-based methods learn to aggregate pixels into *slots* by reconstructing the input as its VAE discrete representation.

Transformer-based architecture is depicted in Fig. 2 first row left. The input image or video frame $\boldsymbol{I}$ is encoded by a primary encoder and aggregated by SlotAttention [20] into slots $\boldsymbol{S}$ under queries $\boldsymbol{Q}$, with object (and background) segmentation masks $\boldsymbol{M}$ as byproducts. Meanwhile, pretrained VAE represents $\boldsymbol{I}$ as discrete representation $\boldsymbol{X}$ and the corresponding code indexes $\boldsymbol{X}_{\mathrm{i}}$. Subsequently, using a Transformer decoder, $\boldsymbol{S}$ is tasked with reconstructing $\boldsymbol{X}_{\mathrm{i}}$ as classification, guided by causal-masked $\boldsymbol{X}$. For videos, current slots $\boldsymbol{S}$ are transformed by a Transformer encoder block into queries for the next frame.

Specifically, discrete representations for Transformer-based OCL are obtained as shown in Fig. 2 second row left:

- Predefine a codebook $\boldsymbol{C}$ containing n c-dimensional learnable codes as template features;
- Transform input $\boldsymbol{I}$ with a dVAE encoder into continuous intermediate representation $\boldsymbol{Z}$;
- Sample $\boldsymbol{Z}$ via Gumbel softmax, yielding one-hot indexes $\boldsymbol{X}_{\mathrm{i}}$ and soft sampling $\boldsymbol{Z}_{\mathrm{s}}$ for dVAE decoding;
- Select template features from $\boldsymbol{C}$ by $\boldsymbol{X}_{\mathrm{i}}$ and compose the discrete representation $\boldsymbol{X}$ to guide OCL training.

Diffusion-based architecture is drawn in Fig. 2 first row right. The key difference is that, with a conditional Diffusion model decoder, $\boldsymbol{S}$ is tasked with reconstructing Gaussian noise $\boldsymbol{N}$ added to $\boldsymbol{X}$ as regression.

Specifically, discrete representations for Diffusion-based OCL are obtained as in Fig. 2 second row right:

- Predefine a codebook $\boldsymbol{C}$ containing n learnable codes as template features;
- Transform input $\boldsymbol{I}$ via VQ-VAE encoder into continuous representation $\boldsymbol{Z}$;
- Find the closest codes' indexes $\boldsymbol{X}_{\mathrm{i}}$ in $\boldsymbol{C}$ for each superpixel in $\boldsymbol{Z}$;
- Form discrete representation $\boldsymbol{X}$ by selecting $\boldsymbol{C}$ using $\boldsymbol{X}_{\mathrm{i}}$, for OCL training.

Remark. These methods' features as discretization units overlooks the composing attributes, thus impeding generalization. Their scalars as code indexes loses sub-feature similarities and differences, thus hindering convergence.

3.2 Naive Grouped Discrete Representation

Our naive GDR decomposes features into attributes via direct channel grouping in VQ-VAE for both Transformer- and Diffusion-based methods.

Beforehand, suppose a dataset is fully described by n c-dimensional template features, which are further decomposed into g attribute groups. Each group consists of a d-dimensional template attributes, $n = a^g$ and $c = g \times d$. Thus, we predefine a set of attribute codebooks $\boldsymbol{C} = \{\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)} ... \boldsymbol{C}^{(g)}\}$, whose parameters are in shape (g, a, d). The combinations of these codes are equivalent to the non-grouped feature-level codebook, whose parameters are in shape (n, c).

Afterwards, we transform the input $\boldsymbol{I}$ with a VAE encoder into continuous intermediate representation $\boldsymbol{Z}$. In VQ-VAE, we sample distances between $\boldsymbol{Z}$ and $\boldsymbol{C}$ via Gumbel noise, yielding tuple code indexes $\boldsymbol{X}_{\mathrm{i}}$:

$$\boldsymbol{D} = \mathrm{l2}(\boldsymbol{Z}^{(1)}, \boldsymbol{C}^{(1)}) \circ \mathrm{l2}(\boldsymbol{Z}^{(2)}, \boldsymbol{C}^{(2)}) ... \circ \mathrm{l2}(\boldsymbol{Z}^{(g)}, \boldsymbol{C}^{(g)}) \tag{1}$$

$$\boldsymbol{D}_{\mathrm{s}} = \mathrm{softmax}(\frac{\boldsymbol{D}^{(1)} + \boldsymbol{G}^{(1)}}{\tau}) \circ \mathrm{softmax}(\frac{\boldsymbol{D}^{(2)} + \boldsymbol{G}^{(2)}}{\tau}) ... \circ \mathrm{softmax}(\frac{\boldsymbol{D}^{(g)} + \boldsymbol{G}^{(g)}}{\tau}) \tag{2}$$

