LRTA: A Transparent Neural-Symbolic Reasoning Framework with Modular Supervision for Visual Question Answering

Weixin Liang
Stanford University
wxliang@stanford.edu

Feiyang Niu
Amazon Alexa AI
nfeiyan@amazon

Aishwarya Reganti
Amazon Alexa AI
areganti@amazon.com

Govind Thattai
Amazon Alexa AI
thattg@amazon.com

Gokhan Tur
Amazon Alexa AI
gokhatur@amazon.com

Abstract

The predominant approach to visual question answering (VQA) relies on encoding the image and question with a "black-box" neural encoder and decoding a single token as the answer like "yes" or "no". Despite this approach's strong quantitative results, it struggles to come up with intuitive, human-readable forms of justification for the prediction process. To address this insufficiency, we reformulate VQA as a full answer generation task, which requires the model to justify its predictions in natural language. We propose LRTA [Look, Read, Think, Answer], a transparent neural-symbolic reasoning framework for visual question answering that solves the problem step-by-step like humans and provides human-readable form of justification at each step. Specifically, LRTA learns to first convert an image into a scene graph and parse a question into multiple reasoning instructions. It then executes the reasoning instructions one at a time by traversing the scene graph using a recurrent neural-symbolic execution module. Finally, it generates a full answer to the given question with natural language justifications. Our experiments on GQA dataset show that LRTA outperforms the state-of-the-art model by a large margin (43.1% v.s. 28.0%) on the full answer generation task. We also create a perturbed GQA test set by removing linguistic cues (attributes and relations) in the questions for analyzing whether a model is having a smart guess with superficial data correlations. We show that LRTA makes a step towards truly understanding the question while the state-of-the-art model tends to learn superficial correlations from the training data.

1 Introduction

A long desired goal for AI systems is to play an important and collaborative role in our everyday lives [29, 31]. Currently, the predominant approach to visual question answering (VQA) relies on encoding the image and question with a black-box transformer encoder [36, 32]. These works carry out complex computation behind the scenes but only yield a single token as prediction output (e.g., "yes", "no"). Consequently, they struggle to provide an intuitive and human readable form of justification consistent with their predictions. In addition, recent study has further demonstrated some unsettling behaviours of those models: they tend to ignore important question terms [33], look at wrong image regions [10], or undesirably adhere to superficial or even potentially misleading statistical associations [1].

4th Knowledge Representation and Reasoning Meets Machine Learning Workshop (KR2ML 2020), at NeurIPS.

To address this insufficiency, we reformulate VQA as a full answer generation task rather than a classification one, i.e. a single token answer. The reformulated VQA task requires the model to generate a full answer with natural language justification. We find that the state-of-the-art model answers a significant portion of the questions correctly for the wrong reasons. To learn the correct problem solving process, We propose LRTA (Look Read Think Answer), a transparent neural-symbolic reasoning framework that solves the problem step-by-step mimicking humans. A human would first (1) look at the image, (2) read the question, (3) think with multi-hop visual reasoning, and finally (4) answer the question. Following this intuition, LRTA deploys four neural modules, each mimicking one problem solving step that humans would take: A scene graph generation module first converts an image into a scene graph; A semantic parsing module parses each question into multiple reasoning instructions; A neural execution module interprets reason instructions one at a time by traversing the scene graph in a recurrent manner and; A natural language generation module generates a full answer containing natural language explanations. The four modules are connected through hidden states rather than explicit outputs. Therefore, the whole framework can be trained end-to-end, from pixels to answers. In addition, since LRTA also produces human-readable output from individual modules during testing, we can easily locate the error by checking the modular output. Our experiments on GQA dataset show that LRTA outperforms the state-of-the-art model by a large margin (43.1% v.s. 28.0%) on the full answer generation task. Our perturbation analyses by removing relation linguistic cues from questions confirm that LRTA makes a step towards truly understanding the question rather than having a smart guess with superficial data correlations. We discuss related work in Appendix A.

