# ViCE: Improving Dense Representation Learning by Superpixelization and Contrasting Cluster Assignment *Supplementary material*

Robin Karlsson[1]
karlsson.robin@g.sp.m.is.nagoya-u.ac.jp
Tomoki Hayashi[1]
hayashi.tomoki@g.sp.m.is.nagoya-u.ac.jp
Keisuke Fujii[1]
fujii@i.nagoya-u.ac.jp
Alexander Carballo[1]
alexander@g.sp.m.is.nagoya-u.ac.jp
Kento Ohtani[1]
ohtani.kento@g.sp.m.is.nagoya-u.ac.jp
Kazuya Takeda[1,2]
kazuya.takeda@nagoya-u.jp

[1] Graduate School of Informatics
Nagoya University
Aichi, Japan

[2] Tier IV Inc.
Tokyo, Japan

## A Pseudocodes

Algorithm 1 explains the generation of $M$ views for a batch of $N$ images. The algorithm samples an image $X^{(n)}$ and computes a superpixel index map $A^{(n)}$. $M$ views are generated from the sampled image and superpixel index map. Each of these views are randomly masked before being resized to the same pixel dimension. Only mutual regions existing in all views are kept. All views are geometrically augmented by random horizontal flipping, and appearance augmented by color distortion and randomly blurred. All generated views are gathered and converted into a 4D tensor.

Algorithm 2 explains the learning algorithm. The model $f_\theta$ generates an embedding map $\hat{Z}$ from the image view tensor $\tilde{X}$. The single tensor $\hat{Z}$ is decomposed into $B$ tensors $\hat{Z}^{(b)}$ each corresponding to a single view. Next, four trees are created to contain the latent visual embeddings $z$ for all elements in each mutual region $i$. A mean vectors $z^*$ is computed to represent regions. Each mean vector gets computed a concept compatibility score $s^*$ as distance to each cluster $C = (c^{(1)}, \ldots, c^{(K)})$. The swapped prediction objective is computed using the score vectors $s^*$ stored in the tree $T_{S^*}$. The model parameters $\theta$ and set of visual concept vectors $C$ are optimized to reduce the loss $\mathcal{L}$.

The swapped prediction objective is explained in Algorithm 3. First, we compute an optimal assignment of visual concepts $Q$ based on the scores in the first view $m = 1$. The

**Algorithm 1** View generation

$\tilde{X} := \{\}$ ▷ Empty sets
$\tilde{A} := \{\}$
**for** $n \in \{1, \ldots, N\}$ **do**
  $X^{(n)} \sim$ dataloader ▷ Sample an image
  $A^{(n)} := \text{superpixels}(X^{(n)})$

  $\tilde{X}^{(n)}, \tilde{A}^{(n)} := \text{gen\_views}(X^{(n)}, A^{(n)})$
  $\# \tilde{X}^{(n)} = \{\tilde{X}^{(1,n)}, \ldots, \tilde{X}^{(M,n)}\}$
  $\# \tilde{A}^{(n)} = \{\tilde{A}^{(1,n)}, \ldots, \tilde{A}^{(M,n)}\}$

  $\tilde{X}^{(n)}, \tilde{A}^{(n)} := \text{mask\_views}(\tilde{X}^{(n)}, \tilde{A}^{(n)})$
  $\tilde{X}^{(n)}, \tilde{A}^{(n)} := \text{resize\_views}(\tilde{X}^{(n)}, \tilde{A}^{(n)})$
  $\tilde{X}^{(n)}, \tilde{A}^{(n)} := \text{mutual\_regions}(\tilde{X}^{(n)}, \tilde{A}^{(n)})$

  $\tilde{X}^{(n)}, \tilde{A}^{(n)} := \text{geometric\_aug}(\tilde{X}^{(n)}, \tilde{A}^{(n)})$
  $\tilde{X}^{(n)} := \text{appearance\_aug}(\tilde{X}^{(n)})$

