# Improving Predicate Representation in Scene Graph Generation by Self-Supervised Learning

Supplementary Materials

So Hasegawa , Masayuki Hiromoto , Akira Nakagawa , and Yuhei Umeda

Fujitsu Limited, Japan {hasegawa.sou,hiromoto,anaka,umeda.yuhei}@fujitsu.com

## A. Implementation Details

### A.1. Merging Technique to Truncate Subject-Object Pairs

A merging technique is employed to merge multiple bounding boxes into a single bounding box so that the merged bounding box contains maximized information about the object. The merging is realized by eliminating small bounding boxes that are mostly overlapped by larger bounding boxes. The detailed procedure is described in Algorithm 1. In these algorithms, a bounding box $b$ is defined as $b = (x_1, y_1, x_2, y_2)$, where $(x_1, y_1)$ is the coordinates of the upper left side of the bounding box, and $(x_2, y_2)$ is those of the lower right side. An input of Algorithms 1 is a collection of bounding boxes for each object class sorted by an area of bounding box in descending order ($B_{\mathrm{cls}}$).

The inclusion of bounding boxes is determined by the overlapping threshold parameter $O_{\mathrm{th}}$. Table 1 shows the effect of the parameter $O_{\mathrm{th}}$ on SePiR's mR@100 scores for SGDet task. From this result, we set $O_{\mathrm{th}} = 0.95$ for all the experiments in this work.

| $O_{\mathrm{th}}$ | mR@100 |
|---|---|
| 0.90 | 18.7 |
| **0.95** | **19.7** |
| 1.00 | 17.2 |

Table 1. Effect of the overlapping threshold parameter $O_{\mathrm{th}}$ on SGDet task. mR@100 values of SePiR+Reweight with different $O_{\mathrm{th}}$ are listed.

**Algorithm 1** Merging Technique

**Input:** $B_{\mathrm{cls}}$
**Output:** $B_{\mathrm{merged}}$
  $B_{\mathrm{del}} \leftarrow []$
  $comb \leftarrow \mathrm{combinations}(B_{\mathrm{cls}})$
  **for** $b_i, b_j \in comb$ **do**
    **if** INCLUDE_CHECK$(b_i, b_j)$ **then**
      $B_{\mathrm{del}}.\mathrm{append}(b_j)$
    **end if**
  **end for**
  $B_{\mathrm{merged}} \leftarrow B_{\mathrm{cls}} - B_{\mathrm{del}}$ ▷ Remove included box

  **function** INCLUDE_CHECK$(b_i, b_j)$
    $A_j = (x_{2j} - x_{1j}) \times (y_{2j} - y_{1j})$ ▷ Area of $b_j$
    $x_{\mathrm{o}1} = \max(x_{1i}, x_{1j}),\ y_{\mathrm{o}1} = \max(y_{1i}, y_{1j})$
    $x_{\mathrm{o}2} = \min(x_{2i}, x_{2j}),\ y_{\mathrm{o}2} = \min(y_{2i}, y_{2j})$
    $A_{\mathrm{o}} = \max(0, x_{\mathrm{o}2} - x_{\mathrm{o}1}) \times \max(0, y_{\mathrm{o}2} - y_{\mathrm{o}1})$
    ▷ Area of overlapping
    **if** $A_{\mathrm{o}}/A_j \geq O_{\mathrm{th}}$ **then**
      **return** *True*
    **else**
      **return** *False*
    **end if**
  **end function**

### A.2. Network Architectures

SePiR consists of various network architectures: a relational encoder, a projector, an object classifier, and a predicate classifier except for an attention-based object detector. The details of the network architectures are shown in Table 2.

Aside from architectures described in the paper, our implementations include a refiner, which consists of 2 fully connected layers, before a relational encoder. An overall architecture containing the refiner is shown in Fig. 1. Since the refiner is trained by self-supervised learning in combination with the relational encoder and the projector, it plays a role in enhancing the visual features by incorporating the information of relational features through the backpropagation of a loss. In training classifiers, enhanced visual features by the refiner are fed to the object classifier.

| Component | Architecture flow |
|---|---|
| Relational encoder | FC → FC → FC → ReLU → FC → BN → ReLU → FC → BN → ReLU → FC |
| Projector | FC → BN → ReLU → FC → BN → ReLU → FC |
| Object classifier | FC → ReLU → FC → ReLU → FC |
| Predicate classifier | FC → ReLU → FC → ReLU → FC |

Table 2. The detailed architecture of each component in SePiR. *FC* is a fully-connected layer, and *BN* is a batch normalization layer.

