Overview of Appendixes

In this supplementary material, we present implementation details and more experimental results. First, more experiments and analysis (e.g., analysis of overall recall performance, experiments using stronger long-tail learning strategy, and extra qualitative results) are described in Sec. 1. We provide the implementation details in Sec. 2. Moreover, we also present the details of graph assembling mechanism and loss function in Sec. 3 and Sec. 4.

In this supplementary material, we present implementation details and more experimental results. First, more experiments and analysis (e.g., analysis of overall recall performance, experiments using stronger long-tail learning strategies, and extra qualitative results) are described in Sec. 1. We provide the implementation details in Sec. 2. Moreover, we also present the details of the graph assembling mechanism and loss function in Sec. 3 and Sec. 4.

1. More Experimental Results

1.1. Overall Recall Analysis

First, we analyze why our SGTR achieves lower overall recall performance than the traditional two-stage design. One potential reason is that the SGTR uses an ResNet-101 backbone (like DETR) for entity detection rather than the Faster-RCNN with an ResNet-101 FPN backbone. According to the experimental results of DETR [1], using the ResNet-101 backbone achieves lower performance than Faster-RCNN with the ResNet-101 FPN on small objects (21.9 versus 27.2 APs on the COCO dataset).

To confirm that, we further study how small entities influence visual relationship detection. We categorize the relationship instances in the Visual Genome dataset into three disjoint sets according to their entity sizes, and plot the statistics of the sets in Fig. 1. The result shows that more than half of the relationships consist of small entities. We also compare the performance of our method (R@100) on three relationship sub-sets with the two-stage approach BGNN [6] and report the performance in Fig. 1. The BGNN with the two-stage design outperforms the SGTR on the relationships with small entities by a large margin due to the inherent limitation of DETR on small entity detection.

The issue of recognizing small objects efficiently and effectively is still under active study [8, 11, 12, 16]. With more sophisticated transformer-based detectors, the overall recall of our method can be further improved.

1.2. Influence of Object Detector

We first compare the performances of DETR and faster RCNN on the head (H), body (B) and tail (T) predicates in Tab.1, which also groups the detector results into H/B/T object classes. The result indicates that DETR achieves a similar detection performance as Faster R-CNN for tail predicates. Moreover, we equip BGNN with DETR and as shown in Tab.2, its performance is on par with its Faster R-CNN counterparts. Finally, we also combine DETR optimized in SGTR with a finetuned BGNN. The results in Tab.2 clearly show the SGTR outperforms this baseline, demonstrating the benefit of our design.

<table>
<thead>
<tr>
<th></th>
<th>Recall (S-DETR / RCNN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rel Head</td>
<td>49.3 / 50.5</td>
</tr>
<tr>
<td>Rel Body</td>
<td>39.5 / 41.3</td>
</tr>
<tr>
<td>Rel Tail</td>
<td>44.3 / 41.7</td>
</tr>
</tbody>
</table>

Table 1. The entity detection performances of DETR and faster RCNN on the head (H), body (B) and tail (T) predicates.
with respect to the instance distribution of relationships. This method improves the performance trade-off between mR@100 and R@100 by increasing mean recall by 1.6 and achieving only a 0.4 performance drop on R@100.

- Moreover, we re-implement the alternative class-balanced retraining strategy (ACBS) [3], which achieves the SOTA on mean recall with the two-stage SGG model. The ACBS re-trains both entity and predicate classifiers using class-based sampling. This method achieves high mean recall performance while sacrificing the performance of the overall performance.

- Finally, we apply the decoupled retraining strategy [4] on the predicate classifier of SGTR. We observe that using additional balanced-sampling retraining results in 6.4 performance gain on mR@100 with a 3.6 drop on R@100. This strategy outperforms the aforementioned methods in terms of mean recall performance.

We also observe that using the re-balance idea on entity classifier does not bring too much performance benefit. To investigate this phenomenon, we report the relationship between the per-class instance frequency and the performance of entity detection in Fig. 2. Despite the fact that the distribution of entity instances obeys the long-tail distribution, SGTR’s entity detection performance is quite balanced, which means that the transformer-based detector is capable of tackling the data imbalanced scenario to some degree, and the additional re-balancing strategy is unnecessary.