To summarize, the main contributions of our paper are three-fold: 1) We formulate VQA as a full answer generation problem (instead of short answer classification) to improve explainability and discourage superficial guess for answering the questions. 2) We propose LRTA, an end-to-end trainable, modular VQA framework facilitating explainability and enhanced error analysis. 3) We create a perturbed GQA test set that provides an efficient way to peak into a model's reasoning capability and validate our approach on the perturbed dataset. The dataset is available for future research - `https://github.com/Aishwarya-NR/LRTA_Perturbed_Dataset`

2 LRTA: Look, Read, Think and Answer

Look: Scene Graph Generation Given an image I, its corresponding scene graph represents the objects in the image (e.g., girl, hamburger) as nodes and the objects' pairwise relationships (e.g., holding) as edges. The first step of scene graph generation is object detection. We use DETR [7] as the object detection backbone since it removes the need for for hand-designed components like non-maximum suppression. DETR [7] feeds the image feature from ResNet50 [14] into a non-autoregressive transformer model, yielding an orderless set of N object vectors $[\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_N]$. Each object vector represents one detected object in the image. Then, for each object vector, DETR uses an object vector decoder (feed-forward network) to predict the corresponding object class

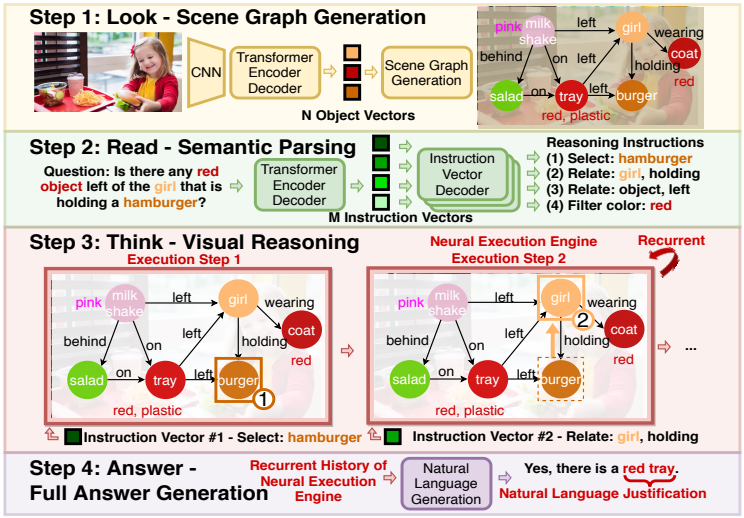

Figure 1: LRTA's four-step workflow (Look, Read, Think, Answer): (1) Convert the image into a scene graph (2) Parse the question into multiple reasoning instructions (3) Executes each instruction step-by-step using a recurrent neural execution engine (4) Generates the full answer with natural language justification.

(e.g., girl), and the bounding box in a multi-task manner. Since the set prediction of N object vectors is order-less, DETR calculates the set prediction loss by first computing an optimal matching between predicted and ground truth objects, and then sum the loss from each object vector. N is fixed to 100 and DETR creates a special class label "`no object`", to represent that the object vector does not represent any object in the image. The object detection backbone learns object classes and bounding boxes, but does not learn object attributes, and the objects' pairwise relationships. We augment the object vector decoder with an additional object attributes predictor. For each attribute meta-concept (e.g., color), we create a classifier to predict the possible attribute values (e.g., red, pink). To predict the relationships, we consider all $N(N-1)$ possible pairs of object vectors,$[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{N(N-1)}]$. The relation encoder transforms each object vector pair to an edge vector through feed-forward and normalization layers as in (1). We then feed each edge vector to the relation decoder to classify its relationship label. Both object attributes and inter-object relationships are supervised in a multi-task manner. To handle the object vector pair that does not have any relationship, we use the "`no relation`" relationship label. We construct the scene graph represented by N object vectors and $N(N-1)$ edge vectors instead of the symbolic outputs, and pass it to downstream modules.

$$\boldsymbol{e}_{i,j} = \text{LayerNorm}(\text{FeedForward}(\boldsymbol{o}_i \oplus \boldsymbol{o}_j)) \tag{1}$$

Read: Semantic Parsing The semantic parser works as a "compiler" that translates the question tokens $(q_1, q_2, \ldots, q_Q)$ into an neural executable program, which consists of multiple instruction vectors. We adopt a hierarchical sequence generation design: a transformer model [39] first parses the question into a sequence of M instruction vectors, $[\boldsymbol{i}_1, \boldsymbol{i}_2, \ldots, \boldsymbol{i}_M]$. The i^{th} instruction vector will correspond exactly to the i^{th} execution step in the neural execution engine. To enable human to understand the semantics of the instruction vectors, we further translate each instruction vector to human-readable text using a transformer-based instruction vector decoder. We pass the M instruction vectors rather than the human-readable text to the neural execution module.

