Iterative Scene Graph Generation

Siddhesh Khandelwal[1,2] **and Leonid Sigal**[1,2,3]
[1]Department of Computer Science, University of British Columbia
[2]Vector Institute for AI
[3]CIFAR AI Chair
{skhandel, lsigal}@cs.ubc.ca

Abstract

The task of scene graph generation entails identifying object entities and their corresponding interaction predicates in a given image (or video). Due to the combinatorially large solution space, existing approaches to scene graph generation assume certain factorization of the joint distribution to make the estimation feasible (*e.g.*, assuming that objects are conditionally independent of predicate predictions). However, this fixed factorization is not ideal under all scenarios (*e.g.*, for images where an object entailed in interaction is small and not discernible on its own). In this work, we propose a novel framework for scene graph generation that addresses this limitation, as well as introduces dynamic conditioning on the image, using message passing in a Markov Random Field. This is implemented as an iterative refinement procedure wherein each modification is conditioned on the graph generated in the previous iteration. This conditioning across refinement steps allows joint reasoning over entities and relations. This framework is realized via a novel and end-to-end trainable transformer-based architecture. In addition, the proposed framework can improve existing approach performance. Through extensive experiments on Visual Genome [30] and Action Genome [25] benchmark datasets we show improved performance on the scene graph generation task. The code is available at github.com/ubc-vision/IterativeSG.

1 Introduction

Scene graphs allow for structured understanding of objects and their interactions within a scene. A *scene graph* is a graph where nodes represent objects within the scene, each detailed by the class label and spatial location, and the edges capture the relationships between object pairs. These relationships are usually represented by a <subject, predicate, object> triple. Effectively generating such graphs, from either images or videos, has emerged as a core problem in computer vision [14, 35, 39, 46, 50, 52, 54]. Scene graph representations can be leveraged to improve performance on a variety of complex high-level tasks like VQA [24, 47], Image Captioning [17, 53], and Image Generation [26].

The task of scene graph generation involves estimating the conditional distribution of the relationship triplets given an image. Naively modelling this distribution is often infeasible as the space of possible relationship triplets is considerably larger than the space of possible subjects, objects, and predicates. To circumvent this issue, existing methods factorize the aforementioned distribution into easy-to-estimate conditionals. For example, *two-stage* approaches, like [11, 47, 55], follow the graphical model image $\rightarrow$ [subject, object] $\rightarrow$ predicate, wherein the subjects and objects are independently obtained via a pre-trained detector like Faster-RCNN [41]. These are then consumed by a downstream network to estimate predicates. Any such factorization induces conditional dependencies (and independencies) that heavily influence model characteristics. For example, in the aforesaid graphical model, errors made during the estimation of the subjects and objects are naturally propagated towards the predicate distribution, which makes the estimation of predicates involving

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

classes with poor detectors challenging. Furthermore, the assumed fixed factorization might not always be optimal. Having information about predicates in an image can help narrow down the space of possible subjects/objects, *e.g.*, predicate `wearing` makes it likely that the subject is a `person`.

Additionally, due to the *two-stage* nature of most existing scene graph approaches (barring very recent methods like [14, 31, 35]), the image feature representations obtained from a pre-trained task-oblivious detector might not be optimally catered towards the scene graph generation problem. Intuitively, one can imagine that the information required to accurately localize an object *might not* necessarily be sufficient for predicate prediction, and by extension, accurate scene graph generation. Morever, two-stage approaches often suffer with efficiency issues as the detected objects are required to be paired up before predicate assignment. Doing so naively, by pairing all possible objects [50], results in quadratic number of pairs that need to be considered. Traditional approaches deal with this using heuristics, such as IoU-based threshold-based pairing [47, 55].

In this work, we aim to alleviate the previously mentioned issues arising from a fixed factorization and potentially limited object-centric feature representations by proposing a *general* framework wherein the subject, objects, and predicates can be inferred jointly (*i.e.*, depend on one another), while simultaneously avoiding the complexity of exponential search space of relational triplets. This is achieved by performing message passing within a Markov Random Field (MRF) defined by components of a relational triplet (Figure 1(a)). Unrolling this message passing is equivalent to an *iterative* refinement procedure (Figure 1(b)), where each message passing stage takes in the estimate from the previous step. Our proposed framework models this iterative refinement strategy by first producing a scene graph estimate using a traditional factorization, and then systematically improving it over multiple refinement steps, wherein each refinement is conditioned on the graph generated in the previous iteration. This conditioning across refinement steps compensates for the conditional independence assumptions and lets our framework jointly reason over the subjects ($\mathbf{s}$), objects ($\mathbf{o}$), and predicates ($\mathbf{p}$).

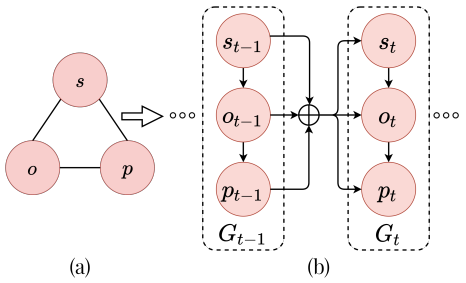

Figure 1: **Iterative Refinement for Scene Graph Generation.** By unrolling the message passing in a Markov Random Field (a), our proposed approach essentially models an iterative refinement process (b) wherein each modification step is conditioned on the previously generated graph estimate.

Contributions. To realize the aforementioned iterative framework, we propose a novel and intuitive transformer [48] based architecture. On a technical level, our model defines three separate multi-layer multi-output synchronized decoders, wherein each decoder layer is tasked with modeling either the subject, object, or predicate components of a relationship triplets. Therefore, the combined outputs from each layer of the three decoders generates a scene graph estimate. The inputs to each decoder layer are conditioned to enable joint reasoning across decoders and effective refinement of previous layer estimates. This conditioning is achieved *implicitly* via a novel joint loss, and *explicitly* via cross-decoder and layer-wise attention. Additionally, each decoder layer is also conditioned on the image features, which are provided by a shared encoder. As our proposed model is end-to-end trainable, it addresses the limitation of two-stage approaches, allowing image features to directly adopt to the scene graph generation task. Finally, to tackle the long-tail nature of the scene graph predicate classes [11], we employ a loss weighting strategy to enable flexible trade-off between dominant (head) and underrepresented (tail) predicate classes in the long-tail distribution. In contrast to data sampling strategies [11, 32], this has a benefit of not requiring additional fine-tuning of models with sampled data post training. We illustrate that our proposed architecture achieves state-of-the-art performance on two benchmark datasets – Visual Genome [30] and Action Genome [25]; and thoroughly analyze effectiveness of the approach as a function of the refinement steps, design choices employed and as a generic add-on to an existing, MOTIF [55], architecture.

2 Related Work

Scene Graph Generation. Scene graph generation has emerged as a popular research area in the vision community [11, 28, 31, 35, 39, 40, 46, 47, 50, 52, 54, 55]. Existing scene graph generation methods can be broadly categorized as either *one-stage* or *two-stage* approaches. The first step of the predominant approach – the *two-stage* methods – involves pre-training a strong object detector

for all object classes in the dataset, usually using detector like Faster-RCNN [41]. The graph generation network is then built on top of the object information (bounding boxes and corresponding features) obtained from the detector. This second step entails *freezing* the detector parameters and *solely* training the graph generation network. The graph generation network is realized via different architectures such as Bi-directional LSTMs [55], Tree-LSTMs [47], Graph Neural Networks [52], and other message passing algorithms [40, 50]. The limitation of all two-stage approaches is apparent – the graph generation network has no influence over the detector and object features. One stage approaches overcome this obstacle by employing architectures that allow end-to-end training, such as fully convolutional networks [35] and transformer based models [7, 14, 31, 43]. The joint optimization enables interaction between the detection and the graph generation modules, leading to better scene graphs. However, all the aforementioned methods operate under the assumption of a fixed factorized model, inducing conditional independencies that might not be ideal under all scenarios.

Long-Tail Recognition. The task of scene graph generation suffers from the challenge of long-tail recognition due to the combinatorial nature of relational triplets (both object and predicate distributions are skewed). Long-tail recognition, in general, is a well studied problem in literature. *Data sampling* is a popular strategy [3, 20, 21, 27, 37, 42, 58] wherein the training data is modified to either over-represent tail classes (oversampling) or under-represent the head classes (undersampling). Specific to the task of scene graph generation, [11, 32] have explored the use of data sampling strategies to improve performance on tail predicate classes. On the contrary, *Loss re-weighting* strategies [1, 9, 13, 27, 34] assign higher weights or impose larger decision boundaries for tail classes. [51] recently adopted this paradigm for the task of scene graph generation by proposing a re-weighting strategy based on correlations between predicate classes. Conceptually similar to [51], we propose a novel re-weighting strategy and illustrate its effectiveness both on its own, and in combination with data resampling.

Transformer Models. In [48], authors introduced a new attention-based architecture called transformers for the task of machine translation, doing away with recurrent or convolutional architectures. Transformers have since widely been adopted for a variety of tasks, such as object detection [2, 38, 57], image captioning [8, 23], and image generation [12]. More recently, scene graph generation methods have also adopted transformer architectures [7, 14, 31, 43], owing to their end-to-end trainable nature and parallelism. Perhaps conceptually closest, [14], which focuses on visual relation detection (VRD) and not scene graph prediction, leverages the composite nature of relationships to simultaneously decode an entire relationship triplet alongside its constituent subject, object, and predicate components. Unlike [14], we do not explicitly model subject-predicate-object "sum" composites, resulting in a simpler architecture, and leverage iterative refinement procedure that relies on a different factorized attention mechanism. In the latest, *concurrent* work, [31] generates a set of entities and predicates separately, and then utilizes a graph assembly procedure to match the predicates to a pair of entities. In contrast to our formulation, this precludes conditioning of entities on predicate estimates; therefore relying on accuracy of less contextualized predictions for subsequent pairing.

3 Formulation

For a given image $\mathbf{I}$, a scene graph $\mathbf{G}$ can be represented as a set of triplets $\mathbf{G} = \{\mathbf{r}_i\}_{i \leqslant n} = \{(\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_i)\}_{i \leqslant n}$, where $\mathbf{r}_i$ denotes the i-th triplet $(\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_i)$, and n denotes the total number of triplets. The subject $\mathbf{s}_i$ denotes a tuple $(\mathbf{s}_{i,c}, \mathbf{s}_{i,b})$, where $\mathbf{s}_{i,c} \in \mathbb{R}^{\eta}$ is the one-hot class label, and $\mathbf{s}_{i,b} \in \mathbb{R}^4$ is the corresponding bounding box coordinates. η is the total number of possible entity classes in the dataset. The object $\mathbf{o}_i$ and predicate $\mathbf{p}_i$ can similarly be represented as tuples $(\mathbf{o}_{i,c}, \mathbf{o}_{i,b})$ and $(\mathbf{p}_{i,c}, \mathbf{p}_{i,b})$ respectively. Note that, $\mathbf{p}_{i,b}$ corresponds to the box formed by the centers' of $\mathbf{s}_{i,b}$ and $\mathbf{o}_{i,b}$ as the diagonally opposite coordinates. Additionally, $\mathbf{p}_{i,c} \in \mathbb{R}^{\upsilon}$ represents the corresponding one-hot predicate label between the pair $(\mathbf{s}_i, \mathbf{o}_i)$, where υ is the total number of possible predicate classes in the dataset. Then the task of scene graph generation can be thought of as modelling the conditional distribution $\Pr(\mathbf{G} \mid \mathbf{I})$. This distribution can be expressed as a product of conditionals,

$$\Pr\left(\mathbf{G} \mid \mathbf{I}\right) = \Pr\left(\{\mathbf{s}_i\} \mid \mathbf{I}\right) \cdot \Pr\left(\{\mathbf{o}_i\} \mid \{\mathbf{s}_i\}, \mathbf{I}\right) \cdot \Pr\left(\{\mathbf{p}_i\} \mid \{\mathbf{s}_i\}, \{\mathbf{o}_i\}, \mathbf{I}\right) \tag{1}$$

where $\{.\}$ denotes a set. For brevity, we omit explicitly mentioning the total number of set elements n throughout the paper. Existing approaches model this product of conditionals by making some underlying assumptions. For example, [11, 47, 55] assume conditional independence between $\mathbf{s}_i$ and $\mathbf{o}_i$ as they rely on heuristics to obtain the entity pairs. However, modelling the conditional in Equation

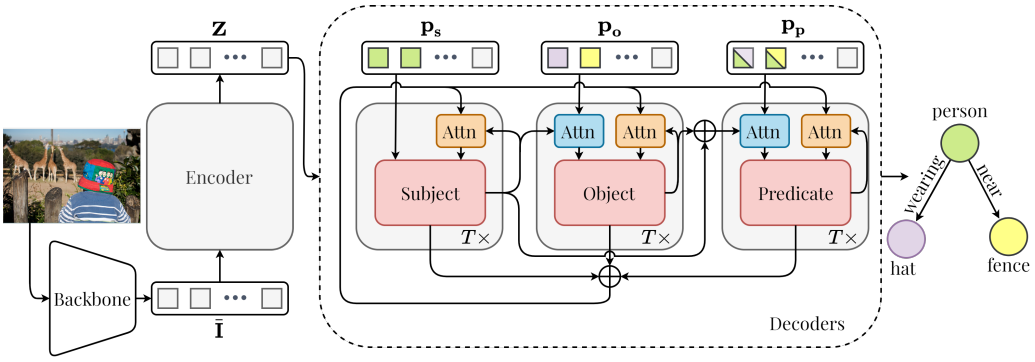

Figure 2: **Transformer Architecture for Iterative Refinement.** For a given image, the model extracts features via a convolutional backbone and a transformer encoder. The individual components of a relationship triplet are generated using separate subject, object, and predicate multi-layer decoders. The inputs to each layer of the decoder is appropriately conditioned. For example, for the predicate decoder, the positional embeddings are conditioned on the outputs generated by the subject and object decoders (blue `Attn` module) and the queries are conditioned on the previously generated graph estimate (orange `Attn` module). The model is additionally implicitly conditioned and trained in an end-to-end fashion using a joint matching loss.

1 (or any other equivalent factorization) in such a "one-shot" manner makes certain assumptions on the flow of information, which, in this case, is from $\mathbf{s}_i \rightarrow \mathbf{o}_i \rightarrow \mathbf{p}_i$. Therefore, any errors made during the estimation of $\mathbf{s}_i$ are naturally propagated towards the estimation of $\mathbf{o}_i$ and $\mathbf{p}_i$. Additionally, the subject (or object) estimation procedure $\Pr\left(\{\mathbf{s}_i\} \mid \mathbf{I}\right)$ is completely oblivious to the estimated predicate $\mathbf{p}_i$. Having access to such information can help the subject (or object) predictor update its beliefs and significantly narrow down the space of feasible entity pairs.

Contrary to existing works, our proposed formulation moves away from the "one-shot" generation ideology described in Equation 1. We instead argue for modelling the task of scene graph generation as an iterative refinement procedure, wherein the scene graph estimate at step t, $\mathbf{G}^t$, is dependent on the previous estimates $\{\mathbf{G}^{t'}\}_{t'<t}$. Formally, our aim is to model the conditional distribution $\Pr\left(\mathbf{G}^t \mid \{\mathbf{G}^{t'}\}_{t'<t}, \mathbf{I}\right)$. Assuming Markov property holds, this can be conveniently factorized as,

$$\begin{aligned}\Pr\left(\mathbf{G}^t \mid \mathbf{G}^{t-1}, \mathbf{I}\right) = &\overbrace{\Pr\left(\{\mathbf{s}_i^t\} \mid \mathbf{G}^{t-1}, \mathbf{I}\right)}^{\text{Subject Predictor}} \cdot \overbrace{\Pr\left(\{\mathbf{o}_i^t\} \mid \{\mathbf{s}_i^t\}, \mathbf{G}^{t-1}, \mathbf{I}\right)}^{\text{Object Predictor}} \\ &\cdot \underbrace{\Pr\left(\{\mathbf{p}_i^t\} \mid \{\mathbf{s}_i^t\}, \{\mathbf{o}_i^t\}, \mathbf{G}^{t-1}, \mathbf{I}\right)}_{\text{Predicate Predictor}}.\end{aligned} \tag{2}$$

Note that even though we assume the flow of information to be from $\mathbf{s}_i^t \rightarrow \mathbf{o}_i^t \rightarrow \mathbf{p}_i^t$ for $\mathbf{G}^t$, conditioning on the previous graph estimate $\mathbf{G}^{t-1}$ allows the subject, object, and predicate predictors to jointly reason and update their beliefs, leading to better predictions. Additionally, the framework described in Equation 2 is model agnostic and can be implemented using any of the existing architectures.

