Multimodal Neural Graph Memory Networks for Visual Question Answering

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

Parmis Naddaf
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
pnaddaf@sfu.ca

Oliver Schulte
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
oschulte@sfu.ca

Abstract

We introduce a new neural network architecture, Multimodal Neural Graph Memory Networks (MN-GMN), for visual question answering (VQA). MN-GMN uses graph structure with different region features as node attributes and applies the recently proposed Graph Network (GN) to reason about objects and their interactions in the scene context. The MN-GMN has four modules. The input module generates a set of visual feature vectors plus a set of encoded region-grounded captions (RGCs) for the image. The RGCs capture object attributes and their relationships. Each visual feature vector/RGC is specified with a bounding-box. Two GNs are constructed from the input module using the visual feature vectors and RGCs. Each node in a GN iteratively computes a question-guided contextualized representation of the visual/textual information assigned to it. To combine the information from both GNs, each node writes the updated representations to a spatial memory. The final state of the memory cells are fed into an answer module to predict an answer. Experiments show MN-GMN outperforms the state-of-the-art on VQA dataset.

1 Introduction

Visual question answering (VQA) has been recently introduced as a grand challenge for AI. This paper proposes a new neural network architecture for VQA based on the recent Graph Network (GN) [3]. The pair-wise 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 surfers are in the picture?* a model may need to aggregate information from multiple, possibly distant, regions. Our new architecture (see Figure 1), Multimodal Neural Graph Memory Network (MN-GMN), uses a 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 which need reasoning about complex arrangements of objects in a scene.

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 Visual Genome dataset. 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, but may not be represented by low-level visual features. For example, in Figure 1, to answer *Is the water calm?*, the caption *a wave in the ocean* is informative; the caption *the water is blue* specifies an attribute of *water*; the captions *a sailboat in the water* and *surfer riding a wave* describe interactions between objects. Captions also incorporate commonsense knowledge that may not be represented by an

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

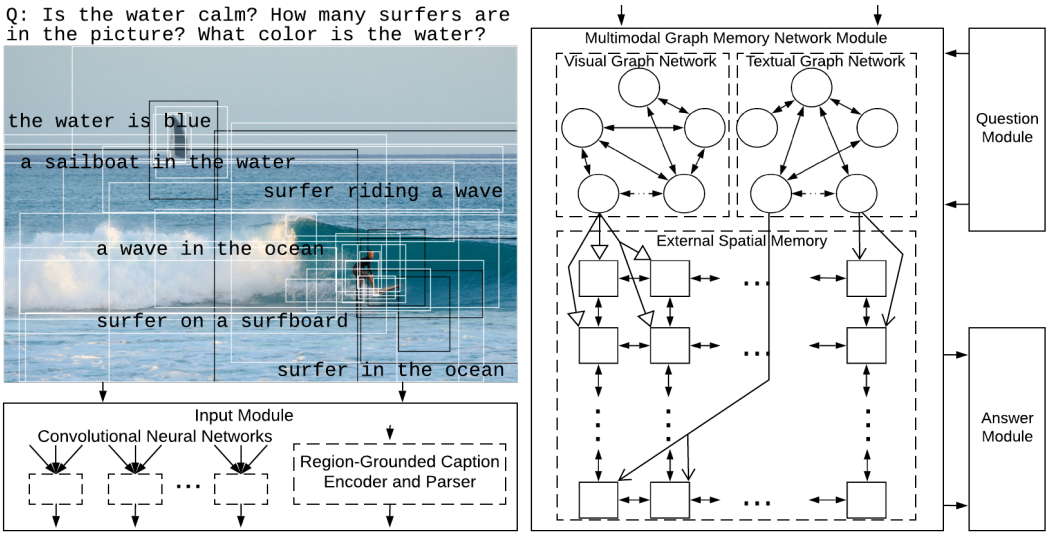

Figure 1: Multimodal Neural Graph Memory Networks for VQA.

end-to-end model especially with currently available small VQA datasets. 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 contextualized representation of the visual/textual information at the bounding-box assigned to it. Our architecture has an external spatial memory which is designed to combine information across the modalities: each node writes the updated representations to the external memory, which is composed of memory cells arranged in a 2D grid. The final state of the memory cells are then fed into the answer module to predict an answer.