$$\boldsymbol{X}_{\mathrm{i}} = \mathrm{argmin}(\boldsymbol{D}_{\mathrm{s}}^{(1)}) \circ \mathrm{argmin}(\boldsymbol{D}_{\mathrm{s}}^{(2)}) ... \circ \mathrm{argmin}(\boldsymbol{D}_{\mathrm{s}}^{(g)}) \tag{3}$$

where $\boldsymbol{Z}^{(1)} ... \boldsymbol{Z}^{(g)}$ are channel groupings of $\boldsymbol{Z}$; $\boldsymbol{G}^{(1)} ... \boldsymbol{G}^{(g)}$ are Gumbel noises; $\circ$ is channel concatenation; $\mathrm{l2}(\cdot, \cdot)$ denotes L2 distances between every vector pair in its two arguments; $\boldsymbol{D}_{\mathrm{s}}$ is soft Gumbel sampling of distances $\boldsymbol{D}$ between continuous representations and codes; $\mathrm{argmin}(\cdot)$ is along the code dimension. For our grouped VAE, multiple code indexes are selected from the attribute groups, forming "tuple indexes". In contrast, the non-grouped VAE selects only one index from a feature-level codebook, forming "scalar indexes".

Subsequently, we select template attributes by $\boldsymbol{X}_{\mathrm{i}}$ from $\boldsymbol{C}$, forming grouped discrete representation $\boldsymbol{X}$, which is the target of Diffusion decoding:

$$\boldsymbol{X} = \mathrm{select}(\boldsymbol{C}^{(1)}, \boldsymbol{X}_{\mathrm{i}}^{(1)}) \circ \mathrm{select}(\boldsymbol{C}^{(2)}, \boldsymbol{X}_{\mathrm{i}}^{(2)}) ... \circ \mathrm{select}(\boldsymbol{C}^{(g)}, \boldsymbol{X}_{\mathrm{i}}^{(g)}) \tag{4}$$

where $\mathrm{index}(\cdot, \cdot)$ selects codes from a codebook given indexes.

Finally, we transform $\boldsymbol{X}_{\mathrm{i}}$ from tuple into scalar, which is the target of Transformer decoding:

$$\boldsymbol{X}_{\mathrm{i}} := a^0 \times \boldsymbol{X}_{\mathrm{i}}^{(1)} + a^1 \times \boldsymbol{X}_{\mathrm{i}}^{(2)} + ... + a^{g-1} \times \boldsymbol{X}_{\mathrm{i}}^{(g)} \tag{5}$$

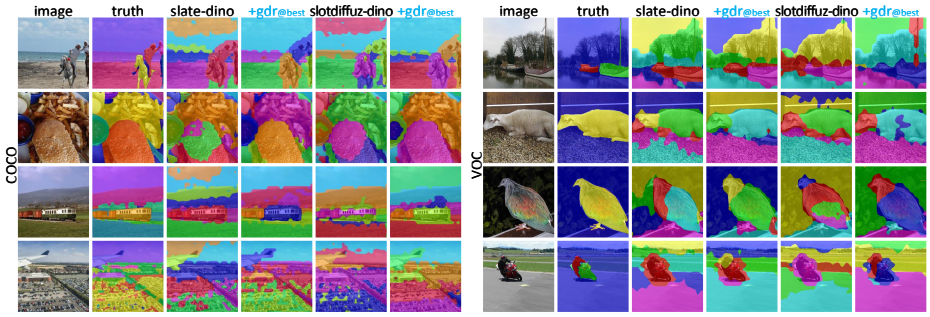

Fig. 3: Object discovery visualization of SLATE and SlotDiffusion plus GDR.

where $\boldsymbol{X}_{\mathrm{i}}^{(1)}...\boldsymbol{X}_{\mathrm{i}}^{(g)}$ are the channel groupings of $\boldsymbol{X}_{\mathrm{i}}$ from Eq. 3.

Besides, we introduce a mild loss to encourage code utilization after grouping

$$l_{\mathrm{u}} = -\mathrm{entropy}(\mathbb{E}[D_{\mathrm{s}}^{(1)}]) - \mathrm{entropy}(\mathbb{E}[D_{\mathrm{s}}^{(2)}])... - \mathrm{entropy}(\mathbb{E}[D_{\mathrm{s}}^{(g)}]) \tag{6}$$

where $\mathbb{E}[\cdot]$ is computed along spatial dimensions while $\mathrm{entropy}(\cdot)$ is computed along the channel dimension.