  $\tilde{X} := \tilde{X} + \tilde{X}^{(n)}$ ▷ Add new views to set
  $\tilde{A} := \tilde{A} + \tilde{A}^{(n)}$
**end for**
$\tilde{X} := \text{to\_tensor}(\tilde{X})$ ▷ $\tilde{X} \in \mathbb{R}^{B \times 3 \times h \times w}$
$\tilde{A} := \text{to\_tensor}(\tilde{A})$ ▷ $\tilde{A} \in \mathbb{R}^{B \times 1 \times h \times w}$

loss is minimized when predicted visual embeddings in secondary views $m \geq 1$ are closer to the optimally assigned visual concept vectors for each region $i$ in all views $m$ of all images $n$. This results in a cross-entropy optimization objective when both assignments $q^{(i)}$ and compatibility scores $s^{(i)*}$ are normalized.

# B Hyperparameter study

We quantify the effect of hyperparameter choices by running a set of high-resolution COCO representation quality experiments for four epochs and linear model evaluation. In each experiment we change only a single parameter in an otherwise static baseline configuration. The experiments are listed in Table 1. Our baseline experiment setup is as follows; view size 512 px, maximal mask coverage 50 %, 128 concepts, queue size of 5K vectors, five views, embedding dimension $D$ equaling 64, and modest view resize range $(0.5, 1.5)$.

The results indicate that modest masking proves to be better than no masking. The ideal number of concepts needs to be found by experiments. Increasing the number of views improves representation learning, as also noted in SwAV [2]. However not by a substantial amount itself explaining the performance gap between ViCE and PiCIE [3] experiments using five and two views, respectively. Larger embedding size $D$ results in more expressive embeddings. The benefit of increasing $D$ is confirmed by an additional experiment using smaller 400 px view sizes to fit training jobs in GPU memory. All benchmark experiments presented in the main paper use the optimal hyperparameters found in this study.

Table 2 present COCO experiments with varying feature dimension $D$ and number of

**Algorithm 2** Learning algorithm

# Generate embedding maps
$\hat{Z} := f_\theta(\tilde{X})$ ▷ $\hat{Z} \in \mathbb{R}^{B\times D\times h\times w}$
$\{\hat{Z}^{(1)},\ldots,\hat{Z}^{(B)}\} := \text{decompose}(\hat{Z})$

# Create embedding and score trees
$T_Z(n,m,i) := \{\}$ ▷ Empty depth-3 trees
$T_{Z*}(n,m,i) := \{\}$
$T_{S*}(n,m,i) := \{\}$
**for** $b \in \{1,\ldots,B\}$ **do**
  $\tilde{Z}^{(b)} := \text{unroll}(\hat{Z}^{(b)})$ ▷ $\tilde{Z}^{(b)} \in \mathbb{R}^{hw\times D}$
  $\tilde{A}^{(b)} := \text{unroll}(\tilde{A}^{(b)})$ ▷ $\tilde{A}^{(b)} \in \mathbb{R}^{hw}$
  $n,m := \text{img\_view\_index}(b)$
  $I := \text{num\_regions}(\tilde{A}^{(b)})$
  **for** $i \in \{1,\ldots,I\}$ **do**
    # Compute mean vectors for region
    $\{\hat{z}^{(j)}\} := \text{extract\_region}(\tilde{Z}^{(b)},\tilde{A}^{(b)},i)$
    $T_Z(n,m,i) := \{\hat{z}^{(j)}\}$
    $z^{(i)*} := \text{mean}(T_Z(n,m,i))$
    $T_{Z*}(n,m,i) := z^{(i)*}$

    # Compute score vectors for region
    $s^{(i)*} = (T_{Z*}(n,m,i))^T C$
    $T_{S^*} := s^{(i)*}$
  **end for**
**end for**