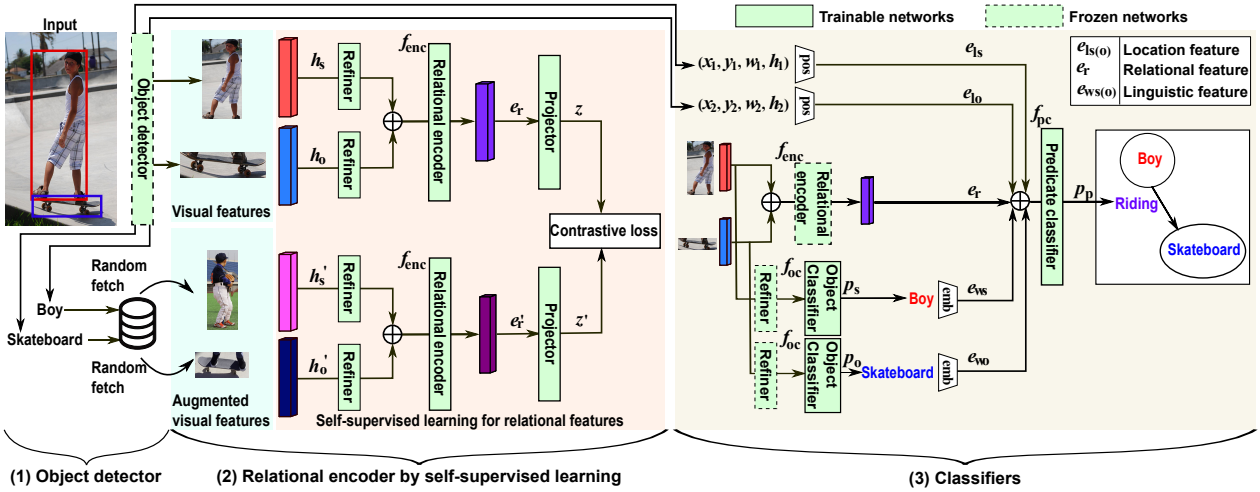


Figure 1. Overall pipeline of the proposed SePiR including a refiner.

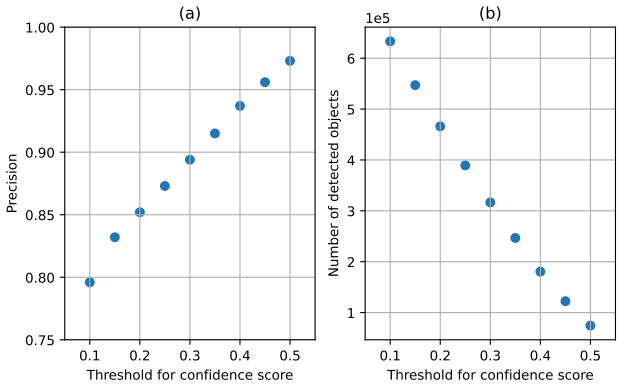


Figure 2. (a) Precision and (b) the number of the detected objects by Conditional DETR as a function of the threshold for the confidence scores of object candidates.

## A.3. Choice of Threshold for Pair Confidence

In the proposed method, subject-object pairs are truncated by using the pair confidence $c_{\text{pair}} = c_{\text{s}} \times c_{\text{o}}$, where $c_{\text{s}}$ and $c_{\text{o}}$ are the confidence scores of the subject and object generated by the pre-trained object detector. Only the pairs satisfying a requirement $c_{\text{pair}} > c_{\text{th}}$ are extracted for the subsequent process. To determine the threshold $c_{\text{th}}$, we examine the confidence scores $c_{\text{s}}, c_{\text{o}}$ of the object detector. Fig 2 (a) shows precision of object detection and (b) shows the number of detected objects as a function of the threshold parameter for the confidence score. We used Conditional DETR [14] for the experiment. Obviously, as the threshold of confidence score increases, the precision also increases while the number of the detected objects decreases. High precision is imperative for preserving relationships between subject-object pairs. On the other hand, the large number of detected objects contributes to increasing variations of the data. Since these two objectives are incompatible, we should take a balanced threshold parameter considering the trade-off. In this case, we want to extract as many objects as possible while keeping the precision at about 0.9. Therefore, we choose a threshold for the object confidence $c_{\text{s}}, c_{\text{o}}$ as 0.3, and set $c_{\text{th}} = 0.09\ (= 0.3^2)$ as a threshold for the pair confidence $c_{\text{pair}}$.

## A.4. Debiasing Methods for Trade-off Curves

With trade-off curves, we are able to judge an advantage of the model if its curve locates in the upper right area where both R@100 and mR@100 are high. To plot such curves, we incorporate six methods to deal with the long-tailed predicate class distribution. We pick them so that these methods cover the range from low R/mR@K to high R/mR@K. The details of these methods are as follows.

- **Cross Entropy** uses a standard cross entropy.
- **CBR** utilizes Class-Balanced Re-weighting loss [7].

- **Bilvl. Samp.** uses a bilevel sampling strategy [11] for re-balancing the data distribution.
- **Reweight** uses a loss function whose weight is computed using counts of each predicate as below.

$$L_{\text{Reweight}}(i) = \frac{1}{n_i} CE_i, \tag{1}$$

where $n_i$ is the number of training samples belonging to the $i$-th predicate class, and $CE_i$ is the cross entropy loss for the $i$-th predicate class.

- **Bilvl. Samp. + Reweight** simultaneously uses Bilvl. Samp. and Reweight above.
- **RTPB (CB)** employs Resistance Training using Prior Bias with a count resistance bias [3].

