arXiv:2001.02314v1 [cs.CV] 7 Jan 2020

Bridging Knowledge Graphs to Generate Scene Graphs

Alireza Zareian, Svebor Karaman, and Shih-Fu Chang
Columbia University, New York, NY, USA
{az2407,sk4089,sc250}@columbia.edu

Abstract

Scene graphs are powerful representations that encode images into their abstract semantic elements, i.e, objects and their interactions, which facilitates visual comprehension and explainable reasoning. On the other hand, commonsense knowledge graphs are rich repositories that encode how the world is structured, and how general concepts interact. In this paper, we present a unified formulation of these two constructs, where a scene graph is seen as an image-conditioned instantiation of a commonsense knowledge graph. Based on this new perspective, we re-formulate scene graph generation as the inference of a bridge between the scene and commonsense graphs, where each entity or predicate instance in the scene graph has to be linked to its corresponding entity or predicate class in the commonsense graph. To this end, we propose a heterogeneous graph inference framework allowing to exploit the rich structure within the scene and commonsense at the same time. Through extensive experiments, we show the proposed method achieves significant improvement over the state of the art.

1. Introduction

Extracting structured, symbolic, semantic representations from data has a long history in Natural Language Processing (NLP), under the umbrella terms of *semantic parsing* at the sentence level [8, 7] and *knowledge extraction* at the document level [20, 34]. The resulting *semantic graphs* or *knowledge graphs* have many applications such as question answering [6, 15] and information retrieval [5, 41]. In computer vision, Xu *et al*. have recently called attention to the task of Scene Graph Generation (SGG) [37], which aims at extracting a symbolic, graphical representation from a given image, where every node corresponds to a localized and categorized object (entity), and every edge encodes a pairwise interaction (predicate). This has inspired two lines of follow-up work, some improving the performance on SGG [22, 28, 42, 38, 21, 36, 10, 9, 1], and others exploiting such rich structures for down-stream tasks such as Visual Question Answering (VQA) [33, 32, 11, 43], image captioning [40, 39], image retrieval [13, 31], and image synthesis [12]. In VQA for instance, SGG not only improves performance, but also promotes interpretability and enables explainable reasoning [32].

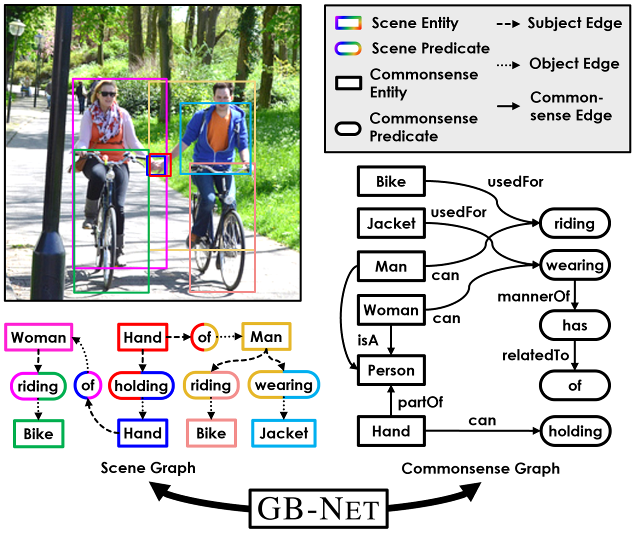

Figure 1. Left: An example of a Visual Genome image and its ground truth scene graph. Right: A relevant portion of the commonsense graph. In this paper we formulate the task of Scene Graph Generation as the problem of creating a bridge between these two graphs. Such bridge not only classifies each scene entity and predicate, but also creates an inter-connected heterogeneous graph whose rich structure is exploited by our method (GB-NET).

Although several methods have been proposed, the state-of-the-art performance for SGG is still far from acceptable. Most recently, [1] achieves only 16% mean recall, for matching the top 100 predicted subject-predicate-object triples against ground truth triples. This suggests the current SGG methods are insufficient to address the complexity of this task. Recently, a few papers have attempted to use external *commonsense* knowledge to advance SGG [42, 9, 1], as well as other domains [2, 14]. This commonsense can range from curated knowledge bases such as ConceptNet [24], ontologies such as WordNet [27], or automatically extracted facts such as co-occurance frequencies [42].

The key message of those works is that a prior knowledge about the world can be very helpful when perceiving a complex scene. If we know the relationship of a Person and a Bike is most likely riding, we can more easily disambiguate between riding, on, and attachedTo, and classify their relationship more accurately. Similarly, if we know a Man and a Woman are both sub-types of Person, even if we only see Man-riding-Bike in training data, we can generalize and recognize a Woman-riding-Bike triplet at test time. Although this idea is intuitively promising, existing methods that implement it have major limitations, as detailed in Section 2, and we address those in the proposed method.

In this paper, we shed light on the similarity of commonsense knowledge graphs and scene graphs, and present a new perspective that unifies those two concepts. Simply put, we formulate both scene and commonsense graphs as knowledge graphs with entity and predicate nodes, and various types of edges. A scene graph node represents an image-specific entity or predicate *instance*, while a commonsense graph node represents an entity or predicate *class*, which is a general concept independent of the image. Similarly, a scene graph edge indicates the participation of an entity instance (*e.g.* as a subject or object) in a predicate instance in a scene, while a commonsense edge states a general fact about the interaction of two concepts in the world. Figure 1 shows an example scene graph and commonsense graph side by side. In the light of this unified perspective, we reformulate the problem of scene graph generation from entity and predicate classification into the problem of bridging the two graphs.