$$[\boldsymbol{i}_1, \boldsymbol{i}_2, \ldots, \boldsymbol{i}_M] = \text{Transformer}(q_1, q_2, \ldots, q_Q) \tag{2}$$

Think: Visual Reasoning with Neural Execution Engine The neural execution engine works in a recurrent manner: At the m^{th} time step, the neural execution engine takes the m^{th} instruction vector ($\boldsymbol{i}_m$) and outputs the scene graph traversal result. Similar to recurrent neural networks, a history vector that summarizes the graph traversal states of all nodes in the current time-step would be passed to the next time-step. The neural execution engine operates with graph neural network. Graph neural network generalizes the convolution operator to graphs using the neighborhood aggregation scheme [6, 42]. The key intuition is that each node aggregates feature vectors of its immediate neighbors to compute its new feature vector as the input for the following neural layers. Specifically, at m^th time step given a node as the central node, we first obtain the feature vector of each neighbor ($\boldsymbol{f}_k^m$) through a feed-forward network with the following inputs: the object vector of the neighbor ($\boldsymbol{o}_k$) in the scene graph, the edge vector between the neighbor node and the central node ($\boldsymbol{e}_{k,\text{central}}$) in the scene graph, the $(m-1)^{th}$ history vector ($\boldsymbol{h}_{m-1}$), and the m^{th} instruction vector ($\boldsymbol{i}_m$).

$$\boldsymbol{f}_k^m = \text{FeedForward}(\boldsymbol{o}_k \oplus \boldsymbol{e}_{k,\text{central}} \oplus \boldsymbol{h}_{m-1} \oplus \boldsymbol{i}_m) \tag{3}$$

We then average each neighbor's feature vector as the context vector of the central node ($\boldsymbol{c}_{\text{central}}^{\boldsymbol{m}}$).

$$\boldsymbol{c}_{\text{central}}^m = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{f}_k^m \tag{4}$$

Next, we perform node classification for the central node, where an "`1`" means that the corresponding node should be traversed at the m^{th} time step and "`0`" otherwise. The inputs of the node classifier are: the object vector of the central node in the scene graph, the context vector of the central node, and the m^{th} instruction vector.

$$s_{\text{central}}^m = \text{Softmax}(\text{FeedForward}(\boldsymbol{o}_{\text{central}} \oplus \boldsymbol{c}_{\text{central}}^m \oplus \boldsymbol{i}_m)) \tag{5}$$

where s_{central}^m is the classification confidence score of central node at m^{th} time step. The node classification results of all nodes constitute a bitmap as the scene graph traversal result. We calculate the weighted average of all object vectors as the history vector ($\boldsymbol{h}_m$), where the weight is each node's classification confidence score.

$$\boldsymbol{h}_m = \sum_{i}^{N} s_i^m \cdot \boldsymbol{o}_i \tag{6}$$

Answer: Full Answer Generation VQA is commonly formulated as a classification problem where the model learns to answers with only one token (e.g., "yes" or "no"). We advocate to formulate VQA as a natural language generation problem, where the model learns to answer the question in a full sentence with justifications. To do this, LRTA deploys a transformer model that takes in the neural execution's history vectors from all time-steps, and generates the full answer tokens $(a_1, a_2, \ldots, a_A)$.

$$(a_1, a_2, \ldots, a_A) = \text{Transformer}(\boldsymbol{h}_1 \oplus \boldsymbol{h}_2 \oplus \cdots \oplus \boldsymbol{h}_M) \tag{7}$$

End-to-End Training: From Pixels to Answers We connect four modules through hidden states rather than symbolic outputs [29]. Therefore, the whole framework could be trained in an end-to-end manner, from pixels to answers. The training loss is simply the sum of losses from all four modules. Each neural module receives supervision not only from the module's own loss, but also from the gradient signals back-propagated by downstream modules. We start from the pre-trained weights of DETR for the object detection backbone and all other neural modules are randomly initialized.

3 Experiments

We evaluate LRTA on the GQA dataset [20], which contains 1.5M questions over 110K images. The details of end-to-end experiment setup are reported in the Appendix.

