Deep Generative Probabilistic Graph Neural Networks for Scene Graph Generation

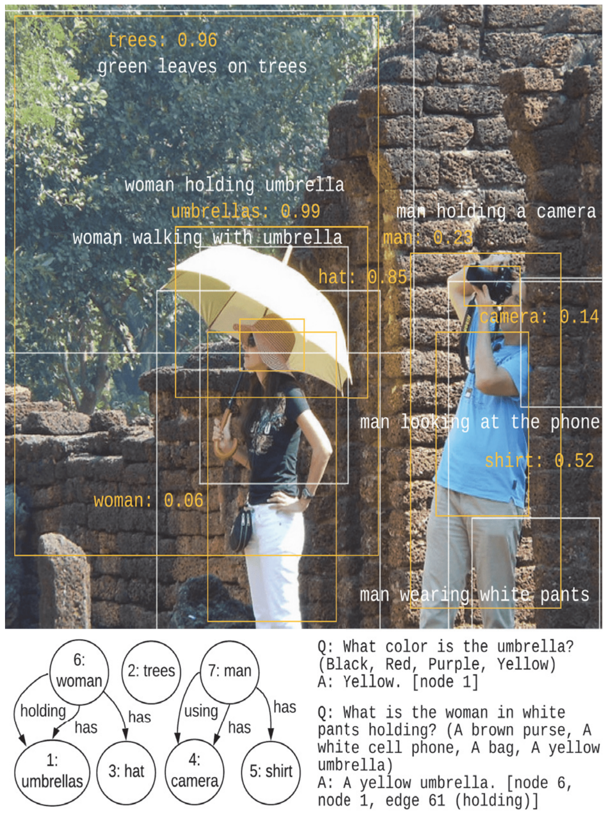
Mahmoud Khademi, Oliver Schulte
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
{mkhademi, oschulte}@sfu.ca

Abstract

We propose a new algorithm, called Deep Generative Probabilistic Graph Neural Networks (DG-PGNN), to generate a scene graph for an image. The input to DG-PGNN is an image, together with a set of region-grounded captions and object bounding-box proposals for the image. To generate the scene graph, DG-PGNN constructs and updates a new model, called a Probabilistic Graph Network (PGN). A PGN can be thought of as a scene graph with uncertainty: it represents each node and each edge by a CNN feature vector and defines a probability mass function (PMF) for node-type (object category) of each node and edge-type (predicate class) of each edge. The DG-PGNN sequentially adds a new node to the current PGN by learning the optimal ordering in a Deep Q-learning framework, where states are partial PGNs, actions choose a new node, and rewards are defined based on the ground-truth. After adding a node, DG-PGNN uses message passing to update the feature vectors of the current PGN by leveraging contextual relationship information, object co-occurrences, and language priors from captions. The updated features are then used to fine-tune the PMFs. Our experiments show that the proposed algorithm significantly outperforms the state-of-the-art results on the Visual Genome dataset for scene graph generation. We also show that the scene graphs constructed by DG-PGNN improve performance on the visual question answering task, for questions that need reasoning about objects and their interactions in the scene context.

1 Introduction

Visual understanding of a scene is one of the most important objectives in computer vision. Over the past decade, there have been great advances in relevant tasks such as image classification and object detection. However, understanding a scene is not only recognizing the objects in the scene. The interactions between objects also play a crucial role in visual understanding of a scene. To represent the semantic content of an image, previous work proposed to build a graph called scene graph (Krishna et al. 2016; Lu et al. 2016; Li et al. 2017), where the nodes represent the objects and the edges show the relationships between them (see Figure 1). This paper therefore introduces a new algorithm, Deep Generative Probabilistic Graph Neural Networks (DG-PGNN), to generate a scene graph for an image.



Q: What color is the umbrella? (Black, Red, Purple, Yellow)
A: Yellow. [node 1]

Q: What is the woman in white pants holding? (A brown purse, A white cell phone, A bag, A yellow umbrella)
A: A yellow umbrella. [node 6, node 1, edge 61 (holding)]

Figure 1: A scene graph represents semantic content of an image via a graph. We leverage region-grounded captions to construct a scene graph. The graph is useful to answer visual questions such as *What color is the umbrella?* (the attributes of node *umbrella*) and *What is the woman in white pants holding?* (the relationship triplet *woman-holding-umbrella*).

An efficient scene graph generation model needs to leverage visual contextual information as well as language priors. Previous work proposed to learn a contextualized representation for nodes and edges either by sending messages across a fixed complete graph (Xu et al. 2017), where each node is potentially an object, or by assuming a fixed linear ordering for the bounding-boxes and applying a bidirectional LSTM (Zellers et al. 2018). The node and edge representations are then used to infer the node-types and edge-types. However, these models cannot leverage the rich latent graph-structured nature of the input. Unlike these approaches, our method is well-designed for graph-structured input data; we simultaneously build the graph and fine-tune the predictions, as we get new information about the scene.

We introduce a new graph neural network model, Probabilistic Graph Network (PGN), to represent a scene graph with uncertainty. The PGN is a probabilistic extension to Graph Network (Battaglia et al. 2018). In a PGN, each node and each edge is represented by a CNN feature vector, and the degree of belief about node-type (object category) of a node and edge-type (predicate class) of an edge is represented by a probability mass function (PMF). Our novel DG-PGNN algorithm constructs a PGN as follows. It sequentially adds a new node to the current PGN by learning the optimal ordering in a Deep Q-learning framework, where states are partial PGNs, actions choose a new node, and rewards are defined based on the ground-truth scene graph of the input image. After adding a new node, DG-PGNN uses message passing to update the feature vectors of the current PGN by leveraging contextual relationship information, object co-occurrences, and language priors from captions. The updated features are then used to fine-tune the PMFs. Our experiments show that DG-PGNN substantially outperforms the state-of-the-art models for scene graph generation on Visual Genome dataset.

We also evaluate the usefulness of DG-PGNN by applying it to the visual question answering (VQA) task. In spite of recent advances in VQA, current VQA models often fail on sufficiently new samples, converge on an answer after listening to only a few words of the question, and do not alter their answers across different images (Agrawal, Batra, and Parikh 2016). Our point of departure is that success at VQA task requires a rich representation of the objects and their visual relationships in an image, which identifies the attributes of these objects, and supports reasoning about the role of each object in the scene context (Figure 1). To apply DG-PGNN to the VQA task, we first construct a scene graph for the image. Then, the predicted scene graph is used to produce a representation of the image and question information by attending to the nodes and edges which are relevant to the question. This enables our model to answer questions which need reasoning about objects and their interactions. In summary, our contributions are the following:

- A new graph neural network model for representing the uncertainty associated with a scene graph.
- A new scene graph construction algorithm that combines deep feature learning with probabilistic message passing in a completely differentiable probabilistic framework. The DG-PGNN sequentially adds a new node to the graph in a reinforcement learning (RL) framework. The motivation for using RL is that a graph can be generated through sequentially choosing components, and Markov decision process is an effective framework for sequential decision making. RL provides a scalable and differentiable approach for structure prediction tasks. Although we focus on the scene graph generation task, the DG-PGNN is a generic neural method for feature learning with latent graph structures, when the structure is unknown in advance, e.g. knowledge graph construction from texts.
- To the best of our knowledge, this is the first work that explicitly exploits textual information of an image to build a scene graph. Textual information is a rich source of information about object attributes and relationships.