More specifically, given an image, our proposed method initializes potential entity and predicate nodes, and then classifies each node by connecting it to its corresponding class node in the commonsense graph. This establishes a bridge between instance-level, visual knowledge and generic, commonsense knowledge. To incorporate the rich combination of visual and commonsense information in the SGG process, we propose a novel graphical neural network, that iteratively propagates messages between the scene and commonsense graphs as well as within each of them, while gradually refining the bridge in each iteration. This leads to a heterogeneous graph inference framework that successively infers edges and nodes, hereafter dubbed Graph Bridging Network (GB-NET)

The proposed method is novel in several aspects. Firstly, it generalizes the formulation of scene graphs into knowledge graphs where predicates are nodes rather than edges. Secondly, it reformulates scene graph generation from object and relation classification into graph linking. Thirdly, our method is able to exploit various forms of knowledge such as co-occurrence statistics, object affordances, and ontology structures, in a unified manner, by encoding them as various types of edges within a single commonsense graph. This is in contrast to existing knowledge-assisted methods, *e.g.* [1], which is explicitly designed to use triplet statistics only. Finally, our message passing framework iteratively uses graph edges to update node representations, and uses nodes to update bridge edges, while conventional graph-based neural networks rely on fixed edges as input.

To evaluate the effectiveness of our method, we conduct extensive experiments on the Visual Genome [18] dataset. The proposed GB-NET outperforms the state of the art consistently in various performance metrics. Through ablative studies, we show how each of the proposed ideas contribute to the results. We also publicly release a comprehensive software package based on [42] and [1], to reproduce the numbers reported in this paper.

2. Related work

2.1. Scene graph generation

Most SGG methods are based on an object detection backbone that extracts region proposals from the input image. They utilize some kind of information propagation module to incorporate context, and then classify each region to an object class, as well as each pair of regions to a relation class [37, 42, 38, 1]. Our method has two key differences with this conventional process: firstly, our information propagation network operates on a larger graph which consists of not only object nodes, but also predicate nodes and commonsense graph nodes, and has a more complex structure. Secondly, we do not classify each object and relation using classifiers, but instead use a pairwise matching mechanism to connect them to corresponding class nodes in the commonsense graph.

More recently, a few methods [42, 9, 1] have used external knowledge to enhance scene graph generation. This external knowledge is sometimes referred to as "commonsense", because it encodes ontological knowledge about classes, rather than specific instances. Despite encouraging results, these methods have major limitations. Specifically, [42] used triplet frequency to bias the logits of their predicate classifier, and [1] used such frequencies to initialize edge weights on their graphs. Neither of these methods can incorporate other types or knowledge, such as semantic similarity of concepts, or object affordances. Gu *et al.* [9] propose a more general way to incorporate knowledge, by retrieving a set of relevant facts for each object from a pool of commonsense facts. However, their method does not utilize the structure of the commonsense graph, and treats knowledge as a set of triplets. Our method considers commonsense as a general graph with several types of edges, explicitly integrates that graph with the scene graph by connecting corresponding nodes, and incorporates the rich structure of commonsense by graphical message passing.

2.2. Graphical neural networks

By Graphical Neural Networks (GNN), we refer to the family of neural networks that take a graph as input, and iteratively update the representation of each node by applying a learnable function (a.k.a., message) on the node's neighbors. Graph Convolution Networks (GCN) [17], Gated Graph Neural Networks (GGNN) [23], and others are all specific implementations of this general model. Most SGG methods use some variant of GNNs to propagate information between region proposals [37, 22, 38, 1]. Our message passing method, detailed in Section 4, resembles GGNN but instead of propagating messages through a static graph, we update (some) edges as well.

Apart from SGG, GNNs have been used in several other computer vision tasks, often in order to propagate context information across different objects in a scene. For instance, [25] injects a GNN into a Faster R-CNN [30] framework to contextualize the features of region proposals before classifying them. This improves the results since the presence of a `table` can affect the detection of a `chair`. On the other hand, some methods utilize GNNs on graphs that represent the ontology of concepts, rather than objects in a scene [26, 35, 19, 14]. This often enables generalization to unseen or infrequent concepts by incorporating their relationship with frequently seen concepts. More similarly to our work, Chen *et al.* [2] were the first to bring those two ideas together, and form a graph by objects in an image as well as object classes in the ontology. Nevertheless, the class nodes in that work were merely an auxiliary means to improve object features before classification. In contrast, we classify the nodes by explicitly inferring their connection to their corresponding class nodes. Furthermore, their task only involves objects and object classes, while we explore a more complex structure where predicates play an important role as well.

3. Problem Formulation

In this section, we first formalize the concepts of knowledge graph in general, and commonsense graph and scene graph in particular. Leveraging their similarities, we then formulate the problem of scene graph generation as bridging these two graphs.

3.1. Knowledge graphs

We define a knowledge graph as a set of entity and predicate nodes $(\mathcal{N}_{\mathrm{E}}, \mathcal{N}_{\mathrm{P}})$, each with a semantic label, and a set of directed, weighted edges $\mathcal{E}$ from a predefined set of types. Denoting by Δ a node type (here, either entity E or predicate P), the set of edges encoding the relation r between nodes of type Δ and Δ' is defined as

$$\mathcal{E}_r^{\Delta \to \Delta'} \subseteq \mathcal{N}_\Delta \times \mathcal{N}_{\Delta'} \to \mathbb{R}. \tag{1}$$

A commonsense graph is a type of knowledge graph in which each node represents the general concept of its semantic label, and hence each semantic label (entity or predicate class) appears in exactly one node. In such a graph, each edge encodes a relational fact involving a pair of concepts, such as `Hand-partOf-Person` and `Cup-usedFor-Drinking`. Formally, we define the set of commonsense entity (CE) nodes $\mathcal{N}_{\mathrm{CE}}$ and commonsense predicate (CP) nodes $\mathcal{N}_{\mathrm{CP}}$ as all entity and predicate classes in our task. Commonsense edges $\mathcal{E}_{\mathrm{C}}$ consist of 4 distinct subsets, depending on the source and destination node type:

$$\begin{aligned} \mathcal{E}_{\mathrm{C}} = &\{\mathcal{E}_r^{\mathrm{CE}\to\mathrm{CP}}\} \cup \{\mathcal{E}_r^{\mathrm{CP}\to\mathrm{CE}}\} \cup \\ &\{\mathcal{E}_r^{\mathrm{CE}\to\mathrm{CE}}\} \cup \{\mathcal{E}_r^{\mathrm{CP}\to\mathrm{CP}}\}. \end{aligned} \tag{2}$$