4 Transformer Based Iterative Generation

As described in Section 3, our proposed iterative scene graph generation formulation is agnostic to the architectural choices used to realise the subject, object, and predicate predictors. In this section we provide a realization of the proposed formulation in Equation 2 using Transformer networks [48]. The choice of using transformer networks is motivated by their natural tendency to model iterative refinement behaviour under the Markov property, wherein each layer of the transformer decoder takes as input the output of the previous layer. Our novel end-to-end trainable transformer based iterative generation architecture builds on top of the DETR [2] framework, which is shown to be effective for the task of object detection. Our proposed model architecture is shown in Figure 3.

Given an image, our proposed approach first obtains corresponding image features using a combination of a convolutional backbone and a multi-layer transformer encoder, akin to DETR [2]. These image features are used as inputs to the subject, object, and predicate predictors, each implemented as

a multi-layer Transformer decoder. To generate a scene graph estimate $\widehat{\mathbf{G}}^t$ at step t in accordance with Equation 2, the queries used in each predictor decoder are appropriately conditioned. For example, in the case of the predicate decoder, its input queries are conditioned on the subject ($\{\widehat{\mathbf{s}}_i^t\}$) and object ($\{\widehat{\mathbf{o}}_i^t\}$) estimates at step t. Additionally, the input to each predictor decoder is infused with *all* decoder estimates $\{(\widehat{\mathbf{s}}_i^{t-1}, \widehat{\mathbf{p}}_i^{t-1}, \widehat{\mathbf{o}}_i^{t-1})\}$ from the previous step $t-1$ via a structured attentional mechanism. The entire model is trained end-to-end, with a novel joint loss applied at each step t to ensure the generation of a valid scene graph at every level. This section describes these components in detail.

4.1 Image Encoder

Similar to DETR [2], for each image $\mathbf{I}$, our proposed architecture uses a deep convolutional network (like ResNet [22]) to obtain image level spatial map $\bar{\mathbf{I}} \in \mathbb{R}^{c \times w \times h}$, where c is the number of channels, and w, h correspond to the spatial dimensions. A multi-layer encoder f_e then transforms $\bar{\mathbf{I}}$ into a position-aware flattened image feature representation $\mathbf{Z} \in \mathbb{R}^{d \times wh}$, where $d < c$.

4.2 Predictor Decoders

Our approach models each of the subject, object, and predicate predictors using a multi-layer transformer decoder [2], which are denoted by $f_{\mathbf{s}}$, $f_{\mathbf{o}}$, and $f_{\mathbf{p}}$ respectively. The t-th layer of each decoder, denoted as $f_{\mathbf{x}}^t; \mathbf{x} \in \{\mathbf{s}, \mathbf{o}, \mathbf{p}\}$, is tasked with generating the step t scene graph $\mathbf{G}^t$. Therefore, at each step t, the decoders output a set of n feature representations $\{\mathbf{q}_{\mathbf{x},i}^t\}; \mathbf{x} \in \{\mathbf{s}, \mathbf{o}, \mathbf{p}\}$, which are transformed into a set of triplet estimates $\{(\widehat{\mathbf{s}}_i^t, \widehat{\mathbf{p}}_i^t, \widehat{\mathbf{o}}_i^t)\}$ via fully-connected feed forward layers.

Specifically, for a decoder $f_{\mathbf{x}}; \mathbf{x} \in \{\mathbf{s}, \mathbf{o}, \mathbf{p}\}$, an arbitrary layer t takes as input a set of queries $\{\mathbf{q}_{\mathbf{x},i}^{t-1}\}$ and a set of learnable positional encodings $\{\mathbf{p}_{\mathbf{x},i}\}$, where $\mathbf{q}_{\mathbf{x},i}^{t-1}; \mathbf{p}_{\mathbf{x},i} \in \mathbb{R}^d$. The output representations $\{\mathbf{q}_{\mathbf{x},i}^t\}$ are then obtained via a combination of self-attention between the input queries, and encoder-decoder attention across the encoder output $\mathbf{Z}$. These attention modules allow the decoder to jointly reason across all queries, while simultaneously incorporating context from the input image.

At a given step t, naively using the inputs $\{\mathbf{q}_{\mathbf{x},i}^{t-1}\}$ and $\{\mathbf{p}_{\mathbf{x},i}\}$ to generate $\mathbf{G}^t$ forgoes leveraging the compositional property of relations. As described Equation 2, for any arbitrary step t, our proposed formulation entails *two* types of conditioning for better scene graph estimation. The first involves conditioning decoders on the step t outputs, specifically the object decoder $f_{\mathbf{o}}^t$ on the subject decoder $f_{\mathbf{s}}^t$, and the predicate decoder $f_{\mathbf{p}}^t$ on both $f_{\mathbf{s}}^t$ and $f_{\mathbf{o}}^t$. The second requires all three decoder layers at step t to be conditioned on the outputs generated at step $t-1$. To effectively implement this design, we modify the inputs to each decoder layer $f_{\mathbf{x}}^t$. Specifically, the positional encodings $\{\mathbf{p}_{\mathbf{x},i}\}$ are modified to condition them on step t outputs, and the queries $\mathbf{q}_{\mathbf{x},i}^{t-1}$ are updated to incorporate information from the previous step $t-1$. Modifying the positional encoding and queries separately allows the model to easily disentangle and differentiate between the two conditioning types.

Conditional Positional Encodings. At a particular step t, the conditional positional encodings for the three decoders are obtained as,

$$\begin{aligned} \{\widehat{\mathbf{p}}_{\mathbf{s},i}^t\} &= \{\mathbf{p}_{\mathbf{s},i}\}; \quad \{\widehat{\mathbf{p}}_{\mathbf{o},i}^t\} = \{\mathbf{p}_{\mathbf{o},i}\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\mathbf{p}_{\mathbf{o},i}\}, \{\widetilde{\mathbf{q}}_{\mathbf{s},i}^t\}, \{\mathbf{q}_{\mathbf{s},i}^t\}\right)\right) \\ \{\widehat{\mathbf{p}}_{\mathbf{p},i}^t\} &= \{\mathbf{p}_{\mathbf{p},i}\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\mathbf{p}_{\mathbf{p},i}\}, \{\widetilde{\mathbf{q}}_{\mathbf{s},i}^t \oplus \widetilde{\mathbf{q}}_{\mathbf{o},i}^t\}, \{\mathbf{q}_{\mathbf{s},i}^t \oplus \mathbf{q}_{\mathbf{o},i}^t\}\right)\right) \end{aligned} \tag{3}$$

where `MultiHead(Q, K, V)` is the Multi-Head Attention module introduced in [48], `FFN(.)` is a fully-connected feed forward network, and $\oplus$ is the concatenation operation. Additionally, $\widetilde{\mathbf{q}}_{\mathbf{x},i}^t = \mathbf{q}_{\mathbf{x},i}^t + \mathbf{p}_{\mathbf{x},i}^t; \mathbf{x} \in \{\mathbf{s}, \mathbf{o}, \mathbf{p}\}$ is the position-aware query.

Conditional Queries. Similarly, for a step t, conditional queries for the subject decoder are defined,

$$\{\widehat{\mathbf{q}}_{\mathbf{s},i}^{t-1}\} = \{\mathbf{q}_{\mathbf{s},i}^{t-1}\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\widetilde{\mathbf{q}}_{\mathbf{s},i}^{t-1}\}, \{\widetilde{\mathbf{q}}_{\mathbf{s},i}^{t-1} \oplus \widetilde{\mathbf{q}}_{\mathbf{o},i}^{t-1} \oplus \widetilde{\mathbf{q}}_{\mathbf{p},i}^{t-1}\}, \{\mathbf{q}_{\mathbf{s},i}^{t-1} \oplus \mathbf{q}_{\mathbf{o},i}^{t-1} \oplus \mathbf{q}_{\mathbf{p},i}^{t-1}\}\right)\right) \tag{4}$$

$\widehat{\mathbf{q}}_{\mathbf{o},i}^{t-1}$ and $\widehat{\mathbf{q}}_{\mathbf{p},i}^{t-1}$ are defined identically. For a decoder layer $f_{\mathbf{x}}^t; \mathbf{x} \in \{\mathbf{s}, \mathbf{o}, \mathbf{p}\}$ we use the conditioned positional encodings $\{\widehat{\mathbf{p}}_{\mathbf{x},i}^t\}$ and queries $\{\widehat{\mathbf{q}}_{\mathbf{x},i}^{t-1}\}$ as input.

4.3 End-To-End Learning

Our proposed transformer based refinement architecture can be trained in an end-to-end fashion. To ensure that a valid scene graph is generated at every level, we propose a novel joint loss that is applied at each step t. Therefore, the combined loss $\mathcal{L}$ can be expressed as $\mathcal{L} = \sum_t \mathcal{L}^t = \sum_t \mathcal{L}_{\mathbf{s}}^t + \mathcal{L}_{\mathbf{o}}^t + \mathcal{L}_{\mathbf{p}}^t$,

where $\mathcal{L}^t_{\mathbf{x}}; \mathbf{x} \in \{\mathbf{s}, \mathbf{o}, \mathbf{p}\}$ represents the loss applied to the t-th layer of the decoder $f_{\mathbf{x}}$. Our approach generates a fixed-size set of n triplet estimates $\{\widehat{\mathbf{r}}^t_i\} = \{(\widehat{\mathbf{s}}^t_i, \widehat{\mathbf{p}}^t_i, \widehat{\mathbf{o}}^t_i)\}$ at each step t, where n is larger than the number of ground truth relations for a given image. Therefore, in order to effectively optimize the proposed model, we obtain an optimal bipartite matching between the predicted and ground truth triplets. Note that, contrary to the matching algorithm in [2], our proposed matching is defined over triplets rather than individual entities. Additionally, instead of independently computing the loss over each decoder layer as in [2], our loss computes a joint matching across all refinement layers.

Let $\mathbf{G} = \{\mathbf{r}_i\} = \{(\mathbf{s}_i, \mathbf{p}_i, \mathbf{o}_i)\}$ denote the ground truth scene graph for an image $\mathbf{I}$. Note that, as the number of ground truth relations is less than n, we convert $\mathbf{G}$ to a n-sized set by padding it with $\varnothing$ (no relation). The goal then is to find a bipartite matching between the ground truth graph $\mathbf{G}$ and the set of all graph estimates $\{\widehat{\mathbf{G}}^t\}$ that minimizes the *joint matching cost*. Specifically, assuming σ to be a valid permutation of n elements,

$$\widehat{\sigma} = \arg\min_{\sigma} \sum_{t} \sum_{i}^{n} \mathcal{L}_{\text{rel}} \left(\mathbf{r}_i, \widehat{\mathbf{r}}^t_{\sigma(i)} \right) \tag{5}$$

where the pair-wise relation matching cost $\mathcal{L}_{\text{rel}}$ is defined as,

$$\mathcal{L}_{\text{rel}} \left(\mathbf{r}_i, \widehat{\mathbf{r}}^t_{\sigma(i)} \right) = -\mathbb{1}_{\{\mathbf{r}_i \neq \varnothing\}} \begin{bmatrix} \widehat{\mathbf{s}}^t_{\sigma(i),c} \cdot \mathbf{s}_{i,c} - \mathcal{L}_{\text{box}} \left(\widehat{\mathbf{s}}^t_{\sigma(i),b}, \mathbf{s}_{i,b} \right) \\ + \widehat{\mathbf{o}}^t_{\sigma(i),c} \cdot \mathbf{o}_{i,c} - \mathcal{L}_{\text{box}} \left(\widehat{\mathbf{o}}^t_{\sigma(i),b}, \mathbf{o}_{i,b} \right) \\ + \widehat{\mathbf{p}}^t_{\sigma(i),c} \cdot \mathbf{p}_{i,c} - \mathcal{L}_{\text{box}} \left(\widehat{\mathbf{p}}^t_{\sigma(i),b}, \mathbf{p}_{i,b} \right) \end{bmatrix} \tag{6}$$

where $\cdot$ is the vector dot product, and $\mathcal{L}_{\text{box}}$ is a combination of the L-1 and generalized IoU losses. Please refer to Section 3 for clarification on the notations. The optimal permutation $\widehat{\sigma}$ can then be computed using the Hungarian algorithm. The loss $\mathcal{L}^t_{\mathbf{s}}$ is then defined as,

$$\mathcal{L}^t_{\mathbf{s}} = \sum_{i=1}^{n} \left[-\log \left(\widehat{\mathbf{s}}^t_{\widehat{\sigma}(i),c} \cdot \mathbf{s}_{i,c} \right) + \mathbb{1}_{\{\mathbf{r}_i \neq \varnothing\}} \mathcal{L}_{\text{box}} \left(\widehat{\mathbf{s}}^t_{\widehat{\sigma}(i),b}, \mathbf{s}_{i,b} \right) \right] \tag{7}$$

$\mathcal{L}^t_{\mathbf{o}}$ and $\mathcal{L}^t_{\mathbf{p}}$ are defined identically. Note that as we use the same permutation $\widehat{\sigma}$ for all refinement layers t, it induces strong *implicit* dependencies between the subject, object, and predicate decoders. The potency of the aforementioned implicit conditioning is highlighted in the experiment section.

4.4 Loss Re-Weighting

Due to the inherent long-tail nature of the scene graph generation task, using an unbiased loss often leads to the model prioritizing the most common (*a.k.a.,* head) predicate classes like `has` and `on`, which have abundant training examples. To afford our proposed model the flexibility to achieve the trade-off between head/tail classes, we integrate a loss-reweighing scheme into the model training procedure. Note that, contrary to existing methods that do this post-hoc via finetuning the final layer of the trained network (see Section 2), we instead train the model with this weighting to allow the internal feature representations to reflect the desired trade-off. Note, to the best of our knowledge, our paper is the first to illustrate effectiveness of such a strategy for the task of scene graph generation. For a particular predicate class $c \in [1, \upsilon]$, we define the class weight w_c as $\max \left\{ (\alpha / f_c)^{\beta}, 1.0 \right\}$, where f_c is the frequency of the predicate class c in the training set, and $\{\alpha, \beta\}$ are scaling parameters. Note that this weighting scheme is similar to the data sampling strategy described in [19, 32]. However, instead of modifying the training set, we scale each class weight by the factor w_c when computing the predicate classification loss $\mathcal{L}^t_{\mathbf{p}}$. Therefore, $\mathcal{L}^t_{\mathbf{p}}$ can be defined similarly to Equation 7,

$$\mathcal{L}^t_{\mathbf{p}} = \sum_{i=1}^{n} \left[-w_c \log \left(\widehat{\mathbf{p}}^t_{\widehat{\sigma}(i),c} \cdot \mathbf{p}_{i,c} \right) + \mathbb{1}_{\{\mathbf{r}_i \neq \varnothing\}} \mathcal{L}_{\text{box}} \left(\widehat{\mathbf{p}}^t_{\widehat{\sigma}(i),b}, \mathbf{p}_{i,b} \right) \right]. \tag{8}$$

5 Experiments

We demonstrate the effectiveness of our proposed approach on two datasets,

Visual Genome [30]. This is a benchmark for scene graph generation. We use the common processed subset from [50], which contains $108k$ images, with 150 object and 50 predicate categories.

Action Genome [25]. This dataset provides scene graph annotations over videos in the Charades dataset [44] for the task of human-object interaction. It contains $9,848$ videos across 35 object

Table 1: **Scene Graph Generation on Visual Genome.** Mean Recall (mR@K), Recall (R@K), and Harmonic Recall (hR@K) shown for baselines and our approach. **B**=Backbone, **D**=Detector. Baseline results are borrowed from [31]. M indicates the number of top-k links used by the baseline [31]. Note that our approach implicitly assumes $M = 1$. For the approaches that use the ResNet-101 backbone [22] (akin to our method), the best result is highlighted in red, second best in blue.

