Multimodal Neural Graph Memory Networks for Visual Question Answering

Mahmoud Khademi
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
mkhademi@sfu.ca

Abstract

We introduce a new neural network architecture, Multimodal Neural Graph Memory Networks (MN-GMN), for visual question answering. The MN-GMN uses graph structure with different region features as node attributes and applies a recently proposed powerful graph neural network model, Graph Network (GN), to reason about objects and their interactions in an image. The input module of the MN-GMN generates a set of visual features plus a set of encoded region-grounded captions (RGCs) for the image. The RGCs capture object attributes and their relationships. Two GNs are constructed from the input module using the visual features and encoded RGCs. Each node of the GNs iteratively computes a question-guided contextualized representation of the visual/textual information assigned to it. Then, to combine the information from both GNs, the nodes write the updated representations to an external spatial memory. The final states of the memory cells are fed into an answer module to predict an answer. Experiments show MN-GMN rivals the state-of-the-art models on Visual7W, VQA-v2.0, and CLEVR datasets.

1 Introduction

Visual question answering (VQA) has been recently introduced as a grand challenge for AI. Given an image and a free-form question about it, the VQA task is to produce an accurate natural language answer. VQA has many applications, such as image retrieval and search. This paper proposes a new neural network architecture for VQA based on the recent Graph Network (GN) (Battaglia et al., 2018).

The pairwise interactions between various regions of an image and spatial context in both horizontal and vertical directions are important to answer questions about objects and their interactions in the scene context. For example, to answer *How many cats are in the picture?* (see Figure 1), a model needs to aggregate information from multiple, possibly distant, regions; hence applying a convolutional neural network may not be sufficient to perform reasoning over the regions. Our new architecture (see Figure 2), Multimodal Neural Graph Memory Network (MN-GMN), uses a

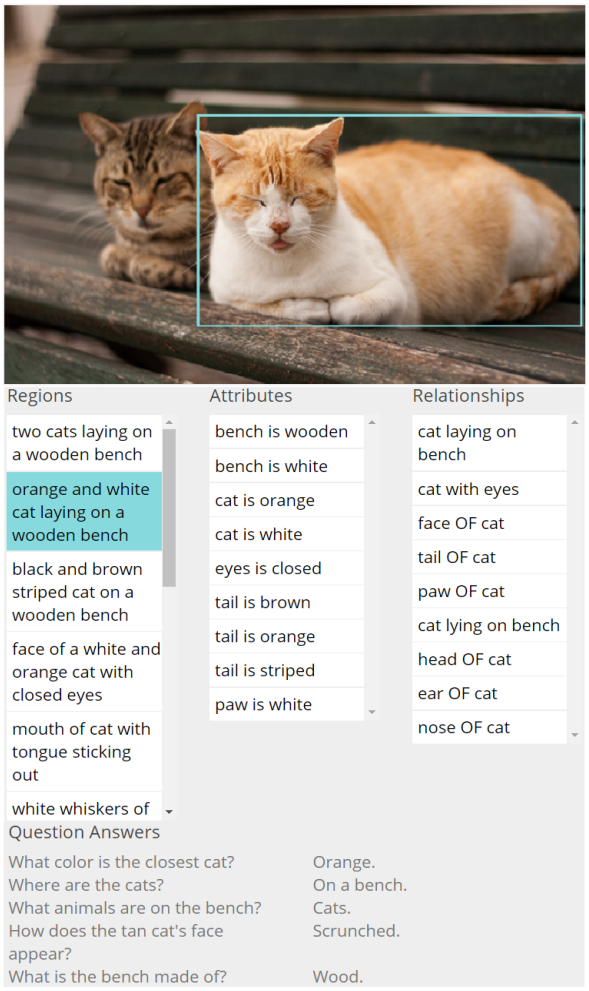

Figure 1: An example from Visual Genome (https://visualgenome.org/). The region-grounded captions provide useful clues to answer questions. For example, to answer *Where are the cats?*, *orange and white cat laying on a wooden bench* is informative.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7177–7188
July 5 - 10, 2020.

graph structure to represent pairwise interactions between visual/textual features (nodes) from different regions of an image. GNs provide a context-aware neural mechanism for computing a feature for each node that represents complex interactions with other nodes. This enables our MN-GMN to answer questions that need reasoning about complex arrangements of objects in a scene.

Previous approaches such as Memory Networks (MN) (Sukhbaatar et al., 2015) and Dynamic Memory Networks (DMN) (Kumar et al., 2015) combined a memory component and an attention mechanism to reason about a set of inputs. The DMN was first proposed for text QA. The text QA task is composed of a question, and a set of statements, called facts, in the order that describes a short story. Only a subset of the facts is required to answer a question. DMN includes four modules: input, question, episodic memory, and answer. The input and question modules encode the question and the facts. Then, the episodic memory takes as input the question and aggregates the facts to produce a vector representation of the relevant information. This vector is passed to the answer module to predict an answer. Previous applications of the MN and DMN for VQA either represent each image region independently as a single visual fact (Xu and Saenko, 2015) or represent the regions of an image like facts of a story with a linear sequential structure (Xiong et al., 2016). But, whereas a linear order may be sufficient for text QA, it is insufficient to represent the 2D context of an image.

The major novel aspect of our approach is that we exploit the flexibility of GNs to combine information from two different sources: visual features from different image regions and textual features based on region-grounded captions (RGCs). An RGC detector is learned by transfer learning from a dataset with region-grounded captions. Like visual features, an RGC is specified with a bounding-box. The RGCs capture object attributes and relationships that are often useful to answer visual questions. For example, in Figure 2, to answer *Is the water calm?*, *a wave in the ocean* is informative; *the water is blue* specifies an attribute of *water*; *surfer riding a wave* describe interactions between objects. Captions also incorporate commonsense knowledge. Our multimodal graph memory network comprises a visual GN and a textual GN, one for each information source. Each node of the two GNs iteratively computes a question-guided