Remark. As illustrated in Fig. 1, by decomposing features into more reusable attributes, ideally any feature can be represented as a combination of these attributes, thus enhancing generalization. By indexing features with tuples rather than scalars, attribute-level similarities and differences can be captured for better object separability, thus benefiting convergence. Notably, when g=1, the above formulation except Eq. 6 reduces to the original non-grouped VAE.

However, directly grouping feature channels into different attributes may separate channels belonging to the same attribute apart or place channels belonging to different attributes together. This can degrade performance.

3.3 Organizing Channel Grouping

In case incorrect channel grouping, we further design a channel organizing mechanism. The key idea is: We use an *invertible projection* to organize the channel order of the continuous representation for grouped discretization, then apply this projection again to recover the (discretized) representation.

Firstly, we project continuous representation $\boldsymbol{Z}$ to a higher channel dimension using the pseudo-inverse of a learnable matrix $\boldsymbol{W}$:

$$\boldsymbol{Z}_{+} = \boldsymbol{Z} \cdot \mathrm{pinv}(\boldsymbol{W}) \tag{7}$$

where $\boldsymbol{Z}$ is in shape (height, width, channel=c); $\mathrm{pinv}(\cdot)$ is pseudo-inverse; and matrix $\mathrm{pinv}(\boldsymbol{W})$ is in shape (channel=c, expanded channel=$8c$). This facilitates channels belonging to the same attribute to be placed together by (i) enabling channel reordering and (ii) generating extra channels to mitigate mis-grouping.

Secondly, we group $\boldsymbol{Z}_{+}$ along the channel dimension and discretize it using the attribute-level codebooks $\boldsymbol{C}$. This yields code indexes $\boldsymbol{X}_{\mathrm{i}}$ and the expanded discrete representation $\boldsymbol{X}_{+}$. This follows the formulation in Eq. 1-5 above.

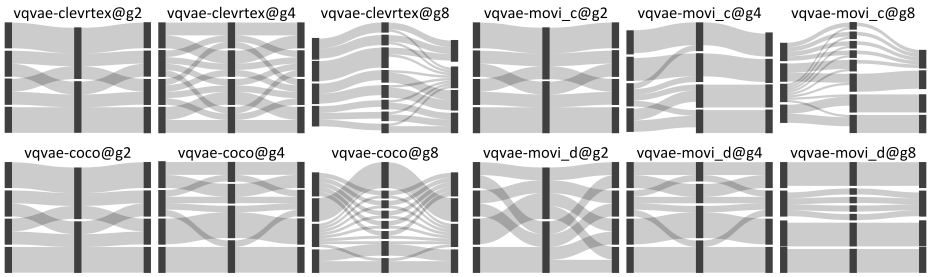

Fig. 4: GDR's invertible projection learns to organize channels' orders for grouped discretization. Every sub-plot has three columns of channels (black bars) and matrix weights among them (grey ribbons). The first column corresponds to continuous representation channels. Ribbons between the first and second columns are the project-up weights. The second column is discretization attribute groups. Ribbons between the second and third columns are the project-down weights. The third column is discretized representation channels.

Meanwhile, we add $\boldsymbol{Z}_+$ to $\boldsymbol{X}_+$:

$$\boldsymbol{X}_+ := \boldsymbol{Z}_+ \times \alpha + \boldsymbol{X}_+ \times (1 - \alpha) \tag{8}$$

where α decays via cosine annealing[4] from 0.5 to 0 in the first half of pretraining and is zeroed 0 afterward. With such residual preserving information through the discretization, VAE can be well pretrained even under mis-grouping.

Thirdly, we project $\boldsymbol{X}_+$ back to obtain the final organized grouped discrete representation:

$$\boldsymbol{X} = \boldsymbol{X}_+ \cdot \boldsymbol{W} \tag{9}$$

where $\boldsymbol{W}$ is the previously introduced learnable matrix in shape (expanded channel=$8c$, channel=c).

Fourthly, to address potential numerical instability arising from matrix pseudo-inverse multiplcation, we normalize $\boldsymbol{X}$:

$$\boldsymbol{X} := \frac{\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}]}{\sqrt{\mathbb{V}[\boldsymbol{X}] + \epsilon}} \tag{10}$$

where $\mathbb{E}$ and $\mathbb{V}$ are the mean and variance over height, width and channel.

3.4 Grouped vs Non-Grouped

Codebook parameters. Compared to the non-grouped, the number of parameters in our grouped codebook is significantly reduced to $\frac{agd}{a^g c} = \frac{ac}{a^g c} = \frac{1}{a^{g-1}}$. E.g., only $\frac{1}{64}$ when a=64, g=2, c=256 and a^g=4096. We increase c to $8c$ and apply normalization plus linear to project it back to c, yielding $\frac{1}{1.6}$ the original number of codebook parameters – still 30% fewer.