A scene graph is a different type of knowledge graph where: (a) each scene entity (SE) node is associated with a bounding box, referring to an image region, (b) each scene predicate (SP) node is associated with an ordered pair of SE nodes, namely a subject and an object, and (c) there are two types of undirected edges which connect each SP to its corresponding subject and object respectively. Here because we define knowledge edges to be directed, we model each undirected subject or object edge as two directed edges in the opposite directions, each with a distinct type. More specifically,

$$\begin{aligned} \mathcal{N}_{\mathrm{SE}} \subseteq &[0,1]^4 \times \mathcal{N}_{\mathrm{CE}}, \\ \mathcal{N}_{\mathrm{SP}} \subseteq &\mathcal{N}_{\mathrm{SE}} \times \mathcal{N}_{\mathrm{SE}} \times \mathcal{N}_{\mathrm{CP}}, \\ \mathcal{E}_{\mathrm{S}} = &\{\mathcal{E}^{\mathrm{SE}\to\mathrm{SP}}_{\texttt{subjectOf}}, \mathcal{E}^{\mathrm{SE}\to\mathrm{SP}}_{\texttt{objectOf}}, \\ &\mathcal{E}^{\mathrm{SP}\to\mathrm{SE}}_{\texttt{hasSubject}}, \mathcal{E}^{\mathrm{SP}\to\mathrm{SE}}_{\texttt{hasObject}}\}, \end{aligned} \tag{3}$$

where $[0,1]^4$ is the set of possible bounding boxes, and $\mathcal{N}_{\mathrm{SE}} \times \mathcal{N}_{\mathrm{SE}} \times \mathcal{N}_{\mathrm{CP}}$ is the set of all possible triples that consist of two scene entity nodes and a scene predicate node. Figure 1 shows an example of scene graph and commonsense graph side by side, to make their similarities clearer. Here we assume every scene graph node has a label that exists in the commonsense graph, since in reality some objects and predicates might belong to background classes, we consider a special commonsense node as background entity and another for background predicate.

3.2. Bridging knowledge graphs

Considering the similarity between the commonsense and scene graph formulations, we make a subtle refinement in the formulation to bridge these two graphs. Specifically, we remove the class from SE and SP nodes and instead encode it into a set of *bridge* edges $\mathcal{E}_{\mathrm{B}}$ that connect each SE or SP node to its corresponding class, *i.e.*, a CE or CP node

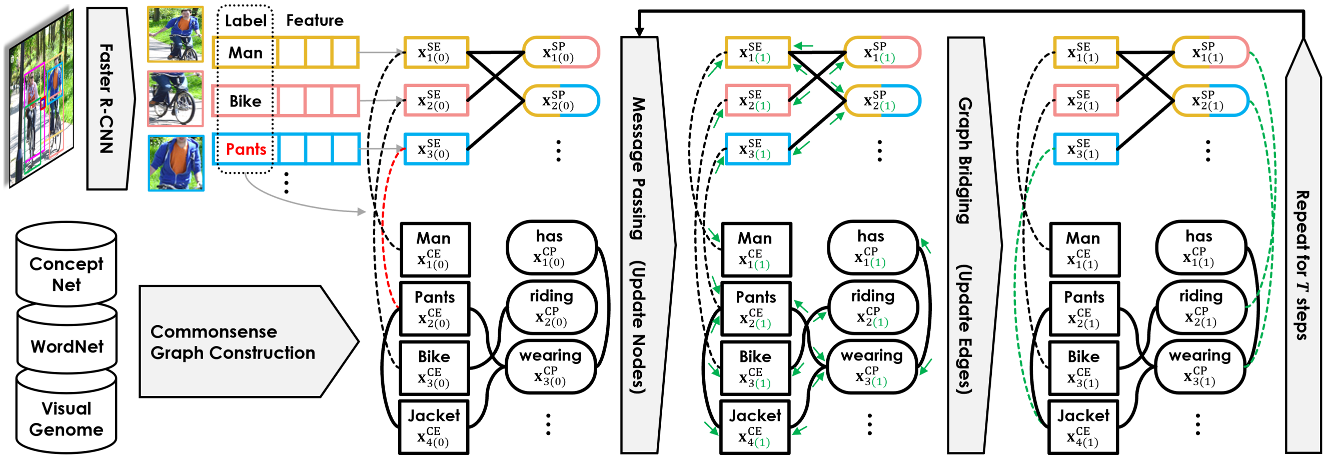
Faster R-CNN
Label
Feature
Man
Bike
Pants
Concept Net
WordNet
Visual Genome
Commonsense Graph Construction
$\mathbf{x}^{\text{SE}}_{1(0)}$
$\mathbf{x}^{\text{SE}}_{2(0)}$
$\mathbf{x}^{\text{SE}}_{3(0)}$
$\mathbf{x}^{\text{SP}}_{1(0)}$
$\mathbf{x}^{\text{SP}}_{2(0)}$
Man $\mathbf{x}^{\text{CE}}_{1(0)}$
Pants $\mathbf{x}^{\text{CE}}_{2(0)}$
Bike $\mathbf{x}^{\text{CE}}_{3(0)}$
Jacket $\mathbf{x}^{\text{CE}}_{4(0)}$
has $\mathbf{x}^{\text{CP}}_{1(0)}$
riding $\mathbf{x}^{\text{CP}}_{2(0)}$
wearing $\mathbf{x}^{\text{CP}}_{3(0)}$
Message Passing (Update Nodes)
$\mathbf{x}^{\text{SE}}_{1(1)}$
$\mathbf{x}^{\text{SE}}_{2(1)}$
$\mathbf{x}^{\text{SE}}_{3(1)}$
$\mathbf{x}^{\text{SP}}_{1(1)}$
$\mathbf{x}^{\text{SP}}_{2(1)}$
Man $\mathbf{x}^{\text{CE}}_{1(1)}$
Pants $\mathbf{x}^{\text{CE}}_{2(1)}$
Bike $\mathbf{x}^{\text{CE}}_{3(1)}$
Jacket $\mathbf{x}^{\text{CE}}_{4(1)}$
has $\mathbf{x}^{\text{CP}}_{1(1)}$
riding $\mathbf{x}^{\text{CP}}_{2(1)}$
wearing $\mathbf{x}^{\text{CP}}_{3(1)}$
Graph Bridging (Update Edges)
Repeat for T steps