2 Related Work

Our work is related to several areas in computer vision, natural language processing and deep learning.

Scene Graph Generation. Johnson et al.(2015) proposed a model based on a conditional random field that reasons about various groundings of potential scene graphs for an image. Liang, Lee, and Xing(2017) proposed a model for visual relationship and attributes detection based on reinforcement learning. In their method, to extract a state feature, they used the embedding of the last two relationships and attributes that were searched during the graph construction. This will lead to limited representational power since the resulting representation depends on the order that the algorithm selects node and edges. Hence, this method may generate different representations for the same graph. Li et al.(2017) described a Multi-level Scene Description Network, to address the object detection, region captioning, and scene graph generation tasks jointly. They aligned object, phrase, and caption regions by applying a dynamic graph using spatial and semantic links. Then they applied a feature refining schema to send messages across features of the phrase, object, and caption nodes via the graph.

Lu et al.(2016) introduced a model to detect a relatively large set of relationships using language priors from semantic word embeddings. Xu et al.(2017) use an RNN for scene graph generation, which learns to improve its predictions iteratively through message passing across a scene graph. Zellers et al.(2018) proposed to learn a contextualized representation for nodes and edges with a bidirectional LSTM, based on a fixed linear ordering for the bounding-boxes. In Newell and Deng(2017), the authors proposed to train a CNN which takes in an image and produces a scene graph in an end-to-end framework. Tang et al.(2019) utilize dynamic tree structures that put the objects in an image into a visual context to improve scene graph generation accuracy.

VQA. Most VQA models compute an attention weight of spatially localized CNN features based on the question. The weighted local features are then used to predict an answer (Xiong, Merity, and Socher 2016). Nam, Ha, and Kim(2016) proposed a joint attention-based model, Dual Attention Networks, for both image and question information. Fukui et al.(2016) use a multimodal compact bilinear (MCB) pooling to get a joint representation for the input image and question. In Teney, Liu, and van den Hengel(2016), authors introduced a VQA model based on structured representations of both scene contents and questions for abstract scenes with a limited number of object classes, a small set of discrete features for each object, and only spatial relationships among them. While highly effective, a limitation is that the scene graph must be given at the test time.

Graph Neural Networks. Battaglia et al.(2018) reviewed recent work on graph neural networks extensively. Graph neural network models have been proposed for feature learning from graph-structured inputs. They have been used for various range of tasks that need rich relational structure such as visual scene understanding (Santoro et al. 2017) and learning to represent programs (Allamanis, Brockschmidt, and Khademi 2017). Li et al.(2015) proposed a model called Gated Graph Neural Network. Given the graph structure and the edge types, their model can be applied to each node to get a node representation which takes into account contextual information from the neighbor nodes. However, the graph structure and edge-types are not often given in tasks such as scene graph generation. This prohibits the use of this model for many real-world tasks.

In this work, we propose a probabilistic framework to build a graph neural network. Recent deep generative models of graphs include Li et al.; Johnson(2018; 2017). Johnson(2017) proposed an extension to Gated Graph Neural Networks which can learn to construct and modify a graph using textual input. They applied the proposed extension to solve some bAbI tasks successfully. A key problem, however, is that their model has to select an ordering to add the nodes to the graph, which may not be optimal.

3 Proposed Algorithm

In this section, we first introduce the Probabilistic Graph Network (PGN). Then, we explain how to construct a complete PGN with N nodes using all bounding-box proposals of the input image, where N is the number of bounding-boxes. This complete PGN provides initial node feature vectors, edge feature vectors, and PMFs for the DG-PGNN algorithm. Next, we describe how to encode the captions. After that, we explain the DG-PGNN algorithm to generate a scene graph for an image. Starting from a null PGN, the algorithm sequentially adds a new node to the current PGN. At each step, the algorithm has two major parts:

- A Q-learning framework which is responsible for selecting the next node.
- PGN updates which update the current PGN after adding the new node and edges.

Whenever we add a new node to the current PGN, we connect the new node to each node of the current PGN using two directed edges (see Figure 2).

The PMFs of the edge between the new node and each node in the current PGN, the feature vector of the new node, and the feature vector of the new edges are initialized by copying the corresponding feature vectors and PMFs from the complete PGN. Then, the node features, edge features, and PMFs of the current graph are updated by exploiting visual contextual information as well as the region-grounded captions of the image, hence reducing uncertainty about objects and predicate classes. We use a GRU to implement an update. After completion of the algorithm, we eliminate the edges which have a probability less than 0.5 and the nodes with background node-type from the last PGN. The remaining graph is the predicted scene graph.

Probabilistic Graph Networks. A Probabilistic Graph Network (PGN) is defined based on a given (scene) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes (bounding-boxes), and $\mathcal{E}$ is a set of probabilistic adjacency matrices. Each $\mid \mathcal{V} \mid \times \mid \mathcal{V} \mid$ matrix $E_k \in \mathcal{E}$ represents a probabilistic adjacency matrix for edge-type (predicate class) k. That is, for each $k \in \{1, \dots, K\}$, $E_k(u, v)$ is the probability that there exists an edge of type k going from node u to node v, where K is the number of edge-types. We write $\mathbf{e}_{u,v} = [E_1(u, v), \dots, E_K(u, v)]^\top \in \mathbb{R}^K$ for the PMF from u to v. The sum of the entries of a PMF must equal 1. We relax this condition for edges, to allow zero or multiple edges with various edge-types from u to v. Each edge $e = (u, v)$ has an edge representation (feature) $\mathbf{h}_{u,v} \in \mathbb{R}^D$. All edges from u to v with different edge-types share the same edge representation. Also, each node v has a node representation $\mathbf{h}_v \in \mathbb{R}^D$ and a node-type PMF $\mathbf{n}_v \in \mathbb{R}^M$, where M is the number of node-types (including "background"), and ith entry of $\mathbf{n}_v$ is the probability that node v has type i.

