Abstract

Scene graph generation is an important task in computer vision aimed at improving the semantic understanding of the visual world. In this task, the model needs to detect objects and predict visual relationships between them. Most of the existing models predict relationships in parallel assuming their independence. While there are different ways to capture these dependencies, we explore a conditional approach motivated by the sequence-to-sequence (Seq2Seq) formalism. Different from the previous research, our proposed model predicts visual relationships one at a time in an autoregressive manner by explicitly conditioning on the already predicted relationships. Drawing from translation models in NLP, we propose an encoder-decoder model built using Transformers where the encoder captures global context and long range interactions. The decoder then makes sequential predictions by conditioning on the scene graph constructed so far. In addition, we introduce a novel reinforcement learning-based training strategy tailored to Seq2Seq scene graph generation. By using a self-critical policy gradient training approach with Monte Carlo search we directly optimize for the (mean) recall metrics and bridge the gap between training and evaluation. Experimental results on two public benchmark datasets demonstrate that our Seq2Seq learning approach achieves strong empirical performance, outperforming previous state-of-the-art, while remaining efficient in terms of training and inference time. Full code for this work is available here: https://github.com/layer6ai-labs/SGG-Seq2Seq.

1. Introduction

Analyzing natural images containing multiple objects and complex interactions between them is a challenging task. We consider a common formulation of this task, scene graph generation (SGG) [51], in which given an image, we need to detect and predict objects and relationships between them in the form of a scene graph [18]. SGG is important for many applications at the intersection of computer vision and language, such as VQA [61, 16, 39, 15, 10, 26], image captioning [56, 14], retrieval [18, 2, 43] and others [1, 53].

Humans successfully understand images by alternating between two primary steps: sequentially attending to different regions of the image and applying high-level reasoning about these regions [47]. The two steps are recurrent until the image is understood, so the overall process is inherently both sequential and conditional. The benefit of sequential conditioning also translates to machine learning models for the SGG task as demonstrated by the example in Figure 1. Given an image of a beach scene, we aim to predict the correct relationship (predicate) between “man” and “surfboard”. At the start, when the model is not shown any other relationships (grayed out portion in the top image) and hence there is no conditioning, the model predicts “on” with high likelihood, as this is one of the most common predicates occurring with “man” and “surfboard” in
the dataset. In the next step we reveal neighboring relationships to the model and condition on them (non-shaded regions in the bottom image). After seeing relationships \( \langle \text{man}, \text{wear}, \text{shoe} \rangle \) and \( \langle \text{man}, \text{on}, \text{beach} \rangle \), the model infers that since the man is wearing shoes and there is an adjacent man on the beach, the more appropriate relationship is “holding”. Consequently the probabilities of “on” and “riding” drop. This example illustrates how sequential conditioning can help resolve ambiguities and reduce bias learned from the training data.

The majority of SGG methods, except for a few notable exceptions such as Neural Motifs (NM) [60], predict the final relationship labels in parallel making a severely limiting assumption of independence among the triplets. In NM, global context from detected objects is encoded via an LSTM, but the final relationship prediction is still done independently for each pair of objects and no conditioning is applied. In contrast, we quantitatively analyze the importance of incorporating both the sequential and conditional properties (§ 2). Based on that analysis and inspired by the Transformer architecture showing strong results both in neural machine translation [45] and computer vision [11, 4], we propose a Seq2Seq model that exploits sequential conditioning. We design a conditional Transformer decoder that sequentially leverages already predicted relationships to adjust its beliefs about future predictions.

Another important limitation of the previous SGG works is related to the way the models are trained and evaluated. In particular, common evaluation metrics, recall [51] and mean recall (mRecall) [6, 44], are not directly tied to the training objective of the SGG models, which typically minimize the cross-entropy loss [60]. The problem is exacerbated by the fact that these metrics focus on different and often conflicting properties, so training a single model that maximizes both metrics is challenging [43, 22]. For example, recall is dominated by frequent relationships [60], while mRecall assigns an equal weight to both frequent and rare relationships [6, 64]. A common method to improve the model on the target metric is to introduce an inductive bias favouring the metric via a carefully designed loss function [23, 29, 41] or features [60, 64]. To improve the model on the target metric, we take a different approach and leverage a reinforcement learning (RL)-based training strategy that enables the direct optimization of the target metric, bridging the gap between training and evaluation. The RL approach also aligns well with our Seq2Seq model as we train our RL policy to make sequential relationship predictions in an optimal order w.r.t. to the target metric (reward).