Figure 2. An illustrative example of the GB-Net process. First, we initialize the scene graph and entity bridges using Faster R-CNN, which leads to possibly incorrect bridges from scene entity nodes to commonsense entity nodes (*e.g.* `Jacket` object incorrectly linked to `Pants` class). Then we propagate messages throughout the graph to update node representations. Next, we create predicate bridges and update entity bridges by computing a pairwise similarity from scene graph nodes to commonsense nodes. This process is repeated T times and the final bridge edges are the output of this model. Red represents mistake and green represents update. (t) represents time step.

respectively:

$$\begin{aligned} &\mathcal{N}^{?}_{\text{SE}} \subseteq [0,1]^4, \\ &\mathcal{N}^{?}_{\text{SP}} \subseteq \mathcal{N}_{\text{SE}} \times \mathcal{N}_{\text{SE}}, \\ &\mathcal{E}_{\text{B}} = \{\mathcal{E}^{\text{SE}\to\text{CE}}_{\texttt{classifiedTo}}, \mathcal{E}^{\text{SP}\to\text{CP}}_{\texttt{classifiedTo}}, \\ &\qquad \mathcal{E}^{\text{CE}\to\text{SE}}_{\texttt{hasInstance}}, \mathcal{E}^{\text{CP}\to\text{SP}}_{\texttt{hasInstance}}\}, \end{aligned} \tag{4}$$

where $.^{?}$ means the nodes are implicit, *i.e.*, their classes are unknown. Each edge of type `classifiedTo`, connects an entity or predicate to its corresponding label in the commonsense graph, and has a reverse edge of type `hasInstance` which connects the commonsense node back to the instance. Based on this reformulation, we can define the problem of SGG as the extraction of implicit entity and predicate nodes from the image (entity and predicate proposal), and then classifying them by connecting each entity or predicate to the corresponding node in the commonsense graph. Given an input image I and a provided and fixed commonsense graph, the goal of SGG with commonsense knowledge is to maximize

$$\begin{aligned} &p(\mathcal{N}_{\text{SE}}, \mathcal{N}_{\text{SP}}, \mathcal{E}_{\text{S}} | I, \mathcal{N}_{\text{CE}}, \mathcal{N}_{\text{CP}}, \mathcal{E}_{\text{C}}) = \\ &\quad p(\mathcal{N}^{?}_{\text{SE}}, \mathcal{N}^{?}_{\text{SP}}, \mathcal{E}_{\text{S}} | I) \times \\ &\quad p(\mathcal{E}_{\text{B}} | I, \mathcal{N}_{\text{CE}}, \mathcal{N}_{\text{CP}}, \mathcal{E}_{\text{C}}, \mathcal{N}^{?}_{\text{SE}}, \mathcal{N}^{?}_{\text{SP}}, \mathcal{E}_{\text{S}}). \end{aligned} \tag{5}$$

In this paper, the first term is implemented as a region proposal network that infers $\mathcal{N}^{?}_{\text{SE}}$ given the image, followed by a simple predicate proposal algorithm that considers all possible entity pairs as $\mathcal{N}^{?}_{\text{SP}}$. The second term is fulfilled by the proposed GB-Net which infers bridge edges by incorporating the rich structure of the scene and commonsense graphs. Note that unlike most existing methods [42, 1], we do not factorize this into predicting entity classes given the image, and then predicate classes given entities. Therefore, our formulation is more general and allows the proposed method to classify entities and predicates jointly.

4. Method

In this section, we first give an overview of our GB-Net method before detailing the heterogeneous graph construction and update process.

4.1. Overview of GB-Net

The proposed method is illustrated in Figure 2. Given an image, our model first applies a Faster R-CNN [30] to detect objects, and represents them as scene entity (SE) nodes. It also creates a scene predicate (SP) node for each pair of entities, which forms an implicit scene graph, yet to be classified. Given this graph and a background commonsense graph, each with fixed internal connectivity, our goal is to create *bridge* edges between the two graphs that connect each instance (SE and SP node) to its corresponding class (CE and CP node). This corresponds to the second term in Eq 5. To this end, our model initializes entity bridges by connecting each SE to the CE that matches the predicted label Faster R-CNN, and propagates messages among all nodes, through every edge type with dedicated message passing parameters. Given the updated node representations, it computes a pairwise similarity between every SP node and every CP node, and find maximal similarity pairs to connect scene predicates to their corresponding classes, via predicate bridges. It also does the same for entity nodes to potentially refine their connections. Given the new bridges, it propagates messages again, and repeats this process for a predefined number of steps. The final state of the bridge determines which class each node belongs to, which leads to a symbolic scene graph.