For the training of the classifiers in SePiR with these methods, an optimization strategy is as follows. The learning rate starts from an initial value $lr_{\text{init}}$ and linearly increases every batch iteration up to $lr_{\max}$ for 500 iterations. After this warm-up phase, the learning rate stays $lr_{\max}$. We set $lr_{\text{init}} = 0.008$ and $lr_{\max} = 0.128$ for *Cross Entropy*, *CBR*, *Bilvl. Samp.*, and *RTPB (CB)* methods, and $lr_{\text{init}} = 0.0005$ and $lr_{\max} = 0.008$ for *Reweight* and *Bilvl. Samp. + Reweight* methods.

### A.5. Acceleration of Data Augmentation Process

We realize augmentation by replacing the visual features of subject-object pairs. A naive implementation of the augmentation leads to a non-negligible computation time because the object detector runs twice to extract visual features for each subject-object pair. Since one iteration consists of about 1,000 subject-object pairs, the augmentation process takes a long time, and the total training speed becomes very slow. To accelerate the augmentation process, we implement two techniques: feature caching and object-wise replacement.

**Feature caching.** Before training self-supervised learning, we create a cached dataset of visual features extracted from all the images of the train set of Visual Genome [10] by utilizing target object labels. The labeled cached features are utilized for the data augmentation in the self-supervised step. With this technique, we can obtain visual features without running the object detector again.

**Object-wise replacement.** We perform feature replacement in an object-wise manner, not pair-wise, because in a single image the number of the objects ($< 100$) is much smaller than that of the subject-object pairs ($\sim 1,000$). Instead of replacing features for each subject-object pair, we replace the features of the objects from which all the subject-object pairs are formed. This can drastically reduce the number of replacements and accelerate the augmentation process. Although this method may reduce the variation of the data augmentation, it is acceptable since a sufficient number of different augmentations are applied while the entire training process.

## B. Quantitative Studies

### B.1. Comparison with the Previous Methods

Recalls and mean recalls @K=20, 50, and 100 are listed in Tables 3 and 4, respectively. Also, the trade-off curves at K=20, 50 are shown in Fig. 3 and 4. A similar trend can be seen at different K values. The proposed method is comparable or higher performance than the previous methods in PredCls and SGDet tasks.

Table 5 shows comparison of mean recalls for "head," "body," and "tail" categories of predicates between SePiR and DTrans. Following [11], above three categories are divided according to the number of instances in the training split as "head (more than 10 k instances)," "body (from 0.5 k to 10 k)," and "tail (less than 0.5 k)." Although the overall mR score of DTrans is higher than SePiR, the result shows that our method is more effective to predict long-tailed rare relationships. Since abstract predicates locate in the head categories, predicting informative predicates on the tail categories leads to decreasing recall of the head categories. The better ability to predict informative predicates the trained model has, the more decrease in the performance on the head categories is observed. This is why the mR score on the head categories of SePiR is lower than that of DTrans.

### B.2. Experiments with Limited Labeled Dataset

**Comparison to state-of-the-art supervised methods.** In the paper, we show that SePiR outperforms state-of-the-art supervised methods (BGNN [11] and DTrans [3]) with the limited labeled dataset on SGDet. We also exhibit the experimental result on PredCls, and the result is shown in Fig. 5. SePiR also achieves higher performance than state-of-the-art supervised methods on PredCls. Since PredCls does not depend on the performance of the object detector, the result indicates that SePiR is able to capture the robust predicate representation.

**Comparison to SePiR in a supervised manner.** We also would like to corroborate that our self-supervised method purely contributes to acquiring the robust predicate representation without depending on the architectures. To ensure this, we compare SePiR with SePiR trained in a supervised manner on PredCls. The result is shown in Fig. 6. Obviously, the gap between the performance of SePiR and