In summary, this paper makes the following contributions:

- Inspired by neural machine translation and our conditional SGG analysis (§ 2), we propose an encoder-decoder model based on Transformers with a sequential autoregressive decoder (§ 4).
- We introduce an RL training strategy that enables the direct optimization of the target metrics, bridging the gap between training and evaluation (§ 4.4.2), by removing the exposure bias. In particular, we employ a Monte Carlo search self-critical policy gradient training approach to accurately estimate the action-value function for our model (§ 4.4.2).
- We obtain state-of-the-art results on both recall and mRecall metrics while maintaining computational efficiency during training and inference (§ 5).

2. Conditional Scene Graph Generation

A scene graph [18] is defined as a set of objects and the relationships between them. We define a categorical triplet using a subject, object and their relationship. For instance \( y_m = \langle \text{man}, \text{on}, \text{surfboard} \rangle \), is the \( m \)th triplet in the image. The scene graph can be viewed as a set of \( M \) such triplets \( \{y_m\}_1^M \). We further assume some canonical order of triplets (e.g. from the left of the image to the right [60]) and define an ordered triplet set \( Y_{1:M} = \{y_1, \ldots, y_M \} \). Applying the chain rule, we define the conditional SGG as a task of sequentially inferring the relationship triplets conditioned on all previously predicted triplets:

\[
p(Y_{1:M}) = \prod_{m=1}^{M} p(y_m | \bigcap_{j=1}^{m-1} y_j)
\]

In the above formulation we ignore visual features assuming that all predictions are conditioned on the image in the way specific to a particular method (see § 4). The majority of SGG methods assume conditional independence of triplets and predict all triplets in parallel. To demonstrate that this assumption is limiting we analyse the relationship co-occurrence in the Visual Genome (VG) dataset [25]. We follow a setup similar to [60], and first compute the co-occurrence likelihoods between pairs of relationship triplets \( p(y_2 | y_1) \) using the training set of VG. We observe a strong co-occurrence bias, with most \( p(y_2 | y_1) \) distributions being highly peaked (Figure 2, top). For example, for \( \langle \text{man}, \text{on}, \text{beach} \rangle \) there are only a few triplets such as \( \langle \text{man}, \text{wearing, shorts} \rangle \) and \( \langle \text{man}, \text{holding, surfboard} \rangle \) that co-occur frequently in the dataset (Figure 2, top left). By extending this example to three triplets \( p(y_3 | y_1, y_2) \), the distribution remains steep but the top co-occurring triplets change. For example, by conditioning on \( \langle \text{man}, \text{on}, \text{beach} \rangle \) and \( \langle \text{horse}, \text{on}, \text{beach} \rangle \) the top triplet changes to \( \langle \text{person}, \text{riding, horse} \rangle \) clearly demonstrating the effect of knowing that both “man” and “horse” are on the beach (Figure 2, top right).

We can expand the sequence of conditioning triplets to arbitrary size \( m \). To avoid the prohibitive cost of computing joint probabilities we use a simple approximation:
Figure 2: VG analysis based on scene graph annotations without considering images or bounding boxes. (top) Triplet co-occurrence in scene graphs has a peaked distribution. The top candidates can be selected reasonably well when conditioned on a single triplet (left) and two triplets (right), including rare relationships (in red). (bottom) Conditioning allows us to accurately predict a triplet given other (already predicted or ground truth) triplets in the image. This motivates our conditional inference pipeline.

\[ p(y_{m+1}|Y_{1:m}) = \sum_{y_{m}} p(y_{m+1}|y_{m}) \]. For small \( m \) we verified that this approximation closely matches the target distribution, particularly when it is used to rank the top co-occurring relationships. We then simulate a sequential prediction process and use these probabilities to predict triplets in a given test image while conditioning on increasingly larger set of ground truth triplets (i.e. assuming a perfect model): \( p(y_{m+1}|Y_{1:m}) \), where \( m \) is the history size. At each step we sample one ground truth triplet per test image and check if it is in the top-100 triplets co-occurring with triplets predicted from \( Y_{1:m} \), we then average this accuracy across all test images. To imitate the effect of the recently introduced mRecall metric [6, 44], we also compute the per-predicate accuracy for all images and then average over all predicate classes. We found that conditioning on more triplets (longer history \( m \)) substantially improves both accuracies (Figure 2, bottom). Our finding indicates that the model is able to make an increasingly better prediction by leveraging the information from the revealed relationships. This motivates our Seq2Seq model described in the following sections.