B	D	#	Method	mR@50/100	R@50/100	hR@50/100	Head	Body	Tail
X101-FPN	Faster RCNN	①	RelDN [56]	6.0 / 7.3	31.4 / 35.9	10.1 / 12.1	-	-	-
		②	MOTIF [55]	5.5 / 6.8	32.1 / 36.9	9.4 / 11.5	-	-	-
		③	VCTree [47]	6.6 / 7.7	31.8 / 36.1	10.9 / 12.7	-	-	-
		④	BGNN [32]	10.7 / 12.6	31.0 / 35.8	15.9 / 18.6	34.0	12.9	6.0
		⑤	VCTree-TDE [46]	9.3 / 11.1	19.4 / 23.2	12.6 / 15.0	-	-	-
		⑥	VCTree-DLFE [6]	11.8 / 13.8	22.7 / 26.3	15.5 / 18.1	-	-	-
		⑦	VCTree-EBM [45]	9.7 / 11.6	20.5 / 24.7	13.2 / 15.8	-	-	-
		⑧	VCTree-BPLSA [18]	13.5 / 15.7	21.7 / 25.5	16.6 / 19.4	-	-	-
		⑨	DT2-ACBS [11]	22.0 / 24.4	15.0 / 16.3	17.8 / 19.5	-	-	-
ResNet-101		⑩	BGNN [32, 31]	8.6 / 10.3	28.2 / 33.8	13.2 / 15.8	29.1	12.6	2.2
		⑪	RelDN [56, 31]	4.4 / 5.4	30.3 / 34.8	7.7 / 9.3	31.3	2.3	0.0
	DETR	⑫	AS-Net [4]	6.1 / 7.2	18.7 / 21.1	9.2 / 10.7	19.6	7.7	2.7
		⑬	HOTR [29]	9.4 / 12.0	23.5 / 27.7	13.4 / 16.7	26.1	16.2	3.4
			Concurrent Work						
		⑭	SGTR$_{M=1}$ [31]	12.0 / 14.6	25.1 / 26.6	16.2 / 18.8	27.1	17.2	6.9
		⑮	SGTR$_{M=3}$ [31]	12.0 / 15.2	24.6 / 28.4	16.1 / 19.8	28.2	18.6	7.1
		⑯	SGTR$_{M=3,\text{BGNN [32]}}$ [31]	15.8 / 20.1	20.6 / 25.0	17.9 / 22.3	21.7	21.6	17.1
		⑰	Ours$_{(\alpha=0.0,\beta=*)}$	8.0 / 8.8	29.7 / 32.1	12.6 / 13.8	31.7	9.0	1.4
		⑱	Ours$_{(\alpha=0.14,\beta=0.5)}$	14.4 / 16.4	27.9 / 30.4	19.0 / 21.3	30.0	17.3	11.2
		⑲	Ours$_{(\alpha=0.07,\beta=0.75)}$	15.7 / 17.8	27.2 / 29.8	19.9 / 22.3	28.5	18.8	13.3
		⑳	Ours$_{(\alpha=0.14,\beta=0.75)}$	15.8 / 18.2	26.1 / 28.7	19.7 / 22.3	28.2	19.4	13.8
		㉑	Ours$_{(\alpha=0.14,\beta=0.75),\text{BGNN [32]}}$	17.1 / 19.2	22.9 / 25.7	19.6 / 22.0	24.4	20.2	16.4
		㉒	Ours$_{(\alpha=0.14,\beta=0.75),M=3}$	19.5 / 23.4	30.8 / 35.6	23.9 / 28.2	32.9	28.1	15.8

(excluding class `person`) and 25 relation categories. As not all video frames are annotated, consistent with prior work [16], we use the annotated frames provides by [25]. Additionally, as in [16], we remove frames without any `person` or object annotations as they do not provide usable scene graphs.

Implementation Details (*transformer-based approach*). We use ResNet-101 [22] as the backbone network for image feature extraction. Each of the subject, object, and predicate decoders have 6 layers, with a feature size of 256. The decoders use 300 queries. For training we use a batch size of 12 and initial learning rate of 10^{-4}, which is gradually decayed. Note that although our model predicts individual relation triplets, a graph is obtained by applying a non-maximum suppression (NMS) strategy [15, 41] to group the predicted subjects and objects into entity instances.

Implementation Details (*MOTIF*). For the re-implementation of the MOTIF [55] baseline and the subsequent augmentation of our proposed framework, we follow the same training procedure as [55]. Specifically, we assume the Faster-RCNN detector [41] with the ResNeXt-101-FPN [49] backbone. The detector is first pre-trained on the Visual Genome dataset [30]. As is the case with two-stage approaches, when learning the scene graph generator, the detector parameters are freezed. The specifics on how our approach is augmented to MOTIF is described in the supplementary (Sec. A).

Evaluation Metrics. To measure performance, we report results using standard scene graph evaluation metrics, namely Recall (R@K) [50] and Mean Recall (mR@K) [5, 47]. While recall is class agnostic, mean recall averages the recalls computed for each predicate category independently. Usually, higher R@K is indicative of better performance on dominant (head) classes, where as higher mR@K suggests better tail class performance. Recent methods [11, 32] have argued for the use of mean recall as it reduces influence of dominant relationships such as `on` and `has` on the metric. However, even for the same model architecture, one can trade-off R@K for mR@K by using long tail recognition techniques described in Section 2. This trade-off is seldom analyzed or reported, making it difficult to compare performance across methods. Therefore, inspired by the generalized zero-shot

Table 2: **Ablation of Model Components.** Mean Recall (mR@K) and Recall (R@K) reported on the Visual Genome dataset. **CAS**=Conditioning Across Steps (Eq. 4), **CWS**=Conditioning Within a Step (Eq. 3), **JL**=Joint Loss (Sec. 4.3).

#	CAS	CWS	JL	mR@20/50	R@20/50
①	✓	✓	✓	**11.8** / **15.8**	**21.0** / **26.1**
②				1.7 / 1.9	2.7 / 4.3
③			✓	11.2 / 14.7	20.1 / 25.4
④		✓	✓	11.5 / 15.3	20.9 / 26.1
⑤	✓		✓	11.7 / 15.4	20.9 / 25.9

Table 3: **Augmentation to MOTIF.** Mean Recall (mR@K) and Recall (R@K) is reported on the Visual Genome dataset for the baseline and the variant with our iterative formulation. † denotes our re-implementation of the method.

Model	t	mR@20/50	R@20/50
MOTIF† [55]		6.0 / 8.0	23.6 / 30.4
MOTIF†	1	6.0 / 8.1	23.7 / 30.6
+	2	6.2 / 8.4	23.9 / 30.7
Ours	3	**6.4** / **8.5**	**24.0** / **30.8**

Table 4: **Ablation of Loss Re-weighting Parameters.** We vary α, β (Sec. 4.4) and report recall, mean recall, and harmonic recall on the Visual Genome test set.

α	β	mR@20/50	R@20/50	hR@20/50
0.0	*	5.8 / 8.0	**24.2** / **29.7**	9.4 / 12.6
0.07	0.75	11.2 / 15.7	21.8 / 27.2	14.8 / 19.9
0.14	0.75	**11.8** / **15.8**	21.0 / 26.1	**15.1** / **19.7**
0.21	0.75	11.3 / 15.5	20.0 / 24.8	14.4 / 19.1
0.14	0.5	10.3 / 14.4	**22.6** / **27.9**	14.2 / 19.0
0.14	0.625	11.0 / 15.1	21.4 / 26.4	14.5 / 19.2
0.14	0.75	**11.8** / 15.8	21.0 / 26.1	**15.1** / **19.7**
0.14	0.875	11.2 / **15.9**	19.3 / 24.5	14.2 / 19.3

Table 5: **Ablation of Model Parameters.** ① = Our proposed transformer approach with 6 decoder layers. ② = transformer with 6 decoder layers devoid of our proposed refinement. ③ = transformer with 1 decoder layer devoid of our proposed refinement.

#	mR@20/50	R@20/50	Model Size
①	**11.8** / **15.8**	**21.0** / **26.1**	1×
②	1.7 / 1.9	2.7 / 4.3	0.9×
③	9.9 / 13.1	17.5 / 21.8	1.1×

learning harmonic average metric [10], we propose a new metric – *harmonic recall* (hR@K), defined as the harmonic mean of mR@K and R@K. Such a metric encourages methods to have a healthy balance between the head and tail class performance. Additionally, we report the performance of our model for different values of the loss weighing parameters described in Section 4.4. Furthermore, we also report the mR@100 on each long-tail category subset, namely *head*, *body*, and *tail*, as in [32].

5.1 Ablation Study

All the subsequent ablation studies are done on the Visual Genome dataset [30].

Model Components. We analyze the importance of each of our model components, namely the conditioning within a particular step t (CWS; Equation 3), the conditioning across steps (CAS; Equation 4), and the joint loss (JL; Section 4.3) in Table 2. For a fair comparison, all models are trained using the *same* loss weighing parameters, $\alpha = 0.14, \beta = 0.75$. It can be seen that our proposed novel joint loss provides a significant improvement in performance (③) when compared to the ablated model that uses separate losses (as in [2]; ②). This is largely due to the joint loss being able to induce strong implicit dependencies between the three decoders. As a consequence, even without CAS and CWS, the proposed joint loss by itself enables refinement. The two forms of conditioning (④-⑤) provide a structured pathway to incorporating knowledge from previously generated estimates or other triplet components in an *explicit* fashion, leading to improved scene graphs. CAS (⑤) builds upon the joint loss, integrating information that the aforementioned implicit conditioning is unable to capture. CWS (④) is complementary to the refinement process, and allows for more consistent graph generation within a step. As these two types of conditionings capture complementary information, using them together (①) further improves on performance.

Refinement Across Steps. To further analyze the effectiveness of our refinement procedure, we contrast the quality of scene graphs generated at each refinement step for a particular trained model ($\alpha = 0.14, \beta = 0.75$). We provide a detailed analysis in the supplementary (Table A1). Visually, Figure 3 highlights steady improvements in the graph quality over refinement steps. Owing to our structured conditioning, the model is able to improve on both predicate detection and entity localization over refinement iterations.

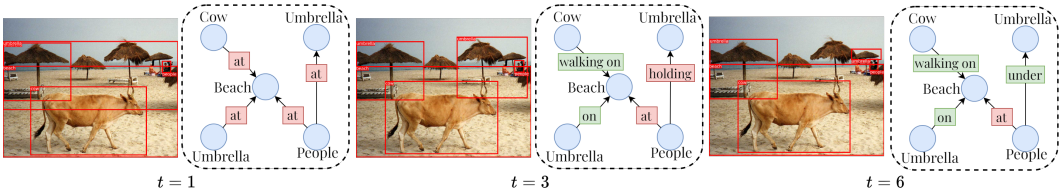

Figure 3: **Qualitative Refinement Analysis.** Graph estimates for different refinement steps shown. Colours red and green indicate incorrect and correct predictions respectively. Additional visualizations are shown in the supplementary (Section C.3).

Loss Re-Weighting. As mentioned in Section 4.4, we employ a simple loss re-weighting scheme, allowing for an easy trade-off between recall for mean recall. Some results with different weightings are reported in Table 1 (⑰–⑳). To investigate this further, we train our proposed model with different values for the parameters α and β, and report the performance in Table 4. We fix β in the top half of the table (first 4 rows), and α in the bottom half (last 4 rows). It can be seen that increasing α and β generally biases the model more towards the tail classes (higher mean recall), whereas lower values tend to favour the head classes (higher recall). As a consequence, our proposed approach is flexibly able to function in a wide range of recall and mean recall values. However, as is the case with every long-tail recognition approach, choosing values that are too high (*e.g.* $\alpha = 0.21, \beta = 0.75$) lead to poor performance due to the model being extremely biased and therefore unable to learn accurate feature representations on the popular classes.

Model Parameters. We additionally argue that the improvements obtained by our proposed refinement procedure are not a direct consequence of having an increased number of parameters. To corroborate this, we contrast our proposed model with alternatives with similar number of parameters but devoid of refinement in Table 5. ① corresponds to our transformer-based approach with 6 decoder layers that uses the proposed joint loss and both forms of conditioning. ② is a similar model with 6 decoder layers but bereft of the aforementioned proposed implicit and explicit conditionings. ③, on the other hand, is a *single* decoder layer transformer model without refinement, wherein the number of parameters are increased to be comparable to our proposed approach. The inferior performance of ② and ③ highlight that increasing model capacity either via making each layer larger or via adding more layers (increasing model depth) does not emulate refinement. Our proposed refinement procedure, realized by the joint loss and explicit conditionings, allows for better learning, and the performance gains observed cannot be attributed to having additional parameters.

5.2 Comparison to Existing Methods

Visual Genome. We report the mean recall, recall, and harmonic recall values contrasting our proposed model with existing scene graph methods, including the concurrent work in [31], in Table 1. Contrary to existing approaches, depending on the loss weighting parameters used, our proposed approach is able to easily operate on a wide spectrum of recall and mean recall values (⑰-⑳). More concretely, compared to the most competitive baseline in SGTR [31] (⑭-⑯) and other one stage methods in AS-Net [4] (⑫) and HOTR [29] (⑬), our approach is able to achieve a considerably better recall on the *head* classes (⑰; 3.5 mR@100 better on head classes compared to [31]). Additionally, when comparing a variant of our approach that has a similar R@100 value to SGTR (⑮ and ⑳), we do significantly better on mean recall (3.8 higher mR@50) highlighting the proficiency of our method to generalize to *tail* classes wherein the training data is limited. Furthermore, our proposed model in ⑳ achieves the best performance across all models (including two-stage methods that use a superior backbone [49]) on hR@50/100, underlining the capability of our method to perform well on both the head and tail classes *simultaneously*. SGTR [31] additionally reports numbers by utilizing a data sampling approach BGNN [32] to bias the model towards the tail classes (⑯). We highlight that, compared to ⑯, our model in ⑳ performs similarly on mR@50 but much better R@50/100 (5.5 R@50 higher). The poorer performance on mR@100 is largely due to SGTR [31] selecting top-3 subjects and objects for each predicate, leading to more relationship triplets being generated. On the contrary, our approach only uses 1 predicate per subject-object pair. For a fairer comparison, we evaluate our model in ⑳ using a similar top-k strategy, wherein we select the top-3 predicates for each subject-object pair. The resulting model (㉒) outperforms the closest baseline by a significant margin (3.3 higher mR@100, 10.6 higher R@100 compared to ⑯). Additionally, we show that our loss re-weighting strategy is complementary to existing data sampling approaches by fine-tuning our trained model in ⑳ using BGNN [32] (㉑). For further analysis, see supplementary Section C.2.

Table 6: **Scene Graph Prediction on Action Genome.** Mean Recall (mR@K), Recall (R@K), and Harmonic Recall (hR@K) shown for baselines and variants of our approach with different loss weighting parameters. **B**=Backbone, **D**=Detector. Baseline results are taken from [16].

B	D	#	Method	R@20 / 50	mR@20/ 50	hR@20/50
X101-FPN	Faster RCNN	⟨1⟩	VRD [36]	10.3 / 10.9	-	
		⟨2⟩	Freq Prior [55]	24.0 / 24.9	-	-
		⟨3⟩	IMP [50]	23.9 / 25.5	-	-
		⟨4⟩	MSDN [33]	24.0 / 25.6	-	-
		⟨5⟩	Graph R-CNN [52]	24.1 / 25.8	-	-
		⟨6⟩	RelDN [56]	25.0 / 26.2	-	-
		⟨7⟩	SimpleBase [16]	27.9 / 30.4	8.3 / 9.1	12.8 / 14.0
R-101	DETR	⟨8⟩	Ours$_{\{\alpha=0,\beta=*\}}$	**31.0** / **37.5**	20.9 / 25.3	24.9 / 30.2
		⟨9⟩	Ours$_{\{\alpha=0.07,\beta=0.75\}}$	30.1 / 36.5	36.9 / 42.8	**33.1** / **39.4**
		⟨10⟩	Ours$_{\{\alpha=0.14,\beta=0.75\}}$	29.2 / 35.3	**37.9** / **44.0**	32.9 / 39.2

Action Genome. We additionally report recall, mean recall, and harmonic recall values on the Action Genome dataset [25] in Table 6. Similar to the observations on Visual Genome, we observe 3.1 higher R@20 and 12.6 higher mR@20 (⟨8⟩) when compared to the closest baseline in [16] (⟨7⟩) despite using an inferior backbone [49]. Furthermore, by effectively biasing our model towards the *tail* classes, we are able to achieve 29.6 better mR@20 while having better R@20/50 values (⟨10⟩) compared to ⟨7⟩.