[4] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingLR.html

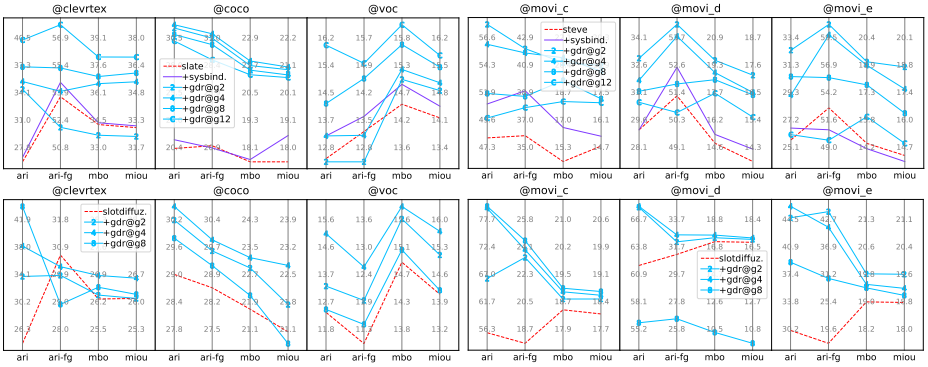

Fig. 5: GDR boosts object discovery performance of both Transformer- (*top*) and Diffusion-based (*bottom*) methods on images (*left*) and videos (*right*). A naive CNN is used as their primary encoder. Titles are datasets; x ticks are metrics while y ticks are metric values in adaptive scope. Higher values are better.

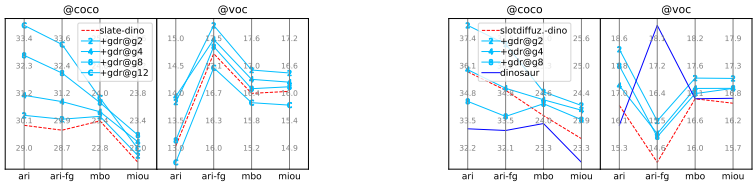

Fig. 6: With DINO1-B/8 for primary encoding, GDR still improves Transformer- (*left*) and Diffusion-based (*right*) methods. Higher values are better.

Codebook computation. Non-grouped computation only involves code matching using inner product for each continuous feature: $c \times n \times 1{=}256 \times 4096{=}2^{20}$. GDR computation involves two projections and code matching: $8c \times c \times 2 + 8c \times \sqrt[g]{n}$, which results in $2^{20} + 2^{17}$ for $g2$ and $2^{20} + 2^{14}$ for $g4$ – computation burden that is nearly identical to the original non-grouped case.

4 Experiments

We conduct experiments using three random seeds to evaluate: (*i*) How well GDR improves mainstream OCL, including Transformer- and Diffusion-based methods; (*ii*) What visual intuitions GDR exhibits in VAE representation; (*iii*) How designs of GDR contribute to its success in the OCL setting.

4.1 Experiment Overview

Models. We use both Transformer-based and Diffusion-based models as our GDR's basis. The former includes SLATE [25] for image and STEVE [27] for video. The latter includes SlotDiffusion [32] and its temporal variant. Upon such basis, we

COCO #slots=7	ARI_{fg}↑	mBO↑	YTVIS #slots=7	ARI_{fg}↑	mBO↑
SPOT	$37.5_{\pm 0.6}$	$34.8_{\pm 0.1}$	VideoSAUR	$39.5_{\pm 0.6}$	$29.0_{\pm 0.4}$
SPOT+GDR@$g2$	$39.7_{\pm 0.5}$	$35.1_{\pm 0.1}$	VideoSAUR+GDR@$g2$	$43.6_{\pm 0.5}$	$31.7_{\pm 0.4}$

COCO #slots=7	class@top1↑	bbox@R2↑
SPOT + MLP	$0.59_{\pm 0.1}$	$0.54_{\pm 0.1}$
SPOT+GDR@$g2$ + MLP	$0.62_{\pm 0.1}$	$0.56_{\pm 0.1}$

Table 1: Object discovery (*upper*) and set prediction (*lower*) of GDR upon state-of-the-arts, SPOT and VideoSAUR. DINO1-B/8 is used for primary encoding.

compare GDR against SysBinder@$g4$ [26]. We also apply GDR to state-of-the-art models, SPOT [15] and VideoSAUR [35], which are also Transformer-based. Methods such as SA [20] and SAVi [16] are excluded due to their low performance or reliance on additional modalities.