Design Validation with Ground Truth Scene Graph Since LRTA deviates from the predominant black-box encoder approach a lot, we first validate the design of LRTA by using a visual oracle for step 1 (ground truth scene graphs). As shown in Table 2 in Appendix B, LRTA with visual oracle achieves a surprisingly high accuracy on both short answers (93.1%) and full answers (85.99%) on the validation set. This shows the great potential and expressivity of LRTA for visual question answering. In addition, if we remove the attributes or the relations in the test data, the performance drops a lot. This shows that scene graph generation beyond object detection is a crucial step and thus we call for more attention to scene graphs for the visual question answering community.

End-to-End Training Experiments Next we train the model end-to-end, from pixels to answers. As shown in Table 1 in Appendix B, LRTA significantly outperforms LXMERT in full answer generation (43% v.s. 28%) and achieves comparable accuracy on short answers (54.48% v.s. 56.2%). Next, we conduct perturbation study to show that the performance of LXMERT comes more from superficial data correlations while LRTA makes a step towards truly understanding the question.

4 Conclusion

We present LRTA, a transparent neural-symbolic reasoning framework for visual question answering, that incorporates [look, read, think and answer] steps to provide a human-readable form of justification at each step. The modular design of our methodology enables the whole framework to be trainable end-to-end. Our experiments on GQA dataset show that LRTA achieves high accuracy on full answer generation task, outperforming the state-of-the-art LXMERT results by a noticeable 15% absolute margin. In addition, LRTA performance drops significantly more than LXMERT, when object attributes and relationships are masked, hence indicating that LRTA makes a step forward, towards truly understanding the question, rather than making a smart guess based on superficial data correlations. In the validation study, we have shown that when provided with an oracle scene graph, LRTA is able to achieve a high accuracy on both short answers (93.1%) and full answers (85.99%), nearing the theoretical bound 96% on short answers [2]. These observations indicate that better scene graph prediction methods offer a great potential in further improving LRTA performance on both short-answer and full-answer tasks.

Acknowledgement We would like to thank Robinson Piramuthu, Dilek Hakkani-Tur, Arindam Mandal, Yanbang Wang and the anonymous reviewers for their insightful feedback and discussions that have notably shaped this work.

References

[1] A. Agrawal, D. Batra, and D. Parikh. Analyzing the behavior of visual question answering models. In *EMNLP*, pages 1955–1960. The Association for Computational Linguistics, 2016.

[2] S. Amizadeh, H. Palangi, O. Polozov, Y. Huang, and K. Koishida. Neuro-symbolic visual reasoning: Disentangling "visual" from "reasoning". *CoRR*, abs/2006.11524, 2020.

[3] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, pages 6077–6086. IEEE Computer Society, 2018.

[4] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. In *HLT-NAACL*, pages 1545–1554. The Association for Computational Linguistics, 2016.

[5] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *CVPR*, pages 39–48. IEEE Computer Society, 2016.

[6] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[8] T. Chen, W. Yu, R. Chen, and L. Lin. Knowledge-embedded routing network for scene graph generation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6163–6171. Computer Vision Foundation / IEEE, 2019.

[9] W. Chen, Z. Gan, L. Li, Y. Cheng, W. Wang, and J. Liu. Meta module network for compositional visual reasoning. *CoRR*, abs/1910.03230, 2019.

[10] A. Das, H. Agrawal, L. Zitnick, D. Parikh, and D. Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions? In *EMNLP*, pages 932–937. The Association for Computational Linguistics, 2016.

[11] Z. Feng, W. Liang, D. Tao, L. Sun, A. Zeng, and M. Song. Cu-net: Component unmixing network for textile fiber identification. *Int. J. Comput. Vis.*, 127(10):1443–1454, 2019.

[12] A. Ghorbani, A. Abid, and J. Y. Zou. Interpretation of neural networks is fragile. In *AAAI*, pages 3681–3688. AAAI Press, 2019.

[13] J. Gu, S. R. Joty, J. Cai, H. Zhao, X. Yang, and G. Wang. Unpaired image captioning via scene graph alignments. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 10322–10331. IEEE, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[15] M. Honnibal and I. Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 2017.

[16] R. Hu, J. Andreas, T. Darrell, and K. Saenko. Explainable neural computation via stack neural module networks. In *ECCV (7)*, volume 11211 of *Lecture Notes in Computer Science*, pages 55–71. Springer, 2018.

[17] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *ICCV*, pages 804–813. IEEE Computer Society, 2017.

[18] R. Hu, A. Rohrbach, T. Darrell, and K. Saenko. Language-conditioned graph networks for relational reasoning. In *ICCV*, pages 10293–10302. IEEE, 2019.