We provide per class relation recall and object AP for both datasets in the supplementary (Section C).

5.3 Generality of Proposed Formulation

Although our transformer model effectively generates better scene graphs, the proposed iterative refinement formulation (Equation 2) can be applied to any existing method. To demonstrate this, we augment this framework to the simple two-stage MOTIF [55] architecture. The details on how this is achieved is deferred to the supplementary (Section A). We contrast the performance of the baseline and our augmented approach on the Visual Genome dataset [30] in Table 3. It can be seen that with each refinement step, the model is able to generate better scene graphs ($\sim$6% higher on mR@20 and mR@50 after 3 steps). Note that, owing to the two-stage nature of MOTIF [55], the image features used by our refinement framework are fixed and oblivious to the task of scene graph generation (as the detector parameters are frozen). Contrary to the proposed transformer model that can query the image at each layer, the performance gains when using the augmented MOTIF variant are largely limited due to the image features being the same across the refinement steps.

6 Conclusion, Limitations, and Societal Impact

In this work we propose a novel and general framework for iterative refinement that overcomes limitations arising from a fixed factorization assumption in existing methods. We demonstrate its effectiveness through a transformer based end-to-end trainable architecture.

Limitations. Limitations of the proposed approach include using a shared image encoder, which improves efficiency, but may limit diversity of representations available for different decoder layers; and limited ability to model small objects inherited from the DETR object detection design [31].

Societal Impact. While our model does not have any direct serious societal implications, its impact could be consequential if it is used in specific critical applications (*e.g.*, autonomous driving). In such cases both the overall performance and biases in object / predicate predictions would require careful analysis and further calibration. Our loss re-weighting strategy is a step in that direction.

Acknowledgments and Disclosure of Funding

This work was funded, in part, by the Vector Institute for AI, Canada CIFAR AI Chair, NSERC CRC and an NSERC DG and Accelerator Grants. Hardware resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute[1]. Additional support was provided by JELF CFI grant and Compute Canada under the RAC award. Finally, we sincerely thank Bicheng Xu and Muchen Li for their valuable feedback on the paper draft.

[1] www.vectorinstitute.ai/#partners

References

[1] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[4] Mingfei Chen, Yue Liao, Si Liu, Zhiyuan Chen, Fei Wang, and Chen Qian. Reformulating hoi detection as adaptive set prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9004–9013, 2021.

[5] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6163–6171, 2019.

[6] Meng-Jiun Chiou, Henghui Ding, Hanshu Yan, Changhu Wang, Roger Zimmermann, and Jiashi Feng. Recovering the unbiased scene graphs from the biased ones. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1581–1590, 2021.

[7] Yuren Cong, Michael Ying Yang, and Bodo Rosenhahn. Reltr: Relation transformer for scene graph generation. *arXiv preprint arXiv:2201.11460*, 2022.

[8] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10578–10587, 2020.

[9] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.

[10] Shabnam Daghaghi, Tharun Medini, and Anshumali Shrivastava. Sdm-net: a simple and effective model for generalized zero-shot learning. In *Uncertainty in Artificial Intelligence*, pages 2103–2113. PMLR, 2021.

[11] Alakh Desai, Tz-Ying Wu, Subarna Tripathi, and Nuno Vasconcelos. Learning of visual relations: The devil is in the tails. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15404–15413, 2021.

[12] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[13] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1851–1860, 2017.

[14] Qi Dong, Zhuowen Tu, Haofu Liao, Yuting Zhang, Vijay Mahadevan, and Stefano Soatto. Visual relationship detection using part-and-sum transformers with composite queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3550–3559, 2021.

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[16] Raghav Goyal and Leonid Sigal. A simple baseline for weakly-supervised human-centric relation detection. In *British Machine Vision Conference*, 2021.

[17] Jiuxiang Gu, Shafiq Joty, Jianfei Cai, Handong Zhao, Xu Yang, and Gang Wang. Unpaired image captioning via scene graph alignments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10323–10332, 2019.

[18] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16383–16392, 2021.

[19] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[20] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[21] H He, Y Bai, EA Garcia, and S ADASYN Li. adaptive synthetic sampling approach for imbalanced learning. ieee international joint conference on neural networks. In *2008 (IEEE World Congress On Computational Intelligence)*, 2008.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Sen He, Wentong Liao, Hamed R Tavakoli, Michael Yang, Bodo Rosenhahn, and Nicolas Pugeault. Image captioning through image transformer. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[24] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[25] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020.

[26] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228, 2018.

[27] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*, 2020.

[28] Siddhesh Khandelwal, Mohammed Suhail, and Leonid Sigal. Segmentation-grounded scene graph generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15879–15889, 2021.

[29] Bumsoo Kim, Junhyun Lee, Jaewoo Kang, Eun-Sol Kim, and Hyunwoo J Kim. Hotr: End-to-end human-object interaction detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 74–83, 2021.

[30] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[31] Rongjie Li, Songyang Zhang, and Xuming He. Sgtr: End-to-end scene graph generation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19486–19496, 2022.

[32] Rongjie Li, Songyang Zhang, Bo Wan, and Xuming He. Bipartite graph network with adaptive message passing for unbiased scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11109–11119, 2021.

[33] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. In *Proceedings of the IEEE international conference on computer vision*, pages 1261–1270, 2017.

[34] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[35] Hengyue Liu, Ning Yan, Masood Mortazavi, and Bir Bhanu. Fully convolutional scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11546–11556, 2021.

[36] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European conference on computer vision*, pages 852–869. Springer, 2016.

[37] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018.

[38] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3651–3660, 2021.

[39] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. *Advances in neural information processing systems*, 30, 2017.

[40] Mengshi Qi, Weijian Li, Zhengyuan Yang, Yunhong Wang, and Jiebo Luo. Attentive relational networks for mapping images to scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3957–3966, 2019.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[42] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *European conference on computer vision*, pages 467–482. Springer, 2016.

[43] Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Sahand Sharifzadeh, Georgios Kaissis, Volker Tresp, et al. Relationformer: A unified framework for image-to-graph generation. *arXiv preprint arXiv:2203.10202*, 2022.

[44] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[45] Mohammed Suhail, Abhay Mittal, Behjat Siddiquie, Chris Broaddus, Jayan Eledath, Gerard Medioni, and Leonid Sigal. Energy-based learning for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13936–13945, 2021.

[46] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3716–3725, 2020.

[47] Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6619–6628, 2019.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[49] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[50] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5419, 2017.

[51] 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 *Proceedings of the 28th ACM International Conference on Multimedia*, pages 265–273, 2020.

[52] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685, 2018.

[53] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.

[54] Alireza Zareian, Svebor Karaman, and Shih-Fu Chang. Bridging knowledge graphs to generate scene graphs. In *European Conference on Computer Vision*, pages 606–623. Springer, 2020.

[55] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5831–5840, 2018.

[56] Ji Zhang, Kevin J Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. Graphical contrastive losses for scene graph parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11535–11543, 2019.

[57] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[58] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All the data we use in our experiments is publicly available. The code for our approach is available at `github.com/ubc-vision/IterativeSG`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We use widely popular data splits for our experiments. We briefly describe this in Section 5. The hyperparameters and additional data details are also mentioned in the supplementary (Section B).
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] As scene graph models are computationally expensive to train, none of the existing literature report error bars. However, we do report the performance of our method on different hyperparameter values to enable better understanding of the performance improvements.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] This is described in the supplementary (Section B).
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] All the datasets we use are publicly available. We cite the creators of Visual Genome (Krishna *et al.* [30]) and Action Genome (Ji *et al.* [25]) throughout the paper.
   (b) Did you mention the license of the assets? [Yes] Visual Genome [30] is licensed under the Creative Commons Attribution 4.0 International License. Action Genome [25] is licensed under the MIT license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] We don't introduce any new assets in the paper.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Both Visual Genome [30] and Action Genome [25] are publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Supplementary: Iterative Scene Graph Generation

Siddhesh Khandelwal[1,2] **and Leonid Sigal**[1,2,3]
[1]Department of Computer Science, University of British Columbia
[2]Vector Institute for AI
[3]CIFAR AI Chair
{skhandel, lsigal}@cs.ubc.ca

The code for our approach is available at github.com/ubc-vision/IterativeSG.

A Augmenting Iterative Framework to MOTIF

As described in Section 3 of the main paper, our proposed iterative refinement formulation is model agnostic and can be integrated to any existing architecture. We demonstrate this in Section 5.3 of the main paper by augmenting our proposed approach to an existing model in MOTIF [55], showing improved scene graph generation performance. In this section we present the details of this integration.

As is the case with most two-stage architectures, MOTIF [55] relies on a pre-trained detector (like Faster R-CNN [41]) to generate bounding box proposals $\mathbf{B} = \{\mathbf{b}_i\}$ and corresponding labels $\mathbf{L} = \{\mathbf{l}_i\}$ for entities $\mathbf{E} = \{\mathbf{e}_i\}$ within a given image. For each of these bounding boxes, the model extracts corresponding image features $\mathbf{z}_i$ via a Region of Interest (ROI) Pooling [41] operation. Additionally, as the task of scene graph prediction requires a pair of entities to associate a relation with, an IOU-based thresholding is used to couple entities into candidate pairs $\{(\mathbf{e}_i, \mathbf{e}_j)\}$. Subsequently, for each of these entity pairs $(\mathbf{e}_i, \mathbf{e}_j)$, features representing the union of their corresponding bounding boxes $(\mathbf{b}_i, \mathbf{b}_j)$ is extracted via a similar ROI pooling operation. We represent these union features as $\mathbf{u}_{i,j}$.

MOTIF [55] utilizes the set of features $\{\mathbf{z}_i\}$ and $\{\mathbf{u}_{i,j}\}$, alongside the bounding box proposals $\{\mathbf{b}_i\}$ and entity label estimates $\{\mathbf{l}_i\}$, to generate object/subject labels and the corresponding predicate between them. MOTIF achieves this by instantiating entity $f_{\mathbf{s},\mathbf{o}}$ and predicate $f_{\mathbf{p}}$ networks, each utilizing bidirectional LSTM. Note that, contrary to our transformer based formulation (Section 4 of the main paper) that has separate decoders for the subject and object (namely $f_{\mathbf{s}}$ and $f_{\mathbf{o}}$, MOTIF uses a single network to predict both the subject and object labels.

Entity Network. To obtain subject/object labels, for each entity $\mathbf{e}_i$, MOTIF [55] uses a bidirectional LSTM to obtain the set of contextual entity representations $\{\mathbf{c}_i^e\}$,

$$\{\mathbf{c}_i^e\} = \texttt{biLSTM}\left(\{\mathbf{z}_i, \mathbf{l}_i\}\right) \tag{A1}$$

where $\mathbf{c}_i^e$ is the concatenation of the final hidden states of the bidirectional LSTM. A decoder LSTM then sequentially generates labels $\{\widehat{\mathbf{l}}_i^e\}$ for each entity as follows,

$$\begin{aligned} \mathbf{h}_i &= \texttt{LSTM}\left(\mathbf{c}_i^e, \widehat{\mathbf{l}}_{i-1}^e\right) \\ \widehat{\mathbf{l}}_i^e &= \texttt{argmax}\left(\mathbf{W}_e \mathbf{h}_i\right) \end{aligned} \tag{A2}$$

where $\mathbf{W}_e$ is a learned parameter.

Predicate Network. Similar to the entity network, the predicate network also uses a bidirectional LSTM to obtain contextual predicate representations $\{\mathbf{c}_i^p\}$ as follows,

$$\{\mathbf{c}_i^p\} = \texttt{biLSTM}\left(\{\mathbf{c}_i^e, \widehat{\mathbf{l}}_i^e\}\right). \tag{A3}$$

For each candidate pair $(\mathbf{e}_i, \mathbf{e}_j)$, the predicate label $\{\widehat{\mathbf{l}}^e_{i,j}\}$ is then obtained as follows,

$$\begin{aligned}\mathbf{g}_{i,j} &= (\mathbf{W}_h \mathbf{c}^p_i) \odot \left(\mathbf{W}_b \mathbf{c}^p_j\right) \odot \mathbf{u}_{i,j} \\ \widehat{\mathbf{l}}^e_{i,j} &= \texttt{argmax}\left(\mathbf{W}_p \mathbf{g}_{i,j}\right)\end{aligned} \tag{A4}$$

where $\mathbf{W}_h, \mathbf{W}_b, \mathbf{W}_p$ are learned parameters, and $\odot$ is the element-wise product.

We refer the reader to [55] for a more detailed explanation of the entity and predicate networks. Our iterative refinement augmented variant of MOTIF defines T entity and predicate networks, denoted by $f^t_{\mathbf{s},\mathbf{o}}$ and $f^t_{\mathbf{p}}$ at layer t. Following our formulation described in Section 3 of the main paper, we condition the inputs to the entity $f^t_{\mathbf{s},\mathbf{o}}$ and predicate $f^t_{\mathbf{p}}$ networks appropriately. The two types of conditionings, namely conditioning within a step (CWS) and conditioning across steps (CAS) are implemented as follows,

Conditioning Within a Step. Contrary to the transformer based model where we need to explicitly condition with a step (Equation 3 of the main paper), this is already implemented in the default MOTIF [55] architecture. This is evident from Equation A3 wherein the contextual predicate representations $\{\mathbf{c}^p_i\}$ are obtained by using the entity contextual representations $\{\mathbf{c}^e_i\}$. This implicitly defines the following conditioning $\mathbf{e}_i \rightarrow \mathbf{p}_i$.

Conditioning Across Steps. In accordance with Equation 2 of the main paper, we want to condition the predictions at layer t on the graph estimate at layer $t-1$. This is achieved by modifying the inputs $\{\mathbf{z}_i\}$ and $\{\mathbf{u}_{i,j}\}$. Specifically, for a particular step t, the input to the entity network $\{\mathbf{z}^t_i\}$ is obtained as follows,

$$\begin{aligned}\{\widehat{\mathbf{z}}^t_i\} &= \{\mathbf{z}^{t-1}_i\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\mathbf{z}^{t-1}_i\}, \{\mathbf{h}^{t-1}_i\}, \{\mathbf{h}^{t-1}_i\}\right)\right) \\ \{\bar{\mathbf{z}}^t_i\} &= \{\widehat{\mathbf{z}}^t_i\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\widehat{\mathbf{z}}^t_i\}, \{\mathbf{g}^{t-1}_{i,j}\}, \{\mathbf{g}^{t-1}_{i,j}\}\right)\right) \\ \{\mathbf{z}^t_i\} &= \{\bar{\mathbf{z}}^t_i\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\bar{\mathbf{z}}^t_i\}, \{\mathbf{u}^{t-1}_{i,j}\}, \{\mathbf{u}^{t-1}_{i,j}\}\right)\right)\end{aligned} \tag{A5}$$

Similarly, for a particular step t, the input to the predicate network $\{\mathbf{u}^t_i\}$ is obtained as follows,

$$\begin{aligned}\{\widehat{\mathbf{u}}^t_{i,j}\} &= \{\mathbf{u}^{t-1}_{i,j}\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\mathbf{u}^{t-1}_{i,j}\}, \{\mathbf{h}^{t-1}_i\}, \{\mathbf{h}^{t-1}_i\}\right)\right) \\ \{\bar{\mathbf{u}}^t_{i,j}\} &= \{\widehat{\mathbf{u}}^t_{i,j}\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\widehat{\mathbf{u}}^t_{i,j}\}, \{\mathbf{g}^{t-1}_{i,j}\}, \{\mathbf{g}^{t-1}_{i,j}\}\right)\right) \\ \{\mathbf{u}^t_{i,j}\} &= \{\bar{\mathbf{u}}^{t-1}_{i,j}\} + \texttt{FFN}\left(\texttt{MultiHead}\left(\{\bar{\mathbf{u}}^{t-1}_{i,j}\}, \{\mathbf{z}^{t-1}_i\}, \{\mathbf{z}^{t-1}_i\}\right)\right)\end{aligned} \tag{A6}$$

Finally, to allow each layer t to refine the estimates of the previous layer $t-1$, the representations $\mathbf{h}^t_i$ and $\mathbf{g}^t_{i,j}$ are modelled as residual connections. Specifically, we modify Equations A2 and A4 as follows,

$$\begin{aligned}\mathbf{h}^t_i &= \mathbf{h}^{t-1}_i + \texttt{LSTM}\left(\mathbf{c}^{t,e}_i, \widehat{\mathbf{l}}^{t,e}_{i-1}\right) \\ \mathbf{g}^t_{i,j} &= \mathbf{g}^{t-1}_{i,j} + \left(\mathbf{W}^t_h \mathbf{c}^{t,p}_i\right) \odot \left(\mathbf{W}^t_b \mathbf{c}^{t,p}_j\right) \odot \mathbf{u}^t_{i,j}\end{aligned} \tag{A7}$$

B Additional Implementation Details

Number of Epochs and Model Selection. The transformer models are trained for 50 epochs on Visual Genome [30] and 18 epochs on Action Genome [25]. The best model is selected by checking the validation set performance in each of the datasets.