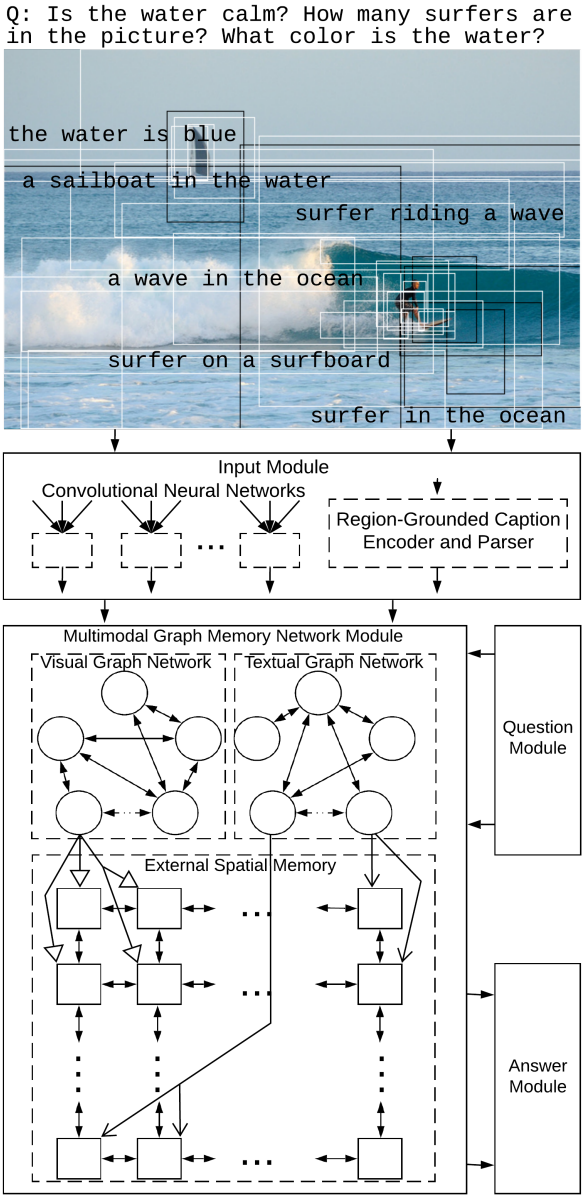

Figure 2: Multimodal Neural Graph Memory Networks for VQA. The visual features/captions are extracted from white/black bounding-boxes and are used as node features to construct the visual/textual Graph Network.

contextualized representation of the visual/textual information at the bounding-box assigned to it. The third component in our multimodal graph memory module is an external spatial memory, which is designed to combine information across the modalities. Each node writes the updated representations to the external spatial memory, which is composed of memory cells arranged in a 2D grid. The final state of the memory cells is then fed into the answer module to predict an answer. The external spatial memory resolves the redundancy introduced by overlapping bounding-boxes, which causes difficulties, for example, with counting questions.

To summarize, our main contributions are:

- We introduce a new memory network architecture, based on graph neural networks, which can reason about complex arrangements of objects in a scene to answer visual questions.
- To the best of our knowledge, this is the first work that explicitly incorporates local textual information (RGCs) of the image via a transfer learning technique into a multimodal memory network to answer visual questions.
- Our architecture, which can be seen as a multimodal relational extension to DMN, rivals the state-of-the-art on three VQA datasets.

2 Related Work

An important part of the VQA task is to understand the given question. Most approaches utilize a neural network architecture that can handle sequences of flexible length and learn complex temporal dynamics using a sequence of hidden states. Such architectures include Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and the Gated Recurrent Unit (GRU). To encode a given image, most VQA approaches employ a Convolutional Neural Network (CNN) pre-trained on ImageNet, such as VGGNet and ResNet, to extract visual information from an image. These two recent trends of applying CNNs and RNNs have been successfully applied to image captioning and visual grounding (Johnson et al., 2015) tasks. Grounding connects words to their visual meaning. Our approach sees VQA as first grounding the question in the image and then predicting an answer.

Most early deep neural-based VQA models produce an answer conditioned on a global visual feature vector and the embedded question. However, since many questions and answers relate to a specific region in an image, these models often cannot predict a precise answer. To overcome this issue, many attention-based models are proposed. The attention-based models compute an attention weight of spatially localized CNN features based on the question to predict an answer (Xu and Saenko, 2015; Xiong et al., 2016). Teney et al. (2018) used the Bottom-Up Attention model (Anderson et al., 2018) to obtain a set of features at different regions of the image and computed an attention weight for each region based on the encoded question to predict an answer. In Lu et al. (2016), the authors proposed a hierarchical co-attention model that jointly implements both image-guided question attention and question-guided visual attention. Fukui et al. (2016) proposed a VQA model based on multimodal compact bilinear (MCB) pooling to get a joint representation for image and question. Similarly, Yu et al. (2018); Kim et al. (2018) utilized higher-order fusion techniques to combine the question with visual features more efficiently. Cadene et al. (2019) proposed a bilinear fusion algorithm to represent interactions between question and image regions.

In Jabri et al. (2016), the authors introduced a model called Relation Networks, which uses multilayer perceptron models to reason over all pairs of local image features extracted from a grid of image regions. Dynamic tree structures have been used in VQA to capture the visual context of image objects (Tang et al., 2019). Yi et al. (2018) proposed a model called neural-symbolic visual question answering (NS-VQA). The NS-VQA uses symbolic structure as prior knowledge to answer questions that need complex reasoning. This model first extracts a structural scene representation from the scene and a program trace from the given question. Then, it applies the program to the scene representation to predict an answer.

Recently, a few models are proposed which can learn the interactions between image regions. The graph learner model (Norcliffe-Brown et al., 2018) merges a graph representation of the image based on the question with a graph convolutional network, to learn visual features that can represent question specific interactions. Yang et al. (2018) proposed to reason over a visual representation of the image called scene graph which represents objects and their relationships explicitly. Li et al. (2019) introduced a VQA model called Relation-aware Graph Attention Network (ReGAT). Guided by the question, ReGAT encodes an image into a graph that represents relations among visual objects. The ReGAT is trained on Visual Genome dataset (Krishna et al., 2016).

Most of the above models need datasets with annotated object relationship triplets for training. Because annotating triplets is difficult, such datasets are relatively small. Instead, our VQA architecture exploits the rich textual information of an image via incorporating the RGCs to learn the attributes of an image region and the interactions between a set of image regions enclosed by an RGC bounding-box.

This information is much easier to obtain because large caption datasets are available.

More recently, Hudson and Manning (2019a) proposed a model called Neural State Machine (NSM) for the visual questions that need compositionality and multi-step inference. Given an image, the NSM first predicts a probabilistic graph as a structured semantic representation of the image. Then, NSM executes sequential reasoning guided by the input question over the predicted graph, by iteratively traversing the nodes of the graph. The authors show that the proposed model can achieve state-of-the-art results on VQA-CP (Agrawal et al., 2018) and GQA (Hudson and Manning, 2019b) datasets. Shrestha et al. (2019) introduced a VQA model called Recurrent Aggregation of Multimodal Embeddings Network (RAMEN), which is suitable for both natural image understanding and the synthetic datasets that need compositional reasoning. The RAMEN processes visual and question features in three steps: early fusion of spatially localized image features with question features, learning bimodal embeddings, and aggregating them across the image by applying a bidirectional GRU to capture the interactions between bimodal embeddings.