2 Related Work

Recently, a few models are proposed which can learn the interactions between image regions. The graph learner model [16], 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. [23] proposed to reason over a visual representation of the input image called scene graph which represents object and their relationships explicitly. [14] introduced a VQA model called Relation-aware Graph Attention Network (ReGAT). Guided by the input question, ReGAT encodes an image into a graph that represents relations among visual objects. The ReGAT is trained on Visual Genome dataset [12]. [7] introduced a graph neural network model called Relation Networks which uses multilayer perceptron models to reason over all pairs of local image features extracted from a spatially localized grid of image regions. Dynamic tree structures have been used in VQA to capture the visual context of image objects [19].

3 Our Proposed Architecture

Figure 1 shows our MN-GMN architecture which has 4 modules. We now describe these modules.

Input Module. The input module has two components: A deep CNN (e.g., Bottom-Up Attention [1], VGGNet, ResNet, etc.), and a region-grounded caption generation model which incorporates a set of RGCs to answer questions about object attributes and their relationships. To extract visual features, we use the Bottom-Up Attention model [1]. The features are obtained via a Faster R-CNN framework 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. We concatenate each feature vector with its bounding-box to obtain a vector denoted by $\mathbf{x}_i$, $(i = 1, \ldots, N)$. The RGCs are generated by a state-of-the-art dense captioning model. Then, they are encoded with a GRU and a parser [18] (see supplementary for more detail). An encoded RGC is denoted by $\tilde{\mathbf{x}}_i \in \mathbb{R}^D$.

Question Module. We encode a question using final hidden state of a GRU, denoted by $\mathbf{q}$, and the same dictionary as we use for captions to match the words in a caption with the words in a question.

Multimodal Graph Memory Network Module. Given a set of visual feature vectors, a set of encoded RGCs and the encoded question, the multimodal graph memory network module produces a vector 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 works [22, 21], 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. 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 (i.e., the Euclidean distance between the normalized center of their bounding-boxes is less than γ). Note that even if two nodes of a GN are not neighbors, they may still communicate via massage passing mechanism of the GN. The external memory is a network of memory cells arranged in a $P \times Q$ two-dimensional grid. Each cell has a fixed location 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. It also resolves the redundancy introduced by overlapping bounding-boxes, which causes difficulties for example with counting questions. 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 multimodal compact bilinear (MCB) pooling [5] 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, and 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. At each iteration, we first update the GNs. Then, we we update the content of the memory cells. We update edge attributes, node attributes and the global attribute of both GNs as described in Algorithm 1. For each GN, we use three different GRUs to implement the functions ϕ^e and ϕ^v, ϕ^u. The $\rho^{e\to v}$ is a simple element-wise summation. The $\rho^{v\to u}$ and $\rho^{e\to u}$ for visual GN are implemented as

$$\bar{\mathbf{v}}' = \psi\big(\textstyle\sum_{i=1}^{N} \sigma(\mathbf{W}_1\mathbf{v}_i + \mathbf{b}_1) \odot \psi(\mathbf{W}_2\mathbf{v}_i + \mathbf{b}_2)\big) \tag{1}$$

$$\bar{\mathbf{e}}' = \psi\big(\textstyle\sum_{k=1}^{M} \sigma(\mathbf{W}_3\mathbf{e}_k + \mathbf{b}_3) \odot \psi(\mathbf{W}_4\mathbf{e}_k + \mathbf{b}_4)\big) \tag{2}$$

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

$$\bar{\mathbf{m}}_{p,q} = f(\mathbf{m}_{p-1,q}, \mathbf{m}_{p,q-1}, \mathbf{m}_{p,q+1}, \mathbf{m}_{p+1,q}) \tag{3}$$

$$\mathbf{m}'_{p,q} = \text{GRU}([\bar{\mathbf{v}}_{p,q}, \bar{\tilde{\mathbf{v}}}_{p,q}, \bar{\mathbf{m}}_{p,q}], \mathbf{m}_{p,q}) \tag{4}$$

where f is a neural network layer which aggregates the memories from the neighboring cells. We repeat these steps for two iterations. As observed by [13], iterating over the inputs allows the memory network to take several reasoning steps, which some questions require.