3. Related Work

SGG. Scene graphs were proposed as a visually grounded graphical structure of an image with localized objects as nodes and pairwise predicates as the edges [18, 52]. The visual relationship detection/scene graph generation task was formalized first by [31]. The standard relationship detection pipeline [31] comprises object detection with off-the-shelf fine-tuned weights to predict objects, and pairwise predicate classification [31, 65, 62, 58, 9, 63, 27, 51, 60, 57, 32, 33]. We follow this protocol to disentangle object detection error from relationship detection and focus on reasoning over the relationships. Recent research [64, 6, 44, 43, 29, 54, 8] address the long-tail issue by improving mean recall as opposed to simple recall dominated by most frequent relationships. Several recent works [43, 41, 23, 22, 30, 20] focus on compositional generalization metrics in SGG, which is an interesting avenue to apply our method in the future.

Contextual Models. Context has been shown to be useful in generating better predictions in several recent works [17, 7, 48]. Our work is closest to Neural Motifs (NM) [60]. In NM, the Bi-LSTMs model is used to capture the global context and structural regularities in scene graphs. NM constructs the global context from all detected objects in a given image. NM then leverages the global context to refine feature-level representations for individual objects and possible relationships between them. In contrast, our approach first encodes global context and then sequentially updates it by leveraging information from triplets that have been decoded so far. This is achieved by applying a Transformer architecture that enables joint conditioning on all predicted history which we demonstrate to be important for maximizing SGG performance.

Attention. Prior work [55, 35] that applies attention in visual relationship detection start by defining a nearest neighbor graph. Attention is used to capture information about the graph structure by encoding it similar to graph attention networks (GAT) [46]. In particular, Graph R-CNN[55] applies GAT over visual similarity while graph self-attention [35] additionally embeds a pair of object features and linguistic relationships jointly. Transformers [45] have been successfully adopted in computer vision [36, 11, 12, 13]. We use Transformers in an encoder decoder based architecture. However, unlike other attention based methods in this domain, our decoder makes sequential predictions conditioned on previous outputs and the model is trained in an auto regressive way. In a parallel work [24] Relational Transformers were applied to visual relationship detection. However, our model is explicitly conditioned on triplet predictions and uses Reinforcement Learning (RL) to optimize for the specific metric.

Reinforcement Learning. Using RL for SGG has remained under explored. [28] built a semantic action graph using language priors and formulated SGG as a single agent decision-making process. CMAT [5] proposed a counterfactual critic model using multi agent policy. DG-PGNN [19] proposed a probabilistic model together with Q-learning to infer a scene graph in a sequential node-by-node fashion. In contrast to [28, 5, 19], our model works on subject-predicate-object triplets and leverages Transformers to capture global context. In addition, our work explores the use of mean recall as a reward to tackle the long-tail distribution on SGG datasets.
4. Our Approach

In this section we describe our encoder-decoder approach. First, we briefly overview the multi-head self-attention (MSA) block and the encoder architecture. Then we describe our novel relationship decoder and optimization approach. We employ the Transformer architecture [45, 11] due to its effectiveness in capturing long-term dependencies and the ability to train in parallel. At the core of the Transformer is the multi-head self-attention:

\[
\text{MSA}(Q, K, V) = \text{Concat}(\text{SA}_1, \text{SA}_2, \ldots, \text{SA}_h)W^O
\]

where \(\text{SA}_i = \text{Softmax} (\frac{Q_iK_i^T}{\sqrt{d}})V_i\)

4.1. Overall Pipeline

First, given an image, the object detector (e.g. Faster-RCNN [37]) returns a set of bounding boxes of \(N\) object instances \(X = \{x_1, \ldots, x_N\}\). The goal of our visual relationship prediction model is to learn the mapping \(f: Y = f(X)\), where \(Y = \{y_1, \ldots, y_M\}\) refers to the ordered set of \(M\) relationship triplets in the image. Representations \(X\) are obtained based on spatial, semantic and visual features following [64]. \(X\) are treated the same way as text tokens in natural language processing, and we apply an encoder-decoder architecture to predict the relationships \(Y\). Objects representations are passed through the Transformer encoder to produce contextualised embeddings that contain both object specific information and global scene context. The decoder then operates in an autoregressive manner, at each time step consuming the previously generated relationship triplets to generate the next triplet. The full architecture diagram is shown in Figure 3.