4.2. Graph construction

The object detection module outputs a set of n detected objects, each with a bounding box b_j, a label distribution p_j and a region-of-interest feature vector $\mathbf{v}_j$. Then we allocate a *scene entity node* (SE) for each object, initialized using the visual features, *i.e.*,

$$\mathbf{x}_j^{\text{SE}} = \phi_{\text{init}}^{\text{SE}}(\mathbf{v}_j), \tag{6}$$

where ϕ_{init}^{n} is a trainable linear projection. We also allocate a *scene predicate node* (SP) for each (ordered) pair of entities and connect such a predicate node with the related entity nodes via subject and object edges. Specifically, we define the following 4 edge types: for a triplet $n_1 - p - n_2$, we connect p to n_1 using a `hasSubject` edge, p to n_2 using a `hasObject` edge, n_1 to p using a `subjectOf` edge, and n_2 to p using an `objectOf` edge. The reason we have two directions as separate types is that in the message passing phase, the way we use predicate information to update entities should be different from the way we use entities to update predicates. Each predicate is initialized using region-of-interest features of a bounding box enclosing the union of its subject and object. That is

$$\mathbf{x}_j^{\text{SP}} = \phi_{\text{init}}^{\text{SP}}(\mathbf{u}_j). \tag{7}$$

On the other hand, we initialize the commonsense graph with *commonsense entity nodes* (CE) and *commonsense predicate nodes* (CP) using a linear projection of their word embeddings:

$$\begin{aligned} \mathbf{x}_i^{\text{CE}} &= \phi_{\text{init}}^{\text{CE}}(\mathbf{e}_i^{n}), \\ \mathbf{x}_i^{\text{CP}} &= \phi_{\text{init}}^{\text{CP}}(\mathbf{e}_i^{p}). \end{aligned} \tag{8}$$

The commonsense graph also has various types of edges that we describe in more detail later in Section 5.2. Our method is independent of the types of commonsense edges.

So far, we have two isolated graphs, scene and commonsense. An SE node created from a detected `Person` intuitively refers to the `Person` concept in the ontology, and hence the `Person` node in the commonsense graph. Therefore, we connect each SE node to the CE node that corresponds the semantic label predicted by Faster R-CNN, via an `classifiedTo` edge type. Instead of a hard classification, we connect each entity to top K_{bridge} classes using p_j (class distribution predicted by Faster R-CNN) as weights. We also create a reverse connection from each CE node to corresponding SE nodes, using an `hasInstance` edge, but with the same weights p_j. As mentioned earlier, this is to make sure information flows from commonsense to scene as well as scene to commonsense, but not in the same way. We similarly define two other edge types, `classifiedTo` and `hasInstance` for predicates, which are initially an empty set, and will be updated to bridge SP nodes to CP nodes as we explain in the following. These 4 edge types can be seen as flexible *bridges* that connect the two fixed graphs, which are considered latent variables to be determined by the model.

This forms a heterogeneous graph with four types of nodes (SE, SP, CE, and CP) and various types of edges: scene graph edges $\mathcal{E}_{\text{S}}$ such as `subjectOf`, commonsense edges $\mathcal{E}_{\text{C}}$ such as `usedFor`, and bridge edges $\mathcal{E}_{\text{B}}$ such as `classifiedTo`. Next, we explain how our proposed method updates node representations and bridge edges, while keeps commonsense and scene edges constant.

4.3. Successive message passing and bridging

Given a heterogeneous graph as described above, we employ a variant of GGNN [23] to propagate information among nodes. First, each node representation is fed into a fully connected network to compute *outgoing* messages, that is

$$\mathbf{m}_i^{\Delta\rightarrow} = \phi_{\text{send}}^{\Delta}(\mathbf{x}_i^{\Delta}), \tag{9}$$

for each i and node type Δ, where ϕ_{send} is a trainable *send head* which has shared weights across nodes of each type. After computing outgoing messages, we send them through all outgoing edges, multiplying by the edge weight. Then for each node, we aggregate incoming messages, by first adding across edges of the same type, and then concatenating across edge types. We compute the *incoming* message for each node by applying another fully connected network on the aggregated messages:

$$\mathbf{m}_j^{\Delta\leftarrow} = \phi_{\text{receive}}^{\Delta}\left(\bigcup_{\Delta'} \bigcup^{\mathcal{E}_k \in \mathcal{E}^{\Delta'\rightarrow\Delta}} \sum_{(i,j,a_{ij}^k)\in\mathcal{E}_k} a_{ij}^k \mathbf{m}_i^{\Delta'\rightarrow} \right), \tag{10}$$

where ϕ_{receive} is a trainable *receive head* and $\cup$ denotes concatenation. Note that the first concatenation is over all 4 node types, the second concatenation is over all edge types from Δ' to Δ, and the sum is over all edges of that type, where i and j are the head and tail nodes, and a_{ij}^k is the edge weight. Given the incoming message for each node, we update the representation of the node using a Gated Recurrent Unit (GRU) update rule, following [3]:

$$\begin{aligned} \mathbf{z}_j^{\Delta} &= \sigma\big(W_z^{\Delta}\mathbf{m}_j^{\Delta\leftarrow} + U_z^{\Delta}\mathbf{x}_j^{\Delta}\big), \\ \mathbf{r}_j^{\Delta} &= \sigma\big(W_r^{\Delta}\mathbf{m}_j^{\Delta\leftarrow} + U_r^{\Delta}\mathbf{x}_j^{\Delta}\big), \\ \mathbf{h}_j^{\Delta} &= \tanh\big(W_h^{\Delta}\mathbf{m}_j^{\Delta\leftarrow} + U_h^{\Delta}(\mathbf{r}_j^{\Delta} \odot \mathbf{x}_j^{\Delta})\big), \\ \mathbf{x}_j^{\Delta} &\Leftarrow (1 - \mathbf{z}_j^{\Delta}) \odot \mathbf{x}_j^{\Delta} + \mathbf{z}_j^{\Delta} \odot \mathbf{h}_j^{\Delta}, \end{aligned} \tag{11}$$

where σ is the sigmoid function, and $W^{\Delta}_{\cdot}$ and $U^{\Delta}_{\cdot}$ are trainable matrices that are shared across nodes of the same type, but distinct for each node type Δ. This update rule can be seen as an extension of GGNN [23] to heterogeneous graphs, with a more complex message aggregation strategy. Note that $\Leftarrow$ means we update the node representation. Mathematically, this means $\mathbf{x}^{\Delta}_{j(t+1)} = U(\mathbf{x}^{\Delta}_{j(t)})$, where U is the aforementioned update rule and (t) denotes iteration number. For simplicity, we drop this subscript throughout this paper.