3 Graph Networks

In this section, we briefly explain the graph networks (GN) framework (Battaglia et al., 2018). The GN extends several other graph neural networks such as message-passing neural networks (Gilmer et al., 2017), and non-local neural networks (Wang et al., 2018). In a GN framework, a graph is represented by a 3-tuple $\mathcal{G} = (\mathbf{u}, \mathcal{V}, \mathcal{E})$, where $\mathbf{u}$ is a graph-level attribute. The $\mathcal{V} = \{\mathbf{v}_i\}_{i=1:N}$ is a set of node attributes, where $\mathbf{v}_i$ is a node attribute of node i, and N is the number of nodes. The $\mathcal{E} = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:M}$ is a set of edges, where $\mathbf{e}_k$ is an edge attribute for the edge going from node s_k to node r_k, and M is the number of edges.

A GN block has three update functions ϕ and three aggregation functions ρ. Given an input graph, a GN block updates the graph using the update and aggregation functions. The computational steps in a GN are represented in Algorithm 1. The function ϕ^e is mapped over entire edges to calculate per-edge updates, ϕ^v is mapped over entire nodes to calculate per-node updates, and ϕ^u is used to update the global attribute. The ρ's should be unvarying to permutations of their inputs and must be flexible to a varying number of arguments, such as maximum, summation, etc.

Algorithm 1: Computational steps in a Graph Network block.

Input : A graph $G = (\mathbf{u}, \mathcal{V}, \mathcal{E})$
Output : Updated graph $G' = (\mathbf{u}', \mathcal{V}', \mathcal{E}')$
Function GraphNetwork($\mathcal{E}$, $\mathcal{V}$, $\mathbf{u}$)
for $k \leftarrow 1$ **to** M **do**
  $\mathbf{e}'_k \leftarrow \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$ ▷ *Compute new edge attributes*
end
for $i \leftarrow 1$ **to** N **do**
  $\mathcal{E}'_i \leftarrow \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i, k=1:M}$
  $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\rightarrow v}(\mathcal{E}'_i)$ ▷ *Aggregate edge attributes for each node*
  $\mathbf{v}'_i \leftarrow \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$ ▷ *Compute new node attributes*
end
$V' \leftarrow \{\mathbf{v}'_i\}_{i=1:N}$
$\mathcal{E}' \leftarrow \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:M}$
$\bar{\mathbf{e}}' \leftarrow \rho^{e\rightarrow u}(\mathcal{E}')$ ▷ *Aggregate edge attributes for the whole graph*
$\bar{\mathbf{v}}' \leftarrow \rho^{v\rightarrow u}(\mathcal{V}')$ ▷ *Aggregate node attributes for the whole graph*
$\mathbf{u}' \leftarrow \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$ ▷ *Compute new global attribute*
return $(\mathbf{u}', \mathcal{V}', \mathcal{E}')$

4 Our Proposed Architecture

Figure 2 shows our MN-GMN architecture, which is composed of four modules: input, question, multimodal graph memory network, and answer. We now describe these modules.

4.1 Input Module

The input module has two components: A deep CNN, e.g., Bottom-Up Attention (Anderson et al., 2018), ResNet (He et al., 2015), etc. and a region-grounded caption (RGC) encoder which encodes the RGCs. The RGCs are generated by a dense captioning model. Then, they are encoded with a GRU and a parser (Schuster et al., 2015). The RGCs are useful to answer questions about object attributes and their relationships. We now describe the details and motivation for these components.

Visual Feature Extraction. To extract visual features, we use the Bottom-Up Attention model. The features are obtained via Faster R-CNN and 101-layer ResNet, which attend to specific image

regions. Using a fixed threshold on object detection, we extract N 2048-dimensional image features from N different regions of the image. The value of N depends on the image and ranges from 10 to 100. Each feature vector has a bounding-box specified by its coordinates $\mathbf{r} = (r_x, r_y, r_{x'}, r_{y'})$, where (r_x, r_y) and $(r_{x'}, r_{y'})$ are the top-left and bottom-right corners of the bounding-box which are normalized to have a values between 0 and 1 based on the height and width of the image. We concatenate each feature vector with its bounding-box to obtain a vector denoted by $\mathbf{x}_i$, $(i = 1, \ldots, N)$. Note that $\mathbf{x}_i$ only describes the image at its bounding-box without exploiting the global spatial context.

Captions. To extract a set of RGCs for the image, we use a dense captioning model proposed by Johnson et al. (2015). This model contains a CNN, a dense localization layer, and an RNN language model that generates the captions (https://github.com/jcjohnson/densecap). The model is trained on RGCs from the Visual Genome dataset. The training set that we use does not include VQA-v2.0/Visual7W test images. Through transfer learning, our model is leveraging the caption annotations. Each RGC has a caption, a bounding-box, and a confidence score. To encode a caption, we first create a dictionary using all words in the captions and questions. We preprocess the captions and questions with basic tokenization by converting all sentences to lower case and throwing away non-alphanumeric characters.

We map the words to a dense vector representation using a trainable word embedding matrix $\mathbf{L} \in L \times D$, where D is the dimensionality of the semantic space, and L is the size of the dictionary. To initialize the word embeddings, we use the pretrained GloVe vectors. The words that don't occur in the pretrained word embedding model are initialized with zeros. We encode a caption using a GRU and a parser. The parser takes a caption and parses it into a set of objects with their attributes and a set of relationship triplets. The encoded RGC is a vector representation denoted by $\tilde{\mathbf{x}} \in \mathbb{R}^D$. See appendix A for more detail about the RGC encoding.

4.2 Question Module

We encode a question using the same dictionary as we use for captions. This enables our model to match the words in a caption with the words in a question and attend to the relevant caption. The final hidden state of a GRU, denoted by $\mathbf{q}$, is used as the representation of the question.