Hardware. The training is done on 4 Tesla T4 for the transformer models. For the MOTIF augmentation, the training for both the baseline and our variant is done on 4 NVidia A100 GPUs.

Generating Graph from Triplets. As our transformer model generates relation triplets, we obtain a graph by applying a non-maximum suppression (NMS) strategy [15, 41] to group the predicted subjects and objects into entity instances. This NMS is applied separately *per class*, and each predicted subject/object is then assigned to a post-NMS bounding box by checking the IoU overlap.

Data Splits. For the Visual Genome dataset [30], we follow the widely popular data splits provided by [55], which can be found at the official release repository of [46]:

github.com/KaihuaTang/Scene-Graph-Benchmark.pytorch. For the Action Genome dataset [25], to facilitate fair comparison with the baselines, we contacted the authors of [16] and obtained their data pre-processing code.

The code for our approach is available at github.com/ubc-vision/IterativeSG.

C Additional Results

C.1 Additional Ablations

Table A1: **Refinement Across Steps.** Results are shown for the model trained using $\alpha = 0.14, \beta = 0.75$ on the Visual Genome test set. To measure the scene graph generation performance, we report recall, mean recall, and harmonic recall. To measure the detection performance, we report the box AP, AP_{50}, and AP_{75} to analyze the detection performance. Finally, we report the relative model size compared to the final model at each refinement layer.

t	mR@20/50	R@20/50	hR@20/50	AP	AP_{50}	AP_{75}	Model Size
1	9.4 / 13.3	18.4 / 23.3	12.4 / 16.9	11.8	24.2	9.7	0.61×
2	10.7 / 14.8	19.8 / 24.8	13.9 / 18.5	13.5	26.3	11.8	0.69×
3	11.1 / 14.8	20.2 / 25.2	14.3 / 18.6	14.0	26.8	12.5	0.76×
4	11.7 / 15.7	20.8 / 26.0	15.0 / 19.6	14.5	27.4	13.0	0.84×
5	11.8 / 15.6	21.0 / 26.1	15.1 / 19.5	14.6	27.6	13.1	0.92×
6	**11.8** / **15.8**	**21.0** / **26.1**	**15.1** / **19.7**	**14.6**	**27.6**	**13.2**	1×

Refinement Across Steps. To further analyze the effectiveness of our refinement procedure, we contrast the quality of scene graphs generated at each refinement step for a particular trained model ($\alpha = 0.14, \beta = 0.75$) in Table A1. From the improvements in recall, mean recall, and harmonic recall at each t, it can be reasoned that the quality of the generated graphs steadily improves with each refinement step, wherein the biggest gains are observed in the initial stages. Furthermore, the improvement in the AP numbers indicate that the detection performance improves simultaneously alongside the graph generation performance, highlighting the ability of our approach to jointly reason over subjects, objects, and predicates. Additionally, as each layer t generates a valid scene graph, an added benefit of our proposed formulation is its ability to allow model compression without the need for re-training. One can simply select the first t refinement layers (and discard the rest) depending on the desired accuracy-memory trade-off. For example, from Table A1, selecting the first 4 layers (and discarding the last 2) gives a model that is 0.84 the size at the expense of a 0.1 drop in hR@50 and hR@100.

Table A2: **Number of Queries Ablation.** We vary the number of queries used by our transformer decoders and report recall, mean recall, and harmonic recall on the Visual Genome test set. This ablation assume $\alpha = 0.14, \beta = 0.75$.

# Queries	mR@20/50	R@20/50	hR@20/50
150	11.7 / 15.0	21.0 / 25.6	15.0 / 18.9
300	**11.8** / **15.8**	**21.0** / 26.1	**15.1** / **19.7**
450	11.6 / 15.7	20.7 / **26.2**	14.9 / 19.6

Number of Queries. We additionally ablate the number of queries used by the decoders of our transformer model. We report performance on recall, mean recall, and harmonic recall on the Visual Genome [55] dataset in Table A2, assuming $\alpha = 0.14, \beta = 0.75$ for all models. It can be seen that setting the number of queries to be too low or too high is detrimental to performance.

Zero Shot Recall. Introduced in [36], zero shot recall (zsR@K) measures recall@K for `<subject, object, predicate>` that are *absent* from the training set. Therefore, this measure allows analysis of a model performance on unseen relationships. We report zsR@50/100 for certain baselines (including concurrent work in [31]) and our approach in Table A3. It can be seen that our approach in

Table A3: **Zero Shot Recall.** We report zero shot recall for baselines and our approach. The baseline numbers are borrowed from [31].

#	Model	zR@50/100
1	BGNN [32]	0.4 / 0.9
2	DT2-ACBS [11]	0.3 / 0.5
3	VCTree-TDE [46]	2.6 / 3.2
4	SGTR$_{M=3}$ [31]	2.4 / **5.8**
5	Ours$_{(\alpha=0.14,\beta=0.75)}$	2.8 / 3.7
6	Ours$_{(\alpha=0.14,\beta=0.75),M=3}$	**3.9** / 5.6

5 provides the best zsR@50 (0.2 higher when compared to 3), which is a stricter metric. Although SGTR [31] (4) has a higher performance on zsR@100, as described in Section 5.2 of the main paper, it selects top-3 subjects and objects for each predicate, leading to more relationship triplets being generated. Therefore, for a fairer comparison, we evaluate our model in 5 using a similar top-k strategy, wherein we select the top-3 predicates for each subject-object pair. This is done by getting the 3 most likely predicate classes from the distribution generated by the predicate decoder. The resulting model (6) provides a much greater margin on zsR@50 (1.5 higher than SGTR [31]) while being comparable on zsR@100 (0.2 lower).

Per Class Recall. We show the per-class predicate recalls on Visual Genome [30] in Figure A1. Additionally, we contrast the per class recalls between steps $t = 1$ (which corresponds to the model with no refinement) and $t = 6$. It can be seen that with more refinement steps, the model is able to perform considerably better across most classes. A similar observation can be made for Action Genome [25], as shown in Figure A2.

Per Class AP. We show the per-class object APs on Visual Genome [30] in Figure A3. Additionally, we contrast the per class APs between steps $t = 1$ (which corresponds to the model with no refinement) and $t = 6$. It can be seen that with more refinement steps, the model is able to perform considerably better across object classes. Note that, for brevity, we only show the top 50 object classes, selected according to the difference between class AP at step $t = 1$ and step $t = 6$. Per-class object APs for the Action Genome [25] dataset is shown in Figure A4.

C.2 Complementarity to Data Sampling Approaches

In Section 5.2 of the main paper, we briefly highlight that our proposed loss weighting strategy is complementary to existing data sampling approaches by fine tuning our model in 20 (Table 1 of the main paper) using BGNN [32]. This fine-tuning is done *only* on the final predicate softmax layer, and all other parameters are freezed. For ease of readability, we copy relevant results from Table 1 of the main paper into Table A4. We show our approach fine-tuned using BGNN [32] in 6 (Table A4) provides a considerable improvement over our approach without BGNN (4; 1.3 mR@50 higher with similar hR@50/100 numbers). In comparison to the baseline SGTR [31] with BGNN (3), our model in 6 provides better mR@50 (1.3 higher), R@50/100 (2.3/0.7 higher), and hR 50 (1.6 higher). As described in Section 5.2, the poorer performance on mR@100 and hR@100 is largely due to SGTR [31] selecting top-3 subjects and objects for each predicate, leading to more relationship triplets being generated. Therefore, for a fairer comparison, we evaluate our model in 6 using a similar top-k strategy, wherein we select the top-3 predicates for each subject-object pair. The resulting model in 7 outperforms the baseline in 3 by a significant margin on all metrics, leading to improved recall on head, body, and tail classes. Therefore, our proposed loss weighting scheme can be used in addition to existing data sampling strategies, making it more flexible.

C.3 Qualitative Results

We provide additional qualitative results for our transformer model at different steps in Figures A5-A14. These highlight steady improvements in the graph quality over refinement steps.

Table A4: **Complementarity to Data Sampling Approaches.** Mean Recall (mR@K), Recall (R@K), and Harmonic Recall (hR@K) shown for the baseline and our approach. **B**=Backbone, **D**=Detector. Baseline results are borrowed from [31]. M indicates the number of top-k links used by the baseline [31].

B	D	#	Method	mR@50/100	R@50/100	hR@50/100	Head	Body	Tail
ResNet-101	DETR	①	SGTR$_{M=1}$ [31]	12.0 / 14.6	25.1 / 26.6	16.2 / 18.8	27.1	17.2	6.9
		②	SGTR$_{M=3}$ [31]	12.0 / 15.2	24.6 / 28.4	16.1 / 19.8	28.2	18.6	7.1
		③	SGTR$_{M=3,\text{BGNN [32]}}$ [31]	15.8 / 20.1	20.6 / 25.0	17.9 / 22.3	21.7	21.6	17.1
		④	Ours$_{(\alpha=0.14,\beta=0.75)}$	15.8 / 18.2	26.1 / 28.7	19.7 / 22.3	28.2	19.4	13.8
		⑤	Ours$_{(\alpha=0.14,\beta=0.75),M=3}$	19.5 / 23.4	**30.8** / **35.6**	**23.9** / 28.2	**32.9**	28.1	15.8
		⑥	Ours$_{(\alpha=0.14,\beta=0.75),\text{BGNN [32]}}$	17.1 / 19.2	22.9 / 25.7	19.6 / 22.0	24.4	20.2	16.4
		⑦	Ours$_{(\alpha=0.14,\beta=0.75),\text{BGNN [32]},M=3}$	**20.2** / **24.1**	29.0 / 34.2	23.8 / **28.3**	27.7	**30.1**	**18.5**

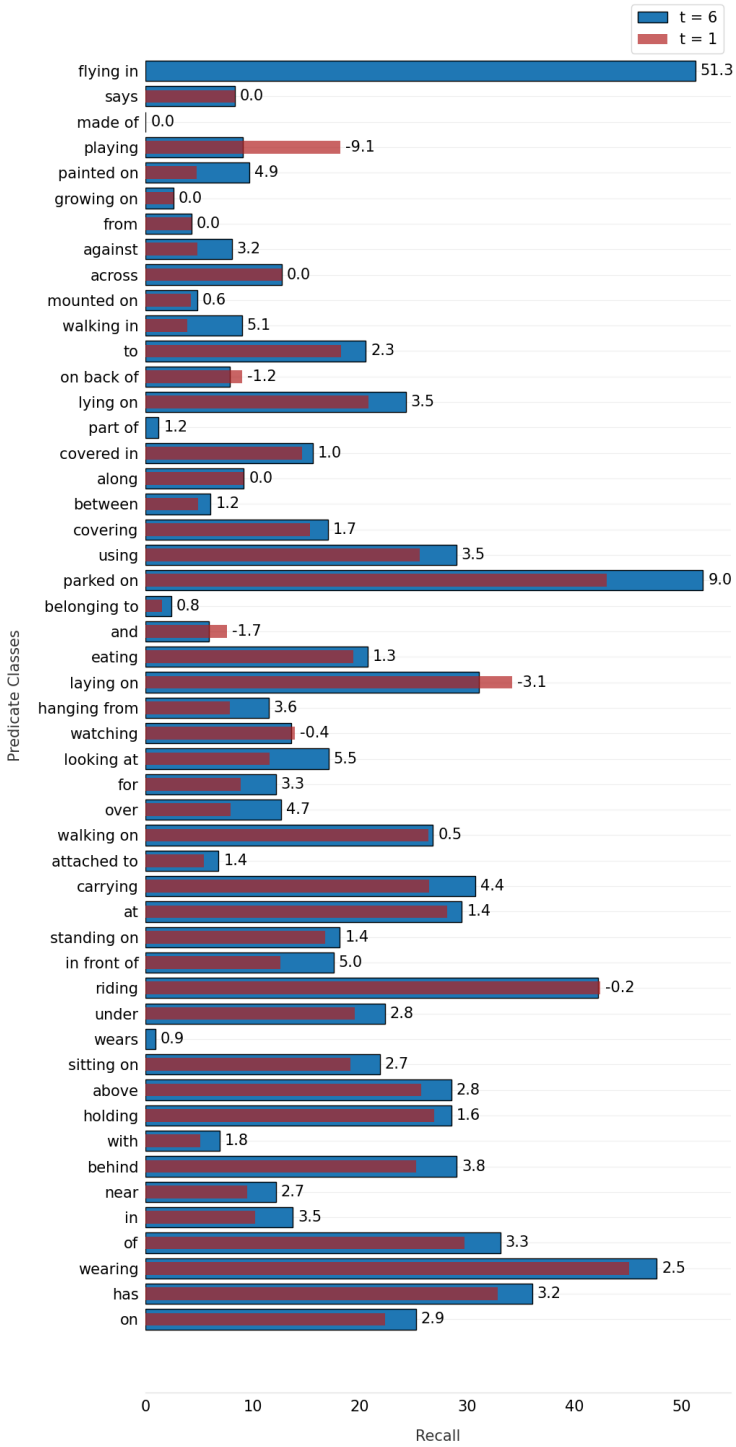

Figure A1: **Per Class Recall on Visual Genome.** Plot shows the recall for the individual predicate classes in Visual Genome [30] for our model trained with $\alpha = 0.14, \beta = 0.75$. The y-axis is sorted in increasing order of predicate frequency in the training set, with the more popular (head) classes at the bottom, and the less popular (tail) classes at the top. We additionally contrast the per class recalls between steps $t = 1$ (indicating no refinement; in red) and $t = 6$ (in blue). The number next to each bar indicates the difference between recall at $t = 6$ and $t = 1$ for a particular predicate class.