Constructing a Complete PGN. Given an image dataset with ground-truth annotations, we first train an object detector. For this purpose, we use the Tensorflow Object Detection API. We use *faster_rcnn_nas* trained on MS-COCO dataset as the pretrained model. Given an image, the output of the object detector is a set of object bounding-boxes with their objectness scores, classification scores, bounding-box coordinates, and feature vectors. We denote the feature vector of bounding-box v by $\mathbf{h}_v^\circ$. Each bounding-box v is specified by its coordinates $\mathrm{B}(v) = (v_x, v_y, v_{x'}, v_{y'})$ and a confidence score $p(v)$, where (v_x, v_y) and $(v_{x'}, v_{y'})$ are the top-left and bottom-right corners of the bounding-box. We use $N = 256$ bounding-boxes per image with the highest objectness scores. We also extract a feature vector for each edge $e = (u, v)$ denoted by $\mathbf{h}_{u,v}^\circ$ from the union of bounding-boxes u, and v. For this purpose, we first feed the image to 152-layer ResNet, pretrained on ImageNet, and obtain 1024 feature maps from the last 14×14 pooling layer. Then, we apply a Region of Interest pooling layer (Girshick 2015), based on the coordinates of each bounding-box, to get 1024 features maps of size 7×7. Next, we use two fully-connected layers with ReLU activation to get a 512-d feature vector denoted by $\mathbf{h}_{u,v}^\circ$.

We may obtain a node-type PMF $\mathbf{n}_v^\circ$ for each candidate bounding-box v based on the classification scores of v. However, to incorporate global context, we sort the bounding-boxes based on their confidence score, and apply a bidirectional LSTM in an Encoder-Decoder architecture (similar to (Zellers et al. 2018)) to obtain a node-type PMF $\mathbf{n}_v^\circ$ for each node, and an edge-type PMF $\mathbf{e}_{u,v}^\circ$ for each pair of the nodes. The Encoder-Decoder architecture is explained in the supplementary materials (https://grlearning.github.io/papers/135.pdf). Note that the order of the bounding-boxes are not crucial here, since the $\mathbf{e}_{u,v}^\circ$, $\mathbf{n}_v^\circ$ are just initial values and they will be updated later during the execution of the DG-PGNN algorithm.

Caption Encoding. We use a region-grounded captioning model from https://github.com/jcjohnson/densecap to extract a set of descriptions for the input image, e.g. *a cloudy sky*, *woman holding umbrella*. Each caption c has a bounding-box $\mathrm{B}(c)$ and a confidence score. We map the one hot representation of each word to a semantic space of dimensionality D via a $D \times n$ word embedding matrix, where n is the size of a dictionary which is created using all training captions (and questions for VQA task). To initialize the word embeddings, we apply skip-gram training to all questions and captions, and concatenate the skip-gram vectors with the pretrained GloVe vectors. The last hidden state of a GRU is used to encode a caption. We reset the GRU after presenting each caption.

Deep Q-learning for Node Selection

Starting from a null graph, we sequentially add a new node to the current graph in a Deep Q-learning framework, where a state is a PGN, each action chooses a new node, and rewards are defined based on the ground-truth scene graph. We use two fully-connected layers with a ReLU activation function to implement the Q-network. Let the current state be a partial PGN denoted by s. The input to the Q-network is defined as

$$\phi = \phi(s) = [\mathbf{g}, \mathbf{p}, \mathbf{d}, \mathbf{h}_1, \ldots, \mathbf{h}_N, \mathbf{n}_1, \ldots, \mathbf{n}_N, \mathbf{o}^{\mathrm{n}}, \mathbf{o}^{\mathrm{e}}] \quad (1)$$

where $\mathbf{g}$, is a global feature vector extracted from the last layer of 152-layer ResNet, $\mathbf{o}^{\mathrm{n}}$ and $\mathbf{o}^{\mathrm{e}}$ are graph-level representations of the current PGN initialized to $\mathbf{0}$, $\mathbf{p} \in \mathbb{R}^N$ is a confidence score vector defined as $\mathbf{p} = [p(1), \ldots, p(N)]$, and vector $\mathbf{d} \in \mathbb{R}^N$ is defined as: $\mathbf{d}(v) = 1$ if v is selected before, otherwise $\mathbf{d}(v) = 0$. The feature vector ϕ provides a history about the nodes, node-types, and edge-types which were selected before. Thus, the Q-network can exploit information about co-occurrence of the objects for selecting the next node. For example, if the current graph has a node with type *street*, it is likely that the image contains object *car*; so the Q-network is more likely to select a bounding-box with a high node-type probability for *car*.

At each step, we select an action from the set $\mathcal{A} = \{v : \mathbf{d}(v) = 0\} \cup \{\mathrm{STOP}\}$, where STOP is a special action that indicates the end of graph construction. The reward function for taking action a at state s is defined as: $r(a, s) = 1$ if there exists a ground-truth box v such that $\mathrm{IoU}(\mathrm{B}(a), \mathrm{B}(v)) \geq 0.5$, otherwise $r(a, s) = -1$, where IoU stands for Intersection over Union.

After each 3000 steps, the Q-network trainable parameters $\boldsymbol{\theta}$ are copied to $\hat{\boldsymbol{\theta}}$ which are used to compute the target Q-values. This helps stabilize the optimization. To reduce correlation between samples and keep experiences from the past episodes, we use an experience replay technique (Mnih et al. 2013; 2015). To update the parameters of the Q-networks, we choose a random minibatch from the replay memory. Let $(\hat{\phi}, \hat{s}, \hat{a}, \hat{r}, \hat{\phi}', \hat{s}')$ be a random transition sample. The target Q-value for the network is obtained as

$$\hat{y} = \hat{r} + \gamma \max_{v \in \hat{\mathcal{A}}'} Q\big(\phi(\hat{s}' + v); \hat{\boldsymbol{\theta}}\big) \quad (2)$$

where $\hat{\mathcal{A}}'$, is a set of actions that can be taken in state $\hat{s}'$, and $\hat{s}' + v$ is obtained by adding node v to $\hat{s}'$. Finally, the parameters of the model are updated as follows:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha\big(\hat{y} - Q\big(\phi(\hat{s} + \hat{a}); \boldsymbol{\theta}\big)\big)\nabla_{\boldsymbol{\theta}} Q\big(\phi(\hat{s} + \hat{a}); \boldsymbol{\theta}\big) \quad (3)$$

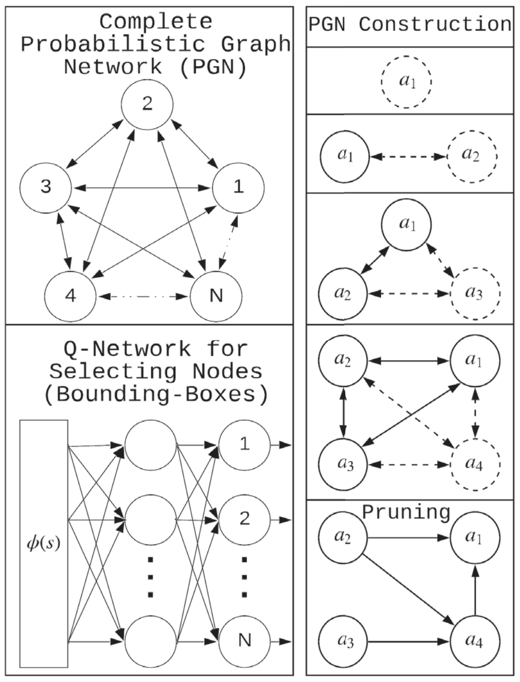

Complete Probabilistic Graph Network (PGN)
2
3
1
4
N
Q-Network for Selecting Nodes (Bounding-Boxes)
$\phi(s)$
1
2
N
PGN Construction
a_1
a_2
a_3
a_4
Pruning

Figure 2: Scene graph generation using DG-PGNN.