4.3 Multimodal Graph Memory Network

Given a set of visual feature vectors, a set of encoded RGCs, and the encoded question, the multimodal graph memory network module produces a representation of the relevant information based on the encoded question. The memory chooses which parts of the inputs to focus on using an attention mechanism. Unlike previous work (Xu and Saenko, 2015; Xiong et al., 2016), our memory network module is multimodal and relational. That is, it employs both textual and visual information of the input image regions, and it exploits pair-wise interactions between each pair of visual/textual features using a visual/textual GN. Similar to visual features, most of the RGCs may be irrelevant to the given question. Thus, the memory module needs to learn an attention mechanism for focusing on the relevant RGCs.

Formally, the multimodal graph memory network is composed of a visual GN $\mathcal{G} = (\mathbf{u}, \mathcal{V}, \mathcal{E})$ with N nodes, a textual GN $\tilde{\mathcal{G}} = (\tilde{\mathbf{u}}, \tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ with $\tilde{N}$ nodes, and an external spatial memory. Each node of the visual GN represents a visual feature with an associated bounding-box. Similarly, each node of the textual GN has a bounding-box corresponds to a detected RGC of the image. In both GNs, we connect two nodes via two forward and backward edges if they are nearby. That is, we connect two nodes if the Euclidean distance between the normalized center of their bounding-boxes is less than $\gamma = 0.5$. Note that even if two nodes of a GN are not neighbors, they may still communicate via the message passing mechanism of the GN.

The external memory is a network of memory cells arranged in a $P \times Q$ grid. Each cell has a fixed location that corresponds to a specific $(H/P) \times (W/Q)$ region in the image, where H and W are height and width of the image. Each node of the visual/textual GN sends its information to a memory cell if its bounding-box covers the location of the cell. Since the bounding-boxes may overlap, a cell may get information from multiple nodes. The external memory network is responsible for aggregating the information from both GNs and eliminating redundancy introduced by overlapping bounding-boxes. This makes our architecture less sensitive to the number of detected bounding-boxes. Since the input to the spatial memory is the output of the GNs, the state of the GN nodes can be seen

as an internal memory, and the state of the spatial memory can be seen as an "external" memory like Neural Turing Machines (Graves et al., 2014).

Initialization. To initialize each node attribute of the visual GN, we combine a visual feature vector extracted from a region of the image with the encoded question using MCB pooling as $\mathbf{v}_i = \mathbf{q} \star \mathbf{x}_i$, where $\star$ represents the MCB pooling. Similarly, we initialize each node attribute of the textual GN as $\tilde{\mathbf{v}}_i = \mathbf{q} \odot \tilde{\mathbf{x}}_i$, where $\odot$ is the element-wise multiplication. We use the MCB to combine the visual features with the encoded question since the question and visual features are from different modalities. The global attribute $\mathbf{u}$ is initialized by a global feature vector of the image extracted from the last layer of the 101-layer ResNet. This helps to answer questions that need the global features of the scene. The global attribute $\tilde{\mathbf{u}}$ is initialized with the encoded question. The edge features of the GNs and memory cells are initialized with zero vectors.

Updates. At each iteration, we first update the GNs. Then, we update the content of the memory cells. We update the edge attributes, node attributes, and global attribute of both GNs as described in Algorithm 1. For each GN, we use three different GRUs to implement the functions ϕ^e, ϕ^v, and ϕ^u. The $\rho^{e\to v}$ is an element-wise summation. The $\rho^{v\to u}$ and $\rho^{e\to u}$ for visual GN are implemented as

$$\bar{\mathbf{v}}' = \psi\big(\textstyle\sum_i \sigma(\mathbf{W}_1\mathbf{v}_i + \mathbf{b}_1) \odot \psi(\mathbf{W}_2\mathbf{v}_i + \mathbf{b}_2)\big)$$
$$\bar{\mathbf{e}}' = \psi\big(\textstyle\sum_k \sigma(\mathbf{W}_3\mathbf{e}_k + \mathbf{b}_3) \odot \psi(\mathbf{W}_4\mathbf{e}_k + \mathbf{b}_4)\big)$$

where, σ and ψ are the sigmoid and tangent hyperbolic activation functions, and $\mathbf{W}_i$, $\mathbf{b}_i, i = 1, \ldots, 4$, are trainable parameters. This allows to incorporate information from the question for computing the attention weights using the sigmoid function for each node/edge. The $\rho^{v\to u}$ and $\rho^{e\to u}$ for the textual GN are implemented in a similar way. Let, $\bar{\mathbf{v}}_{p,q} = \frac{1}{|\mathcal{N}_{p,q}|}\sum_{i\in\mathcal{N}_{p,q}} \mathbf{v}_i$ and $\bar{\tilde{\mathbf{v}}}_{p,q} = \frac{1}{|\tilde{\mathcal{N}}_{p,q}|}\sum_{i\in\tilde{\mathcal{N}}_{p,q}} \tilde{\mathbf{v}}_i$, where $\mathcal{N}_{p,q}$ and $\tilde{\mathcal{N}}_{p,q}$ are the set of nodes which are connected to the memory cell (p, q) in the visual and textual GNs, respectively. Each memory cell is updated as

$$\bar{\mathbf{m}}_{p,q} = f(\mathbf{m}_{p-1,q}, \mathbf{m}_{p,q-1}, \mathbf{m}_{p,q+1}, \mathbf{m}_{p+1,q})$$
$$\mathbf{m}'_{p,q} = \text{GRU}([\bar{\mathbf{v}}_{p,q}, \bar{\tilde{\mathbf{v}}}_{p,q}, \bar{\mathbf{m}}_{p,q}], \mathbf{m}_{p,q})$$

where f is a neural network layer which aggregates the memories from the neighboring cells. We repeat these steps for two iterations. Applying one iteration decreases the accuracy by about 2.0 points. As observed by Kumar et al. (2015), iterating over the inputs allows the memory network to take several reasoning steps which some questions require.

4.4 Answer Module

The answer module predicts an answer using a GN called answer GN. The nodes of the answer GN are the external spatial memory cells. However, there is an edge between every ordered pair of the nodes (cells), hence the answer GN is a complete graph. This supports reasoning across distant regions of the image. Let $\mathbf{m}^{\circ}_{p,q}$ be the final state of the memory cell at location (p, q). We initialize the node attributes of the answer GN denoted by $\mathbf{v}^{\circ}_{p,q}$ as $\mathbf{v}^{\circ}_{p,q} = \mathbf{m}^{\circ}_{p,q}$. The edge attributes are initialized using the one-hot representation of the location of the sender and receiver memory cells. That is, the edge attribute of the edge going from the memory cell at location (p, q) to (p', q'), is initialized with a vector of size $2P + 2Q$ which is computed by concatenating the one-hot representation of p, q, p', and q'. The global attribute of the answer GN is initialized with a vector of zeros.