[19] D. A. Hudson and C. D. Manning. Compositional attention networks for machine reasoning. *CoRR*, abs/1803.03067, 2018.

[20] D. A. Hudson and C. D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, pages 6700–6709. Computer Vision Foundation / IEEE, 2019.

[21] D. A. Hudson and C. D. Manning. Learning by abstraction: The neural state machine. In *NeurIPS*, pages 5901–5914, 2019.

[22] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. B. Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, pages 3008–3017. IEEE Computer Society, 2017.

[23] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and F. Li. Image retrieval using scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3668–3678. IEEE Computer Society, 2015.

[24] B. Knyazev, H. de Vries, C. Cangea, G. W. Taylor, A. C. Courville, and E. Belilovsky. Graph density-aware losses for novel compositions in scene graph generation. *CoRR*, abs/2005.08230, 2020.

[25] R. Koner, P. Sinhamahapatra, and V. Tresp. Relation transformer network. *CoRR*, abs/2004.06193, 2020.

[26] L. Li, Z. Gan, Y. Cheng, and J. Liu. Relation-aware graph attention network for visual question answering. In *ICCV*, pages 10312–10321. IEEE, 2019.

[27] Q. Li, J. Fu, D. Yu, T. Mei, and J. Luo. Tell-and-answer: Towards explainable visual question answering using attributes and captions. In *EMNLP*, pages 1338–1346. Association for Computational Linguistics, 2018.

[28] Q. Li, Q. Tao, S. R. Joty, J. Cai, and J. Luo. VQA-E: explaining, elaborating, and enhancing your answers for visual questions. In *ECCV (7)*, volume 11211 of *Lecture Notes in Computer Science*, pages 570–586. Springer, 2018.

[29] W. Liang, Y. Tian, C. Chen, and Z. Yu. MOSS: end-to-end dialog system framework with modular supervision. In *AAAI*, pages 8327–8335. AAAI Press, 2020.

[30] W. Liang, J. Zou, and Z. Yu. ALICE: active learning with contrastive natural language explanations. In *EMNLP (1)*, pages 4380–4391. Association for Computational Linguistics, 2020.

[31] W. Liang, J. Zou, and Z. Yu. Beyond user self-reported likert scale ratings: A comparison model for automatic dialog evaluation. In *ACL*, pages 1363–1374. Association for Computational Linguistics, 2020.

[32] J. Lu, V. Goswami, M. Rohrbach, D. Parikh, and S. Lee. 12-in-1: Multi-task vision and language representation learning. In *CVPR*, pages 10434–10443. IEEE, 2020.

[33] P. K. Mudrakarta, A. Taly, M. Sundararajan, and K. Dhamdhere. Did the model understand the question? In *ACL (1)*, pages 1896–1906. Association for Computational Linguistics, 2018.

[34] J. Shi, H. Zhang, and J. Li. Explainable and explicit visual reasoning over scene graphs. In *CVPR*, pages 8376–8384. Computer Vision Foundation / IEEE, 2019.

[35] A. Shin, Y. Ushiku, and T. Harada. The color of the cat is gray: 1 million full-sentences visual question answering (FSVQA). *CoRR*, abs/1609.06657, 2016.

[36] H. Tan and M. Bansal. LXMERT: learning cross-modality encoder representations from transformers. In *EMNLP/IJCNLP (1)*, pages 5099–5110. Association for Computational Linguistics, 2019.

[37] K. Tang, Y. Niu, J. Huang, J. Shi, and H. Zhang. Unbiased scene graph generation from biased training. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 3713–3722. IEEE, 2020.

[38] D. Teney, L. Liu, and A. van den Hengel. Graph-structured representations for visual question answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3233–3241. IEEE Computer Society, 2017.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[40] R. Vedantam, K. Desai, S. Lee, M. Rohrbach, D. Batra, and D. Parikh. Probabilistic neural symbolic models for interpretable visual question answering. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6428–6437. PMLR, 2019.

[41] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3097–3106. IEEE Computer Society, 2017.

[42] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*. OpenReview.net, 2019.

[43] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh. Graph R-CNN for scene graph generation. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, volume 11205 of *Lecture Notes in Computer Science*, pages 690–706. Springer, 2018.

[44] X. Yang, K. Tang, H. Zhang, and J. Cai. Auto-encoding scene graphs for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10685–10694. Computer Vision Foundation / IEEE, 2019.