Datasets. We evaluate those models on ClevrTex[5], COCO[6] and VOC[7] for image OCL tasks, while MOVi-C/D/E[8] for video. We also use YTVIS[9], YouTube video instance segmentation. These encompass both synthetic and real-world cases, featuring multiple objects and complex textures. Except for those two state-of-the-arts, the input size is unified to 128×128 and other data processing follows the convention. Note that we use COCO panoptic instead of instance segmentation and the high-quality YTVIS[10] for strict evaluation.

Hyperparameters. The codebook size is $n{=}a^g{=}4096$ for all. GDR's group number is set to GDR@$g2$, $g4$, $g8$ and $g12$. Correspondingly, the attribute group sizes are (64, 64), (8, 8, 8, 8), (2, 2, 2, 2, 4, 4, 4, 4) and (2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2), ensuring 4096 combinations. The number of slots is roughly set to the maximum/average object count plus one: 10+1 for ClevrTex, COCO, VOC and MOVi-C; 20+1 for MOVi-D; and 23+1 for MOVi-E. However, for those two state-of-the-arts, we strictly follow their official experiment settings.

4.2 Performance

Object discovery. We use common object discovery metrics: Adjusted Rand Index (ARI)[11], ARI foreground (ARI_{fg}), mean Best Overlap (mBO)[12] and mean Intersection-over-Union (mIoU)[13]. As shown in Fig. 5, GDR significantly enhances accuracy across both synthetic and real-world images and videos. With the naive

[5] https://www.robots.ox.ac.uk/~vgg/data/clevrtex
[6] https://cocodataset.org/#panoptic-2020
[7] http://host.robots.ox.ac.uk/pascal/VOC
[8] https://github.com/google-research/kubric/tree/main/challenges/movi
[9] https://youtube-vos.org/dataset/vis
[10] https://github.com/SysCV/vmt?tab=readme-ov-file#hq-ytvis-high-quality-video-instance-segmentation-dataset
[11] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html
[12] https://ieeexplore.ieee.org/document/7423791
[13] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_score.html

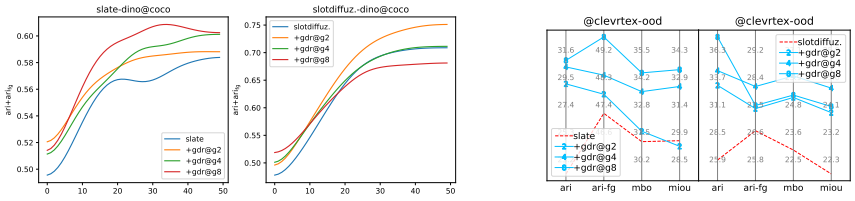

Fig. 7: (*left*) GDR accelerates model convergence. The x axis is val epochs while y is accuracy in ARI+ARI_{fg}. Smoothed with a Gaussian kernel size 5. (*right*) GDR improves model generalization. Models are trained on Clevrtex and tested on its out-of-distribution version. Higher values are better.

CNN [16] for primary encoding, it boosts both Transformer-based methods and Diffusion-based methods. GDR always outperforms the competitor SysBinder by a large margin. We further evaluate GDR's effectiveness with vision foundation model DINO1-B/8 [6] for strong primary encoding. As shown in Fig. 6 and 3, on both SLATE and SlotDiffusion, GDR improves accuracy across all metrics in most cases.

Applying to state-of-the-art. Following the original settings, we apply GDR to SPOT [15] and VideoSAUR [35], by replacing SPOT's VAE with GDR and by replacing VideoSAUR's reconstruction target (continuous DINO features) with GDR discretized DINO features, respectively. As shown in Table 1 upper, GDR is still able to boost state-of-the-art methods' performance further.

Set prediction. Following [24], we employ OCL to represent dataset COCO as slots, and use a small MLP to predict the object class and bounding box corresponding to each slot, with metrics of top-1 accuracy and the R2 score respectively. As shown in Table 1 lower, our GDR improves SLATE in set prediction, demonstrating is superior quality in object representation.

Convergence. The validation curves of ARI+ARI_{fg} in Fig. 7 left demonstrate that GDR consistently accelerates the basis' convergence in OCL training. Along with Fig. 9, forming VAE discrete representation with tuple indexes captures attribute-level similarities and differences among super-pixels, thereby guiding OCL models to converge better.

Generalization. We transfer models from ClevrTex to its out-of-distribution version ClevrTex-OOD without finetuning. As shown in Fig. 7 right, GDR consistently improves basis methods' generalization. This confirms that GDR's decomposition from features into attributes helps the model learn more fundamental representations that are robust to distribution shifts.