$\mathcal{L} = \text{swapped\_prediction}(T_{S^*})$

optimize$(\theta, C, \mathcal{L})$

**Algorithm 3** Swapped prediction objective

$\mathcal{L} := 0$
$Q := \text{optimal\_assignment}(T_{S^*})$
**for** $n \in \{1,\ldots,N\}$ **do**
  **for** $m \in \{2,\ldots,M\}$ **do**
    **for** $i \in \{1,\ldots,I\}$ **do**
      $q^{(i)} := Q(n,i)$
      $s^{(i)*} := T_{S^*}(n,m,i)$
      $p^{(i)} := \sigma\left(\frac{1}{\tau}s^{(i)*}\right)$
      $\mathcal{L} -= q^{(i)} log\, p^{(i)}$
    **end for**
    $\mathcal{L} := \mathcal{L}/I$
  **end for**
**end for**
$\mathcal{L} := \mathcal{L}/(N(M-1))$

Table 1: Hyperparameter experiments

| Hyperparameter change | Δ mIoU |
|---|---|
| Masking ratio 50% → 25% | +1.52 (+8.3%) |
| Masking ratio 50% → 0% | +1.35 (+7.4%) |
| #Concepts 128 → 64 | -0.45 (-2.5%) |
| #Concepts 128 → 256 | -0.59 (-3.2%) |
| Queue size 5K → 10K | -0.73 (-4.0%) |
| #Views 5 → 2 | -1.22 (-6.6%) |
| Emb. size $D$ 64 → 32 | -1.48 (-8.1%) |
| Resize range (0.5, 1.5) → (0.15, 2.0) | -1.86 (-10.1%) |

Table 2: Effect of varying feature dimension $D$ and prototype count $K$

| $(D,K)$ | $(64,64)$ | $(64,128)$ | $(128,128)$ | $(128,256)$ | $(256,128)$ | $(256,256)$ |
|---|---|---|---|---|---|---|
| mIoU | 26.34 | 26.91 | 27.20 | 26.36 | **27.25** | 26.08 |

prototypes $K$. Each model uses the same RN 50 backbone and is trained for 4 epochs. Increasing $D$ consistently results in better performance. However, increasing $K$ beyond 128 prototypes leads to worse result, at least for the same amount of training iterations. The possibility of further improving maximum performance by increasing $D$ and $K$ with additional training epochs remain to be explored.

## C Superpixel vs. grid experiments

The left plot in Fig. 1 demonstrates consistent gains from using superpixels instead of grids. The right plot shows how performance converges for very small and large base element sizes with linear model evaluation. The result indicates that there exists a sweet spot for base element size in terms of effective learning.