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 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 and ϕ^v, 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 1 and 2, but with different set of parameters. The answer module predicts an answer as

$$\hat{\mathbf{p}} = \sigma\big(\mathbf{W}g(\mathbf{u}^\circ) + \tilde{\mathbf{W}}\tilde{g}(\mathbf{u}^\circ) + \mathbf{b}\big) \tag{5}$$

where, $\mathbf{u}^\circ$ is the updated global attribute of the answer GN, $\tilde{\mathbf{W}} \in \mathbb{R}^{Y\times 300}$, $\mathbf{W} \in \mathbb{R}^{Y\times 2048}$, $\mathbf{b} \in \mathbb{R}^Y$ are trainable parameters, g, $\tilde{g}$ are non-linear layers, and Y is the number of possible answers. 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 $p_i = 1.0$ if $A \geq 3$ annotators give the ith answer word, otherwise $p_i = A/3$.

4 Experiments

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

Dataset. VQA v2.0 [2] 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.

Baseline Methods. For VQA dataset, we compare our model with several architectures developed recently including the state-of-the-art models ReGAT, BAN, VCTREE, and MuRel. The ReGAT uses Visual Genome relationships as external training data. MAN [15] is a memory-augmented neural network which attends to each training exemplar to answer visual questions, even when the answers happen infrequently in the training set. The Count [25] is a neural network model designed to count objects from object proposals. We implement several lesion architectures. 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 $\mathbf{u}^\circ$ in Eq. 5 to generate an answer. 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 $\mathbf{u}^\circ$ in Eq. 5. Finally, MN-GMN is our full model.

Results and Discussion. Our experimental results on VQA dataset are reported in Table 1. MN-GMN$^-$ outperforms N-GMN. This shows that RGCs help to improve the 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. The supplement reports similar results on Visual7W and CLEVR datasets, as well as qualitative evaluation.

	Test-dev				Test-std			
Model	Y/N	Num	Other	All	Y/N	Num	Other	All
MAN [15]	-	-	-	-	79.2	39.5	52.6	62.1
Count [25]	83.1	51.6	59.0	68.1	83.6	51.4	59.1	68.4
MFH [24]	84.3	50.7	60.5	68.8	-	-	-	-
Buttom-Up [20, 1]	81.8	44.2	56.1	65.3	-	-	-	65.7
G-learner [16]	-	-	-	-	82.9	47.1	56.2	66.2
v-AGCN [23]	82.4	45.9	6.5	65.9	82.6	45.1	56.7	66.2
MuRel [4]	84.8	49.8	57.9	68.0	-	-	-	68.4
VCTREE [19]	84.3	47.8	59.1	68.2	84.6	47.4	59.3	68.5
BAN [11]	85.4	54.0	60.5	70.0	-	-	-	70.4
ReGAT [14]	86.1	54.4	60.3	70.3	-	-	-	70.6
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 of various models on VQA-v2.0.

5 Conclusions

Multi-modal Neural Graph Memory Networks are a new architecture for VQA. MN-GMN represents bi-modal local features as node attributes in a graph. It leverages Graph Networks to reason about objects and their interactions in the scene. MN-GMN showed superior quantitative and qualitative performance, compared to both state-of-the-art comparison and lesion approaches on VQA v2.0.

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 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, 2018.

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

[3] 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. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

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

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

[6] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 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, 2017.

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

[8] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 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, 2017.

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

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

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

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

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

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

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

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

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

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

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

[20] Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 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, 2018.

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

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

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

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

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

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

Supplementary Materials

1 Qualitative Evaluation

Figure S1 and Figure S2 show how MN-GMN can answer a question correctly by incorporating RGCs, whereas N-GMN gives the wrong answer. Figure S3 and Figure S4 illustrate visualization of the attention weights with MN-GMN to answer a Number question. For this example, we compute the attention weights which are used to obtain $\bar{\mathbf{v}}'$ for each memory cell. (More precisely, the magnitude of the sigmoid output which implements $\rho^{v \to u}$ for the external spatial memory; cf. Eq. 1). Each attention weight shows the importance of a fixed region in a 14×14 grid of cells to the question. Figure S5 and Figure S6 show VQA examples on the CLEVR dataset.