4.2. Object Encoder

To capture the global context of a scene, we follow the standard Transformer encoder architecture and form a cascade of \(B\) identical blocks, where each block applies multi-head self-attention. Formally, the new object embeddings produced by the \((b + 1)\)-th Transformer block are given by:

\[
X_{b+1} = \text{FFN}(\text{MSA}(X_b, X_b, X_b)) + X_b
\]
Table 1: VG mRecall results.

<table>
<thead>
<tr>
<th>mRecall@</th>
<th>PRDCLS</th>
<th>SGCLS</th>
<th>SGDET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>MOTIFS+TDE(GATE) [43]</td>
<td>18.5</td>
<td>24.9</td>
<td>28.3</td>
</tr>
<tr>
<td>GB-NET [50]</td>
<td>18.9</td>
<td>25.3</td>
<td>28.4</td>
</tr>
<tr>
<td>VTrans+TDE(GATE) [43]</td>
<td>18.4</td>
<td>25.4</td>
<td>28.7</td>
</tr>
<tr>
<td>VTree+TDE(SUM) [43]</td>
<td>21.3</td>
<td>26.1</td>
<td>30.2</td>
</tr>
<tr>
<td>Seq2Seq - teacher forcing</td>
<td>6.3</td>
<td>8.5</td>
<td>10.2</td>
</tr>
<tr>
<td>Seq2Seq - scheduled sampling</td>
<td>7.1</td>
<td>8.5</td>
<td>9.9</td>
</tr>
</tbody>
</table>

Table 2: VRD recall results.

<table>
<thead>
<tr>
<th>Recall@</th>
<th>Relationship Detection</th>
<th>Phrase Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>k = 1</td>
<td>50 100 100</td>
<td>50 100 100</td>
</tr>
<tr>
<td>free k</td>
<td>20 50 100</td>
<td>20 50 100</td>
</tr>
</tbody>
</table>

4.3. Relationship Decoder

Our proposed relationship decoder predicts visual relationships one at a time in an autoregressive manner. Given the contextualized object features $X_B \in \mathbb{R}^{N \times D}$ and $m$ already predicted visual relationships $Y_{1:m}$, the goal of the relationship decoder is to learn the conditional probability of the $(m+1)$-th visual relationship $p(\hat{y}_{m+1} | X_B, Y_{1:m})$ that maximizes the probability of $(m+1)$-th triplet in the ground truth sequence. Note that during training, the predicted sequence $\hat{Y}_{1:m}$ can be replaced with the ground truth sequence $Y_{1:m}$, but during inference the model conditions only on its own predictions.

**Decoder Input.** Given the predicted visual relationships sequence $\hat{Y}_{1:m}$, we first convert this sequence to vector representations to be used as input into the decoder model. For each predicted triplet $\hat{y} \in \hat{Y}_{1:m}$, we concatenate the embeddings of $(X_B[i], E[r], X_B[j])$, where $i$ and $j$ are object and subject indices in $\hat{y}$. $E$ is the learned embeddings for all predicates in the training set, and $r$ is the predicate index in $\hat{y}$. The concatenated embeddings is then fed into a fully-connected layer and projected into $D$-dimensional triplet representations. At the beginning of the decoding, we use a learned $D$-dimensional vector as the first input embedding to start the decoding. This is equivalent to the special $<$SOS$>$ (start of sequence) token in traditional sequence transduction tasks, where the $<$SOS$>$ token informs the decoder to start generating the output sequence.

The projected $D$-dimensional triplet representations are fed into a stack of $K$ identical Transformer decoder blocks. Each Transformer decoder block consists of MSA (Equation 5) followed by cross-attention between contextualized object embeddings $X_B$ and the intermediate triplet representations $H$ (Equation 6):

$$H_{k+1} = \text{MSA}(Y_k, Y_k, Y_k) + Y_k$$  \hspace{1cm} (5)

$$Y_{k+1} = \text{FFN}(H_{k+1} + \text{MSA}(H_{k+1}, X_B, X_B))$$  \hspace{1cm} (6)

where $H_{k+1}$ is the intermediate output from the $(k+1)$-th self-attention block. The cross-attention enables the model to correlate the current relationship predictions in $\hat{Y}_{1:m}$ with all detected objects in the visual scene, and update its beliefs about what other relationships are present. After $K$ blocks, the final representation $Y_K$ is used to predict the next relationship triplet. We concatenate the history embedding in $Y_K$ with all possible remaining object pairs that can have a relationship, predict the relationship for each pair, and take the triplet with the highest probability:

$$p(\hat{y} | X_B, Y_{1:m}) = \text{Softmax}_\gamma(\text{Concat}(X_B[i], Y_K[m], X_B[j]) \ast W_r)$$  \hspace{1cm} (7)