### 1.4. Zero-shot Recall

We compare our method with VCTree-TDE in Tab. 4, achieving a gain of 2.6 on zR@100. The result shows the generalization capability of SGTR to unseen relationships. We greatly appreciate R2's suggestions and will leave the further exploration to future work.

### 1.5. Model Selection

We present experiments for selecting the hyper-parameters ($N_r$, $K$) on validation set of VG in Tab. 5. The performance saturates at $N_r = 160$ and $K = 3$.

<table>
<thead>
<tr>
<th>$N_r$</th>
<th>mR@100</th>
<th>R@100</th>
<th>$K$</th>
<th>mR@100</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>16.1</td>
<td>27.5</td>
<td>1</td>
<td>16.4</td>
<td>26.2</td>
</tr>
<tr>
<td>130</td>
<td>16.3</td>
<td>27.3</td>
<td>2</td>
<td>17.3</td>
<td>28.2</td>
</tr>
<tr>
<td>160</td>
<td>17.7</td>
<td>28.5</td>
<td>3</td>
<td>17.7</td>
<td>28.5</td>
</tr>
<tr>
<td>190</td>
<td>16.1</td>
<td>27.0</td>
<td>4</td>
<td>17.5</td>
<td>28.2</td>
</tr>
</tbody>
</table>

Table 5. The performance of different choices of hyper-parameters $N_r$. ⌂ 

---

Table 2. 'D' means DETR, 'S' means DETR optimized in SGTR. † means bi-level resampling, 'AP': mAP50, and 'mR': mRecall

---

Table 3. The performance of SGTR by adopting the advanced long-tail learning strategy on the VG dataset. "*" denotes the bi-level sampling proposed in [6]; "P" denotes the modified bi-level sampling; The "cRT" denotes the decoupled retraining strategy on predicate classifier proposed by [4]; the "DisAlign" denotes the retraining strategy for logits adjustment proposed by [14]; the "ACBS" refers to the alternative class balanced retraining strategy proposed by [3].

---

Figure 2. The frequency statistics and per-class detection performance of entities on the VG dataset. (a): The frequency of each entity category; (b) The per-class entity detection performance (AP).
Figure 3. Comparison of entity indicator of predicate node and entity node. We use the different colored bounding boxes of entities to distinguish between the entity node (red) and the entity indicator (pink). The yellow arrow indicates the predicate between the entities. (best viewed in color)

Figure 4. Qualitative comparison between our method and BGNN† in SGG. Both methods predict many reasonable relationships which are not annotated in GT. We mark the relationships of rare semantic predicate categories retrieved by SGTR in red (best viewed in color).

1.6. Qualitative Results

Visualization of Entity Indicator and Entity Node To demonstrate the effectiveness of our proposed graph assembling mechanism, we visualize predictions of entity indicators of our predicate representation and entity nodes after the graph assembling. As shown in Fig. 3, the entity indicator only provides a rough localization and classification of entities rather than precise bounding boxes. This information can be refined into more accurate entity results with graph assembling, which significantly improves the quality of the generated scene graph.

Prediction Comparison between Different Design We compare different method (e.g., BGNN [6]) by visualizing the relationship predictions. In Fig. 4, we mark the different relationship predictions between BGNN and SGTR with red color. It shows that SGTR retrieves more relationships of less frequent semantic categories than BGNN.

2. Implementation Details

We implement our method based on the PyTorch 1.8 [9] and cvpods [15]. Our training process consists of two phases: 1) entity detector pre-training and 2) SGTR joint training.

Entity Detector Pre-training Phase: We follow the DETR training configuration to learn the entity detector on Visual Genome and Openimage datasets. We train the entity detector with the AdamW optimizer with a learning rate of 1e-5, a batch size of 16, and the model takes 100 epochs for convergence on 4 TITAN V GPUs. We use the same scale augmentation with DETR, resizing the input images such that the shortest side is at least 480 and at most 600 pixels, while the longest is at most 1000. The hyper-parameters of
Transformer (e.g., number of attention heads, drop-out rate) are also kept the same with the DETR.

**Joint Learning Phase:** In the joint training phase, we adopt the same optimizer, learning rate, and batch size configuration as in the entity detector pre-training stage. In contrast with the two-stage SGG model, we refine the parameters of the detector in the joint learning rather than freezing the detector. We empirically observe that this refinement further improves the performance of entity detection. We train the SGTR for the Visual Genome dataset for 8.39e4 iterations without learning rate decay by using an early-stopping strategy. For the Openimage dataset, we train the model with the indicator’s bounding boxes, and the learning rate is decreased by 0.1x after 1e5 iterations.