To train the model, we use *ϵ-greedy learning*, that is, with probability ϵ a random action is selected, and with probability $1 - \epsilon$ an optimal action is selected, as indicated by the Q-networks. For test images, we construct the graph by sequentially selecting the optimal actions (bounding-boxes) based on the highest Q-values until it selects the STOP.

Probabilistic Graph Network Updates

Node Representation Updates. We compute a new node representation $\mathbf{h}'_v$ for each node v, every time that we extend the graph. This allows to exploit information about co-occurrence of objects. For example, if the current graph has a node with type *building*, it is likely that the graph contains *window*. For each node v in the current PGN, an ideal node representation must take into account the information from every node which has an edge to v based on the strength of the edge. The update equations are as follows

$$\mathbf{a}_v = \textstyle\sum_{k=1}^{K} \sum_{u \in \mathcal{V}} E_k(u, v)\mathbf{h}_{u,v} \quad (4)$$

$$\mathbf{h}'_v = \mathrm{GRU}(\mathbf{a}_v, \mathbf{h}_v) \quad (5)$$

where, $\mathcal{V}$ is the set of the nodes in the current PGN. Intuitively, $\mathbf{a}_v$ is the aggregation of *massages* from all nodes to v weighted by the strength of their outgoing edge to v. The update mechanism first computes the node activations $\mathbf{a}_v$ for each node v. Then, a (one-step) GRU is used to update node representation for each node by incorporating information from the previous node representation $\mathbf{h}_v$. Note that unlike (Li et al. 2017; Xu et al. 2017), our message propagation mechanism is based on strength of the edges, not the predefined structure of the graph, since E_k is probabilistic. Also, our message passing scheme is part of a larger RL-based architecture. Thus, unlike these work which use all candidate boxes, in our DG-PGNN the massages propagate through the partial graph that has been constructed by DG-PGNN, not the complete graph.

Edge Representation Updates. After updating the node representations, we compute a new edge representation $\mathbf{h}'_{u,v}$ for each edge $e = (u, v)$. This allows to exploit contextual relationship information. For example, if the type of node u is *man*, and the type of node v is *horse*, it is likely that the edge-type of $e = (u, v)$ be *riding*. We use a (one-step) GRU to update the edge representations as

$$\mathbf{h}'_{u,v} = \mathrm{GRU}([\mathbf{h}_u, \mathbf{h}_v], \mathbf{h}_{u,v}) \tag{6}$$

Caption-Guided Updates. If a word of a caption refers to a node-type, then it is useful to update the node representation of a node which is located at the region of the caption based on the encoded caption. For example, for *a yellow umbrella*, it is useful to update the node representation of a node that is located at the region of the caption, such that the new representation increases the degree of the belief that the node-type of the node is *umbrella*; or *man is riding a horse* increases the chance that an edge that is located at the region of the caption has type *riding*. Given a new node v, if there is a caption c such that $\mathrm{IoU}(\mathrm{B}(c), \mathrm{B}(v)) \geq 0.5$, we update the node representation of v as $\mathbf{h}'_v = \mathrm{GRU}([\mathbf{c}, \mathbf{n}_v], \mathbf{h}_v)$, where, $\mathbf{c}$ is the encoded caption. Similarly, if there is a caption c and a node u such that $\mathrm{IoU}(\mathrm{U}(u, v), \mathrm{B}(c)) \geq 0.5$, where U stands for union of two boxes, we update the edge representation of (u, v) and (v, u) as

$$\mathbf{h}'_{u,v} = \mathrm{GRU}([\mathbf{c}, \mathbf{e}_{u,v}], \mathbf{h}_{u,v}) \tag{7}$$

$$\mathbf{h}'_{v,u} = \mathrm{GRU}([\mathbf{c}, \mathbf{e}_{v,u}], \mathbf{h}_{v,u}) \tag{8}$$

Graph-Level Representation. We obtain a graph-level representation $\mathbf{o}^{\mathrm{n}}$ using the node representations as

$$\mathbf{o}^{\mathrm{n}} = \psi\big(\textstyle\sum_v \sigma\big(f^{\mathrm{n}}(\mathbf{n}_v, \mathbf{h}_v)\big) \odot \psi\big(g^{\mathrm{n}}(\mathbf{n}_v, \mathbf{h}_v)\big)\big) \tag{9}$$

where, f^{n} and g^{n} are two neural networks, ψ is the hyperbolic tangent function, and $\sigma\big(f(\mathbf{n}_v, \mathbf{h}_v)\big)$ is an attention weight vector that decides which nodes are related to the current graph-level representation. Similarly, we obtain a graph-level representation $\mathbf{o}^{\mathrm{e}}$ using edge representations as

$$\mathbf{o}^{\mathrm{e}} = \psi\big(\textstyle\sum_{u,v} \sigma\big(f^{\mathrm{e}}(\mathbf{e}_{u,v}, \mathbf{h}_{u,v})\big) \odot \psi\big(g^{\mathrm{e}}(\mathbf{e}_{u,v}, \mathbf{h}_{u,v})\big)\big) \tag{10}$$

These two representations provide a summary of the current graph for the Q-network.

Node-Type Probability Updates. To compute a new node-type PMF for node v, we apply a GRU and a softmax layer

$$\mathbf{n}'_v = \mathrm{softmax}\big(\mathrm{GRU}([\mathbf{h}_v, \mathbf{o}^{\mathrm{n}}, \mathbf{o}^{\mathrm{e}}], \mathbf{n}_v)\big) \tag{11}$$

where, the softmax function guarantees that sum of the probability of all node-types for node v, is 1.

Edge-Type Matrix Updates. We update the edge-type probability vectors as follows

$$\mathbf{r}_{u,v} = \sigma\big(\mathbf{W}^{(1)}\psi\big(\mathbf{W}^{(2)}\mathbf{x} + \mathbf{b}^{(1)}\big) + \mathbf{b}^{(2)}\big) \tag{12}$$

$$\mathbf{z}_{u,v} = \sigma\big(\mathbf{W}^{(3)}\psi\big(\mathbf{W}^{(4)}\mathbf{x} + \mathbf{b}^{(3)}\big) + \mathbf{b}^{(4)}\big) \tag{13}$$

$$\mathbf{e}'_{u,v} = \mathbf{e}_{u,v} \odot (1 - \mathbf{r}_{u,v}) + (1 - \mathbf{e}_{u,v}) \odot \mathbf{z}_{u,v} \tag{14}$$

where $\mathbf{x} = [\mathbf{h}_{u,v}, \mathbf{n}_v, \mathbf{h}_v, \mathbf{n}_u, \mathbf{h}_u]$, and the weight matrices and biases are trainable parameters. This update grantees that the updated probabilities are between 0 and 1.