Here, $i$ and $j$ are subject and object indices in $\hat{y}$ and $r$ is the predicate between them. $Y_K[m]$ is the $m$-th embedding in the $m \times D$ output $Y_K$ of the decoder, and represents the contextualised embedding of the last predicted relationship triplet at step $m$. The triplet with the highest probability $\hat{y}_{m+1}$ is taken as the decoder prediction at step $m + 1$, and the process is repeated until termination criteria is reached.

4.4. Model Optimization

We consider two approaches for training our model. One leverages the standard “teacher forcing” framework, while the other is our proposed strategy based on reinforcement learning.

4.4.1 Teacher Forcing

Sequence-to-sequence models are typically trained with the teacher forcing strategy [50]. At each time step, instead of conditioning on the model’s own predictions in an autoregressive fashion, the model is provided with the ground truth from the previous step to learn the conditional probability of the next ground truth triplet $p(y_{m+1} | X_B, Y_{1:m})$. To remove the bias from introducing a particular order during training, we randomly shuffle the ground truth relationship triplets for each image to form the input sequence, and this shuffling is repeated for each batch. The teacher forcing objective only maximizes the probability of positive examples, i.e., pairs of objects that have a relationship. However, in the task of visual relationship prediction, learning to predict which objects do not have a relationship is equally critical to model performance [50]. To incorporate negative examples, at each decoding step, we randomly sample $L$ object pairs that do not have relationships, and train the model to
predict the “no_relationship” predicate for these pairs. Denoting the $l$-th sampled negative triplet as $y_l^-$, the teacher forcing objective for our model can be formulated as:

$$L = - \sum_{m=1}^{M} \left[ \log p(y_{m+1}\mid X_B, Y_{1:m}) + \sum_{l=1}^{L} \log p(y_l^-\mid X_B, Y_{1:m}) \right]$$  

(8)

### 4.4.2 Reinforcement Learning

The teacher forcing objective generally leads to stable and fast learning. However, optimizing with maximum likelihood does not necessarily translate to optimal performance on the target metric such as recall. There are two main reasons for this. First, during training the model is provided with the ground truth as input history, while during inference the model has to rely on its own predictions. Second, the maximum likelihood objective does not directly optimize for the target metric, resulting in the discrepancy between training and evaluation. We address both of these problems by proposing an RL optimization approach. RL enables the model to explore different policies during training to learn the one that yields maximum reward at inference time. We define rewards based on the target metric which allows for direct optimization of non-differentiable metrics and reduces the gap between training and inference.

We focus on two common metrics, recall and mRecall, but analogous approach may be extended to other metrics. Previous works noted a trade-off between recall and mRecall [43, 59, 22]. Therefore, we design our reward function as a convex combination of recall and mRecall scores, and use a hyperparameter $\alpha \in [0, 1]$ to control their relative importance. Suppose that $\hat{Y}_{1:M'}$ is the model’s predicted triplets for a given image, where $M'$ is the length of predicted sequence. We denote the recall and mRecall scores for $\hat{Y}_{1:M'}$ as $r(\hat{Y}_{1:M'})$ and $mr(\hat{Y}_{1:M'})$ respectively. The reward is then defined as:

$$R(\hat{Y}_{1:M'}) = \alpha \cdot r(\hat{Y}_{1:M'}) + (1 - \alpha) \cdot mr(\hat{Y}_{1:M'})$$  

(9)

A major challenge in applying RL to the SGG task is the lack of intermediate reward, since the reward can only be computed on the final predicted triplet set, while we aim for the model to learn the optimal action at each decoding step. Following [42], the RL objective with no intermediate reward can be defined as:

$$L_{RL} = \mathbb{E} \left[ R(\hat{Y}_{1:M'})|s_0 \right] = \sum_{y_1 \in \mathcal{V}} p(y_1|s_0) Q(s_0, y_1)$$  

(10)

where $s_0$ is the initial state, and $Q(s, a)$ is the action-value function defined as the expected accumulative reward starting from state $s$, taking action $a$, and then following the policy specified by the model. $\mathcal{V}$ is the set of all possible triplets that the model can predict in the first decoding step.