| Models | PredCls | | | SGCls | | | SGDet | | |
|---|---|---|---|---|---|---|---|---|---|
| | R@20 | R@50 | R@100 | R@20 | R@50 | R@100 | R@20 | R@50 | R@100 |
| KERN [5] | | 65.8 | 67.6 | | 36.7 | 37.4 | | 27.1 | 29.8 |
| GPS-Net [12] | 60.7 | 66.9 | 68.8 | 36.1 | 39.2 | 40.1 | 22.6 | 28.4 | 31.7 |
| PCPL [20] | | 50.8 | 52.6 | | 27.6 | 28.4 | | 14.6 | 18.6 |
| BGNN [11] | | 59.2 | 61.3 | | 37.4 | 38.5 | | 31.0 | 35.8 |
| Seq2Seq-RL [13] | 60.3 | 66.4 | 68.5 | 34.5 | 38.3 | 39.0 | 22.1 | 30.9 | 34.4 |
| DTrans + RTPB (CB) [3] | 60.3 | 66.4 | 68.5 | 34.5 | 38.3 | 39.0 | 22.1 | 30.9 | 34.4 |
| Motifs [21, 19] | 58.5 | 65.2 | 67.1 | 32.9 | 35.8 | 36.5 | 21.4 | 27.2 | 30.3 |
| Motifs + TDE [18] | 38.7 | 50.8 | 55.8 | 21.8 | 27.2 | 29.5 | 5.9 | 7.4 | 8.4 |
| Motifs + BA-SGG [9] | 44.4 | 50.7 | 52.5 | 26.9 | 30.1 | 31.0 | 16.8 | 23.0 | 26.9 |
| Motifs + RTPB (CB) [3] | | 40.4 | 42.5 | | 26.0 | 26.9 | | 19.0 | 22.5 |
| VCTree [19] | 60.1 | 66.4 | 68.1 | 35.2 | 38.1 | 38.8 | 22.0 | 27.9 | 31.3 |
| VCTree + TDE [18] | 39.1 | 49.9 | 54.5 | 22.8 | 28.8 | 31.2 | 14.3 | 19.6 | 23.3 |
| VCTree + BA-SGG [9] | 43.9 | 50.0 | 51.8 | 30.2 | 34.0 | 35.0 | 15.8 | 21.7 | 25.5 |
| VCTree + RTPB (CB) [3] | | 41.2 | 43.3 | | 28.7 | 30.0 | | 18.1 | 21.3 |
| **SePiR + Bilevel sampling** | 55.1 | 62.3 | 64.6 | 31.9 | 35.6 | 36.8 | 20.5 | 27.5 | 32.1 |
| **SePiR + RTPB (CB)** | 24.0 | 29.7 | 31.8 | 14.2 | 17.2 | 18.1 | 8.9 | 12.8 | 15.6 |
| **SePiR + Reweight** | 20.2 | 26.3 | 28.9 | 11.7 | 14.7 | 15.9 | 9.5 | 13.6 | 16.6 |

Table 3. Recalls of PredCls, SGCls, and SGDet on VG. The scores of the existing methods are referred from the cited papers.

| Models | PredCls | | | SGCls | | | SGDet | | |
|---|---|---|---|---|---|---|---|---|---|
| | mR@20 | mR@50 | mR@100 | mR@20 | mR@50 | mR@100 | mR@20 | mR@50 | mR@100 |
| KERN [5] | | 17.7 | 19.2 | | 9.4 | 10.0 | | 6.4 | 7.3 |
| GPS-Net [12] | 17.4 | 21.3 | 22.8 | 10.0 | 11.8 | 12.6 | 6.9 | 8.7 | 9.8 |
| PCPL [20] | | 35.2 | 37.8 | | 18.6 | 19.6 | | 9.5 | 11.7 |
| BGNN [11] | | 30.4 | 32.9 | | 14.3 | 16.5 | | 10.7 | 12.6 |
| Seq2Seq-RL [13] | 21.3 | 26.1 | 30.5 | 11.9 | 14.7 | 16.2 | 7.5 | 9.6 | 12.1 |
| DTrans + RTPB (CB) [3] | 30.3 | 36.2 | 38.1 | 19.1 | 21.8 | 22.8 | **12.7** | **16.5** | 19.0 |
| Motifs [21, 19] | 10.8 | 14.0 | 15.3 | 6.3 | 7.7 | 8.2 | 4.2 | 5.7 | 6.6 |
| Motifs + TDE [18] | 18.5 | 24.9 | 28.3 | 11.1 | 13.9 | 15.2 | 6.6 | 8.5 | 9.9 |
| Motifs + BA-SGG [9] | 24.8 | 29.7 | 31.7 | 14.0 | 16.5 | 17.5 | 10.7 | 13.5 | 15.6 |
| Motifs + RTPB (CB) [3] | 28.8 | 35.3 | 37.7 | 16.3 | 19.4 | 20.6 | 9.7 | 13.1 | 15.5 |
| VCTree [19] | 14.0 | 17.9 | 19.4 | 8.2 | 10.1 | 10.8 | 5.2 | 6.9 | 8.0 |
| VCTree + TDE [18] | 17.2 | 23.3 | 26.6 | 8.9 | 11.8 | 13.4 | 6.3 | 8.6 | 10.3 |
| VCTree + BA-SGG [9] | 26.2 | 30.6 | 32.6 | 17.2 | 20.1 | 21.2 | 10.6 | 13.5 | 15.7 |
| VCTree + RTPB (CB) [3] | 27.3 | 33.4 | 35.6 | **20.6** | **24.5** | **25.8** | 9.6 | 12.8 | 15.1 |
| **SePiR + Bilevel sampling** | 25.2 | 30.8 | 33.2 | 13.9 | 17.0 | 18.5 | 8.2 | 11.0 | 13.1 |
| **SePiR + RTPB (CB)** | 31.7 | 37.8 | 40.3 | 17.0 | 19.7 | 20.7 | 11.2 | 14.2 | 16.4 |
| **SePiR + Reweight** | **32.1** | **39.9** | **43.2** | 17.6 | 21.6 | 23.6 | 12.0 | **16.5** | **19.7** |

Table 4. Mean recalls of PredCls, SGCls, and SGDet on VG. The scores of the existing methods are referred from the cited papers. The bold font indicates the best mR for each task.