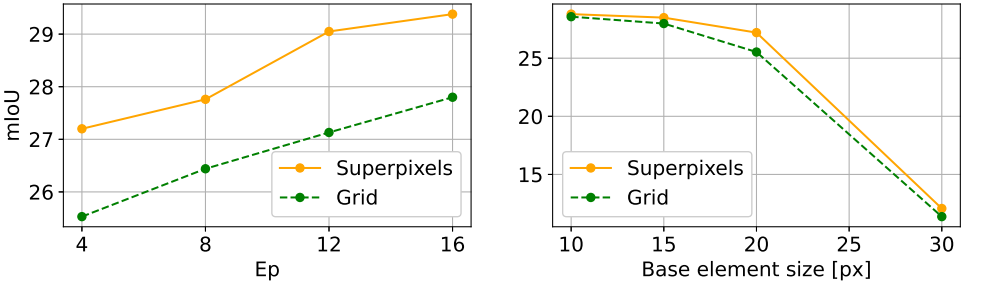


Figure 1: Superpixel and grid performance compared on high-resolution COCO

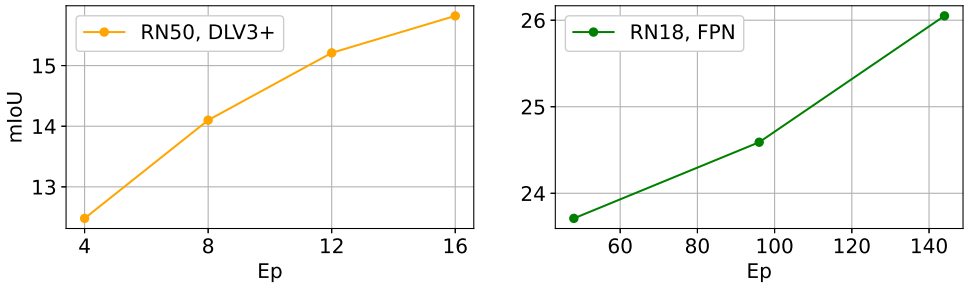


Figure 2: Performance when starting from random initialization on high-resolution COCO (left) and low-resolution Cityscapes (right) images

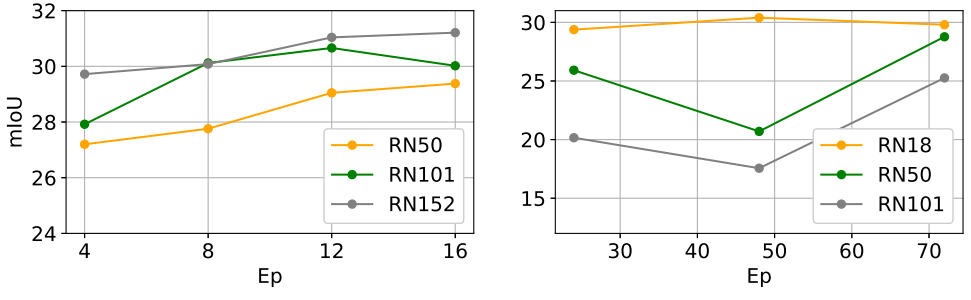


Figure 3: Performance with different backbones on high-resolution COCO (left) and Cityscapes (right) images

# D Representation learning from random initialization

In Fig. 2 we show that ViCE is capable to learn visual concepts from scratch using both high- and low-resolution images and linear model evaluation. In particular, the low-resolution Cityscapes model shows linear improvement and achieves 26.05 mIoU after 144 epochs, approaching the best result 30.84 mIoU obtained after 24 epochs starting with pretrained weights. Thus differently from STEGO [5], our method is thus not fundamentally reliant on weight initialization from other supervised or self-supervised pretraining tasks, though using pretrained weights effectively bootstraps learning.

# E Effect of backbone complexity

In Fig. 3 we show how performance change with increasing backbone complexity with linear model evaluation. Our results on COCO indicate that performance per epoch consistently improves with increased backbone complexity. In contrast, the results on Cityscapes indicate worse performance. A plausible explanation is that Cityscapes is smaller and less general than COCO, making larger self-supervised models prone to overfit patterns that do not generalize beyond the training sample distribution.

Table 3: Average training step time per high-resolution image batch

| Phase | Forward | Loss comp. | Backward | Optimization | Tot. |
|---|---|---|---|---|---|
| [msec] | 429 | 166 | 4167 | 43 | 4824 |

Table 4: Average inference time for a high-resolution image

| | Segmentation model | Cluster model | Linear model |
|---|---|---|---|
| [msec] | 57 | 2395 | 15 |

# F Timing information

We present training step timing information in Table 3. The summary is compiled by the framework VISSL [4], and represent average values for a training process involving 32 V100 GPUs distributed over 8 nodes. Table 4 shows average inference time per image for cluster and linear evaluation models using a single 3080Ti GPU in a desktop machine. Note that for high-resolution images, linear model evaluation is 160 times quicker than the k-NN cluster evaluation implemented using FAISS [6].

# G Additional visualization results

In Fig. 5, the center image shows how visual concept embeddings in the output embedding map can be clustered into coherent regions. The right image demonstrates how to semantically interpret the image by assigning each cluster a semantic meaning or class. The fact that this is possible depends on the consistent semantic interpretability of the discovered clusters over different samples.