[45] T. Yao, Y. Pan, Y. Li, and T. Mei. Exploring visual relationship for image captioning. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, volume 11218 of *Lecture Notes in Computer Science*, pages 711–727. Springer, 2018.

[46] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi. Neural motifs: Scene graph parsing with global context. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5831–5840. IEEE Computer Society, 2018.

[47] C. Zhang, W. Chao, and D. Xuan. An empirical study on leveraging scene graphs for visual question answering. In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, page 288. BMVA Press, 2019.

Appendix A: Related Work

Explainable VQA Existing black-box visual question answering models attempt to directly map inputs to outputs using black-box architectures without explicitly modeling the underlying reasoning processes. To mitigate the black-box nature of these models, several model interpretation techniques have been developed to improve model transparency and explainability [10, 12, 28, 27, 30, 11]. One of the most popular approaches is attention map visualization, which highlights the important image regions for answering the question. However, recent study shows that the visualized attention regions correlate poorly with humans [10, 12]. Another line of research propose to generate natural language justifications [28, 27] along with the short answer. Similar to the GQA dataset [20], the FSVQA dataset [35] provides full answer annotations for VQA, but the full answer generation task remains unexplored. To the best of our knowledge, LRTA is the first full answer generation framework for GQA [20], and possibly for the visual quesiton answering task. However, this approach still do not reveal the model's step-by-step problem solving process. This approach makes a step towards more explainable VQA models, but still does not reveal the internal problem solving process of the model.

Neural Module Networks Another active line of research on visual question answering explores neural module networks (NMN) [5, 4, 17, 22, 16, 34, 40], which composes the model's neural architecture on the fly based on the given question. Instead of training a model with static neural architecture, they hand-defined a set of small neural networks (i.e., neural modules), each dedicated for a specific kind of logical operation. Given a question, NMN first uses a semantic parser to parse the question into a series of logical operations (similar to our reasoning instructions). Then, given a sereis of logical operations, NMN dynamically layouts the small neural networks. For example, to answer "What color is the metal cube?", NMN dynamically composes four modules: (1) a module that finds things made of metal, (2) a module that localizes cubes, (3) a module that determines the color of objects. However, NMN models are challenging to optimize by its nature. Therefore, its success is mostly restricted to the synthetic CLEVR dataset [22] and how to extend NMN to real-world datasets is still an open research problem [9]. Different from NMNs, our framework is conceptually simple and could be easily trained in an end-to-end manner, from pixels to answers.

Our work is also related to the sporadic attempts in scene graph based VQA. Hudson et al. [21] propose neural state machines that simulates the computation of an automaton on probabilistic scene graphs. [18, 26] models the interaction between objects using graph attention mechanism. Different from their work, our work is end-to-end trainable, from pixels to answers, and transparently provides the execution result of each step.

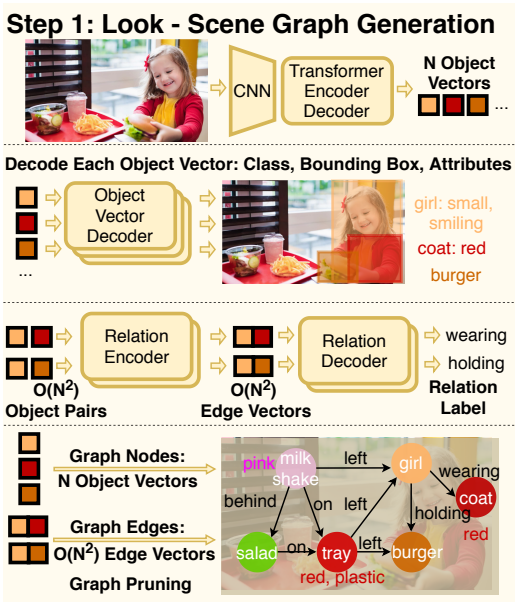

Figure 2: LRTA's Scene Graph Generation Workflow.