To estimate $Q(s, a)$, we note that in the final $M'$-th step the model outputs $\hat{y}_{M'}$, so we have $Q(s = Y_{1:M'-1}, a = \hat{y}_{M'}) = R(\hat{Y}_{1:M'})$. However, to evaluate the intermediate step, the action-value should reflect not only the quality of the already predicted relationships, but also the quality of the predictions that the model can potentially generate in the future. To this end we apply a Monte Carlo search [40] with roll-out to sample the remaining predictions. For each intermediate state $s = Y_{1:m}$, we sample the remaining $M' - m$ visual relationships $T$ times. The sampling is done according to the Softmax probabilities (Equation 7) at each decoding step from $m + 1$ to $M'$. We then concatenate each sample $\hat{Y}_{m+1:M'}$ with the already predicted visual relationships $Y_{1:m}$ to form the complete prediction and compute the reward. The action-value $Q(s = Y_{1:m}, a = \hat{y}_{m+1})$ for an intermediate state $s = Y_{1:m}$ where $m < M'$ can thus be

| Table 3: VG Recall results for the SGDET, SGCLS and PRDCLS tasks with and without graph constraints. *We omit RelDN results on SGCLS and PRDCLS where they evaluate using subject and object pairs from the ground truth, which is inconsistent with other work. |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| **Recall:**     | **SGDET**       | **Graph**       | **SGCLS**       | **PRDCLS**      | **SGDET**       | **Graph**       |
| 20   | 50   | 100   | 20   | 50   | 100   | 20   | 50   | 100   | 20   | 50   | 100   | 20   | 50   | 100   | 20   | 50   | 100   |
| L    | L    | L    | L    | L    | L    | L    | L    | L    | L    | L    | L    | L    | L    | L    | L    | L    |
| 6.5  | 8.1  | 8.2  | 18.2 | 21.8 | 22.6 | 47.9 | 54.1 | 55.4 | 9.7  | 11.3 | 26.5 | 30.0 | 68.0 | 75.2 |
| Message Passing[51] | -    | 3.4  | 4.2  | -    | 21.7 | 24.4 | -    | 44.8 | 53.0 | -    | -    | -    | -    | -    | -    |
| Graph R-CNN[55] | -    | 11.4 | 13.7 | -    | 29.6 | 31.6 | -    | 54.2 | 59.1 | -    | -    | -    | -    | -    | -    |
| Message Passing+[60] | 14.6 | 20.7 | 24.5 | 31.7 | 34.6 | 35.4 | 52.7 | 59.3 | 61.3 | 22.0 | 27.4 | 43.4 | 47.2 | 75.2 | 83.6 |
| Frequency+Overlap[60] | 20.1 | 26.2 | 30.1 | 29.3 | 32.3 | 32.9 | 53.6 | 60.6 | 62.2 | 28.6 | 34.9 | 40.0 | 40.3 | 43.4 | 75.7 | 82.9 |
| MotifNet-LeftRight[60] | 21.4 | 27.2 | 30.3 | 32.9 | 35.8 | 36.5 | 58.5 | 65.2 | 67.1 | 30.5 | 35.8 | 44.5 | 47.7 | 81.1 | 88.3 |
| RelDN[64] | 21.1 | 28.3 | 32.7 | -    | -    | -    | -    | -    | -    | 30.4 | 36.7 | -    | -    | -    | -    |
| VCTree[44] | 22.0 | 27.9 | 31.3 | 35.2 | 38.1 | 38.8 | 60.1 | 66.4 | 68.1 | -    | -    | -    | -    | -    | -    |
| HetHi[49] | 21.6 | 27.5 | 30.9 | 33.8 | 36.6 | 37.3 | 59.8 | 66.3 | 68.1 | -    | -    | -    | -    | -    | -    |
| GB-Net[59] | -    | 26.4 | 30.0 | -    | 38.0 | 38.8 | -    | 66.6 | 68.2 | 29.4 | 35.1 | 47.7 | 51.1 | 83.6 | 90.5 |
| Seq2Seq - RL (ours) | 22.1 | 30.9 | 34.4 | 34.5 | 38.3 | 39.0 | 60.3 | 66.4 | 68.5 | 30.9 | 37.0 | 46.9 | 51.2 | 83.6 | 90.8 |