Then, we update the edge attributes, the node attributes and the global attribute of the answer GN as described in Algorithm 1. As before, we use three different GRUs to implement functions ϕ^e, ϕ^v, and ϕ^u. The $\rho^{e\to v}$ is a simple element-wise summation. The $\rho^{v\to u}$ and $\rho^{e\to u}$ are implemented as before, but with different set of parameters. The answer module predicts an answer as $\hat{\mathbf{p}} = \sigma\big(\mathbf{W}g(\mathbf{u}^{\circ}) + \tilde{\mathbf{W}}\tilde{g}(\mathbf{u}^{\circ}) + \mathbf{b}\big)$ where, $\mathbf{u}^{\circ}$ is the updated global attribute of the answer GN, $\mathbf{W} \in \mathbb{R}^{Y\times 2048}, \tilde{\mathbf{W}} \in \mathbb{R}^{Y\times 300}, \mathbf{b} \in \mathbb{R}^{Y}$ are trainable parameters, g, $\tilde{g}$ are non-linear layers, and Y is the number of possible answers.

Following Teney et al. (2018), to exploit prior linguistic information about the candidate answers, the GloVe embeddings of the answer words are used to initialize the rows of the $\tilde{\mathbf{W}}$. Initialization with the Glove embeddings improves the performance by about 1.0 point. Similarly, to utilize prior visual information about the candidate answers, a visual embedding is used to initialize the rows of $\mathbf{W}$. The visual embedding is obtained by retrieving 10 image from Google Images for each word. Then, the images are encoded using the ResNet-101 pretrained on ImageNet to obtain a feature vector of size 2048. For each word, the average of the feature vectors is used to initialize a

row of $\mathbf{W}$. The loss for a single sample is defined as $\mathcal{L} = -\sum_{i=1}^{Y} p_i \log(\hat{p}_i) + (1 - p_i) \log(1 - \hat{p}_i)$ where, $\hat{p}_i$ is the ith element of $\hat{\mathbf{p}}$, and p_i is the ith element of the ground-truth vector $\mathbf{p}$ ($p_i = 1.0$ if $A \geq 3$ annotators give the ith answer word, otherwise $p_i = A/3$). For multiple choice task, the candidate answers are encoded by the last state of a GRU and concatenated with $\mathbf{u}^\circ$ using a neural network layer as $\acute{p} = \sigma\big(\acute{\mathbf{w}} \acute{f}([\mathbf{u}^\circ, \mathbf{a}]) + \acute{b}\big)$ where, $\mathbf{a}$ is an encoded answer choice, $\acute{f}$ is a non-linear layer, and $\acute{\mathbf{w}}, \acute{b}$ are trainable parameters. For multiple choice task, the binary logistic loss $-p \log(\acute{p}) - (1 - p) \log(1 - \acute{p})$ 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 MN-GMN is implemented in TensorFlow. We use a library from `https://github.com/deepmind/graph_nets` to implement the GNs. We follow VQA tips in Teney et al. (2018) to train our models. More specifically, to apply an ensemble technique, 20 instances of the model is trained with various initial random seeds. For test images, the scores for the answers by all models are summed, and the answer is predicted using the highest summed score. To minimize the loss, we apply the RMSprop optimization algorithm with a learning rate of 0.0001 and minibatches of size 100.

Dropout with probability 0.5 and early stopping are applied to prevent overfitting. Dropout is used after the layer that computes the updated global attribute of the answer GN. During training, all parameters are tuned except for the weights of the CNN and RGC detector to avoid overfitting. For VQA-v2.0 and Visual7W datasets, we augment the training dataset with Visual Genome/GQA images and QA pairs. The training set that we use does not include the VQA-v2.0/Visual7W test or Visual7W validation images. The output dimension of the MCB and the dimension of the hidden layer in both RGC and question GRUs are set to 512. Also, we set $P, Q = 14$ and $D = 512$. The full model takes around 6 hours to train on two Titan X GPUs.

5 Experiments

We explain the datasets, baseline models, and evaluation metric that we use in our experiments. Then, the experimental results are discussed.

Datasets. VQA-v2.0 (Antol et al., 2015) includes $82,783$ training images, $40,504$ validation images, and $81,434$ testing images. There are $443,757$ training questions, $214,354$ validation questions, and $447,793$ test questions in this dataset. A subset of the standard test set, called test-dev, contains $107,394$ questions. Each question has 10 candidate answers generated by humans. We choose correct answers that appear more than 8 times. This makes $Y = 3,110$ candidate answers. We use the standard metric (Antol et al., 2015), which is an answer is correct if at least 3 people agree.

Visual7W dataset (Zhu et al., 2015) includes $47,300$ images. We train and evaluate our model on *telling* questions of the Visual7W which includes $28,653$ images. This set uses six types of questions: *what* (6%), *where* (48%), *when* (16%), *who* (5%), *why* (10%), *how* (15%). The training, validation and test splits, contain 50%, 20%, 30% of the QA pairs, respectively. For evaluation, Visual7W provides four candidate answers. The Visual7W has fewer language biases compared to VQA.

We also experiment on 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 by the Tensorflow Object Detection API. We use Faster R-CNN NasNet trained on the 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 feature vectors. For CLEVR, we omit the textual GN, since CLEVR images do not have rich textual information.

Baselines. We compare our model with several architectures developed recently, including the state-of-the-art models ReGAT, BAN, VCTREE, and MuRel. For comparison, we also include three related models in Table 1 that have been proposed more recently in Arxiv preprints during the preparation of this work: LXRT, MSM@MSRA, and MIL@HDU. The ReGAT exploits supervision from Visual Genome relationships. MAN is a memory-augmented neural network which attends to each training exemplar to answer visual questions, even when the answers infrequently happen in the training set. The Count (Zhang et al., 2018) is a neural network model designed to count objects from object proposals. For Visual7W, we

compare our models with Zhu et al. (2015), MCB, MAN, and MLP. The MCB leverages the Visual Genome QA pairs as additional training data and the 152-layer ResNet as a pretrained model. The MLP method uses (image,question,answer) triplets to score answer choices. For CLEVR, we compare our models with several baselines proposed by Johnson et al. (2017a) as well as the state-of-the-art models RAMEN, PROGRAM-GEN, and NS-VQA. N2NMN learns to predict a layout based on the question and compose a network using a set of neural modules. The CNN+LSTM+RN learns to infer a relation using a neural network model called Relation Networks. The PROGRAM-GEN exploits supervision from functional programming, which is used to generate CLEVR questions.