Parameter Updates. We update the parameters of the proposed model after updating the node-type PMFs and the edge-type matrices. For these updates, we minimize the following loss functions

$$\mathcal{L}^{\mathrm{n}} = -\textstyle\sum_v \mathbf{n}^\star_v \cdot \log(\mathbf{n}'_v) + (1 - \mathbf{n}^\star_v) \cdot \log(1 - \mathbf{n}'_v) \tag{15}$$

$$\mathcal{L}^{\mathrm{e}} = -\textstyle\sum_{u,v} \mathbf{e}^\star_{u,v} \cdot \log(\mathbf{e}'_{u,v}) + (1 - \mathbf{e}^\star_{u,v}) \cdot \log(1 - \mathbf{e}'_{u,v}) \tag{16}$$

where, $\mathbf{n}^\star_v$ and $\mathbf{e}^\star_{u,v}$ denote the ground-truth for PMF of node v, and edge-type probabilities of edge $e = (u, v)$. That is, $\mathbf{n}^\star_v(m) = 1$ if there is a ground-truth bounding-box u of node-type m such that $\mathrm{IoU}(u, v) \geq 0.5$, otherwise $\mathbf{n}^\star_v(m) = 0$. Similarly, $\mathbf{e}^\star_{u,v}(k) = 1$ if there is a ground-truth edge $e = (u', v')$ of type k such that $\mathrm{IoU}(\mathrm{U}(u, v), \mathrm{U}(u', v')) \geq 0.5$, otherwise $\mathbf{e}^\star_{u,v}(k) = 0$.

DG-PGNN for VQA. We use the last hidden state of a GRU to obtain an encoded question denoted by $\mathbf{q}$. Then, we obtain question-guided graph-level representations as

$$\mathbf{o}^{\mathrm{n}}_{\mathrm{q}} = \psi\big(\textstyle\sum_v \sigma(\mathbf{h}_v \star \mathbf{q}) \odot \psi(\mathbf{h}_v \star \mathbf{q})\big) \tag{17}$$

$$\mathbf{o}^{\mathrm{e}}_{\mathrm{q}} = \psi\big(\textstyle\sum_{u,v} \sigma(\mathbf{h}_{u,v} \star \mathbf{q}) \odot \psi(\mathbf{h}_{u,v} \star \mathbf{q})\big) \tag{18}$$

where, we used MCB to implement $\star$. This allows to incorporate information from the question for computing the attention weights for each node and edge using the sigmoid function. The candidate answers are also encoded by the last hidden state of a GRU and concatenated with $\mathbf{o}^{\mathrm{n}}_{\mathrm{q}}$ and $\mathbf{o}^{\mathrm{e}}_{\mathrm{q}}$ using a neural network layer as $\hat{p} = \sigma\big(\mathbf{w}f([\mathbf{o}^{\mathrm{n}}_{\mathrm{q}}, \mathbf{o}^{\mathrm{e}}_{\mathrm{q}}, \mathbf{a}]) + b\big)$ where, $\mathbf{a}$ is the encoded answer choice, f is a ReLU non-linear layer, and $\mathbf{w}, b$ are trainable parameters. The binary logistic loss $-p\log(\hat{p}) - (1-p)\log(1-\hat{p})$ is used, where p is 1.0 for an image-question-answer triplet, if the answer choice is correct, otherwise p is 0.

Training Details and Optimization. The discount factor γ is set to 0.85 and ϵ is annealed from 1 to 0.05 during the first 50 epochs, and is fixed after epoch 50. For VQA, the loss is minimized using RMSprop optimization algorithm with learning rate 0.0001 and minibatches of size 100. To prevent overfitting, dropout with probability 0.5 and early stopping are applied. D is set to 512. For each image, we use up to 10 captions with a confidence score greater than 1.0. During training, all parameters are tuned except for the weights of the CNN and caption generation component to avoid overfitting. We use default values for all hyper-parameters of the object detector API. Our model takes two days to train on two NVIDIA Titan X GPUs.

4 Experiments

For each task, we introduce the datasets, baseline models and evaluation metric that we use in our experiments. Then, the results are presented and discussed.

Scene Graph Generation Task

Dataset. The Visual Genome dataset (Krishna et al. 2016) contains 108,077 images. We used Visual Genome version 1.4 release. This release contains cleaner object annotations (Xu et al. 2017). Annotations provide subject-predicate-object triplets. A triplet means that there is an edge between the subject and the object and the type of the edge is indicated by the predicate. Following (Xu et al. 2017), we use the most frequent 150 object categories and 50 predicates for scene graph prediction task. This results in a scene graph of about 11.5 objects and 6.2 relationships per image. The training and test splits contains 70% and 30% of the images, respectively.

Metrics. Top-K recall (Rec@K) is used as the metric, which is the fraction of the ground truth relationship triplets subject-predicate-object hit in the top-K predictions in an image. Predictions are ranked by the product of the node-type probability of the subject, the node-type probability of the object, and the edge-type probability of the predicate. Following (Xu et al. 2017), we evaluate our model based on three tasks as follows:

- Predicate classification (PRED-CLS) task: to predict the predicates of all pairwise relationships of a set of objects, where the location and object categories are given.
- Scene graph classification (SG-CLS) task: to predict the predicate and the object categories of the subject and object in all pairwise relationships, where the location of the objects are given.
- Scene graph generation (SG-GEN) task: to detect a set of object bounding-boxes and predict the predicate between each pair of the objects, at the same time. An object is considered to be properly detected, if it has at least 0.5 Intersection over Union overlap with a bounding-box annotation with the same category.

Baseline Models. We compare our model with several baseline models including the state-of-the-art models (Tang et al. 2019; Zellers et al. 2018; Newell and Deng 2017). Lu et al.(2016) uses language priors from semantic word embeddings, while Xu et al.(2017) uses an RNN and learns to improves its predictions iteratively through message passing across the scene graph. After a few steps the learned features are classified. Moreover, three ablation models help evaluate the impact of each component of DG-PGNN:

- DG-PGNN$^{\circ}$ is the initial complete PGN without applying the DG-PGNN ($e = (u, v)$ is pruned if $\mathbf{e}^{\circ}_{u,v}$ is less than 0.5, and v is deleted if $\mathbf{n}^{\circ}_{v}$ is less than 0.5)
- DG-PGNN^{-} is the same as DG-PGNN except that it does not use captions.
- DG-PGNN† is the same as the full model (DG-PGNN), but it uses the VGG features instead of ResNet.