So far, we have explained how to update node representations using graph edges. Now using the new node representations, we should update the bridge edges $\mathcal{E}_{\text{B}}$ that connect scene nodes to commonsense nodes. To this end, we compute a pairwise similarity from each SE to all CE nodes, and from each SP to all CP nodes.

$$\mathbf{a}^{\text{EB}}_{ij} = \frac{\exp\langle \mathbf{x}^{\text{SE}}_i, \mathbf{x}^{\text{CE}}_j \rangle_{\text{EB}}}{\sum_{j'} \exp\langle \mathbf{x}^{\text{SE}}_i, \mathbf{x}^{\text{CE}}_{j'} \rangle_{\text{EB}}}, \tag{12}$$

where

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\text{EB}} = \phi^{\text{SE}}_{\text{att}}(\mathbf{x})^T \phi^{\text{CE}}_{\text{att}}(\mathbf{y}), \tag{13}$$

and similarly for predicates,

$$\mathbf{a}^{\text{PB}}_{ij} = \frac{\exp\langle \mathbf{x}^{\text{SP}}_i, \mathbf{x}^{\text{CP}}_j \rangle_{\text{PB}}}{\sum_{j'} \exp\langle \mathbf{x}^{\text{SP}}_i, \mathbf{x}^{\text{CP}}_{j'} \rangle_{\text{PB}}}, \tag{14}$$

where

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\text{PB}} = \phi^{\text{SP}}_{\text{att}}(\mathbf{x})^T \phi^{\text{CP}}_{\text{att}}(\mathbf{y}). \tag{15}$$

Here $\phi^{\Delta}_{\text{att}}$ is a fully connected network that resembles *attention head* in transformers. Note that since $\phi^{\Delta}_{\text{att}}$ is not shared across node types, our similarity metric is asymmetric. We use each $\mathbf{a}^{\text{EB}}_{ij}$ to set the edge weight of the `classifiedTo` edge from $\mathbf{x}^{\text{SE}}_i$ to $\mathbf{x}^{\text{CE}}_j$, as well as the `hasInstance` edge from $\mathbf{x}^{\text{CE}}_j$ to $\mathbf{x}^{\text{SE}}_i$. Similarly we use each $\mathbf{a}^{\text{PB}}_{ij}$ to set the weight of edges between $\mathbf{x}^{\text{SP}}_i$ and $\mathbf{x}^{\text{CP}}_j$. In preliminary experiments we realised that such fully connected bridges hurt performance in large graphs. Hence, we only keep the top K_{bridge} values of $\mathbf{a}^{\text{EB}}_{ij}$ for each i, and set the rest to zero. We do the same thing for predicates, keeping the top K_{bridge} values of $\mathbf{a}^{\text{PB}}_{ij}$ for each i. Given the updated bridges, we propagate messages again to update node representations, and iterate for a fixed number of steps, T. The final values of $\mathbf{a}^{\text{EB}}_{ij}$ and $\mathbf{a}^{\text{PB}}_{ij}$ are the outputs of our model, which can be used to classify each entity and predicate in the scene graph.

4.4. Training

We closely follow [1] which itself follows [42] for training procedure. Specifically, given the output and ground truth graphs, we align output entities and predicates to ground truth counterparts. To align entities we use IoU and predicates will be aligned naturally since they correspond to aligned pairs of entities. Then we use the output probability scores of each node to define a cross-entropy loss. The sum of all node-level loss values will be the objective function to be minimized using Adam [16].

Due to the highly imbalanced predicate statistics in Visual Genome, we observed that best-performing models usually concentrate their performance merely on the most frequent classes such as `on` and `wearing`. To alleviate this, we modify the basic cross-entropy objective that is commonly used by assigning an importance weight to each class. We follow the recently proposed class-balanced loss [4] where the weight of each class is inversely proportional to its frequency. More specifically, we use the following loss function for each predicate node:

$$\mathcal{L}^P_i = -\frac{1-\beta}{1-\beta^{n_j}} \log \mathbf{a}^{\text{PB}}_{ij}, \tag{16}$$

where j is the class index of the ground truth predicate aligned with i, n_j is the frequency of class j in training data, and β is a hyperparameter. Note that $\beta = 0$ leads to a regular cross-entropy loss, and the more it approaches 1, the more strictly it suppresses frequent classes. To be fair in comparison with other methods, we include a variant of our method without reweighting, which still outperforms all other methods.

5. Experiments

Following the literature, we use the large-scale Visual Genome benchmark [18] to evaluate our method. We first show how our GB-Net compares to the state of the art, by extensively evaluating it on 24 performance metrics. Then we present an ablation study to illustrate how each innovation contributes to the performance.

5.1. Task description

Visual Genome [18] consists of 108,077 images with annotated objects (entities) and pairwise relationships (predicates), which is then post-processed by [37] to create scene graphs. They use the most frequent 150 entity classes and 50 predicate classes to filter the annotations. Figure 1 shows an example of their post-processed scene graphs which we use as ground truth. We closely follow their evaluation settings such as train and test splits.

The task of scene graph generation, as described in Section 4, is equivalent to the SGGen scenario proposed by [37] and followed ever since. Given an image, the task of SGGen is to jointly infer entities and predicates from scratch. Since this task is complicated and involves too many degrees of freedom, [37] also introduced two other tasks that are mainly for performance analysis and diagnosis. In SGCls, we take localization (here region proposal network) out of the picture, by providing the model