Fig. 6 presents additional output visualizations of high-resolution COCO images for clustering and linear evaluation models with 256 clusters or linear model predictions. Each image is interpreted by five different models and arranged in groups. Each group display the input image in the top-left corner with the PiCIE output visualization bellow for comparison. The remaining visualizations display the output of clustering and linear evaluation models trained on high- and low-resolution COCO images. Ground truth labels are visualized in the right column. We find that high-resolution models produce better segmentation borders and less noise. Linear evaluation model output also displays better segmentation borders and less noise, in addition to 160 times faster evaluation time.

# References

[1] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, PAMI-8(6):679–698, 1986.

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, volume 33, 2020.

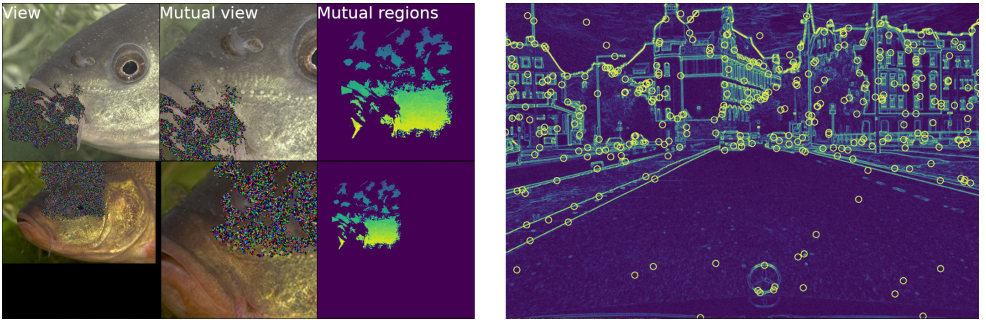


Figure 4: (Left) Examples of two generated view pairs. The first image displays the actual view feed to the model. The second image illustrates the mutual image region. The third image shows mutual superpixel regions colored by region index. (Right) View generation centers sampled from a probability mask representing image complexity measured by the Canny edge detection algorithm [0].

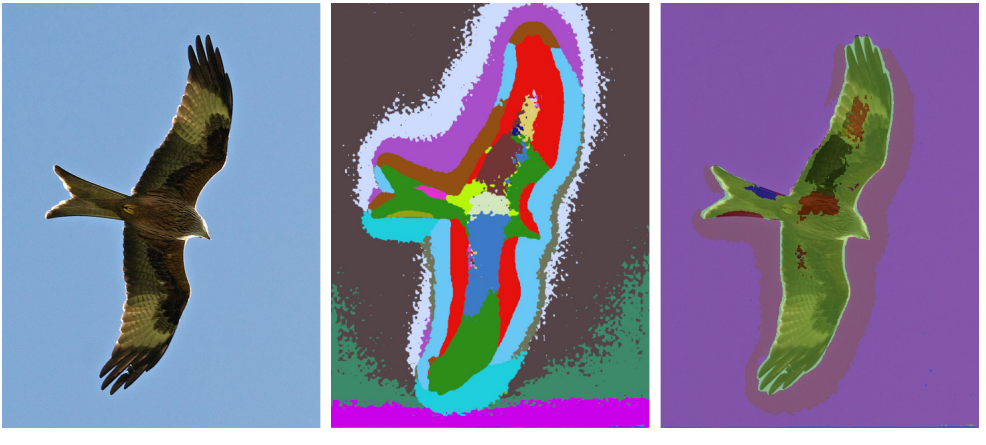

Figure 5: Visualization of output clustering. The center image shows clusters with random colors. The right image shows how clusters are mapped to a semantic classes.