Results and Discussion. Our experimental results are reported in Table 1. The results show that DG-PGNN outperforms the state-of-the art models for scene graph generation task. Both DG-PGNN and Xu et al.(2017) leverage the power of RNNs to learn a feature vector for each bounding-box. DG-PGNN incorporates captions and contextual information of the image via information propagation to construct precise scene graphs. However, Xu et al.(2017) suffers from imbalanced classification problem (often there is no edge between a pair of objects). DG-PGNN† results (comparing to DG-PGNN) show the effect of backbone CNN (VGG versus ResNet) is insignificant. This is because our algorithm uses strong contextual information to compensate for the detection errors caused by applying the VGG.

	PRED-CLS		SG-CLS		SG-GEN	
Model	R@50	R@100	R@50	R@100	R@50	R@100
SG-LP[1]	27.88	35.04	11.79	14.11	00.32	00.47
SG-IMP[2]	44.75	53.08	21.72	24.38	03.44	04.24
MSDN[3]	63.12	66.41	19.30	21.82	7.73	10.51
GRCNN[4]	54.2	59.1	29.6	31.6	11.4	13.7
PIX2GR[5]	68.0	**75.2**	26.5	30.0	9.7	11.3
MOTIF[6]	65.2	67.1	35.8	36.5	27.2	30.3
VCTREE[7]	66.4	68.1	38.1	38.8	27.9	31.3
DG-PGNN$^{\circ}$	63.5	65.3	34.2	34.8	26.1	28.8
DG-PGNN^{-}	67.3	70.1	38.6	39.2	30.2	31.8
DG-PGNN†	69.0	72.1	39.3	40.1	31.2	32.5
DG-PGNN	**70.1**	73.0	**39.5**	**40.8**	**32.1**	**33.1**

Table 1: Recall for predicate classification, scene graph classification, and scene graph generation tasks on Visual Genome dataset. The References are: Lu et al.(2016)[1], Xu et al.(2017)[2], Li et al.(2017)[3], Yang et al.(2018)[4], Newell and Deng(2017)[5], Zellers et al.(2018)[6], and Tang et al.(2019)[7].

VQA Task

Datasets. Visual7W (Zhu et al. 2015) includes $47,300$ images which occurs in both Visual Genome and MS-COCO datasets. The training, validation and test splits, contains 50%, 20%, 30% of the QA pairs, respectively. We train and evaluate our model on *telling* questions of the Visual7W. This set uses six types of questions: *what*, *where*, *when*, *who*, *why*, *how*. For evaluation, Visual7W provides four candidate answers.

We also evaluated our model on the test set of CLEVR (https://cs.stanford.edu/people/jcjohns/clevr/) dataset (Johnson et al. 2017a). CLEVR evaluates different aspects of visual reasoning such as attribute recognition, counting, comparison, logic, and spatial relationships. Each object in an image has the following attributes: shape (*cube*, *sphere*, or *cylinder*), size (*large* or *small*), color (8 colors), and material (*rubber* or *metal*). An object detector with 96 classes is trained using all combinations of the attributes. The predicates are *left*, *right*, *in front*, *behind*, *same size*, *same color*, *same shape*, and *same material*. We created a scene graph for each training image using the provided annotations. We concatenate the normalized coordinates of each bounding-box with the features vector that we extract from Region of Interest pooling layer. This helps to detect spatial relationships between objects.

Baseline Models. For Visual7W, we compare our models with Zhu et al.(2015), MCB (Fukui et al. 2016), MAN (Ma et al. 2018), and MLP (Jabri, Joulin, and Van Der Maaten 2016). MCB leverages the Visual Genome QA pairs as additional training data and the 152-layer ResNet as a pretrained model. This model took the first place in the VQA Challenge workshop 2016. MAN utilized a memory-augmented neural network which attend to each training exemplar to answer visual questions, even when the answers happen infrequently in the training set. The MLP method uses image-question-answer triplets to score answer choices. For

CLEVR, we compare our DG-PGNN$^-$ with several baselines proposed by Johnson et al.(2017a) as well as state-of the art models PROGRAM-GEN (Johnson et al. 2017b) and CNN+LSTM+RN (Santoro et al. 2017). N2NMN (Hu et al. 2017) learns to predict a layout based on the question and compose a network using a set of neural modules. CNN+LSTM+RN learns to infer a relation using a neural network model called Relation Networks. PROGRAM-GEN exploits supervision from functional programming which is used to generate CLEVR questions.

Model	What	Where	When	Who	Why	How	Avg
HUMAN[1]	96.5	95.7	94.4	96.5	92.7	94.2	95.7
LSTM-ATT[1]	51.5	57.0	75.0	59.5	55.5	49.8	54.3
CONCAT+ATT[2]	47.8	56.9	74.1	62.3	52.7	51.2	52.8
MCB+ATT[2]	60.3	70.4	79.5	69.2	58.2	51.1	62.2
MAN[3]	62.2	68.9	76.8	66.4	57.8	52.9	62.8
MLP[4]	64.5	75.9	82.1	72.9	68.0	56.4	67.1
DG-PGNN$^-$	65.4	77.1	83.0	74.1	68.8	57.3	68.0
DG-PGNN	**66.8**	**78.3**	**84.4**	**75.6**	**70.0**	**58.2**	**69.3**

Table 2: Accuracy Percentage on Visual7W. The references are: Zhu et al.(2015)[1], Fukui et al.(2016)[2], Ma et al.(2018)[3], Jabri, Joulin, and Van Der Maaten(2016)[4].

Model	All	Exist	Count	Cmp-Int	Q-At	Cmp-At
HUMAN[1]	92.6	96.6	86.7	86.5	95.0	96.0
Q-TYPE MODE[1]	41.8	50.2	34.6	51.0	36.0	51.3
LSTM[1]	46.8	61.1	41.7	69.8	36.8	51.3
CNN+BOW[1]	48.4	59.5	38.9	51.1	48.3	51.8
CNN+LSTM[1]	52.3	65.2	43.7	67.1	49.3	53.0
CNN+LSTM+MCB[1]	51.4	63.4	42.1	66.4	49.0	51.0
CNN+LSTM+SA[1]	68.5	71.1	52.2	73.5	85.3	52.3
N2NMN[2]	83.3	85.7	68.5	85.0	90.0	88.8
CNN+LSTM+RN [3]	95.5	97.8	90.1	93.6	97.9	97.1
PROGRAM-GEN[4]	**96.9**	97.1	**92.7**	**98.7**	**98.2**	**98.9**
DG-PGNN$^-$	96.1	**98.0**	91.2	94.2	98.1	97.8

Table 3: Accuracy Percentage on CLEVR. The references are: Johnson et al.(2017a)[1], Hu et al.(2017)[2], Santoro et al.(2017)[3], and Johnson et al.(2017b)[4].

Results. Tables 2 and 3 reports our experimental results on Visual7W and CLEVR datasets, respectively. For Visual7W, the results show that our algorithm outperforms MCB, MAN, and MLP for VQA task. This is because the scene graph provides a rich intermediate representation to answer questions involving objects and their relationships. For CLEVR, our algorithm achieves comparable results with the state of the art. We emphasize that, unlike PROGRAM-GEN, our algorithm does not exploit supervision from functional programming.