Ablation Study. We implement several lesion architectures. The MN+ResNet model does not use any GNs and is designed to evaluate the effect of using GN. This model is similar to MN (Sukhbaatar et al., 2015). It applies a soft attention for 14×14 ResNet feature maps (the last 14×14 pooling layer) and generates a representation $\mathbf{u}^{\circ} = h(\mathbf{q}) \star h'(\sum_{i=1}^{196} \alpha_i \mathbf{x}_i)$. Here h, h' are non-linear layers, and α_i is an attention weight computed as $\alpha_i = \text{softmax}\big(\mathbf{w}h''([\mathbf{x}_i, \mathbf{q}])\big)$, where $\mathbf{w}$ is a learned parameter vector and h'' is a non-linear layer. Then, an answer is predicted as described before.

The N-GMN model only uses the visual GN (no textual GN nor spatial memory). This model evaluates the effect of incorporating RGCs. After two iterations, the global feature vector of the visual GN is used as $\mathbf{u}^{\circ}$ to generate an answer. The N-GMN^{+} model only uses the visual GN and the external spatial memory components (no textual GN). This model is used for the CLEVR dataset since CLEVR images do not have rich textual information. The MN-GMN^{-} model does not use the external spatial memory. After two iterations, the global feature vector of the visual and textual GNs are concatenated and fed into a non-linear layer to generate $\mathbf{u}^{\circ}$. Finally, MN-GMN is our full model.

Results and Discussion. Our experimental results on VQA-v2.0 dataset are reported in Table 1. For LXRT, MSM@MSRA, and MIL@HDU, the numbers are reported from the VQA Challenge 2019 Leaderboard (using an ensemble of models). Across all question types, N-GMN outperforms MN+ResNet. This shows that applying the visual GN with explicit object bounding-boxes provides a usefully richer representation than a grid of fixed visual features. MN-GMN^{-} outperforms N-GMN. This shows that RGCs help to improve accuracy. RGCs are especially useful for answering the Other and Yes/No question types. Our full model MN-GMN outperforms MN-GMN^{-}. This shows that applying external spatial memory is effective, especially for Number questions. The full model's accuracy is higher than the baselines.

	Test-dev				Test-std			
Model	Y/N	Num	Other	All	Y/N	Num	Other	All
MAN[1]	-	-	-	-	79.2	39.5	52.6	62.1
Count[2]	83.1	51.6	59.0	68.1	83.6	51.4	59.1	68.4
MFH[3]	84.3	50.7	60.5	68.8	-	-	-	-
Buttom-Up[4]	81.8	44.2	56.1	65.3	-	-	-	65.7
G-learner[5]	-	-	-	-	82.9	47.1	56.2	66.2
v-AGCN[6]	82.4	45.9	56.5	65.9	82.6	45.1	56.7	66.2
RAMEN[7]	-	-	-	66.0	-	-	-	-
MuRel[8]	84.8	49.8	57.9	68.0	-	-	-	68.4
VCTREE[9]	84.3	47.8	59.1	68.2	84.6	47.4	59.3	68.5
BAN[10]	85.4	54.0	60.5	70.0	-	-	-	70.4
ReGAT[11]	86.1	54.4	60.3	70.3	-	-	-	70.6
LXRT[12]	89.3	56.9	65.1	74.2	89.5	56.7	65.2	74.3
MSM@MSRA [13]	89.8	58.9	65.4	74.7	89.8	58.4	65.7	74.9
MIL@HDU[14]	**90.1**	**59.2**	**65.7**	**75.0**	**90.4**	**59.2**	**65.8**	**75.2**
MN+ResNet	84.2	43.4	58.1	67.3	84.5	44.0	58.1	67.5
N-GMN	86.1	53.5	61.2	70.6	86.7	53.6	61.8	71.2
MN-GMN^{-}	88.0	53.5	63.8	72.6	88.5	53.7	64.2	73.1
MN-GMN	88.2	56.0	64.2	73.2	88.3	56.1	64.5	73.5

Table 1: Accuracy percentage on the VQA-v2.0 dataset. The references are Ma et al. (2018)[1], Zhang et al. (2018)[2], Yu et al. (2018)[3], Teney et al. (2018); Anderson et al. (2018)[4], Norcliffe-Brown et al. (2018)[5], Yang et al. (2018)[6], Shrestha et al. (2019)[7], Cadene et al. (2019)[8], Tang et al. (2019)[9], Kim et al. (2018)[10], Li et al. (2019)[11], Tan and Bansal (2019)[12], Liu et al. (2019)[13], and Yu et al. (2019)[14].

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
N-GMN	66.2	77.2	83.3	74.0	69.2	58.5	68.6
MN-GMN^{-}	67.1	77.4	84.0	**75.1**	70.1	59.2	69.3
MN-GMN	**67.3**	**77.4**	**84.0**	75.0	**70.3**	**59.4**	**69.5**

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

Our results on Visual7W are reported in Table 2. Our N-GMN, MN-GMN^{-}, and MN-GMN outperform the baselines MLP, MAN, and MCB+ATT. The results for our N-GMN^{+} on CLEVR in Table 3 are competitive with the state-of-the-art RAMEN, PROGRAM-GEN, and NS-VQA. We emphasize that, unlike PROGRAM-GEN, our algorithm does

not exploit supervision from functional programming. Also, unlike NS-VQA, our model is not tailored to synthetic datasets only, since it performs well on both natural and artificial datasets that need multi-step compositional reasoning.

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
RAMEN[5]	96.9	98.9	94.1	88.5	98.9	99.3
NS-VQA[6]	**99.8**	**99.9**	**99.7**	**99.9**	**99.8**	**99.8**
N-GMN$^-$	95.6	97.7	90.3	93.5	98.0	97.3
N-GMN$^+$	96.3	98.0	91.8	94.8	98.1	98.1

Table 3: Accuracy on CLEVR dataset. The references are Johnson et al. (2017a)[1], Hu et al. (2017)[2], Santoro et al. (2017)[3], Johnson et al. (2017b)[4], Shrestha et al. (2019)[5], and Yi et al. (2018)[6].