We re-implement the previous two-stage methods (BGNN [6] and RelDN [13]) in our codebase by using the same configuration as the released codes. For fair comparison, we replace the ResNeXt-101 FPN backbone with the ResNet-101 backbone to learn the model. We also apply the one-stage HOI works (AS-Net [2] and HOTR [5]) on the SGG. We use the hyper-parameters of the models reported by the authors. All experiments are conducted by training the model until convergence.

### 3. Correspondence Matrix for Assembling

For clarity, we will use the correspondence matrix between the subject entity and predicate $M_s$ to introduce the details. The correspondence matrix is determined by the distance function, which takes the semantic outputs (e.g., bounding boxes $B$, classification $P$ of the entity detector and entity indicator of the predicate structural decoder) as input. Specifically, the distance function consists of two parts: spatial matching distance $d_{loc} \in \mathbb{R}^{N_r \times N_e}$ and category matching distance $d_{cls} \in \mathbb{R}^{N_r \times N_e}$, as shown in Eq. 1.

$$M_s^{*} = d_{loc}(B_s, B_e) \cdot d_{cls}(P_s, P_e)$$

Each element in the correspondence matrix $M_s^{*}$ is calculated by pairing the $N_r$ predicate predictions with $N_e$ entity predictions, as shown in the following equations:

$$M_{i,j}^{s} = d_{loc}(B_s(i), B_e(j)) \cdot d_{cls}(P_s(i), P_e(j))$$

$$= d_{loc}(b_s,i, b_e,j) \cdot d_{cls}(p_s,i, p_e,j)$$

where $i \in [0, N_e]$, $j \in [0, N_r]$ for enumerating each pair between the predicate proposal and entity set.

Then we present the two components of the distance function, $d_{loc}$ and $d_{cls}$. Specifically, the $d_{loc}$ consists of the $d_{giou} \in \mathbb{R}^{N_r \times N_e}$ and $d_{center} \in \mathbb{R}^{N_r \times N_r}$, as show in Eq. 4.

$$d_{loc}(b_s, b_e) = \frac{d_{giou}(b_s, b_e)}{d_{center}(b_s, b_e)}$$

Concretely, the $d_{giou}$ is the clipped GIOU of the entity and the indicator’s bounding boxes, and $d_{center}$ is the L1 distance between the bounding boxes’ centers in Eq. 5, 6. The center points-based matching has also been adopted in HOI methods [2, 7, 10].

$$d_{giou}(b_s, b_e) = \max(\min(GIOU(b_s, b_e), 0), 1)$$

$$d_{center}(b_s, b_e) = \| [x_s, y_s] - [x_e, y_e] \|_1$$

Here $[x_s, y_s]$ and $[x_e, y_e]$ are the normalized center coordinates of the bounding box in $b_s$ and $b_e$, respectively. For the $d_{cls}$, we use the cosine distance to calculate the similarity of the classification distribution between two entity predictions, as shown in following equation:

$$d_{cls}(p_s, p_e) = \frac{p_s \cdot p_e^T}{||p_s|| \cdot ||p_e||}$$

### 4. Matching Cost and Loss Function

#### 4.1. Triplets Matching Cost

We use the set-matching strategy to supervise the relationship predictions $T = \{(b_s^e, p_s^e, b_e^p, p_p^e, b_p)\}$. To obtain the matches, we need to calculate and minimize the matching cost $C \in \mathbb{R}^{N_{gt} \times N_{gt}}$ between the $N_r$ relationship predictions and the $N_{gt}$ GT relationships. Concretely, the matching cost $C$ includes two parts: the predicate cost $C_p$ and the entity cost $C_e$, as:

$$C = \lambda_p C_p + \lambda_e C_e$$

where $\lambda_p, \lambda_e$ is the coefficients of two cost terms.

The predicate cost, $C_p(i,j)$ between the $i$-th predicate prediction and the $j$-th ground-truth relationship is computed according to the predicate classification distribution and location prediction in Eq. 9:

$$C_p(i,j) = \exp(-p_{p,i}^{gt} \cdot p_{p,j}^T) + \| b_{p,i} - b_{p,j}^{gt} \|_1$$

where $p_{p,i} \in \mathbb{R}^{1 \times C_p}$ is the $i$-th predicate label of the $j$-th ground truth relationship. Similarly, $b_{p,i} \in \mathbb{R}^{1 \times 4}$ and $b_{p,j}^{gt} \in \mathbb{R}^{1 \times 4}$ are the center coordinates of the entity pair from the $i$-th relationship prediction and the $j$-th ground truth relationship, respectively.