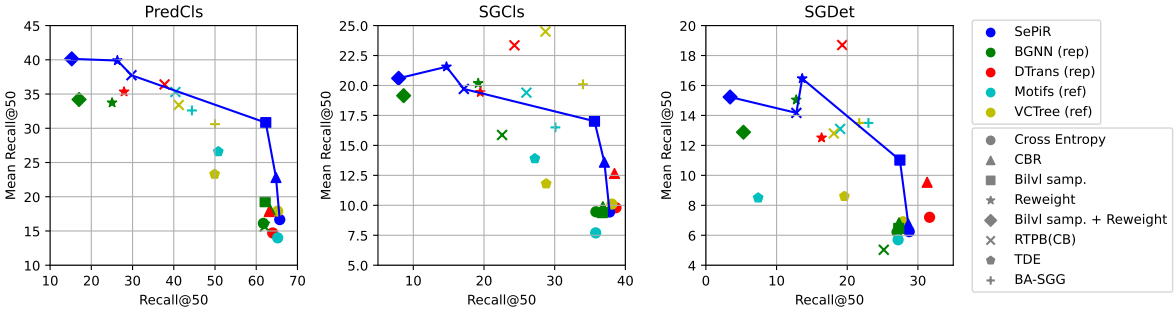


Figure 3. Trade-off curves between R@50 and mR@50 comparing SePiR and the existing methods (Motifs [21], VCTree [19], BGNN [11], and DTrans [3]) for PredCls, SGCls, and SGDet tasks on VG. *rep* means our reproduction, and *ref* means reference from the original papers.

SePiR (SL) becomes larger as the number of training images decreases. Since the architectures of these two methods are the same, the result indicates that our proposed self-supervised learning is beneficial for capturing the robust predicate representation.

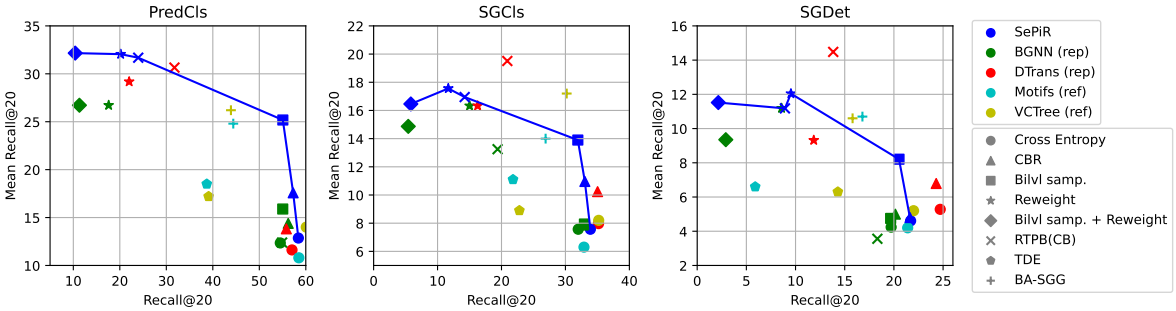


Figure 4. Trade-off curves between R@20 and mR@20 comparing SePiR and the existing methods (Motifs [21], VCTree [19], BGNN [11], and DTrans [3]) for PredCls, SGCls, and SGDet tasks on VG. *rep* means our reproduction, and *ref* means reference from the original papers.

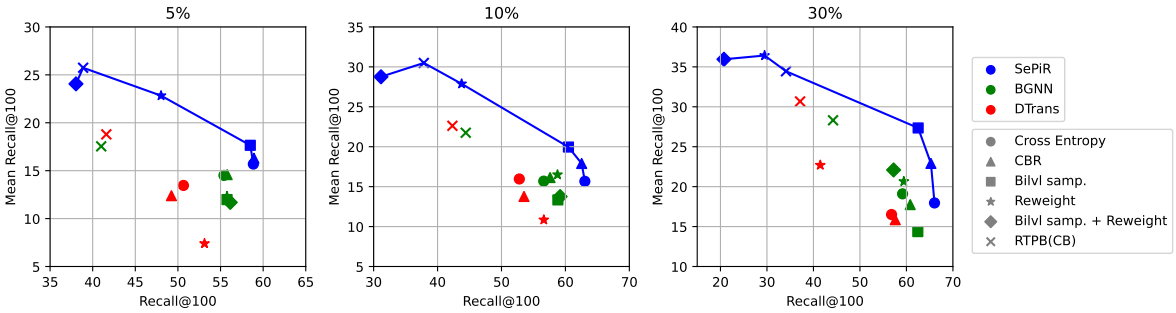


Figure 5. The comparison of PredCls performance to BGNN [11] and DTrans [3] with limited (5%, 10%, and 30%) labeled dataset.

| Category | SePiR + Reweight | DTrans + RTPB(CB) |
|---|---|---|
| Head | 19.1 | 25.5 |
| Body | 19.2 | 23.2 |
| Tail | 20.3 | 17.2 |
| All | 19.7 | 20.9 |

Table 5. Comparison of mR@100 between SePiR and DTrans for head, body, and tail sub-categories of predicate classes.