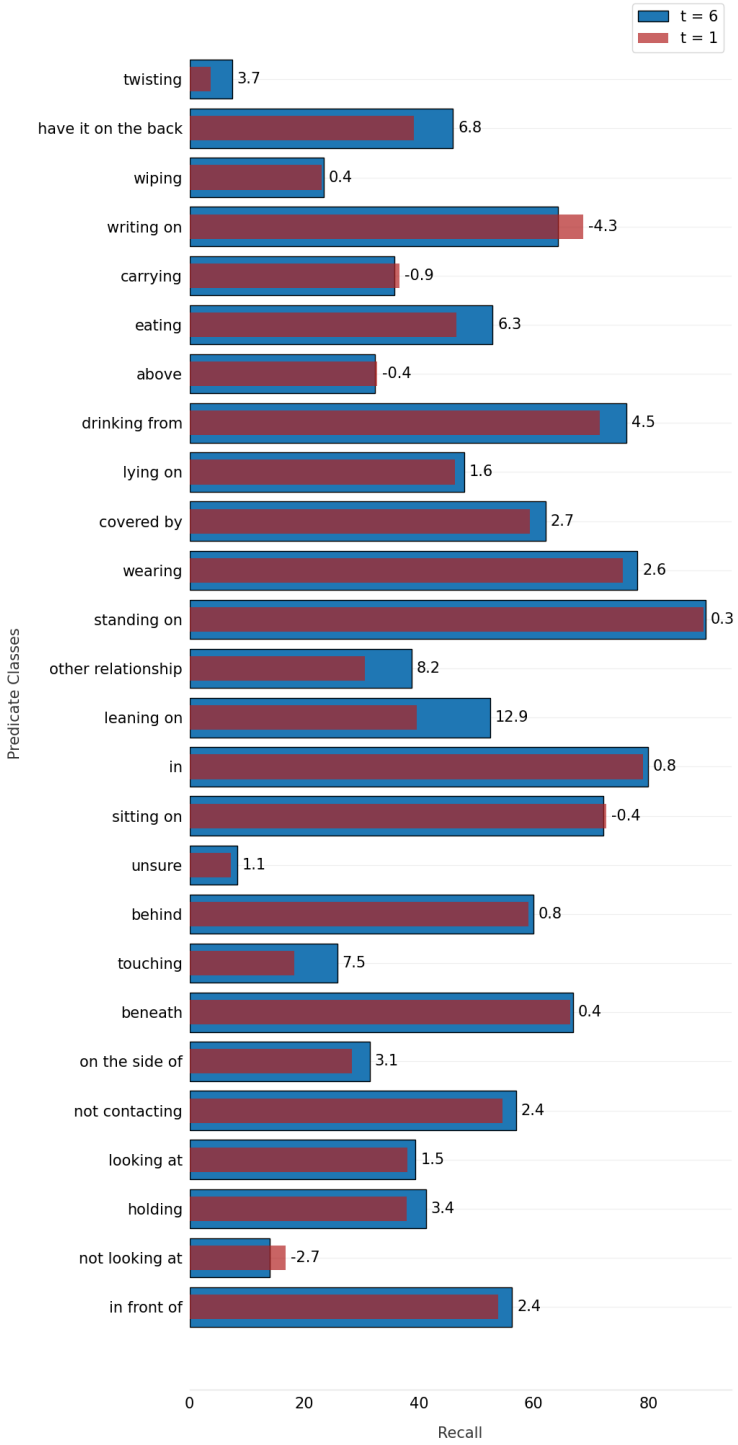

Figure A2: **Per Class Recall on Action Genome.** Plot shows the recall for the individual predicate classes in Action Genome [25] for our model trained with $\alpha = 0.14, \beta = 0.75$. The y-axis is sorted in increasing order of predicate frequency in the training set, with the more popular (head) classes at the bottom, and the less popular (tail) classes at the top. We additionally contrast the per class recalls between steps $t = 1$ (indicating no refinement; in red) and $t = 6$ (in blue). The number next to each bar indicates the difference between recall at $t = 6$ and $t = 1$ for a particular predicate class.

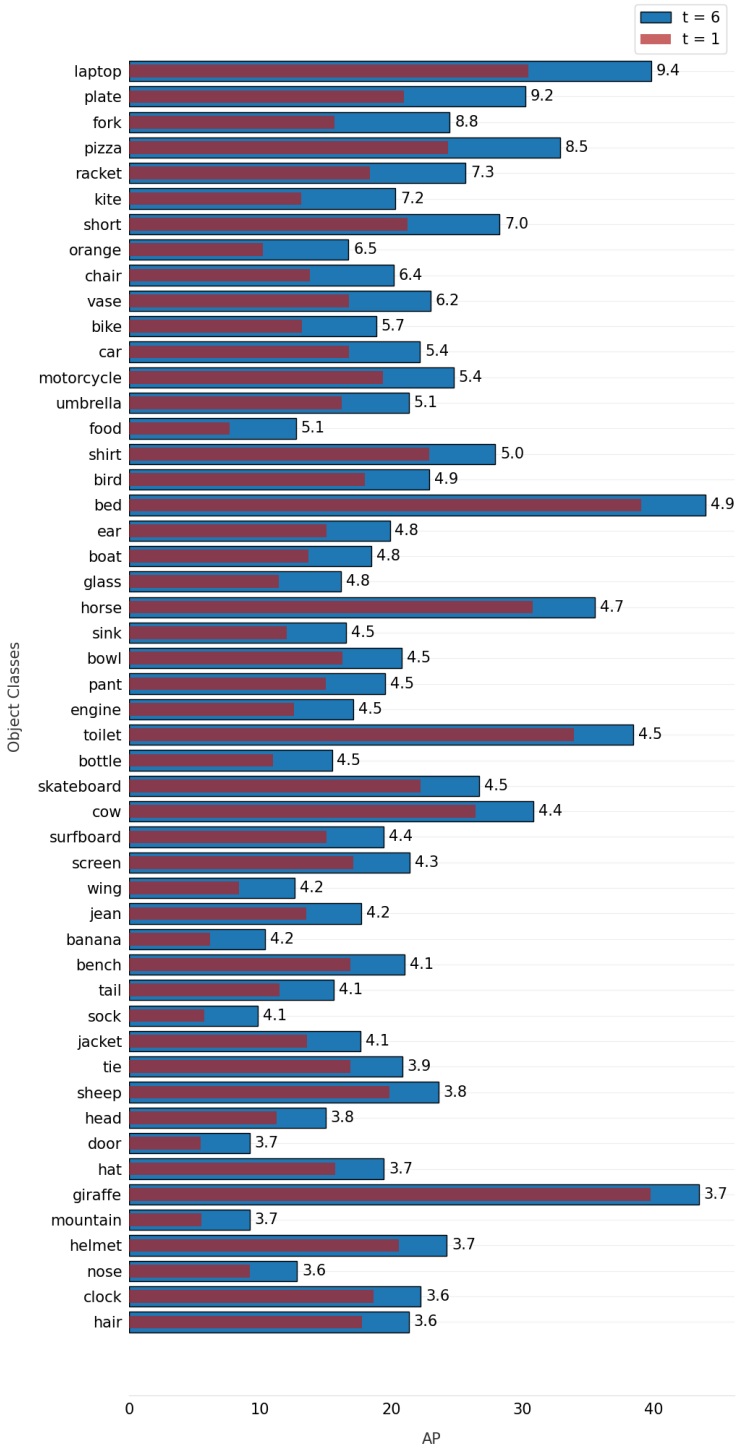

Figure A3: **Per Class AP on Visual Genome.** Plot shows the AP for the top 50 individual object classes in Visual Genome [30] (out of 150 total object classes) for our model trained with $\alpha = 0.14, \beta = 0.75$. We contrast the per class APs between steps $t = 1$ (indicating no refinement; in red) and $t = 6$ (in blue). The number next to each bar indicates the difference between AP values at $t = 6$ and $t = 1$ for a particular object class. The y-axis is sorted by this difference value.

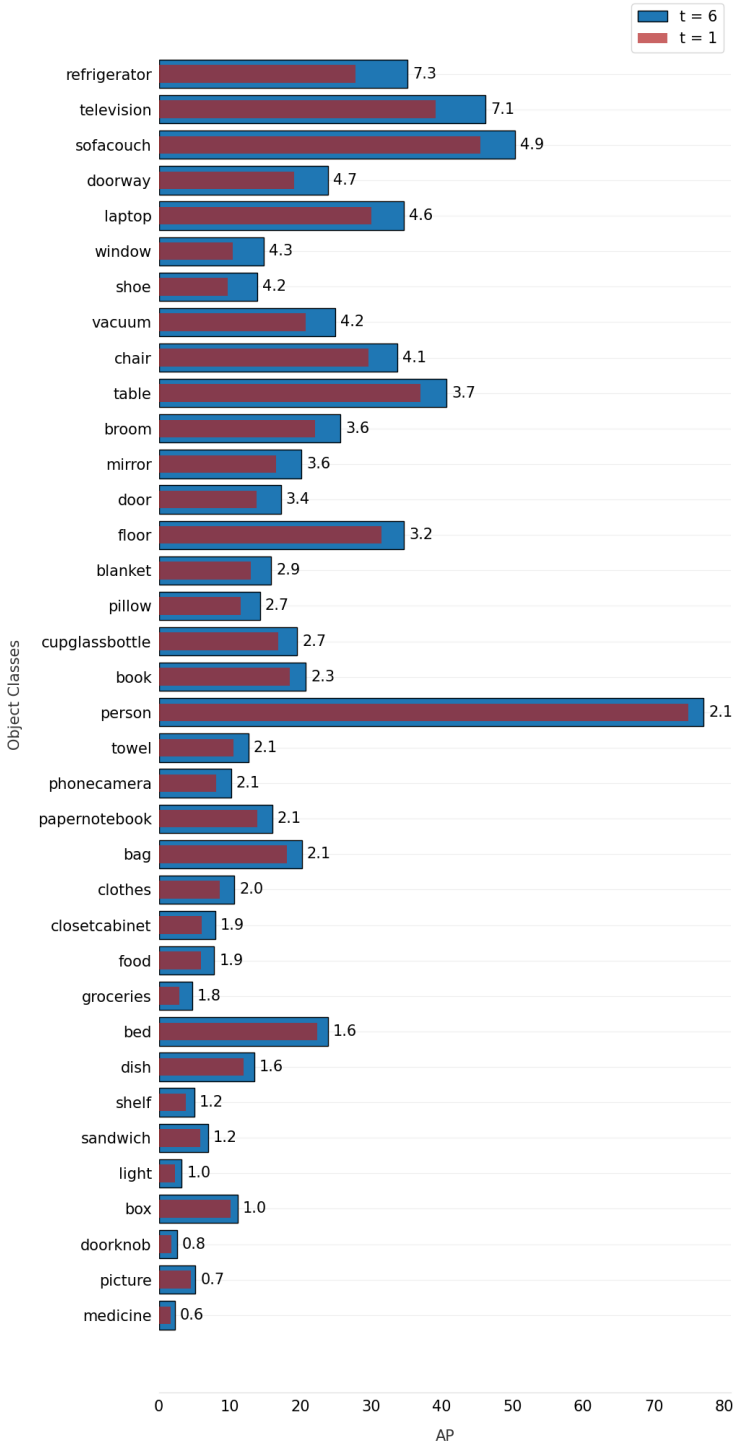

Figure A4: **Per Class AP on Action Genome.** Plot shows the AP for the individual object classes in Action Genome [25] for our model trained with $\alpha = 0.14, \beta = 0.75$. We contrast the per class APs between steps $t = 1$ (indicating no refinement; in red) and $t = 6$ (in blue). The number next to each bar indicates the difference between AP values at $t = 6$ and $t = 1$ for a particular object class. The y-axis is sorted by this difference value.