The entity cost $C_e(i,j)$ between the $i$-th predicted relationship and $j$-th ground-truth relationship is given by:

$$C_e(i,j) = w_{giou} \cdot \prod_{* \in \{s,o\}} \exp\left(-d_{giou}(b_{e,i}^*, b_{gt,j}^*)\right)$$

$$+ w_1 \cdot \sum_{* \in \{s,o\}} \| b_{e,i}^* - b_{gt,j}^* \|_1$$

$$+ w_{cls} \cdot \prod_{* \in \{s,o\}} \exp\left(-p_{e,j}^{(s,gt)} \cdot p_{e,j}^T\right)$$
where the $b_{s,i}^*$ and $p_{o,i}^*$ are the $i$-th subject/object entity box and category distribution of relationship $\mathcal{T}$ after graph assembling, respectively. The $b_{s,j}^*$ and $p_{o,j}^*$ is the subject/object bounding boxes and one-hot entity category label from $j$-th ground truth relationships.

### 4.2. Loss Calculation

Our total loss $L$ is composed of entity detector loss $L_{enc}$ and predicate node generator loss $L_{pre}$:

$$L = L_{enc} + L_{pre}$$  \hspace{1cm} (13)

The entity detector loss $L_{enc}$ is calculated independently by following the same design in DETR [1].

The loss of predicate node generator $L_{pre}$ is determined by the prediction and ground-the relationships according to the matching index $I_{tri} \in \mathbb{N}^{N_{gt}}$. The $I_{tri}$ stores the index of matched predictions for each GT relationship. Specifically, the predicate node generator loss consists of entity indicator loss $L_{i}^{pre}$ and predicate sub-decoder loss $L_{p}^{pre}$:

$$L_{pre} = L_{i}^{pre} + L_{p}^{pre}$$  \hspace{1cm} (14)

For loss of predicate sub-decoder loss $L_{p}^{pre}$, we have:

$$L_{p}^{pre} = \sum_{i}^{N_{gt}} \left( \|b_{p,i}^*-b_{gt,i}^*\|_1 + CE(p_{p,i}^*, I_{tri}^*) \right)$$  \hspace{1cm} (15)

where $b_{p,i}^*$ and $b_{gt,i}^*$ is the subject/object entity box and category distribution of the GT relationship and prediction, respectively. The CE denotes the cross entropy loss between the predicate classification $b_{p,i}^*$ and the GT predicate category one-hot vector $p_{p,i}^*$.

For the entity indicator loss $L_{i}^{pre}$:

$$L_{i}^{pre} = \sum_{s,o} (L_{ent,loc}^* + L_{ent,cls}^*)$$  \hspace{1cm} (16)

where $\{ s,o \}$ indicates the subject/object role of an entity in relationships. The indicator loss is composed of two factors, $L_{ent,loc}^*$ and $L_{ent,cls}^*$, for two types of semantic representation: bounding boxes $b_s$, $b_o$ and classification $p_s$, $p_o$.

$$L_{ent,loc}^* = \sum_{i}^{N_{gt}} \left( \|b_s^*, I_{tri}^* - b_o^*\|_1 + 1 - GIOU(b_s^*, I_{tri}^*, b_o^*) \right)$$  \hspace{1cm} (17)

$$L_{ent,cls}^* = \sum_{i}^{N_{gt}} CE(p_s^*, I_{tri}^*, p_o^*)$$  \hspace{1cm} (18)

The $L_{ent,loc}^*$ is computed by the $L1$ distance and GIoU loss between the bounding box outputs $b_s^*, I_{tri}^*$ and ground-truth entity boxes $b_{gt,i}^*$. The $L_{ent,cls}^*$ is calculated from the cross entropy loss of classification prediction $p_s, I_{tri}^*$ according to ground-truth entities' category $p_{o,i}^*$.

### 5. Social Impacts

Our method has no direct potential negative impact, one possible negative impact is that SGG may serve as a base module for surveillance abuse.

### References