Task	Metric	Graph	Method					
		Constraint	IMP+ [37]	FREQ [42]	SMN [42]	KERN [1]	GB-Net	GB-Net-β
SGGen	mR@50	Yes	3.8	4.3	5.3	*6.4*	6.1	**7.1**
		No	5.4	5.9	9.3	*11.7*	9.8	**11.7**
	mR@100	Yes	4.8	5.6	6.1	7.3	*7.3*	**8.5**
		No	8.0	8.9	12.9	*16.0*	14.0	**16.6**
	R@50	Yes	20.7	23.5	**27.2**	*27.1*	26.4	26.3
		No	22.0	25.3	*30.5*	**30.9**	29.4	29.3
	R@100	Yes	24.5	27.6	**30.3**	29.8	*30.0*	29.9
		No	27.4	30.9	*35.8*	**35.8**	35.1	35.0
SGCls	mR@50	Yes	5.8	6.8	7.1	9.4	*9.6*	**12.7**
		No	12.1	13.5	15.4	19.8	*21.4*	**25.6**
	mR@100	Yes	6.0	7.8	7.6	10.0	*10.2*	**13.4**
		No	16.9	19.6	20.6	26.2	*29.1*	**32.1**
	R@50	Yes	34.6	32.4	35.8	36.7	**38.0**	*37.3*
		No	43.4	40.5	44.5	45.9	**47.7**	*46.9*
	R@100	Yes	35.4	34.0	36.5	37.4	**38.8**	*38.0*
		No	47.2	43.7	47.7	49.0	**51.1**	*50.3*
PredCls	mR@50	Yes	9.8	13.3	13.3	17.7	*19.3*	**22.1**
		No	20.3	24.8	27.5	36.3	*41.1*	**44.5**
	mR@100	Yes	10.5	15.8	14.4	19.2	*20.9*	**24.0**
		No	28.9	37.3	37.9	49.0	*55.4*	**58.7**
	R@50	Yes	59.3	59.9	65.2	65.8	**66.6**	*66.6*
		No	75.2	71.3	81.1	81.9	**83.6**	*83.5*
	R@100	Yes	61.3	64.1	67.1	67.6	**68.2**	*68.2*
		No	83.6	81.2	88.3	88.9	**90.5**	*90.3*

Table 1. The performance of our method (mean and overall triplet recall, at top 50 and top 100, with and without graph constraint, for the three tasks of SGGen, SGCls and PredCls) compared to the state of the art on the original VG split [37]. Numbers are in percentage. All baseline numbers were borrowed from [1]. Top two methods for each metric is shown in **bold** and *italic* respectively. Both the original and balanced versions of our method outperform the state of the art consistently on all metrics.

with ground truth bounding boxes even during test. The task is hence to label bounding boxes with entity classes and infer predicates. In PredCls, we take object detection for granted, and provide the model with not only ground truth bounding boxes, but also their true entity class. The task is thus merely predicate inference. In each task, the main evaluation metric is average per-image recall of the top K subject-predicate-object triplets. The confidence of a triplet that is used for ranking is computed by multiplying the classification confidence of all three elements. Given the ground truth scene graph, each predicate forms a triplet, which we match against the top K triplets in the output scene graph. A triplet is matched if all three elements are classified correctly, and the bounding boxes of subject and object match with an IoU of at least 0.5.

There are three aspects of variation in computing the recall. The first is K, for which we conventionally choose 50 and 100. The second aspect is whether or not to enforce the so-called *graph constraint*, which limits the top K triplets to only one predicate for each ordered entity pair. The third aspect, as introduced by [1], is whether to compute the recall for each predicate class separately and take the mean (mR), or compute a single recall for all triplets (R). For each of the three SGG tasks, we report overall recall at $K = 50$ and $K = 100$ (R@50 and R@100), mean recall at the same values of K: (mR@50 and mR@100), and we report all these four metrics with and without enforcing graph constraint, leading to 8 metric for each task, 24 overall.

5.2. Implementation details

We use a fully connected network with one hidden layer and ReLU activation for each ϕ_{send}, ϕ_{receive} and ϕ_{att}. We use a linear layer for each ϕ_{init}. We set the dimension of node representations to 1024, and perform 3 message passing steps, except in ablation experiments where we try 1, 2 and 3. We tried various values for β. Generally the higher it is, mean recall improves and recall is hurt. We find that 0.999 is a good trade-off. Also we used $K_{\text{bridge}} = 5$. All hyperparameters are tuned using a validation set randomly selected from training data. We borrow the Faster R-CNN backbone trained by [42], which predicts 128 proposals.

In our commonsense graph, the nodes are the 151 en-

Method	SGGen				PredCls			
	mR@50	mR@100	R@50	R@100	mR@50	mR@100	R@50	R@100
No Knowledge	5.5	6.6	25.3	28.8	15.4	16.8	62.5	64.5
$T = 1$	5.6	6.7	24.9	28.5	15.6	17.1	62.1	64.2
$T = 2$	5.7	6.9	26.1	29.7	18.2	19.7	66.7	68.4
GB-Net	**6.1**	**7.3**	**26.4**	**30.0**	**18.2**	**19.7**	**67.0**	**68.6**

Table 2. Ablation study on Visual Genome. All numbers are in percentage, and graph constraint is enforced.

tity classes and 51 predicate classes that are fixed by [37], including background. We use the GloVE [29] embedding of category titles to initialize their node representation (via ϕ_{init}). We compile our commonsense edges from three sources, WordNet [27], ConceptNet [24], and Visual Genome. To summarize, there are three groups of edge types in our commonsense graph. We have `SimilarTo` from WordNet hierarchy, we have `PartOf`, `RelatedTo`, `IsA`, `MannerOf`, and `UsedFor` from ConceptNet, and finally from VG training data we have conditional probabilities of subject given predicate, predicate given subject, subject given object, *etc*. We explain this process in detail in the supplementary material. The process of graph generation involves manual effort, thus we make the created commonsense graph publicly available as a part of our code.

5.3. Main results

Table 1 summarizes our results in comparison to the state of the art. IMP+ refers to the re-implementation of [37] by [42] using their new Faster R-CNN backbone. That method does not use any external knowledge and only uses message passing among the entities and predicates and then classifies each. Hence, it can be seen as a strong, but knowledge-free baseline. FREQ is a simple baseline proposed by [42], which predicts the most frequent predicate for any given pair of entities, solely based on statistics from the training data FREQ surprisingly outperforms IMP+, confirming the importance of commonsense knowledge in SGG.