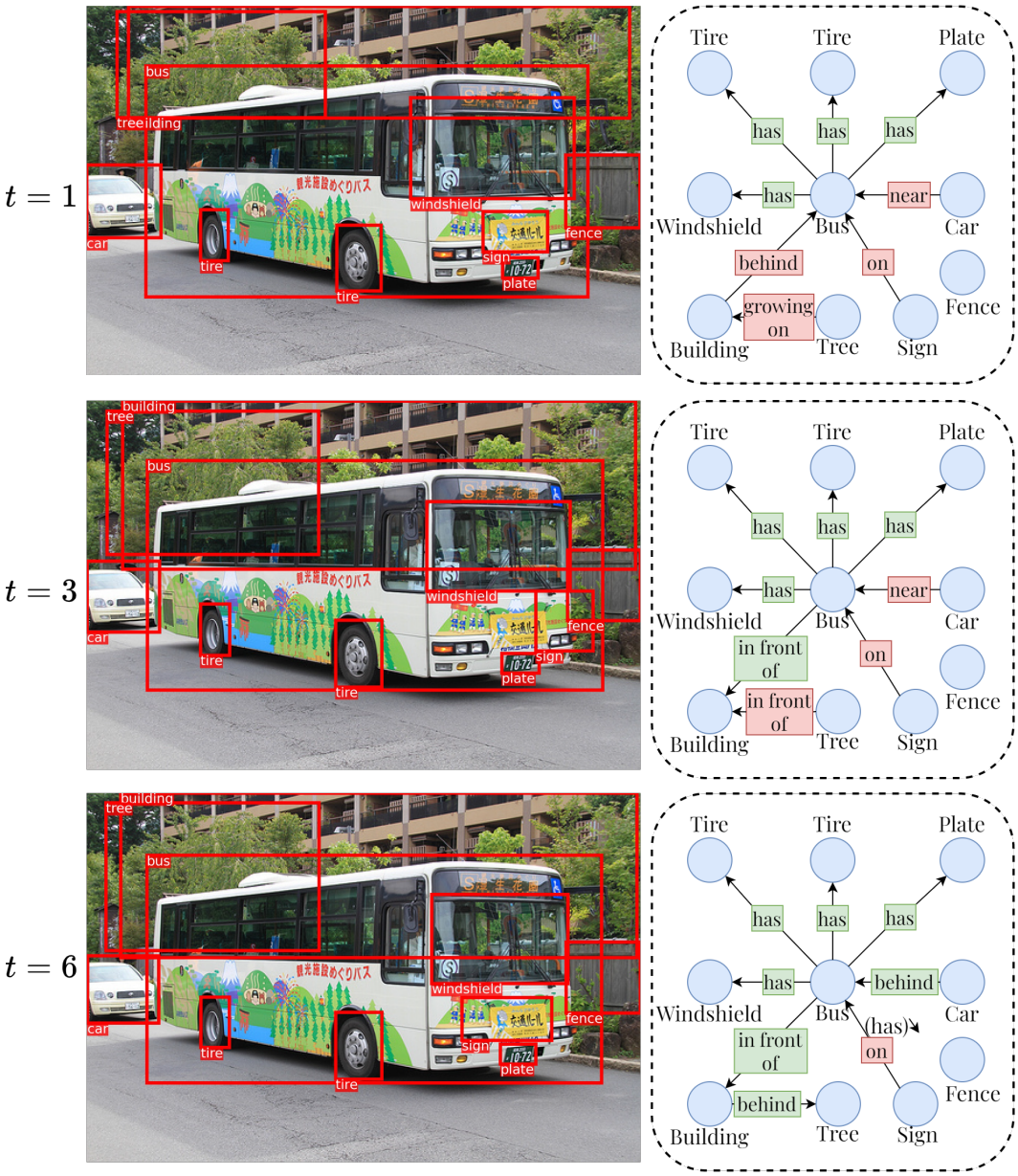

Figure A5: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

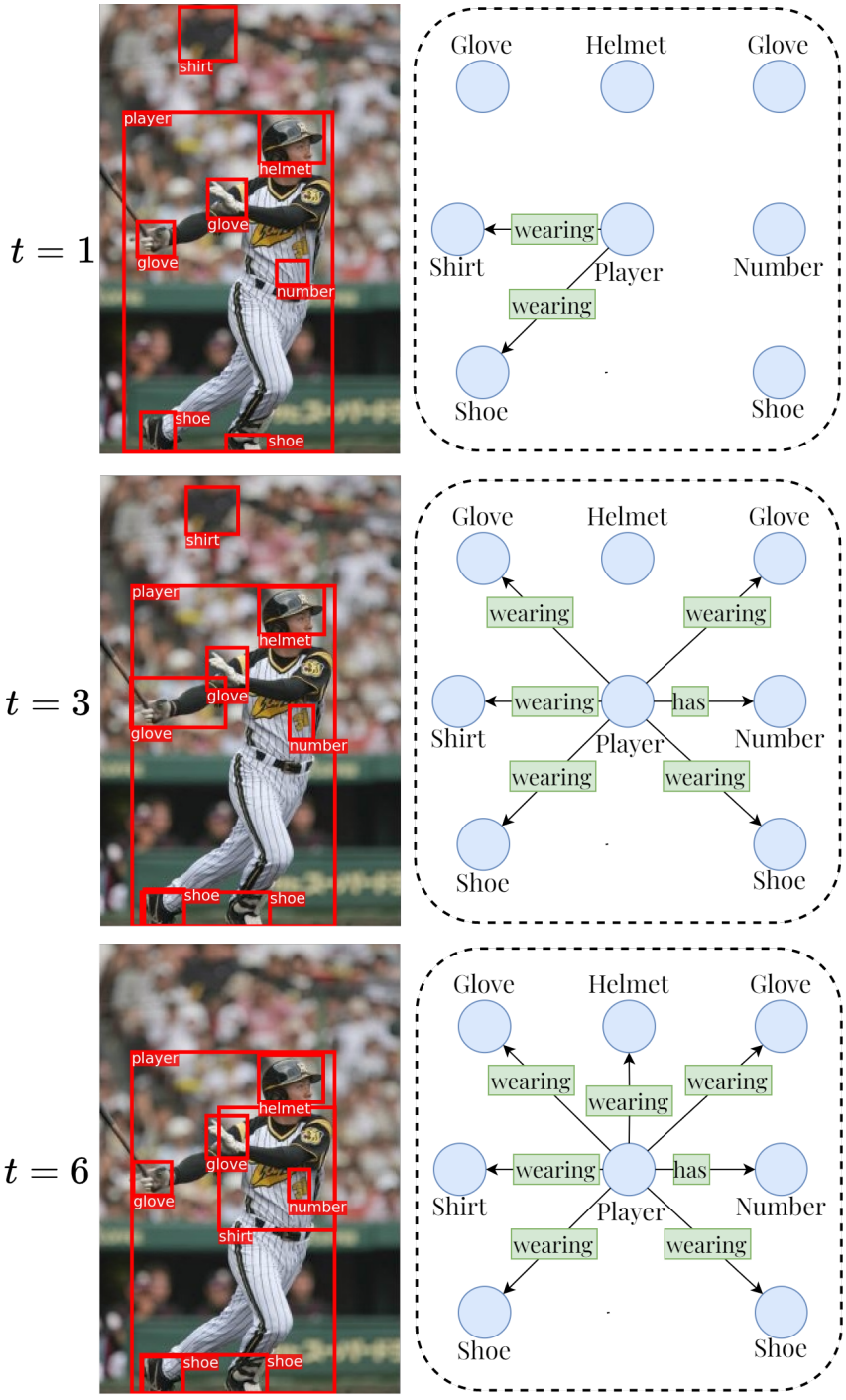

Figure A6: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

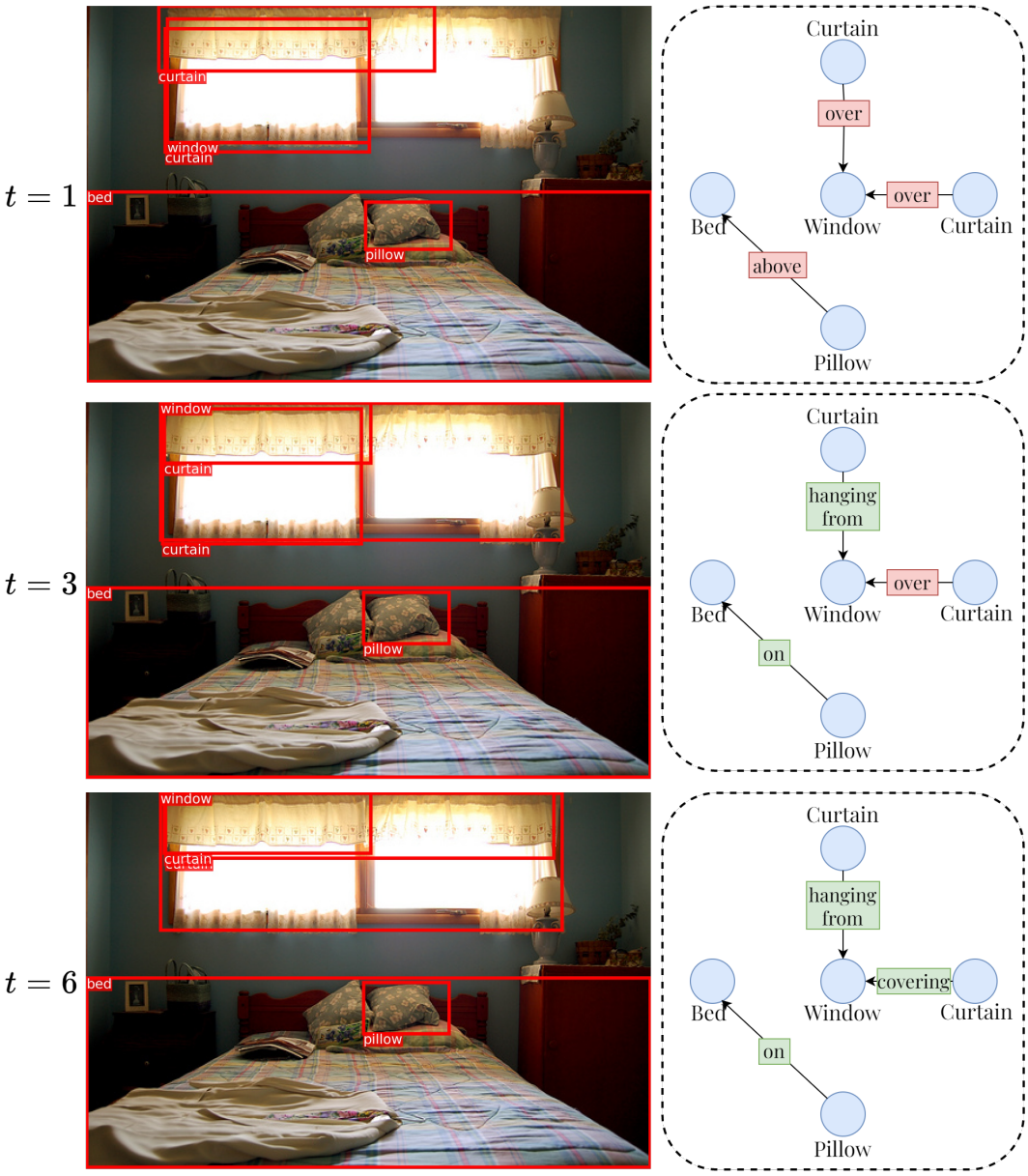

Figure A7: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

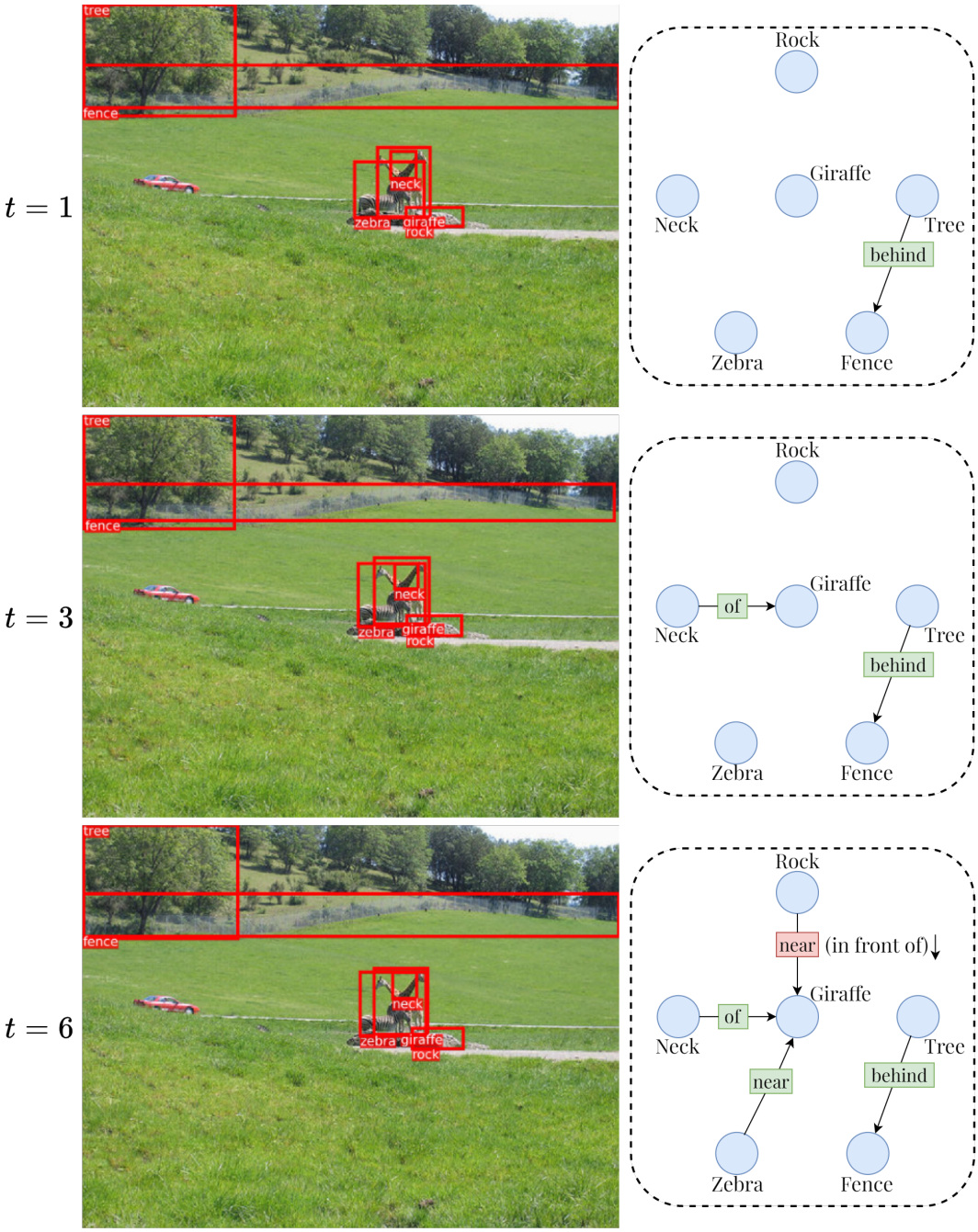

Figure A8: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

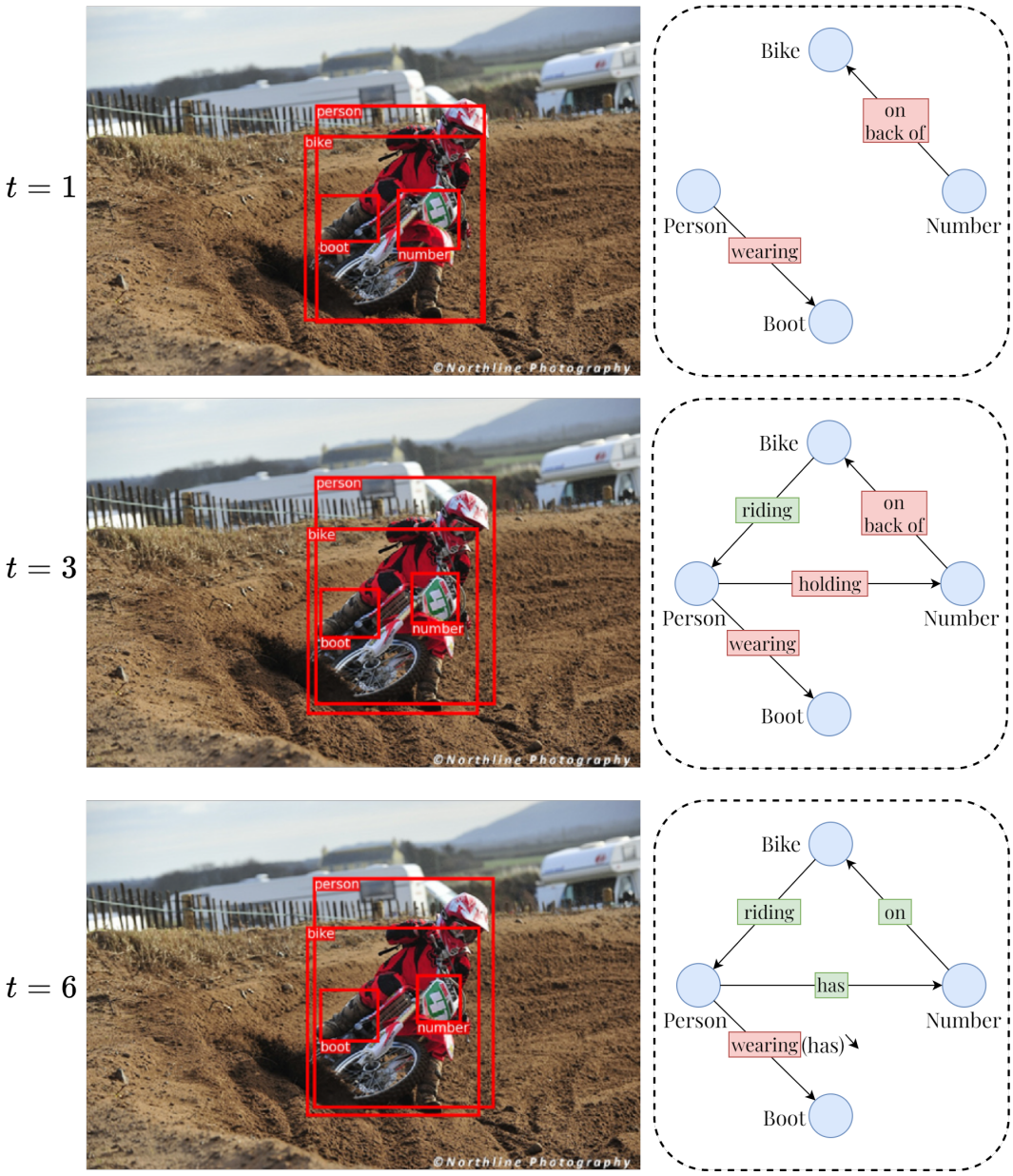

Figure A9: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

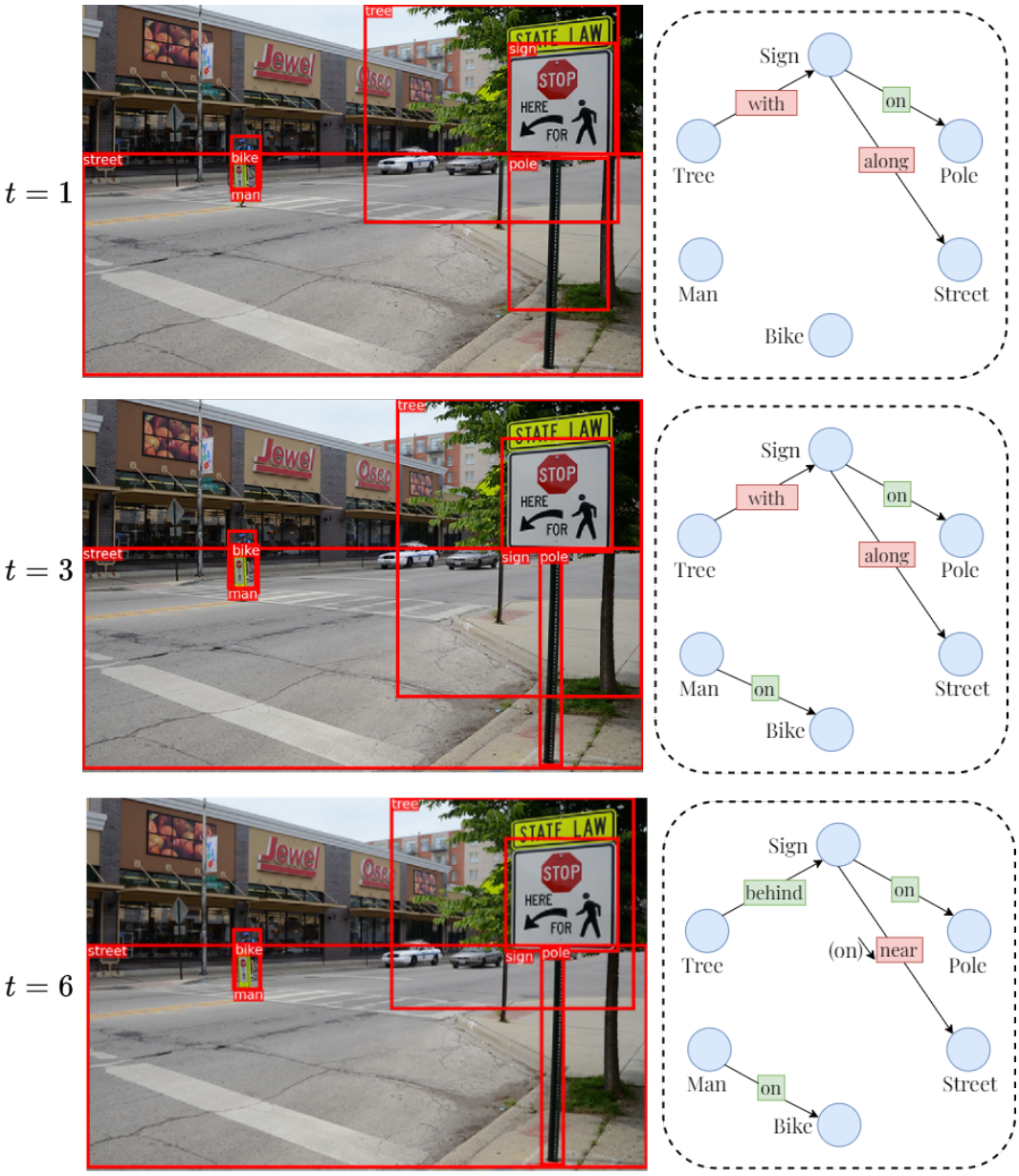

Figure A10: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

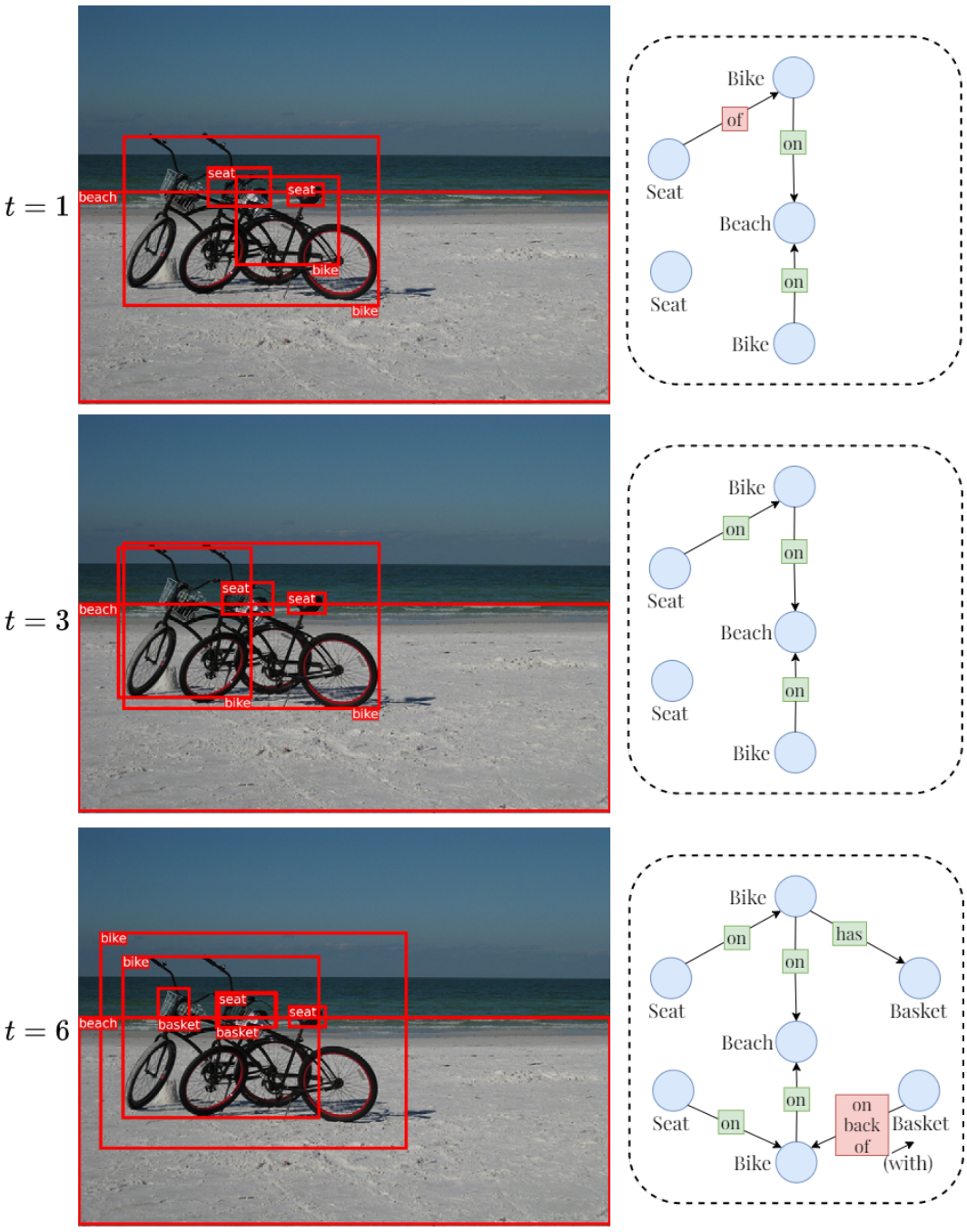