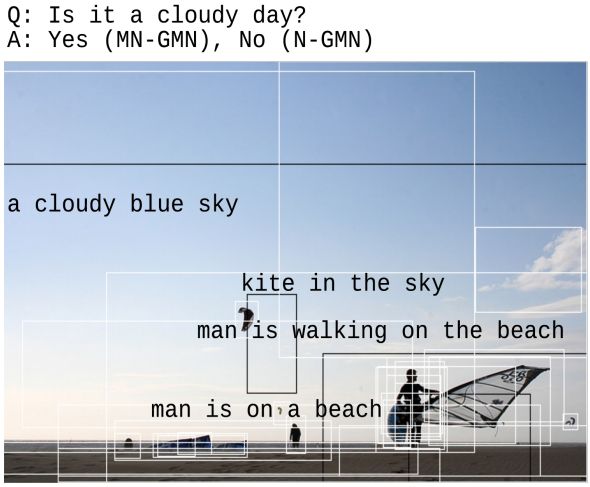

Figure 3: N-GMN versus MN-GMN. The MN-GMN provides the correct answer using *a cloudy blue sky*.

Figure 3 shows how MN-GMN can answer a question correctly by incorporating RGCs, whereas N-GMN gives the wrong answer. Figure 4 illustrates the visualization of the attention weights with MN-GMN to answer a Number question. We compute the attention weights that are used to obtain $\bar{\mathbf{v}}'$ for each spatial memory cell. More precisely, the magnitude of the sigmoid output that implements $\rho^{v \to u}$ for the spatial memory is visualized. Each attention weight shows the importance of a fixed region in a 14×14 grid of cells to the question. Figure 5 shows a VQA example on the CLEVR dataset. Appendix B provides more examples.

Figure 4: Visualization of the attention weights for a 14×14 grid of cells. Red regions get higher attention.

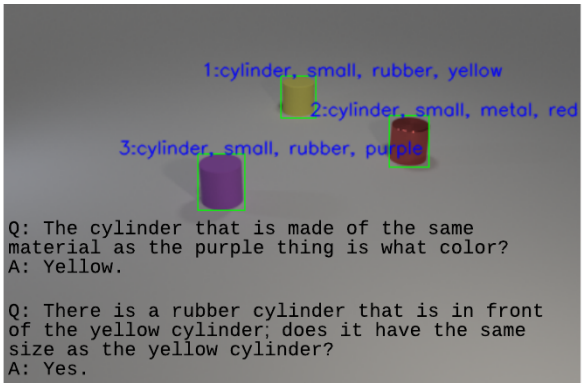

Figure 5: Example VQA with N-GMN$^+$ on CLEVR.

6 Conclusions

Multi-modal Neural Graph Memory Networks are a new architecture for the VQA task. The MN-GMN represents bimodal local features as node attributes in a graph. It leverages a graph neural network model, Graph Network, to reason about objects and their interactions in a scene. In experiments on three datasets, the MN-GMN showed superior quantitative and qualitative performance compared to the lesion approaches and rivals the state-of-the-art models. A future research direction is to combine RGCs with distant supervision by an external knowledge base to answer the visual questions that need external knowledge; for example *Which animal in this photo can climb a tree?*

References

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4971–4980.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Remi Cadene, Hedi Ben-Younes, Matthieu Cord, and Nicolas Thome. 2019. Murel: Multimodal relational reasoning for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1989–1998.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813.

Drew Hudson and Christopher D Manning. 2019a. Learning by abstraction: The neural state machine. In *Advances in Neural Information Processing Systems*, pages 5901–5914.

Drew A Hudson and Christopher D Manning. 2019b. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6700–6709.

Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. 2016. Revisiting visual question answering baselines. In *European conference on computer vision*, pages 727–739. Springer.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017a. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017b. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2015. Densecap: Fully convolutional localization networks for dense captioning. *arXiv preprint arXiv:1511.07571*.

Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Relation-aware graph attention network for visual question answering. *arXiv preprint arXiv:1903.12314*.

Bei Liu, Zhicheng Huang, Zhaoyang Zeng, Zheyu Chen, and Jianlong Fu. 2019. Learning rich image region representation for visual question answering. *arXiv preprint arXiv:1910.13077*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Chao Ma, Chunhua Shen, Anthony Dick, Qi Wu, Peng Wang, Anton van den Hengel, and Ian Reid. 2018. Visual question answering with memory-augmented networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6975–6984.

Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. 2018. Learning conditioned graph structures for interpretable visual question answering. In *Advances in Neural Information Processing Systems*, pages 8334–8343.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. 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.

Robik Shrestha, Kushal Kafle, and Christopher Kanan. 2019. Answer them all! toward universal visual question answering models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10472–10481.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*.

Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. 2019. Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6619–6628.

Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 2018. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4223–4232.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. *arXiv preprint arXiv:1603.01417*.

Huijuan Xu and Kate Saenko. 2015. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *arXiv preprint arXiv:1511.05234*.

Zhuoqian Yang, Jing Yu, Chenghao Yang, Zengchang Qin, and Yue Hu. 2018. Multi-modal learning with prior visual relation reasoning. *arXiv preprint arXiv:1812.09681*.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1031–1042.

Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. 2019. Deep modular co-attention networks for visual question answering. *arXiv preprint arXiv:1906.10770*.

Zhou Yu, Jun Yu, Chenchao Xiang, Jianping Fan, and Dacheng Tao. 2018. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE transactions on neural networks and learning systems*, 29(12):5947–5959.

Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. 2018. Learning to count objects in natural images for visual question answering. *arXiv preprint arXiv:1802.05766*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2015. Visual7w: Grounded question answering in images. *arXiv preprint arXiv:1511.03416*.

A Encoding Region-Grounded Captions

Given a caption, the hidden state of the caption GRU for the i-th word is computed by $\mathbf{c}_i = \text{GRU}(\mathbf{s}_i, \mathbf{c}_{i-1})$, where $\mathbf{s}_i$ is the semantic representation of the i-th word in the caption. The final hidden state of the GRU, denoted by $\mathbf{c} \in \mathbb{R}^D$, is used as a vector representation of the caption. We reset the GRU after feeding each caption.

The encoded captions may not properly represent the objects in an image, the relationships between them, and the object attributes, especially for a caption with a complex parse tree. Thus, we enrich this representation by utilizing a parser developed by Schuster et al. (2015) that takes a single caption and parses it into a set of relationship triplets (possibly empty), a set of objects, and their attributes. For example, given *orange and white cat laying on a wooden bench*, the parser outputs a triplet *cat-lay on-bench*, objects *cat* and *bench*, and attributes *cat-white*, *cat-orange* and *bench-wooden*.