4.3 Interpretability

Decomposition from features to attributes. Although without explicit supervision models can hardly learn concepts [26] as human-readable as in Fig. 1, we can still analyze GDR's decomposition from features to attributes as follows. Given GDR discrete representation $\boldsymbol{X}$, we replace the attributes of objects' superpixels with

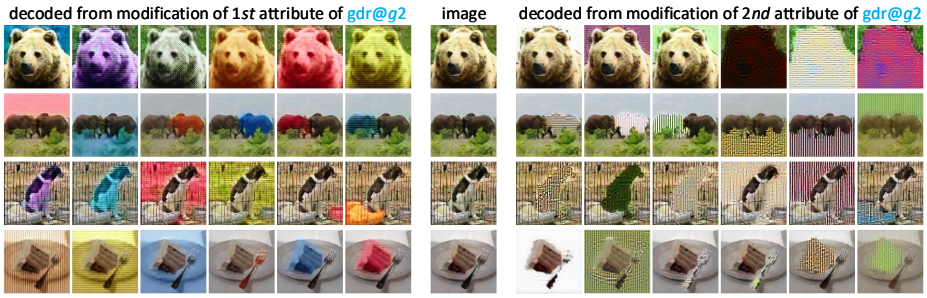

Fig. 8: For GDR@$g2$, one attribute group roughly learns colors, while the other roughly learns textures. The original image is at the center. The left and right are images decoded from the modified VAE discrete representation.

arbitrary attribute codes then decode them into images to observe the changes. As shown in Fig. 8, under $g2$ setting, modifying one attribute group roughly alters the colors, whereas modifying the other destroys the textures. This suggests that the first group learns colors while the second learns textures.

Attribute-level similarities and differences. Basis methods' scalar index tensor and GDR's tuple index tensor $\boldsymbol{X}_{\text{i}}$ can be visualized by mapping different indexes to distinct colors. For our only competitor, SysBinder, we assign different colors to its different attention groups. As shown in Fig. 9, scalar indexes mix all attributes together, whereas our tuple indexes highlight similarities (identical colors) and differences (distinct colors) among superpixels in each attribute group. In contrast, SysBinder also captures such attribute-level information but with very limited diversity and details.

Object separability. The visualization of $\boldsymbol{X}$ for both the basis VQ-VAE and GDR can be achieved by coloring the different distances between each superpixel and the reference point (the average of all superpixels). As shown in Fig. 10, GDR consistently exhibits better object separability across all g settings, suggesting GDR's superior guidance to OCL. However, using an excessive number of groups in GDR may result in the omission of certain objects.

4.4 Ablation

The effects of different designs in GDR are listed in Tab. 2. We use ARI+ARI_{fg} since ARI largely indicates how well the background is segmented while ARI_{fg} reflects the discovery quality of foreground objects.

Number of groups, formulated in Eq. 1-6: g=2, 4, 8 or 12. As shown in Fig. 5 and 6, the optimal g depends on the specific dataset. However, $g12$ and $g8$ tend to result in suboptimal performance while $g4$ consistently leads to guaranteed performance gains over the basis methods.

Channel expansion rate, formulated in Eq. 7: c, $2c$, $4c$ or $8c$. Although $8c$ generally performs best, the expansion rate has a nearly saturated impact on

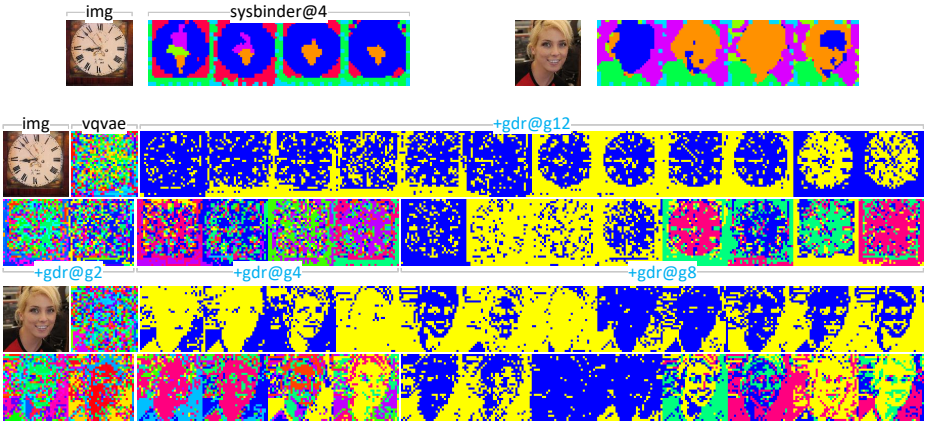

Fig. 9: SysBinder's attribute groups (*upper*), i.e., attention maps, and GDR's attribute groups (*lower*), i.e., tuple code indexes. GDR captures attribute-level similarities and differences among superpixels, whereas the non-grouped "vqvae" mixes all together. SysBinder lacks too much diversity and detail.