Qualitative Examples

Figures 3, 4, and 5 illustrate some scene graphs generated by our model. The DG-PGNN predicts a rich semantic representation of the given image by recognizing objects, their locations, and relationships between them. For example, DG-PGNN can correctly detect spatial relationship (*bowl on table*, *tree behind girl*, and *cube behind cylinder*), and interactions (*girl riding horse* and *girl wearing shirt*). The DG-PGNN can correctly recognize *girl* (instead of *woman*), and *zebra in snow* (instead of *zebra in grass*).

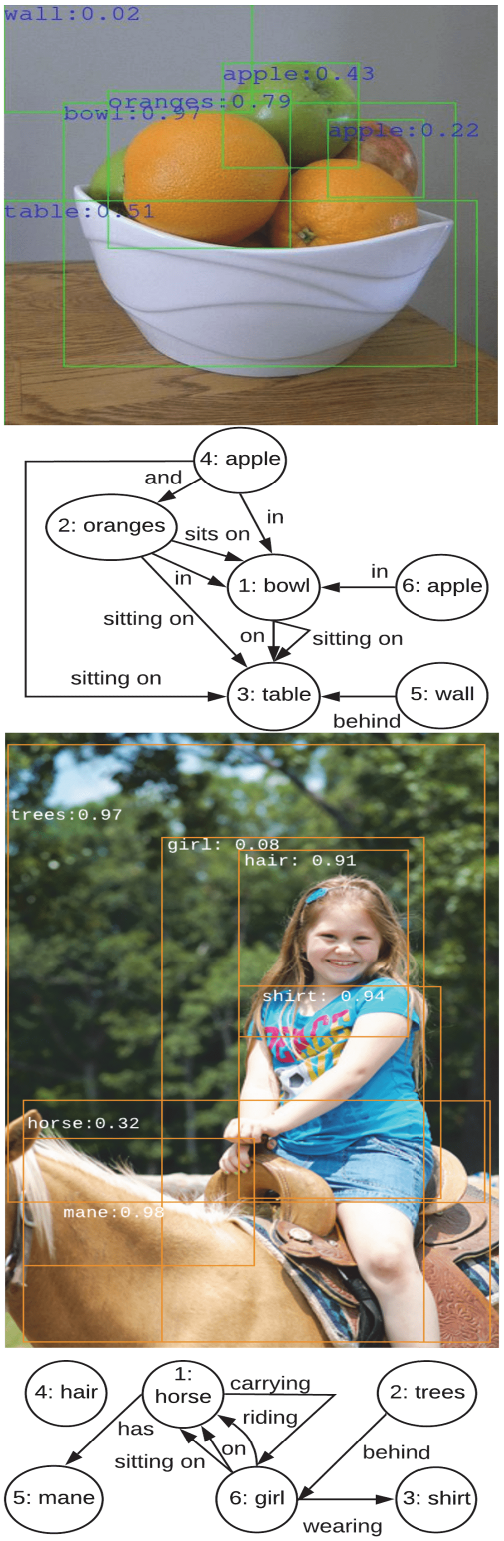
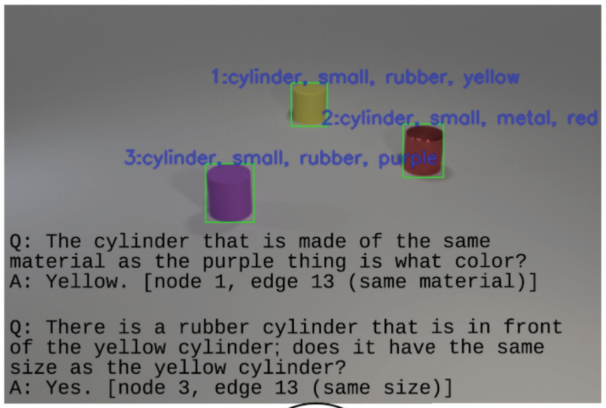


Figure 3: Examples of scene graphs generated by DG-PGNN on Visual Genome dataset.

Figure 4 shows two examples of VQA by DG-PGNN† on CLEVR dataset. Figures 5 shows examples of VQA by DG-

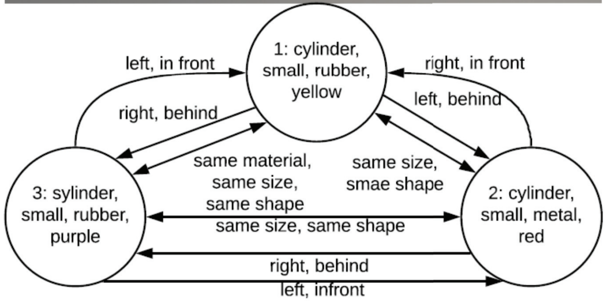
PGNN on Visual7W dataset. For each question, the nodes and edges with the highest attention weight are specified.



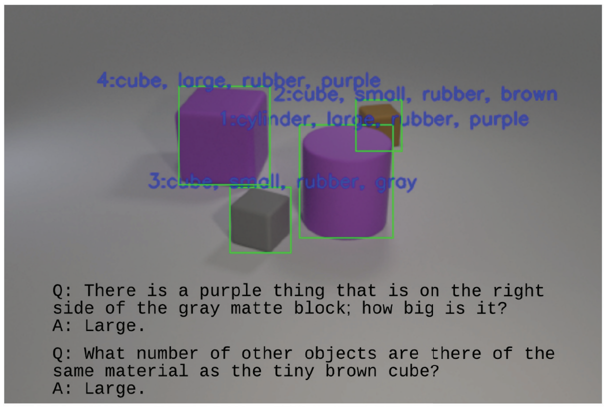
1:cylinder, small, rubber, yellow
2:cylinder, small, metal, red
3:cylinder, small, rubber, purple

Q: The cylinder that is made of the same material as the purple thing is what color?
A: Yellow. [node 1, edge 13 (same material)]

Q: There is a rubber cylinder that is in front of the yellow cylinder; does it have the same size as the yellow cylinder?
A: Yes. [node 3, edge 13 (same size)]



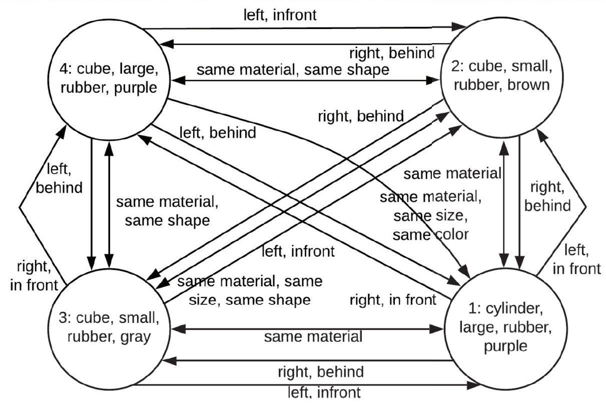
1: cylinder, small, rubber, yellow
2: cylinder, small, metal, red
3: sylinder, small, rubber, purple
left, in front
right, in front
right, behind
left, behind
same material, same size, same shape
same size, smae shape
same size, same shape
right, behind
left, infront