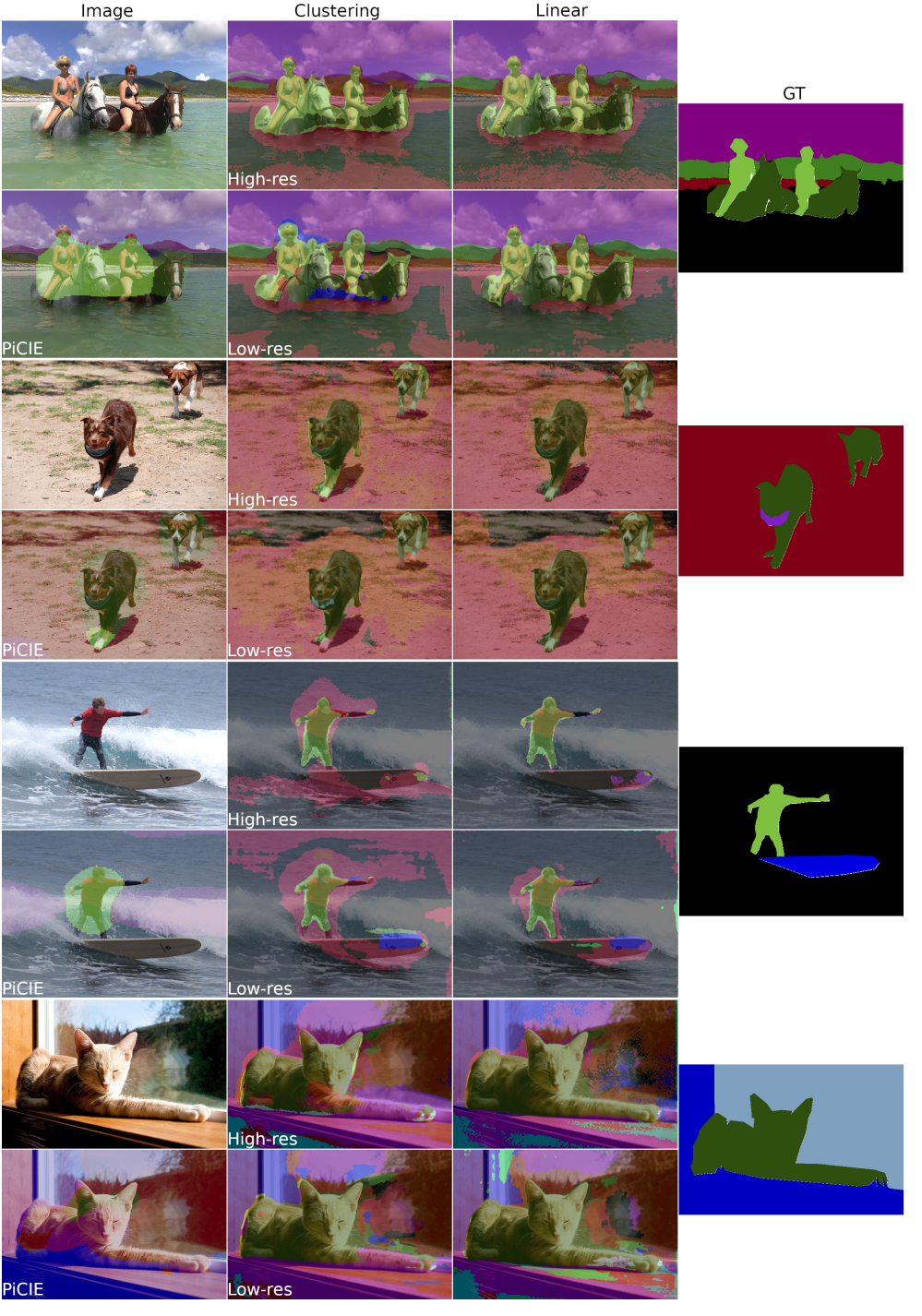


Figure 6: Output visualizations of cluster and linear evaluation models trained on low- and high-resolution COCO images.

[3] Jang Hyun Cho, Utkarsh Mall, Kavita Bala, and Bharath Hariharan. PiCIE: Unsupervised semantic segmentation using invariance and equivariance in clustering. In *CVPR*, pages 16794–16804, 2021.

[4] Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. VISSL. https://github.com/facebookresearch/vissl, 2021.

[5] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan Noah Snavely, and William T. Freeman. Unsupervised semantic segmentation by distilling feature correspondances. In *ICLR*, Apr. 2022.

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.