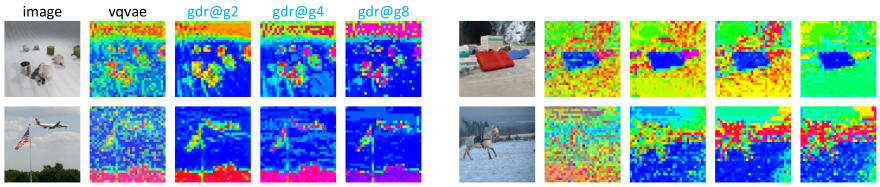

Fig. 10: GDR improves object separability in VAE discrete representation. More groups improve object separability but increase the risk of losing some objects.

GDR's performance. This suggests that our channel organizing mechanism is effective, reducing the necessity for a higher channel expansion rate.

The *channel organizing* based on our *invertible projection* designed in Sect. 3.3 is crucial for OCL model performance. If we disrupt it by replacing W pseudo-inverse in project-up with specified weights, as formulated in Eq. 7, the object discovery accuracy drops significantly.

We also visualize how the invertible project-up and project-down organize channels for grouping. As shown in Fig. 4, some input channels are mixed, switched or split into different attributes for discretization, then the pseudo-inverse recovers them in the form of discrete representations. Such patterns are clearly observed across most datasets and grouping configurations.

Using *annealing residual connection* during training, formulated in Eq. 8, consistently yields better performance than without.

Normalization at last, formulated in Eq. 10, is generally beneficial, though its effect is not highly significant.

expansion rate	$8c$	$4c$	$2c$	$1c$
ARI+ARI_{fg}	89.47	89.29	88.93	88.16

utiliz. loss	invertible project	residual connection	final normaliz.	ARI+ARI_{fg}
✓	✓	✓	✓	89.47
✗				84.78
	✗			81.52
	✗ Wpinv			32.25
		✗		88.84
			✗	89.16

Table 2: Effects of expansion rate, utilization loss, invertible projection (and replacing W pinv with specified weights), residual connection in training and final normalization. Experimented SLATE+GDR@$g4$ on ClevrTex.

5 Conclusion

We propose grouped discrete representation in VAE to guide OCL training better. This technique improves the mainstream Transformer- and Diffusion-based OCL methods in both convergence and generalization. Although self-supervision cannot guarantee different groups learn different human-readable attributes, our method still exhibits interesting and interpretable patterns in attribute-level discrete representations. Fundamentally, we only modify the VAE part of OCL models, indicating broader applicability to other VAE-based models.

Acknowledgment

We acknowledge the support of the Finnish Center for Artificial Intelligence (FCAI) and the Research Council of Finland through its Flagship program. Additionally, we thank the Research Council of Finland for funding the projects ADEREHA (grant no. 353198), BERMUDA (362407) and PROFI7 (352788). We also appreciate CSC-IT Center for Science, Finland, for granting access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking and hosted by CSC (Finland) in collaboration with the LUMI consortium. Furthermore, we acknowledge the computational resources provided by the Aalto Science-IT project through the Triton cluster. Finally, the first author expresses his heartfelt gratitude to his wife for her unwavering support and companionship.

References

1. Bahdanau, D., Cho, K.H., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. International Conference on Learning Representations (2015)
2. Bar, M.: Visual Objects in Context. Nature Reviews Neuroscience **5**(8), 617–629 (2004)