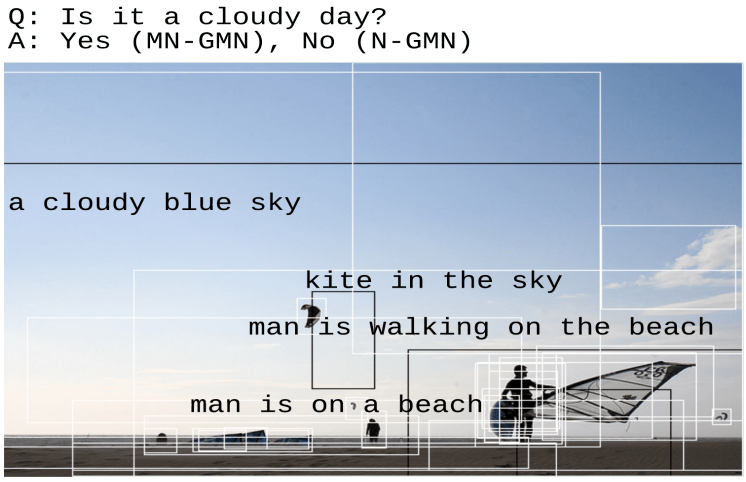

Figure S1: MN-GMN provides the correct answer using *a cloudy blue sky*.

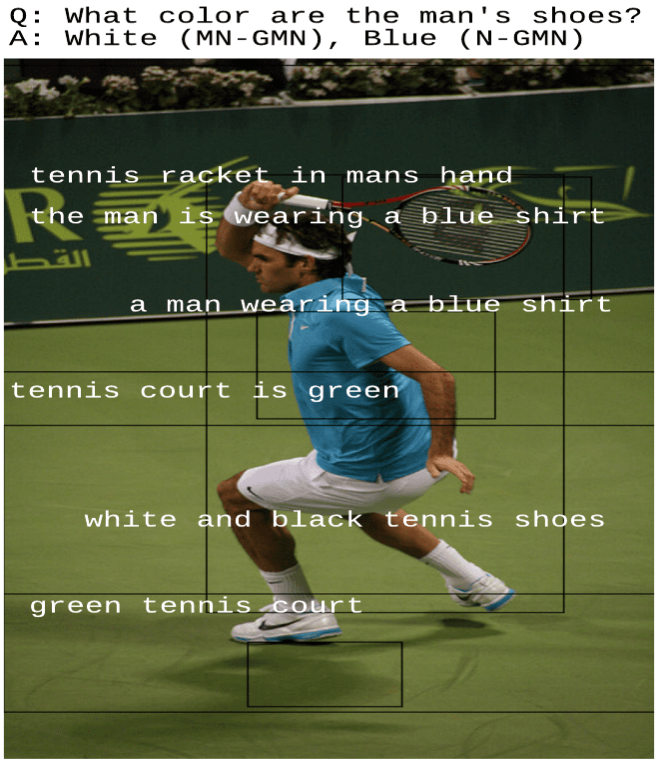

Figure S2: MN-GMN provides the correct answer using *white and black tennis shoes*