For the Visual Genome dataset, the parser produces less than three relationships for about %98

of the captions. We obtain a fixed-length representation for the output of the parser, denoted by $\tilde{\mathbf{c}} \in \mathbb{R}^{14D}$ by allocating the embedding of 14 words: 6 words for up to two relationship triplets and 8 words for up to 4 objects and their attributes. For the aforementioned example, the fixed-length representation is the concatenation of the embedding of each word in sequence $\prec$*cat-lay on-bench,x-x-x,bench-wooden,cat-orange,cat-white,x-x*$\succ$, where *x* is a special token to represent an empty slot. To create the sequences, we use a fixed arbitrary order. Each RGC has also a bounding-box specified by its coordinates $\tilde{\mathbf{r}} = (\tilde{r}_x, \tilde{r}_y, \tilde{r}_{x'}, \tilde{r}_{y'})$, where $(\tilde{r}_x, \tilde{r}_y)$ and $(\tilde{r}_{x'}, \tilde{r}_{y'})$ are the top-left and bottom-right corners of the bounding-box which are normalized to have a value between 0 and 1 based on the height and width of the image. For each RGC, we project the concatenation of $\tilde{\mathbf{r}}$, $\mathbf{c}$ and $\tilde{\mathbf{c}}$ to a space of dimensionality D using a densely-connected layer with ReLU activation function to obtain a vector representation denoted by $\tilde{\mathbf{x}} \in \mathbb{R}^D$.

B More Examples for VQA Task

Figure 6 shows a VQA example on CLEVR dataset. Figure 7 shows how MN-GMN can answer a question correctly by incorporating the region-grounded captions, whereas N-GMN gives the wrong answer. Figure 8 illustrates the visualization of the attention weights with MN-GMN to answer a Number question. For this example, we compute the attention weights that are used to obtain $\bar{\mathbf{v}}'$ for each spatial memory cell. More precisely, the magnitude of the sigmoid output that implements $\rho^{v \to u}$ for the external spatial memory is visualized. Each attention weight shows the importance of a fixed region in a 14×14 grid of memory cells to the question.

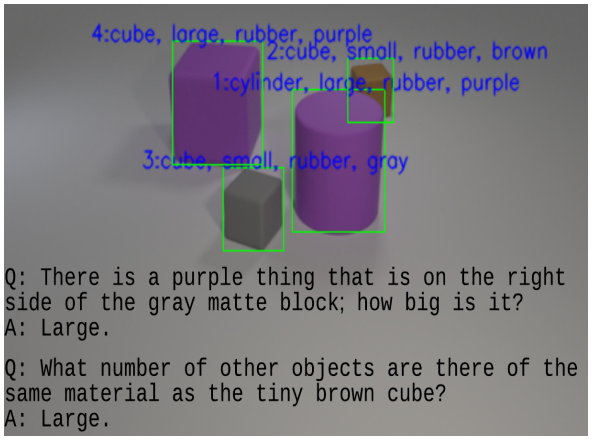

Figure 6: Example VQA with N-GMN$^+$ on CLEVR.

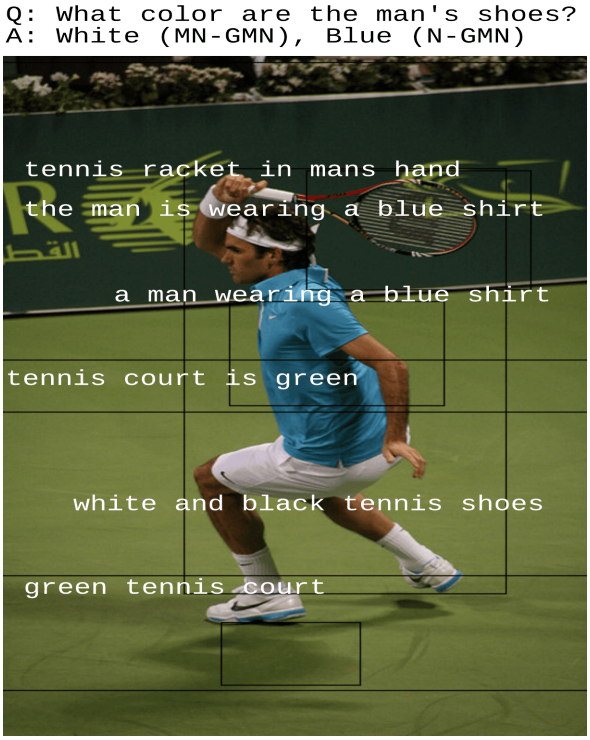

Figure 7: The MN-GMN provides the correct answer using *white and black tennis shoes*.

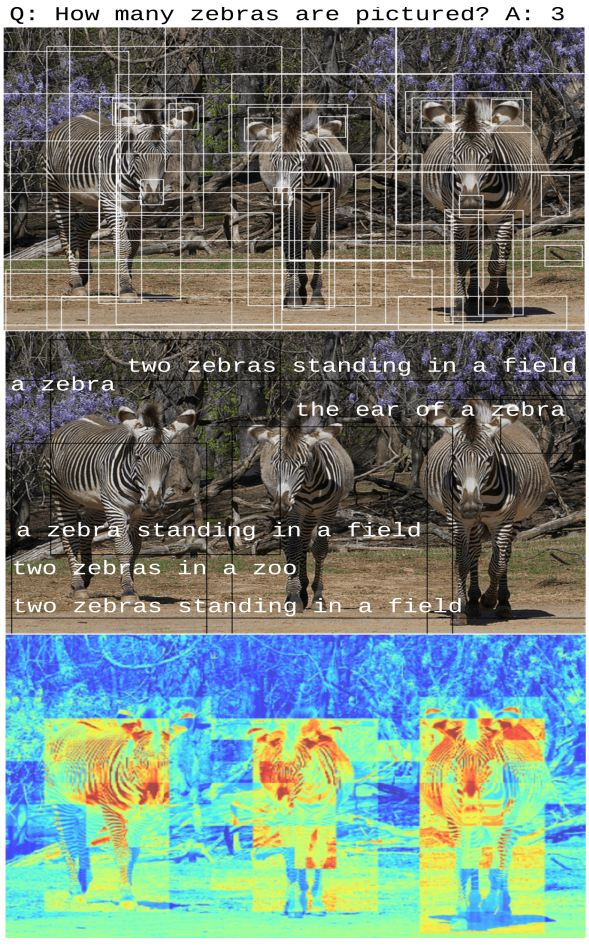

Figure 8: Visualization of the attention weights for a 14×14 grid of cells. Red regions get higher attention.