Figure A11: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

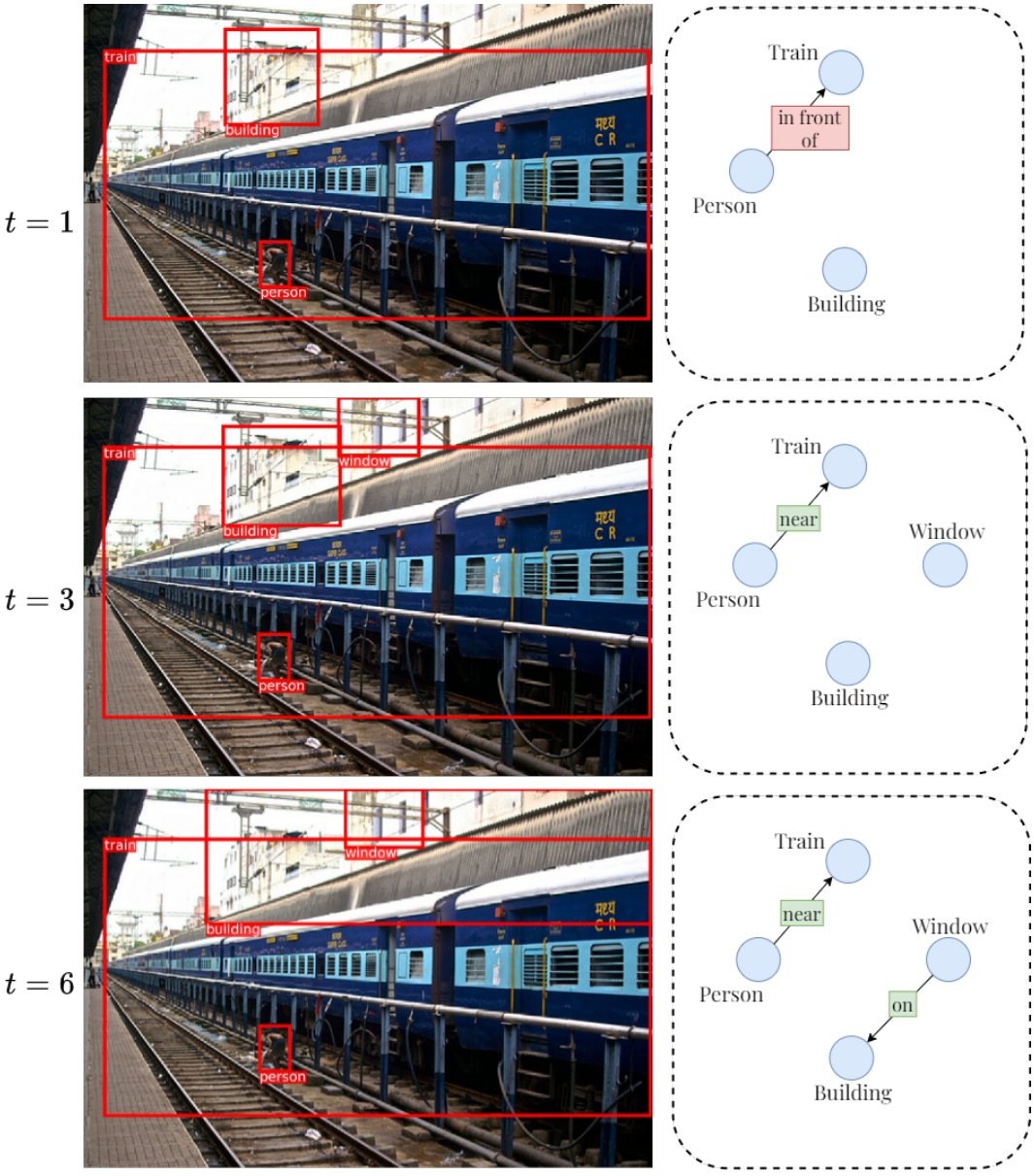

Figure A12: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

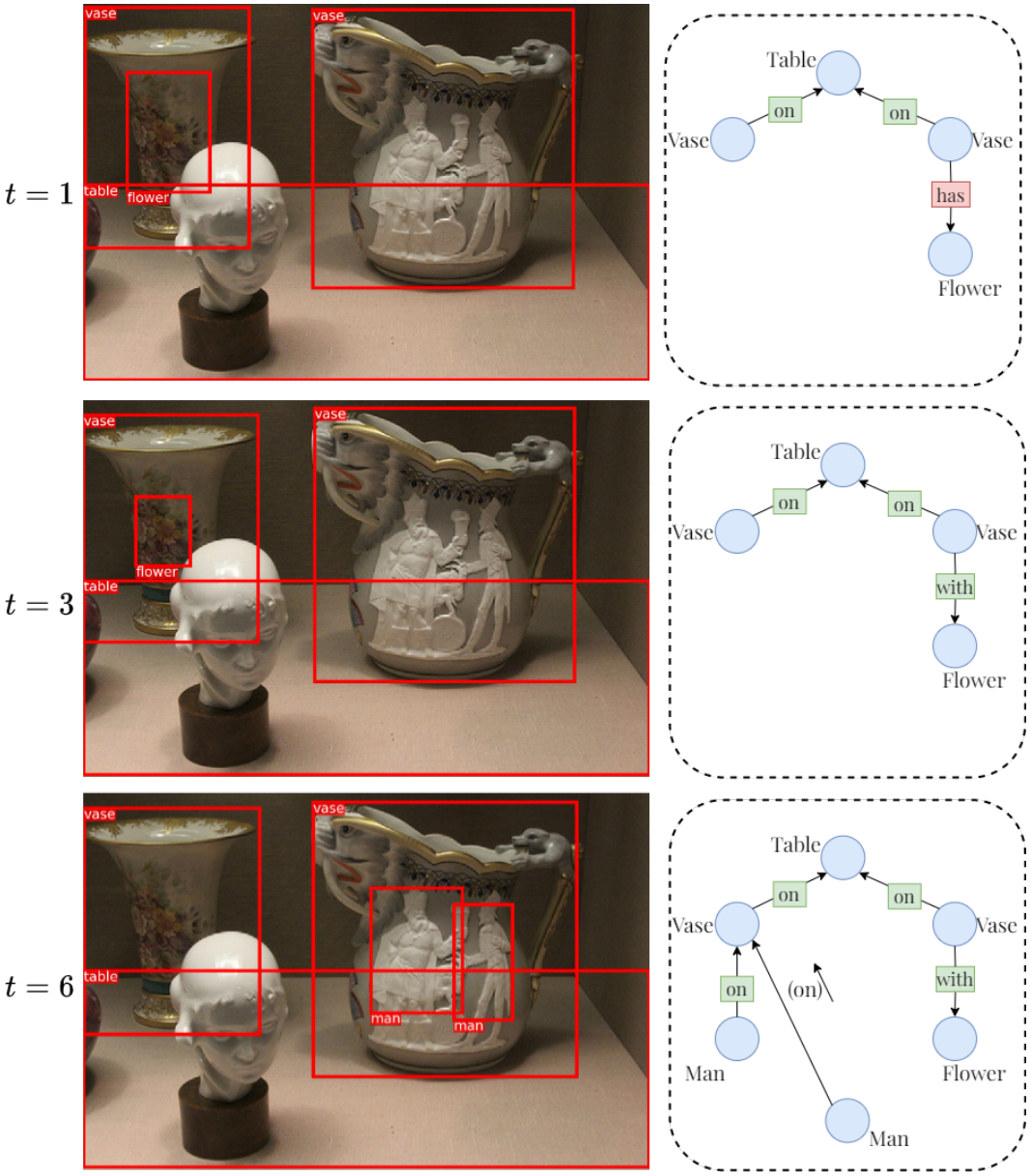

Figure A13: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.

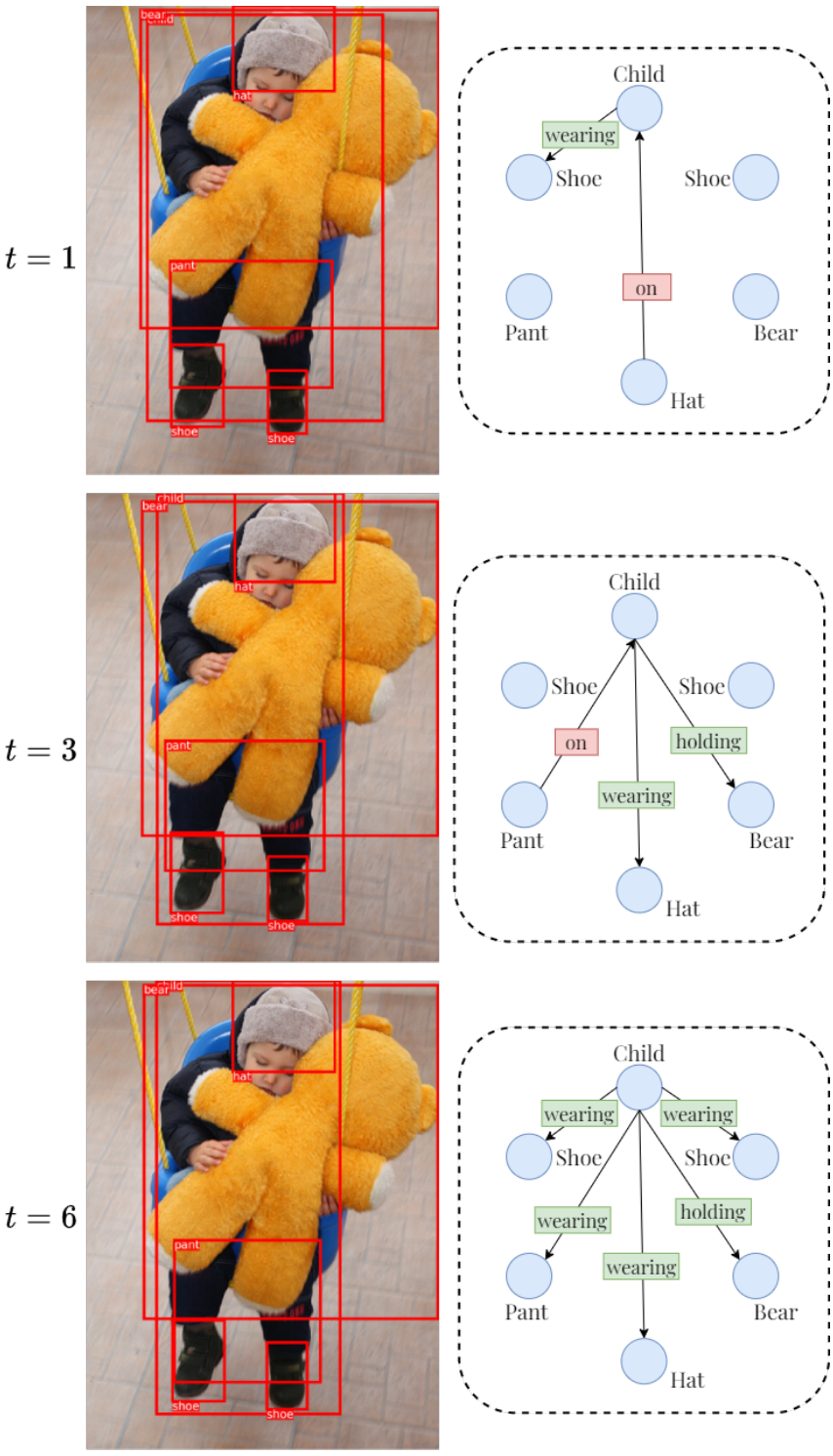

Figure A14: **Qualitative Results.** Graph estimates for different refinement steps ($t = 0, 3,$ and6 are shown. Colors red and green indicate incorrect and correct predictions respectively. For the incorrect predictions at $t = 6$, we additionally mention the correct predicate label in parenthesis next to it, and also show direction of the relation.