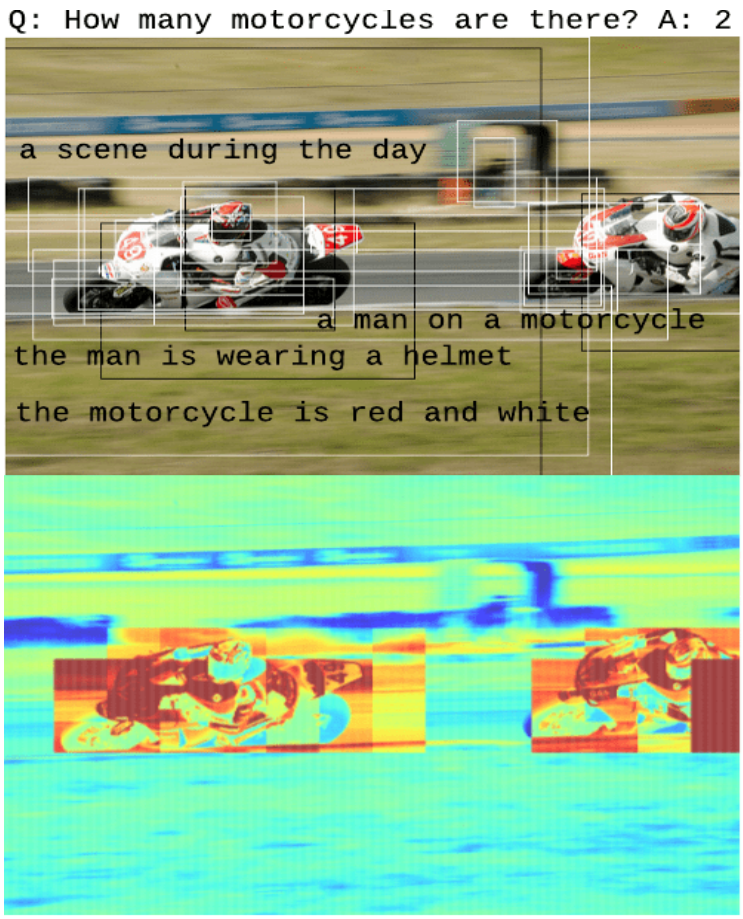

Figure S3: Visualization of the attention weights for a 14×14 grid of memory cells. The red regions get higher attention.