4:cube, large, rubber, purple
2:cube, small, rubber, brown
1:cylinder, large, rubber, purple
3:cube, small, rubber, gray

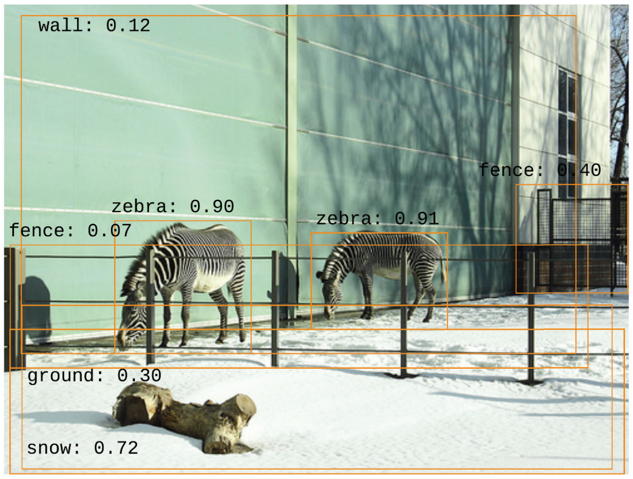
Q: There is a purple thing that is on the right side of the gray matte block; how big is it?
A: Large.

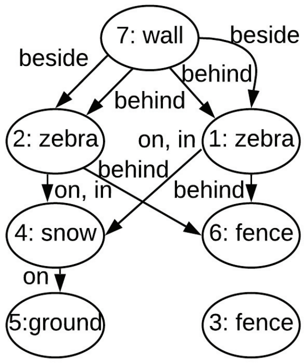
Q: What number of other objects are there of the same material as the tiny brown cube?
A: Large.



4: cube, large, rubber, purple
2: cube, small, rubber, brown
3: cube, small, rubber, gray
1: cylinder, large, rubber, purple
left, infront
right, behind
same material, same shape
right, behind
left, behind
left, behind
same material, same shape
same material
same material, same size, same color
right, behind
left, infront
right, in front
left, in front
same material, same size, same shape
right, in front
same material
right, behind
left, infront

Figure 4: Examples of VQA by DG-PGNN† on CLEVR dataset. For each question, the nodes and edges with the highest attention weight are specified.





Q: What animals are pictures?
(Lion, Tiger, Bear, Zebra)
A: Zebra. [node 1, node 2]

Q: What are the zebras standing on?
(Dirt, On a road, Each other, Snow)
A: Snow. [node 4, node 1, edge 14 (on)]

Figure 5: An example of scene graph generation and VQA by DG-PGNN on Visual7W dataset. For each question, the nodes and edges with the highest attention weight are specified. The DG-PGNN can correctly detect spatial relationships *wall beside zebra* and *zebra behind fence*. Also, the DG-PGNN can correctly recognize *zebra in snow* (instead of *zebra in grass* which occurs more frequently).

5 Conclusion

We presented a new generative graph neural network, Deep Generative Probabilistic Graph Neural Networks, which generates a scene graph for an image. Our experiments show that the proposed algorithm significantly outperforms the state-of-the-art results on Visual Genome dataset for scene graph generation task. The scene graphs constructed by DG-PGNN improve performance on VQA task on Visual7W and CLEVR datasets. A future research direction is to train an *end-to-end* model which learns to answer visual question about an image by constructing a scene graph for the image.

References

Agrawal, A.; Batra, D.; and Parikh, D. 2016. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*.

Allamanis, M.; Brockschmidt, M.; and Khademi, M. 2017. Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740*.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Fukui, A.; Park, D. H.; Yang, D.; Rohrbach, A.; Darrell, T.; and Rohrbach, M. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Girshick, R. 2015. Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

Hu, R.; Andreas, J.; Rohrbach, M.; Darrell, T.; and Saenko, K. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 804–813.

Jabri, A.; Joulin, A.; and Van Der Maaten, L. 2016. Revisiting visual question answering baselines. In *European conference on computer vision*, 727–739. Springer.

Johnson, J.; Krishna, R.; Stark, M.; Li, L.-J.; Shamma, D.; Bernstein, M.; and Fei-Fei, L. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3668–3678.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017a. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2901–2910.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Hoffman, J.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017b. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2989–2998.

Johnson, D. D. 2017. Learning graphical state transitions. In *International Conference on Learning Representations (ICLR)*.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.-J.; Shamma, D. A.; et al. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Li, Y.; Ouyang, W.; Zhou, B.; Wang, K.; and Wang, X. 2017. Scene graph generation from objects, phrases and region captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1261–1270.

Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; and Battaglia, P. 2018. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*.

Liang, X.; Lee, L.; and Xing, E. P. 2017. Deep variation-structured reinforcement learning for visual relationship and attribute detection. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 4408–4417. IEEE.

Lu, C.; Krishna, R.; Bernstein, M.; and Fei-Fei, L. 2016. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 852–869. Springer.

Ma, C.; Shen, C.; Dick, A.; Wu, Q.; Wang, P.; van den Hengel, A.; and Reid, I. 2018. Visual question answering with memory-augmented networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6975–6984.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Nam, H.; Ha, J.-W.; and Kim, J. 2016. Dual attention networks for multimodal reasoning and matching. *arXiv preprint arXiv:1611.00471*.

Newell, A., and Deng, J. 2017. Pixels to graphs by associative embedding. In *Advances in neural information processing systems*, 2171–2180.

Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, 4967–4976.

Tang, K.; Zhang, H.; Wu, B.; Luo, W.; and Liu, W. 2019. Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6619–6628.

Teney, D.; Liu, L.; and van den Hengel, A. 2016. Graph-structured representations for visual question answering. *CoRR, abs/1609.05600* 3.

Xiong, C.; Merity, S.; and Socher, R. 2016. Dynamic memory networks for visual and textual question answering. *arXiv preprint arXiv:1603.01417*.

Xu, D.; Zhu, Y.; Choy, C. B.; and Fei-Fei, L. 2017. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.

Yang, J.; Lu, J.; Lee, S.; Batra, D.; and Parikh, D. 2018. Graph r-cnn for scene graph generation. *arXiv preprint arXiv:1808.00191*.

Zellers, R.; Yatskar, M.; Thomson, S.; and Choi, Y. 2018. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5831–5840.

Zhu, Y.; Groth, O.; Bernstein, M.; and Fei-Fei, L. 2015. Visual7w: Grounded question answering in images. *arXiv preprint arXiv:1511.03416*.