3. Barnes, C., Rizvi, S., Nasrabadi, N.: Advances in Residual Vector Quantization: A Review. IEEE Transactions on Image Processing **5**(2), 226–262 (1996)
4. Bouchacourt, D., Tomioka, R., Nowozin, S.: Multi-Level Variational Autoencoder: Learning Disentangled Representations from Grouped Observations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
5. Burgess, C., Matthey, L., Watters, N., et al.: MONet: Unsupervised Scene Decomposition and Representation. arXiv preprint arXiv:1901.11390 (2019)
6. Caron, M., Touvron, H., Misra, I., et al.: Emerging Properties in Self-Supervised Vision Transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9650–9660 (2021)
7. Cavanagh, P.: Visual Cognition. Vision Research **51**(13), 1538–1551 (2011)
8. Chen, Y., Fan, H., Xu, B., et al.: Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks With Octave Convolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3435–3444 (2019)
9. Elsayed, G., Mahendran, A., Van Steenkiste, S., et al.: SAVi++: Towards End-to-End Object-Centric Learning from Real-World Videos. Advances in Neural Information Processing Systems **35**, 28940–28954 (2022)
10. Greff, K., Kaufman, R.L., Kabra, R., et al.: Multi-Object Representation Learning with Iterative Variational Inference. In: International Conference on Machine Learning. pp. 2424–2433. PMLR (2019)
11. Huang, G., Liu, S., Van der Maaten, L., Weinberger, K.: CondenseNet: An Efficient DenseNet Using Learned Group Convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2752–2761 (2018)
12. Im Im, D., Ahn, S., Memisevic, R., Bengio, Y.: Denoising criterion for variational auto-encoding framework. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31 (2017)
13. Jang, E., Gu, S., Poole, B.: Categorical Reparameterization with Gumbel-Softmax. International Conference on Learning Representations (2017)
14. Jiang, J., Deng, F., Singh, G., Ahn, S.: Object-Centric Slot Diffusion. Advances in Neural Information Processing Systems (2023)
15. Kakogeorgiou, I., Gidaris, S., Karantzalos, K., Komodakis, N.: SPOT: Self-Training with Patch-Order Permutation for Object-Centric Learning with Autoregressive Transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22776–22786 (2024)
16. Kipf, T., Elsayed, G., Mahendran, A., et al.: Conditional Object-Centric Learning from Video. International Conference on Learning Representations (2022)
17. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems **25** (2012)
18. Lim, K.L., Jiang, X., Yi, C.: Deep Clustering with Variational Autoencoder. IEEE Signal Processing Letters **27**, 231–235 (2020)
19. Liu, X., Yuan, J., An, B., Xu, Y., Yang, Y., Huang, F.: C-Disentanglement: Discovering Causally-Independent Generative Factors under an Inductive Bias of Confounder. Advances in Neural Information Processing Systems **36**, 39566–39581 (2023)
20. Locatello, F., Weissenborn, D., Unterthiner, T., et al.: Object-Centric Learning with Slot Attention. Advances in Neural Information Processing Systems **33**, 11525–11538 (2020)
21. Oquab, M., Darcet, T., Moutakanni, T., et al.: DINOv2: Learning Robust Visual Features without Supervision. Transactions on Machine Learning Research (2023)

22. Palmeri, T., Gauthier, I.: Visual Object Understanding. Nature Reviews Neuroscience **5**(4), 291–303 (2004)
23. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022)
24. Seitzer, M., Horn, M., Zadaianchuk, A., et al.: Bridging the Gap to Real-World Object-Centric Learning. International Conference on Learning Representations (2023)
25. Singh, G., Deng, F., Ahn, S.: Illiterate DALL-E Learns to Compose. International Conference on Learning Representations (2022)
26. Singh, G., Kim, Y., Ahn, S.: Neural Systematic Binder. International Conference on Learning Representations (2022)
27. Singh, G., Wu, Y.F., Ahn, S.: Simple Unsupervised Object-Centric Learning for Complex and Naturalistic Videos. Advances in Neural Information Processing Systems **35**, 18181–18196 (2022)
28. Van Den Oord, A., Vinyals, O., Kavukcuoglu, K.: Neural Discrete Representation Learning. Advances in Neural Information Processing Systems **30** (2017)
29. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention Is All You Need. Advances in Neural Information Processing Systems **30** (2017)
30. Watters, N., Matthey, L., Burgess, C., Alexander, L.: Spatial Broadcast Decoder: A Simple Architecture for Disentangled Representations in VAEs. ICLR 2019 Workshop LLD (2019)
31. Wu, Z., Dvornik, N., Greff, K., Kipf, T., Garg, A.: SlotFormer: Unsupervised Visual Dynamics Simulation with Object-Centric Models. International Conference on Learning Representations (2023)
32. Wu, Z., Hu, J., Lu, W., Gilitschenski, I., Garg, A.: SlotDiffusion: Object-Centric Generative Modeling with Diffusion Models. Advances in Neural Information Processing Systems **36**, 50932–50958 (2023)
33. Yang, D., Liu, S., Huang, R., et al.: Hifi-Codec: Group-Residual Vector Quantization for High Fidelity Audio Codec. arXiv preprint arXiv:2305.02765 (2023)
34. Yi, K., Gan, C., Li, Y., Kohli, P., et al.: CLEVRER: CoLlision Events for Video REpresentation and Reasoning. International Conference on Learning Representations (2020)
35. Zadaianchuk, A., Seitzer, M., Martius, G.: Object-Centric Learning for Real-World Videos by Predicting Temporal Feature Similarities. Advances in Neural Information Processing Systems **36** (2024)
36. Zhao, R., Li, J., Wu, Z.: Convolution of Convolution: Let Kernels Spatially Collaborate. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 651–660 (2022)
37. Zhao, R., Wu, Z., Zhang, Q.: Learnable Heterogeneous Convolution: Learning both Topology and Strength. Neural Networks **141**, 270–280 (2021)