**Comparison for each predicate.** The result is shown in Fig. 9–7 and Table 6. Clearly, SePiR outperforms DTrans on tail categories with all the limited labeled dataset. The result indicates that our self-supervised learning method contributes to improving predicate representation including tail categories.

## B.3. More Ablation Study

In this section, we investigate three components of SePiR that are not described in the paper: (1) self-supervised methods, (2) word embedding methods, and (3) object detectors.

**Self-supervised methods.** To select a loss function in the self-supervised learning step, we pick up three self-supervised learning methods: SimCLR [4], SimSiam [6], and VICReg [1]. While SimCLR is the representative method of contrastive learning methods, SimSiam and VICReg are the representatives of non-contrastive methods. To see only the effects of the above self-supervised learning methods to the relational features, we do not utilize location features and linguistic features for predicate classification. We test on PredCls with the limited (5%, 10%, and 30%) labeled dataset.

The result is shown in Fig. 10. In the case of 5% and 10%, there seems little difference among SimCLR, SimSiam, and VICReg because the plots locate in similar trade-off curves. For 30%, however, the performance of SimCLR is better than the others. This is why we select SimCLR as the self-supervised method in SePiR.

**Word embedding methods.** GloVe [15] is the most widely used word embedding method in scene graph generation as linguistic feature. We investigate the validity of GloVe by comparing to other word embedding methods: random vectors, FastText [2], Numberbatch [17], and BERT [8]. To generate the random vectors, we sample 300-dim embedding vectors from a standard normal distribution for each object label. As queries for BERT, we construct a sentence with object labels of each subject-object pair as `[Subject label] is [MASK] [Object label]`. Then, BERT embedding is extracted from $i$-th index of features of the last layer, where $i$ corresponds to the index of `[MASK]`.

We investigate the influence of each word embedding method on PredCls with Visual Genome. The result is

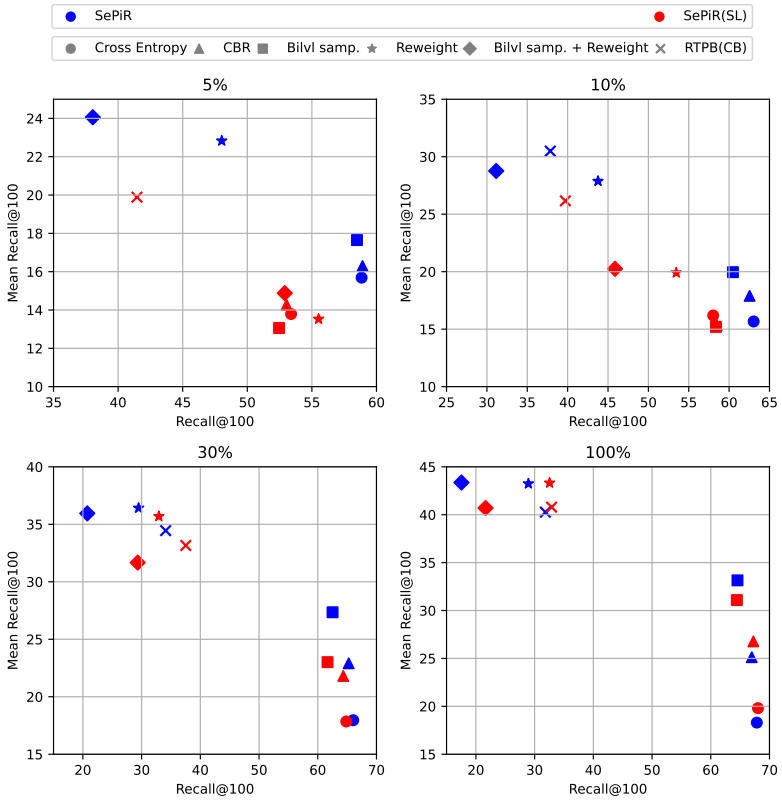


Figure 6. The comparison of PredCls performance to SePiR trained in a supervised manner (SePiR(SL)) with limited (5%, 10%, and 30%) labeled dataset.

| | SePiR + Reweight | | | DTrans + RTPB(CB) | | |
|---|---|---|---|---|---|---|
| Category | 5 | 10 | 30 | 5 | 10 | 30 |
| Head | 17.82 | 15.48 | 10.99 | 16.99 | 25.12 | 25.19 |
| Body | 17.09 | 21.09 | 20.72 | 1.82 | 5.18 | 11.94 |
| Tail | 3.99 | 8.63 | 13.17 | 0.13 | 1.23 | 2.78 |
| All | 11.32 | 14.82 | 16.04 | 3.20 | 6.23 | 9.76 |

Table 6. Comparison of mR@100 between SePiR and DTrans for head, body, and tail sub-categories of predicate classes with limited (5%, 10%, and 30%) labeled dataset.

shown in Fig 12. Actually, there are no big differences between word embedding methods. Hence, we select GloVe as a linguistic feature in SePiR following conventional scene graph generation methods. Rather, it is important to denote that utilizing random vectors achieves a similar performance to using the other word embedding methods, and that it achieves slightly higher performance than FastText and Numberbatch. The result implies that a naive introduction of word embedding cannot utilize information more than discrete points of the object labels, and that intrinsic linguistic information that object labels contain are not fully used.