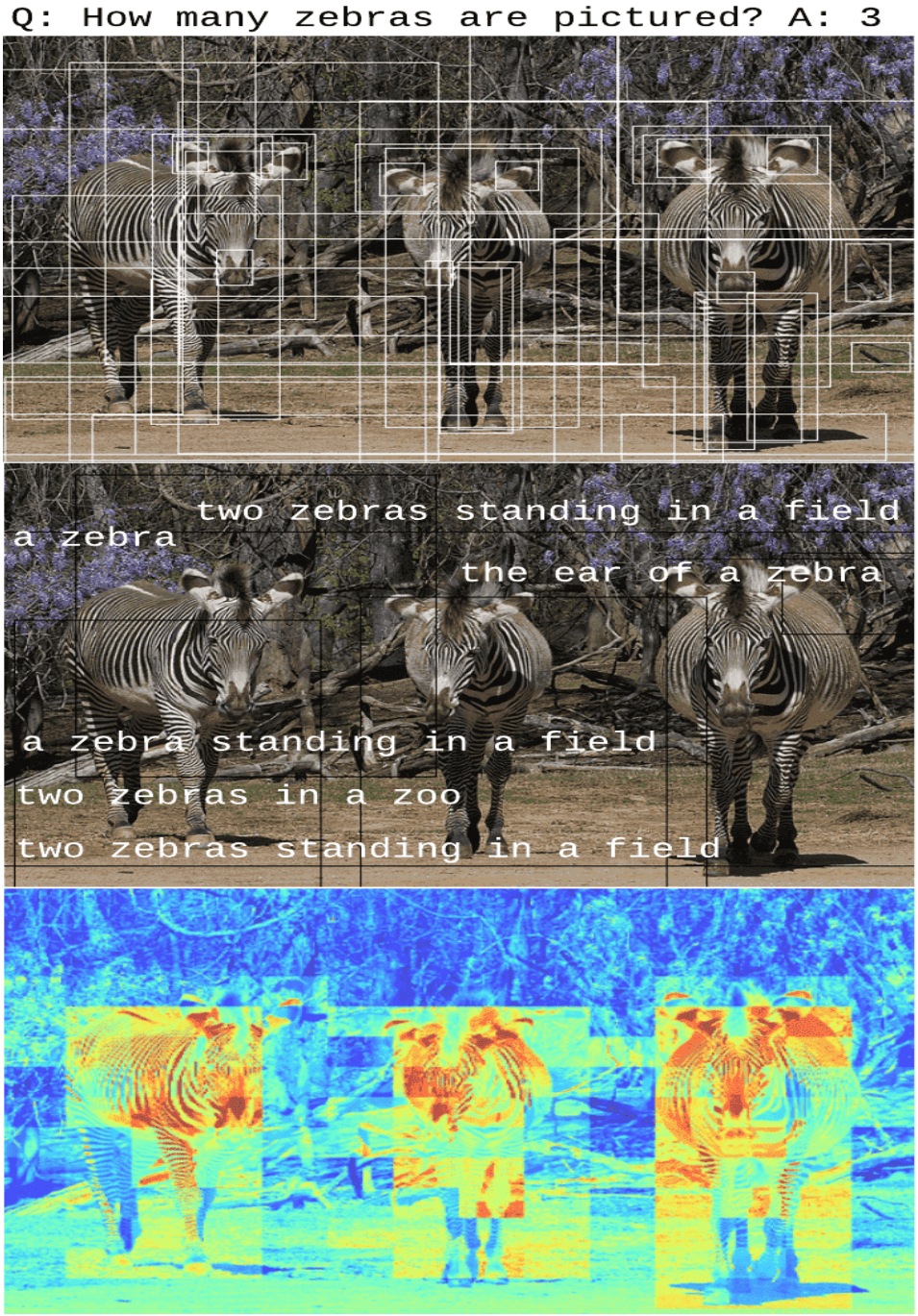

Figure S4: Visualization of the attention weights for a 14×14 grid of memory cells. The red regions get higher attention.

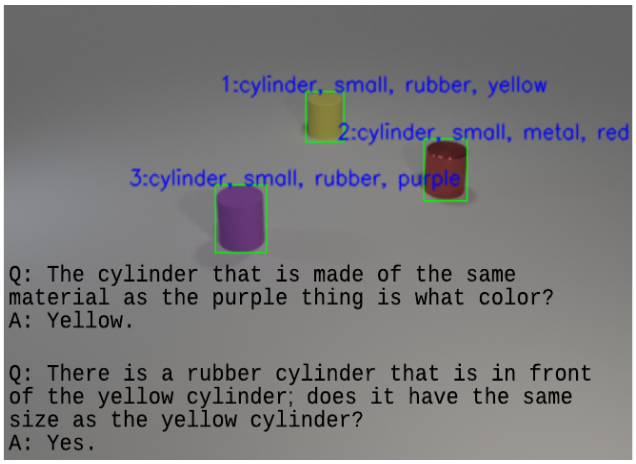

Figure S5: Example VQA with N-GMN on CLEVR dataset.

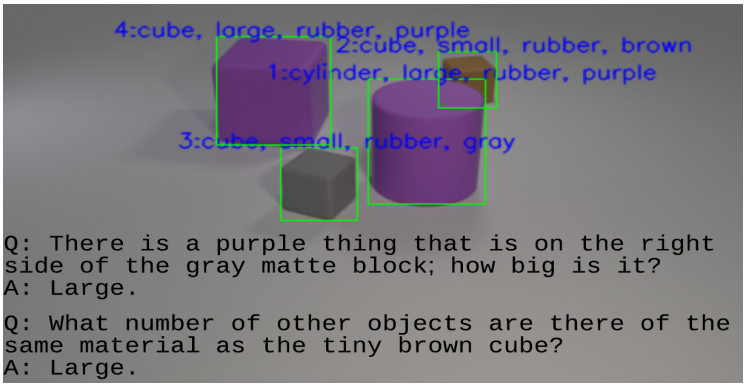

Figure S6: Example VQA with N-GMN on CLEVR dataset.

2 Graph Networks

In this section, we briefly explain the graph networks (GN) framework [3]. In 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 (global) 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 ϕ^e is mapped over entire edges to calculate per-edge updates, ϕ^v is mapped over entire nodes to calculate per-node updates, 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 varying number of arguments, e.g. maximum, summation, etc.