Scene Graph Generation Scene graph generation (SGG) [41] is a visual detection task that aims to predict objects and their relations from an image. In recent years, it has drawn increasing attentions that greatly advance the interface of vision and language. By extracting the concepts and contextual relations from pixels, scene graph provides an intuitive high-level summary of a raw image and facilitates downstream reasoning tasks such as image captioning [13, 44, 45], VQA [38, 20, 21, 47] or image retrieval [23]. Previous works [41, 21, 46, 43, 8] have predominantly relied on Faster R-CNN based detectors that typically generate a potentially large set of bounding box proposals whose contextualized representation is then fed through a subsequent sequence to sequence network (e.g. LSTM [46] or Transformers [25]) to predict object labels and their relations. A key disadvantage of those detectors is that they normally need many hand-designed components like a non-maximum suppression procedure or anchor generation. In that

Model	Full Acc	Short Acc
Prior [20]	-	28.93%
Human [20]	-	89.30%
Bottom-up [3]	-	49.74%
MAC [19]	-	54.06%
LXMERT [36]	28.00%	**56.20%**
LRTA	**43.10%**	**54.48%**

Table 1: End-to-end training experiment on testdev set

Model	Full Acc	Short Acc
LRTA trained w/ visual oracle		
Evaluated w/o attributes	67.79%	78.21%
Evaluated w/o relations	67.95%	75.47%
Evaluated w/o attributes & relations	50.15%	61.15%
Evaluated w/ visual oracle	**85.99%**	**93.10%**
LRTA trained w/ reading oracle		
Evaluated w/ reading oracle	55.45%	64.36%

Table 2: Validation study on valid set

Model	**Short Acc Drop** (from → to)
VB & PRPN masked	
LXMERT [36]	19.43% (56.20% → 36.77%)
LRTA	**26.20%** (54.48% → 28.28%)
Attributes masked	
LXMERT [36]	9.41% (56.20% → 46.79%)
LRTA	**21.03%** (54.48% → 33.45%)

Table 3: Perturbation analysis on testdev set.

Model	**Short Acc Drop** (from → to)
VB & PRPN masked	
LXMERT [36]	4.40% (64.30% → 59.90%)
LRTA	**16.67%** (62.79% → 46.12%)
Attributes masked	
LXMERT [36]	9.24% (64.30% → 55.06%)
LRTA	**16.57%** (62.79% → 46.22%)

Table 4: Perturbation analysis on valid set.

regard, we adopted DETR [7], a recently proposed method that streamlines the detection process and makes our whole pipeline end-to-end trainable. Despite the growing research interest, SGG remains as a challenging task largely due to the training bias, e.g. `<human, on, beach>` appears more frequently than `<human, lay on, beach>`. As such, dummy models predicting solely based on frequency is embarrassingly not far from the state-of-the-art as reported in [37, 24]. An unbiased SGG method [37] was recently proposed that sheds some promising light on the data bias issue. Thanks to the modular design of LRTA, we can easily incorporate such unbiased SGG into our pipeline.

Appendix B: Experiments Results

Setup We evaluate LRTA on the GQA dataset. To the best of our knowledge, LRTA is the first full answer generation model on GQA [20]. We use the standard dataset split. During training, we use the ground truth for scene graphs, reasoning instructions, scene graph traversal results for each step, and full answers. During testing, we only use images and questions. We add transformer decoder to the state-of-the-art short answer model LXMERT [36] as a full answer generation baseline. We report accuracy on both short answers and full answers for both LXMERT and LRTA. Full answers are evaluated with string match accuracy since the full answers follows pre-defined templates. We delay improving the metric as future work.

Perturbed GQA Dataset and Adversarial experiments In order to probe whether a model has effectively leveraged linguistic cues, we design a perturbation study by systematically removing the cues such as attributes and relationships from the questions and evaluate if the model's performance changes significantly. Specifically, the better a model understands the language cues, the more drop we expect the model's performance on the cues stripped questions. We use a comprehensive list of attributes obtained by [9] and mask them using a predefined mask token. For effectively masking relationships, we use Spacy POS-Tagger [15] and mask *verbs* (VB) and *prepositions* (PRPN) from the question. We evaluate LXMERT and LRTA for short answer accuracy and report the results on the testdev set and public valid set in Table 3 and Table 4, respectively. We can deduce from the results that LRTA results drop more significantly than LXMERT in both masking scenarios on both sets. On testdev set, we see that for relationships LRTA performance drops by **26.20%** as compared to 19.43% drop in LXMERT, while for attributes, the margin is more significant at **21.03%** and 9.41% respectively, thus providing us a strong convergent evidence for our hypothesis that LRTA truly takes

a leap forward while trying to systematically understand the question and its components rather than using peripheral correlations. On the valid set, we notice a similar pattern to the testdev set. LRTA has a higher drop in performance when the attributes/relationships are masked as compared to LXMERT [36].