**Object detectors.** In the paper, we adopt an attention-based object detector to capture object-specific visual features that are effective for the proposed augmentation. Since object-specific visual features exclude unrelated information (*e.g.* backgrounds or other objects) as much as possible, the features are beneficial for predicate predictions. On the other hand, the visual features extracted by the conventional object detector based on a region-proposal network (RPN) sometimes include this meaningless information because they extract the visual features via bounding boxes. To ensure the advantage of the attention-based object detector, we compare the visual features by the attention-based object detector with those by the RPN-based object detector. We select Conditional DETR [14] as an attention-object detector and Faster RCNN [16] as an RPN-based object detector. Using the architectures of SePiR, we evaluate on PredCls in a supervised manner and do not utilize location feature

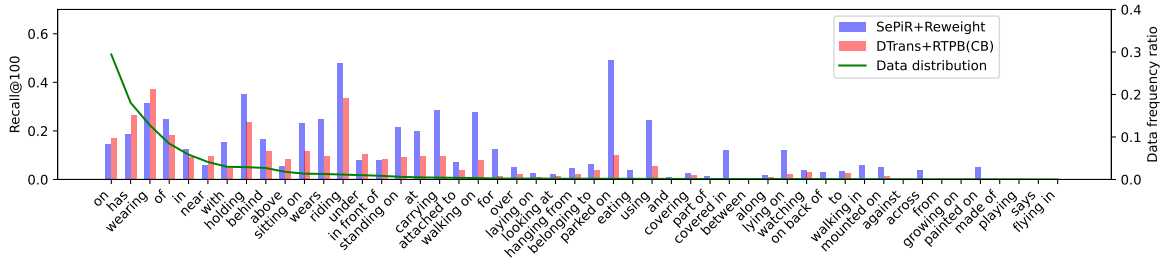


Figure 7. The comparison of Recall@100 for each predicate between SePiR+Reweight and DTrans+RTPB(CB) with limited (5%) labeled dataset.

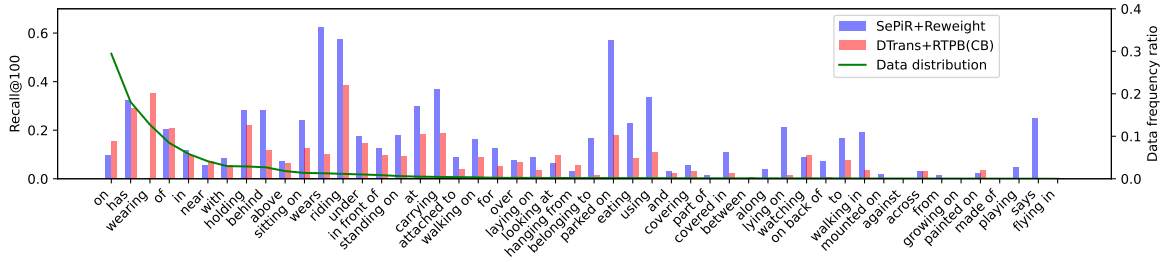


Figure 8. The comparison of Recall@100 for each predicate between SePiR+Reweight and DTrans+RTPB(CB) with limited (10%) labeled dataset.

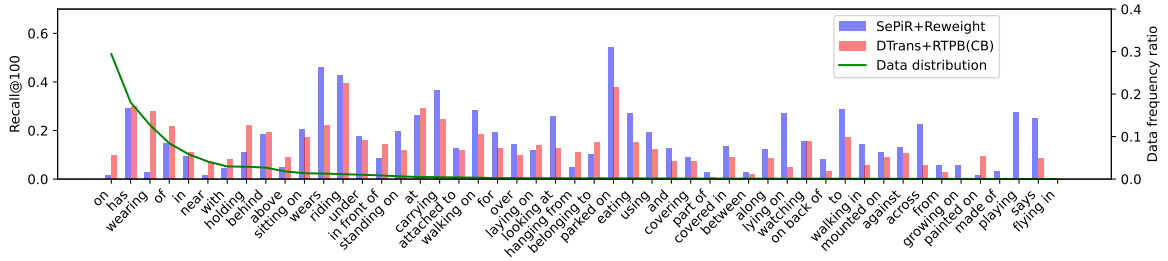


Figure 9. The comparison of Recall@100 for each predicate between SePiR+Reweight and DTrans+RTPB(CB) with limited (30%) labeled dataset.