Algorithm 1: Computational steps in a Graph Network block [3]

Input : A graph $G = (\mathbf{u}, \mathcal{V}, \mathcal{E})$
Output The 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 $(\mathbf{u}', \mathcal{V}', \mathcal{E}')$

3 Encoding the Region-Grounded Captions

The second component of the input module applies a dense captioning model from `https://github.com/jcjohnson/densecap` to extract a set of RGCs for the input image [10]. This model contains a CNN (VGGNet), a dense localization layer, and an RNN language model that generates the captions. The model has been trained on RGCs from the Visual Genome dataset. Through transfer learning, our model is leveraging the caption annotations. In contrast with other datasets which only include full image captions, or ground words of image captions in regions, the Visual Genome provides individual region captions. Each RGC has a caption, a bounding-box, and a confidence score. For each image, we select all RGCs with a confidence score greater than 1.5. 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 not occur in the pretrained word embedding model are initialized with zeros. We encode a caption using a GRU. Given a caption, the hidden state of the 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, specially for caption with many words and a complex parse tree. Thus, we enrich this representation by utilizing a parser [18] 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 caption *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 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 example, 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. We use an arbitrary order but fixed for all RCGs, to create the sequences. 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 $\tilde{\mathbf{x}} \in \mathbb{R}^D$.

4 Training Details and Optimization

The MN-GMN is implemented in TensorFlow framework. We use a library from `https://github.com/deepmind/graph_nets` to implement the GNs. To minimize the loss, we apply 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. During training, all parameters are tuned except for the weights of the CNN and RGC detector components to avoid overfitting. We set the γ to 0.5 in our experiments. We choose correct answers that appear more than 8 times. This makes $Y = 3,110$ candidate answers. We use the standard metric [2]: an answer is correct if at least 3 people agree. The output dimension of the MCB is set to 512. The dimension of the hidden layer in both RGC and question GRUs is set to 512. Also, we set $P, Q = 14$ and $D = 512$. Our model takes around 6 hours to train on two NVIDIA Titan X GPUs. Following [20], to exploit prior linguistic information about the candidate answers, the GloVe embeddings of the answer words are used to initialize the rows of the $\tilde{\mathbf{W}}$ in the answer module. 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 $\mathbf{W}$.

5 Results on Visual7W and CLEVR

We also experiment on Visual7W [26] dataset. The Visual7W dataset includes $47,300$ images which occur in both Visual Genome and MS-COCO datasets. 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.

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.

We also experiment on CLEVR dataset [8]. 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. 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 the CLEVR images do not have rich textual information.

Our results on Visual7W are reported in Table S1. 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 S2 are competitive with the state-of-the-art PROGRAM-GEN. We emphasize that, unlike PROGRAM-GEN, our algorithm does not exploit supervision from functional programming.

Model	What	Where	When	Who	Why	How	Avg
Human [26]	96.5	95.7	94.4	96.5	92.7	94.2	95.7
LSTM-ATT [26]	51.5	57.0	75.0	59.5	55.5	49.8	54.3
Concat+ATT [5]	47.8	56.9	74.1	62.3	52.7	51.2	52.8
MCB+ATT [5]	60.3	70.4	79.5	69.2	58.2	51.1	62.2
MAN [15]	62.2	68.9	76.8	66.4	57.8	52.9	62.8
MLP [7]	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 S1: Accuracy Percentage on Visual7W.

Model	All	Exist	Count	Cmp-Int	Q-At	Cmp-At
HUMAN [8]	92.6	96.6	86.7	86.5	95.0	96.0
Q-TYPE MODE [8]	41.8	50.2	34.6	51.0	36.0	51.3
LSTM [8]	46.8	61.1	41.7	69.8	36.8	51.3
CNN+BOW [8]	48.4	59.5	38.9	51.1	48.3	51.8
CNN+LSTM [8]	52.3	65.2	43.7	67.1	49.3	53.0
CNN+LSTM+MCB [8]	51.4	63.4	42.1	66.4	49.0	51.0
CNN+LSTM+SA [8]	68.5	71.1	52.2	73.5	85.3	52.3
N2NMN [6]	83.3	85.7	68.5	85.0	90.0	88.8
CNN+LSTM+RN [17]	95.5	97.8	90.1	93.6	97.9	97.1
PROGRAM-GEN [9]	**96.9**	97.1	**92.7**	**98.7**	**98.2**	**98.9**
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 S2: Accuracy Percentage on CLEVR. The N-GMN$^+$ model only uses the visual GN and the external spatial memory components (no textual GN).