SMN [42] applies bi-directional LSTMs on top of the entity features, then classifies each entity and each pair. They bias their classifier logits using statistics from FREQ, which improves their total recall significantly, at the expense of higher bias against less frequent classes, as revealed by [1]. More recently, KERN [1] encodes VG statistics into the edge weights of the graph, which is then incorporated by propagating messages. Since it encodes statistics more implicitly, KERN is less biased compared to SMN, which improves mR. Our method, GB-Net, encodes those statistics as well as other forms of knowledge such as WordNet, ConceptNet, and word embeddings, within a rich, heterogeneous knowledge graph, and classifies the scene graph by dynamically refining its bridges into the commonsense graph, while simultaneously passing messages between and within the two graphs. This improves both R and mR significantly. Further, our class-balanced model, GB-Net-β, further enhances mR significantly without hurting R by much.

We observed that the state of the art performance has been saturated in the SGGen setting, especially for overall recall. This is partly because object detection performance is a bottleneck that limits the performance. It is worth noting that mean recall is a more important metric than overall recall, since most SGG methods tend to score a high overall recall by investing on few most frequent classes, and ignoring the rest [1]. As shown in Table 1, our method achieves significant improvements in mean recall. We provide in-depth performance analysis by comparing our recall per predicate class with that of the state of the art, as well as qualitative analysis in the supplementary material.

5.4. Ablation study

To further explain our performance improvement, Table 2 compares our full method with its weaker variants. Specifically, to investigate the effectiveness of commonsense knowledge, we remove the commonsense graph and instead classify each node in our graph using a 2-layer fully connected classifier after message passing. This negatively impacts performance in all metrics, proving our method is able to exploit commonsense knowledge through the proposed bridging technique. Moreover, to highlight the importance of our proposed message passing and bridge refinement process, we repeated the experiments with fewer steps. We observe the performance improves consistently with more steps, proving the effectiveness of our model.

6. Conclusion

We proposed a new method for Scene Graph Generation that incorporates external commonsense knowledge in a novel, graphical neural framework. We unify the formulation of scene graphs and commonsense graphs as two types of knowledge graph, which are fused into a single graph through a dynamic message passing and bridging algorithm. Our method iteratively propagates messages to update nodes, then compares nodes to update bridge edges, and repeats until the two graphs are carefully connected. Through extensive experiments on the Visual Genome dataset, we showed our method, KG-Fuse, outperforms the state of the art in various metrics.

References

[1] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6163–6171, 2019. 1, 2, 3, 4, 6, 7, 8

[2] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7239–7248, 2018. 1, 3

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 5

[4] 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 Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019. 6

[5] Laura Dietz, Alexander Kotov, and Edgar Meij. Utilizing knowledge graphs for text-centric information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1387–1390. ACM, 2018. 1

[6] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM, 2014. 1

[7] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, 2014. 1

[8] Matt Gardner, Pradeep Dasigi, Srinivasan Iyer, Alane Suhr, and Luke Zettlemoyer. Neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 17–18, 2018. 1

[9] Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. Scene graph generation with external knowledge and image reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1969–1978, 2019. 1, 2

[10] Roei Herzig, Moshiko Raboh, Gal Chechik, Jonathan Berant, and Amir Globerson. Mapping images to scene graphs with permutation-invariant structured prediction. In *Advances in Neural Information Processing Systems*, pages 7211–7221, 2018. 1

[11] Drew A Hudson and Christopher D Manning. Learning by abstraction: The neural state machine. *arXiv preprint arXiv:1907.03950*, 2019. 1

[12] 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. 1

[13] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015. 1

[14] Keizo Kato, Yin Li, and Abhinav Gupta. Compositional learning for human object interaction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 234–251, 2018. 1, 3

[15] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. Question answering as global reasoning over semantic abstractions. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 1

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3

[18] 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. 2, 6

[19] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Multi-label zero-shot learning with structured knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1576–1585, 2018. 3

[20] Manling Li, Ying Lin, Joseph Hoover, Spencer Whitehead, Clare Voss, Morteza Dehghani, and Heng Ji. Multilingual entity, relation, event and human value extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 110–115, 2019. 1

[21] Yikang Li, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. Factorizable net: an efficient subgraph-based framework for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 335–351, 2018. 1

[22] 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. 1, 3

[23] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 3, 5, 6

[24] Hugo Liu and Push Singh. Conceptneta practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004. 1, 8

[25] Yong Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Structure inference net: Object detection using scene-level context and instance-level relationships. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pages 6985–6994, 2018. 3

[26] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016. 3

[27] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 1, 8

[28] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. In *Advances in neural information processing systems*, pages 2171–2180, 2017. 1

[29] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 8

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 3, 4

[31] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, pages 70–80, 2015. 1

[32] Jiaxin Shi, Hanwang Zhang, and Juanzi Li. Explainable and explicit visual reasoning over scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8376–8384, 2019. 1

[33] Damien Teney, Lingqiao Liu, and Anton van den Hengel. Graph-structured representations for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2017. 1

[34] David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. Entity, relation, and event extraction with contextualized span representations. *arXiv preprint arXiv:1909.03546*, 2019. 1

[35] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6857–6866, 2018. 3

[36] Sanghyun Woo, Dahun Kim, Donghyeon Cho, and In So Kweon. Linknet: Relational embedding for scene graph. In *Advances in Neural Information Processing Systems*, pages 560–570, 2018. 1

[37] 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. 1, 2, 3, 6, 7, 8

[38] 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. 1, 2, 3

[39] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019. 1

[40] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 684–699, 2018. 1

[41] Jing Yu, Yuhang Lu, Zengchang Qin, Weifeng Zhang, Yanbing Liu, Jianlong Tan, and Li Guo. Modeling text with graph convolutional network for cross-modal information retrieval. In *Pacific Rim Conference on Multimedia*, pages 223–234. Springer, 2018. 1

[42] 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. 1, 2, 4, 6, 7, 8

[43] Cheng Zhang, Wei-Lun Chao, and Dong Xuan. An empirical study on leveraging scene graphs for visual question answering. *arXiv preprint arXiv:1907.12133*, 2019. 1