**Ablation Results**
- **Seq2Seq - encoder only**: 21.0 | 28.5 | 31.9 | 32.8 | 35.4 | 36.2 | 59.2 | 64.8 | 67.1 |
- **Seq2Seq - teacher forcing**: 21.4 | 29.6 | 33.0 | 33.1 | 36.9 | 37.4 | 59.4 | 65.3 | 67.2 |
- **Seq2Seq - scheduled sampling**: 21.7 | 30.1 | 34.1 | 33.6 | 37.6 | 38.4 | 59.6 | 65.8 | 67.9 |

*We omit RelDN results on SGCLS and PRDCLS where they evaluate using subject and object pairs from the ground truth, which is inconsistent with other work.*

---

[50x105]M

importance. Suppose that $\alpha \in [0, 1]$. Therefore, we design our reward function defined as the expected accumulative reward starting from state $s$, taking action $a$, and then following the policy specified by the model. $\mathcal{V}$ is the set of all possible triplets that the model can predict in the first decoding step.

To estimate $Q(s, a)$, we note that in the final $M'$-th step the model outputs $\hat{y}_{M'}$, so we have $Q(s = Y_{1:M'-1}, a = \hat{y}_{M'}) = R(\hat{Y}_{1:M'})$. However, to evaluate the intermediate step, the action-value should reflect not only the quality of the already predicted relationships, but also the quality of the predictions that the model can potentially generate in the future. To this end we apply a Monte Carlo search [40] with roll-out to sample the remaining predictions. For each intermediate state $s = Y_{1:m}$, we sample the remaining $M' - m$ visual relationships $T$ times. The sampling is done according to the Softmax probabilities (Equation 7) at each decoding step from $m + 1$ to $M'$. We then concatenate each sample $\hat{Y}_{m+1:M'}$ with the already predicted visual relationships $Y_{1:m}$ to form the complete prediction and compute the reward. The action-value $Q(s = Y_{1:m}, a = \hat{y}_{m+1})$ for an intermediate state $s = Y_{1:m}$ where $m < M'$ can thus be
where \( \hat{Y}_{1:M} \) is iteratively defined as the next-state value starting from \( Y_{1:m} \). It can be observed that the action-value function is iteratively defined as the next-state value starting from \( Y_{1:m} \) and rolling out to the end.

The gradient of the objective function \( L_{RL} \) with respect to the model parameters is derived using policy gradient [42] as:

\[
\nabla L_{RL} \approx \sum_{m=1}^{M'} \sum_{\hat{y}_{m+1} \in \mathcal{Y}} \nabla p(\hat{y}_{m+1}|\hat{Y}_{1:m}) \cdot Q(\hat{Y}_{1:m}, \hat{y}_{m+1}) \\
= \sum_{m=1}^{M'} \sum_{\hat{y}_{m+1} \in \mathcal{Y}} p(\hat{y}_{m+1}|\hat{Y}_{1:m}) \nabla \log p(\hat{y}_{m+1}|\hat{Y}_{1:m}) \cdot Q(\hat{Y}_{1:m}, \hat{y}_{m+1})
\]  

(12)

Note that if we directly use the weighted sum of recall and mRecall in Equation 9, most sequences will get a positive reward even if they are highly sub-optimal. To provide a stronger signal to the model, we instead use self-critical training [38], and take the difference between rewards for sampled and greedily decoded sequences as the reward. This encourages the model to explore policies that lead to better samples than greedy decoding.

5. Experiments

We evaluate our model on two public SGG benchmarks, Visual Relationship Detection (VRD) [31] and Visual Genome (VG) [25]. On both datasets we compare our approach to an extensive set of leading baselines described in the Related Work section.

**VRD.** We use the dataset split from [58] and report recall@50 and 100. Following [58], we benchmark our model on two standard tasks, Relationship Detection and Phrase Detection, with and without the graph constraint denoted as \( k = 1 \) and free-\( k \) respectively. The graph constraint limits prediction to one relationship predicate for each object pair, while no graph constraint accepts an arbitrary number of predicates.

**VG.** We use the dataset split from [51] and the VGG detection model weights from [64]. Following the evaluation protocol from [51], we compute recall and mRecall on three tasks: scene graph detection (SGDET), scene graph classification (SGCLS), and predicate classification (PRDCLS). As in VRD we benchmark performance with and without the graph constraint.