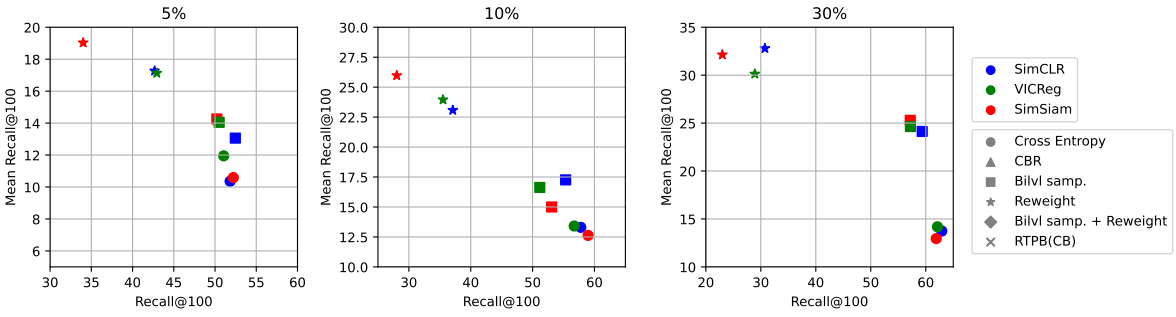


Figure 10. The comparison of the performance on PredCls among three self-supervised methods with the limited (5%, 10%, and 30%) labeled dataset.

and linguistic feature because these features depend on the performances of the object detector.

The result is shown in Fig. 11. Clearly, Conditional DETR achieves higher performance than Faster RCNN. The result indicates that the object-specific visual features has abilities to improve predicate predictions. More impor-

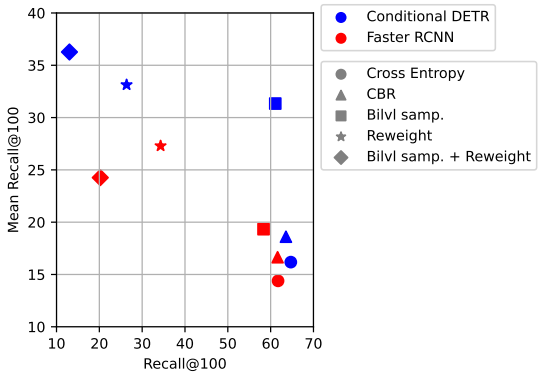


Figure 11. The comparison of performance on PredCls between visual features by Conditional DETR [14] and Faster RCNN [16]

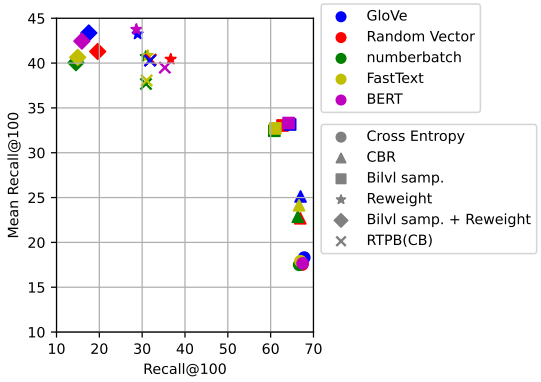


Figure 12. The comparison of the performance on PredCls using five kinds of word embedding methods.

tantly, these visual features would be useful for preserving predicates via augmentation in SePiR.

## References


[1] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] Chao Chen, Yibing Zhan, Baosheng Yu, Liu Liu, Yong Luo, and Bo Du. Resistance training using prior bias: Toward unbiased scene graph generation. In *AAAI*, volume 36, pages 212–220, 2022.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.

[5] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *CVPR*, pages 6163–6171, 2019.

[6] Xinlei Chen and Kaiming He. Exploring simple Siamese representation learning. In *CVPR*, pages 15750–15758, 2021.

[7] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, pages 9268–9277, 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, volume 1, pages 4171–4186, 2019.

[9] Yuyu Guo, Lianli Gao, Xuanhan Wang, Yuxuan Hu, Xing Xu, Xu Lu, Heng Tao Shen, and Jingkuan Song. From general to specific: Informative scene graph generation via balance adjustment. In *ICCV*, pages 16383–16392, 2021.

[10] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv:1602.07332*, 2016.

[11] Rongjie Li, Songyang Zhang, Bo Wan, and Xuming He. Bipartite graph network with adaptive message passing for unbiased scene graph generation. In *CVPR*, pages 11109–11119, 2021.

[12] Xin Lin, Changxing Ding, Jinquan Zeng, and Dacheng Tao. GPS-Net: Graph property sensing network for scene graph generation. In *CVPR*, pages 3746–3753, 2020.

[13] Yichao Lu, Himanshu Rai, Jason Chang, Boris Knyazev, Guangwei Yu, Shashank Shekhar, Graham W. Taylor, and Maksims Volkovs. Context-aware scene graph generation with Seq2Seq Transformers. In *ICCV*, pages 15931–15941, 2021.

[14] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for fast training convergence. In *ICCV*, pages 3651–3660, 2021.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Conf. Empirical Methods in Natural Lang. Process.*, pages 1532–1543, 2014.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, volume 28, 2015.

[17] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451, 2017.

[18] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased Scene Graph Generation From Biased Training. In *CVPR*, 2020.

[19] Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. Learning to compose dynamic tree structures for visual contexts. In *CVPR*, 2019.

[20] Shaotian Yan, Chen Shen, Zhongming Jin, Jianqiang Huang, Rongxin Jiang, Yaowu Chen, and Xian-Sheng Hua. PCPL: Predicate-correlation perception learning for unbiased scene graph generation. In *ACM MM*, pages 265–273, 2020.

[21] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *CVPR*, pages 5831–5840, 2018.