We train our model using the Adam optimizer [21] with \( \beta_1 = 0.9, \beta_2 = 0.999 \), batch size of 4096, and learning rate of 1e-3. We apply linear learning rate warmup over the first 1K steps, and cosine learning rate decay afterwards. We use the standard Transformer blocks for both encoder and decoder [45], with four encoder blocks and two decoder blocks. All blocks have embedding size of 128 and 4 attention heads. The model is trained for 500 epochs on VRD and 2K epochs on VG. We train the model with teacher forcing over the first half of the epochs, and alternate between teacher forcing and reinforcement learning over the second half of the epochs. We set the number of playout samples \( T \) to be 16 for the reinforcement learning loss.

5.1. Results and Analysis

mRecall results on VG are shown in Table 1, and recall results on the VRD and VG datasets are shown in Tables 2 and 3 respectively. We observe that our approach, denoted as Seq2Seq, outperforms the baselines on most tasks and recall thresholds. We improve over the prior state-of-the-art by 4.6% (+1.6) on all metrics on average, with most improvements observed on the VG SGDet task.

To investigate the contribution from each component of our model we conduct extensive ablation experiments, and ablation results are shown at the bottom of each table. We first remove the sequential decoder and ablate the non-sequential architecture (Seq2Seq-encoder only) where we only use the Transformer encoder to obtain the contextualized bounding box representations. For each pair of objects, we concatenate their contextualized representations, and use an FNN layer followed by a Softmax layer over the pos-
We observe in Tables 1 and 2 that on both VRD and VG datasets removing the sequential decoder from the model leads to considerable performance degradation on all tasks and recall thresholds. This demonstrates the effectiveness of conditional sequential decoding for scene graph generation. Similarly, training with teacher forcing also hurts performance relative to the full RL training. Scheduled sampling partially closes the gap between teacher forcing and RL but doesn’t eliminate it completely, and performance still drops by over a point on some tasks. These results indicate that optimizing for the target metric while simultaneously learning to condition on predictions instead of ground truth is highly beneficial for the SGG task. Lastly, to estimate the effect that training sequence sampling has on performance, we repeated SGDET Seq2Seq-RL training 10 times with different seeds. We observed that the variation in performance across training runs was very small with a standard deviation of 0.13.

Recall vs mRecall. We perform a hyperparameter sensitivity analysis for α in our reward function in Equation 9. We vary α from 0 to 1, and report recall and mRecall @100 results on the VG SGDET task shown in Figure 4. We observe that the recall and mRecall metrics are inversely correlated, i.e., improvement in recall results in degradation of mRecall, and vice versa. This is consistent with previously reported findings by other works in this area [43, 59, 22]. An additional advantage of our RL approach is that it allows to directly control the degree to which each metric contributes to the reward, and thus directly optimize the model to achieve the desired balance between the two metrics.

Training and Inference Speed. We evaluate and compare the average per image training and inference time for our Seq2Seq approach and several leading baseline models, results are shown in Figure 5. For fair comparison all models are trained and timed on the same server. For our Seq2Seq approach, we report training times for the three ablation architectures described above and the full RL model. When the model is trained with teacher forcing only, the Transformer architecture enables parallel decoding via causal masking which significantly accelerates forward and backward passes. Training with scheduled sampling or RL requires sampling the predicted relationship triplets one at a time, and is thus less efficient. At inference time, we benchmark block decoding where multiple top predicted triplets (block size) are taken at each decoding step. Block decoding can significantly accelerate inference by requiring fewer forward passes through the model. But, as seen in Figure 5, it can also lead to accuracy degradation by reducing the effectiveness of sequential conditioning. In all cases, even decoding one triplet at a time (block size = 1), inference time in our model is highly competitive with leading baselines while we also substantially improve recall performance.

Qualitative Results. Figure 6 shows two scenes with motorcycles. In both scenes, independent prediction has difficulty distinguishing “riding” vs “near” relationships between “person” and “motorcycle”. On the right we show a conditional decoding sequence. In both cases our model first identifies easier grounding relationships for motorcycle and person, which then enables it to correctly predict “riding” and “near” for the top and bottom images respectively. We see a very substantial increase in probability relative to the independent prediction (15% → 59% for “riding” and 31% → 58% for “near”).
References


computer vision and pattern recognition, pages 1347–1356, 2017. 3, 5


