It's not about the Journey; It's about the Destination: Following Soft Paths under Question-Guidance for Visual Reasoning Supplemental Material

Monica Haurilet Alina Roitberg Rainer Stiefelhagen
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
{haurilet, alina.roitberg, rainer.stiefelhagen}@kit.edu

1. Configuration Details

In this section, we provide more details for implementing our model, as well as further learning procedure specifics.

Question-based Guide. For better comparability on the AI2D dataset, we employ the same input representation as related work [8, 9], *i.e.* GloVe [13] features pre-trained on 6B tokens from Wikipedia. As previous work [8, 9] uses LSTMs for embedding the question sentence, we also use a single-layered LSTM with 256 hidden units for our question-based guide. The final representation of the question is set as the last hidden state of the LSTM.

For CLEVR and COG, we experienced LSTM convergence instabilities linked to significantly longer sentences (over 15 tokens). This LSTM-related issue was previously reported by others [5]. However, various works show strong potential of using 1-D convolutions instead of recurrent neural networks. Such convolution-based models oftentimes exceed the accuracy of LSTMs, while having a more stable learning procedure [1, 3, 2]. We therefore opt to embed the questions using 1-D convolutions with an attention module.

The question-based guide embeds the question words represented as one-hot vectors using a 1D convolutional neural network with self-attention. We use six convolution layers (with 32 output filters of size 3, stride 1) each with zero padding and ReLU activation. In case of COG and CLEVR, we do not share weights between the guides for each question type, however the structure of each guide is identical. Finally, we obtain the attention module with a further 1D convolution and a sigmoid non-linearity. Even though softmax is the widespread way to normalize an attention module, sigmoid has the benefit of weighting individual elements (*i.e.* words) independently of their total amount. Our goal is that *e.g.* the word 'sphere' is weighted in the same way in both 'How many *spheres* are there?' and 'How many *green spheres* are there?'. This would not be the case for softmax, as softmax would give both 'sphere' and 'green' a high weight and thus, sphere would automatically have a smaller weight as it has to share the amount with green after the normalization. In the final layer, the number of hidden units corresponds to the maximal path length T (*i.e.* we get a different attention map for each step t in the path). The final representation of traversal directions for each $t \in \{1, \dots, T\}$ is therefore a weighted sum of the words in the question based on the weights obtained from these attention maps.

Visual Graph. As previously mentioned, we use GloVe features for representing the words of each question in AI2D. The same word embedding model is then used to represent each node in the graph, while the edges are set to 1 if the corresponding node pair is connected and 0 otherwise (as done in previous work on AI2D [9]). For CLEVR and COG, we employ one-hot encodings for the nodes, where we specify the shape, color, material, size and frame number (in case of the COG video dataset) for each object instance. The edges are defined as a directed relation representation of the source-target node pair, which consists of location information, whether the linked nodes share the same properties or not, as well as the target node embedding. Including the target node in the edge representation is very important, as we often do not only have to find nodes that are *e.g.* left of an object, but we have to filter specific targets *e.g.* *green* objects left of the source node.

In COG, the graphs for each image are included in the dataset for both training and testing. This is, however, not the case for CLEVR where the graphs are only available for training. For the test set, we built the graph by making use of an off-the-shelf object detector – an SSD [12] with a ResNet152 [6] backbone provided by [7]. The produced graphs have an overall accuracy of over 98% on the validation set, where the most common mistake is due to strongly occluded objects in the image. We did not find any other types of errors *e.g.* missing edges or including too many nodes in the graph. Such 'imperfect' graphs however cause only a small drop in performance of around 0.4% on the validation set in the VQA task. We make the predicted graphs publicly available to foster further research in graph-based VQA on the CLEVR dataset.

Graph Traveler. The graph-traveler uses the direction embeddings produced by the guide to traverse the graph in search of the destination nodes. To obtain the confidence $\tau^t(n)$ for each time step t and node n being in the path, we concatenate each node for $t = 1$ and edge for $t > 1$ with the direction embeddings, followed by two fully connected layers. The first layer has a size of 256 and ReLU activation, while the second layer is a single hidden unit representing the confidence of the starting nodes and transition probabilities, respectively. As we constrain $\tau^t(n)$ to be in the interval $[0, 1]$, we use either sigmoid (for counting and existence tasks) or softmax (for other tasks) for confidence normalization. The nonlinearities are both applied on either the starting nodes *i.e.* $\tau^1(n)$ or on each edge between each node pair. In case of sigmoid normalization, we replace the sum operation of Equation 5 with the maximum function applied on the input edge confidences, in order to hold the premise of $\tau^t(n) \leq 1$.

Prediction Module. Visual reasoning datasets cover a variety of task formats (*e.g.* counting, diagram question-answering) and therefore differ greatly in their solution modes (*e.g.* multiple choice vs. free result format), which should be taken into account in the prediction module output. Since AI2D is a dataset in a multiple-choice setting, we need to input the multiple choice answers to the model. As an input to our prediction module, we concatenate each of the possible multiple-choice answers with the final representation of the question and of the destination node individually. This is followed by two fully connected layers, with the last layer containing a single hidden unit corresponding to the confidence of the current answer being the correct one.

COG and CLEVR are both open-ended datasets, and thus we obtain the final prediction by using the set of all possible answers in the test set *i.e.* the prediction module consists of a fully-connected layer with the number of hidden units equal to the number all of possible answers across all tasks. However, for COG we use two different streams in the same way as related work [15] for each answer type: pointing to an object in the graph and other type of answers in text format (*e.g. yes, circle, red*). In case of pointing questions, the prediction module consists of a fully connected layer with a single hidden unit for each node in the graph, which are normalized using softmax (*i.e.* sum over all nodes will equal to one). The other type of answers have either are yes/no or attribute-based ones. As we previously specified, for yes and no answers, we use sigmoid activation for the soft paths, while for the attribute-based questions we make use of softmax.

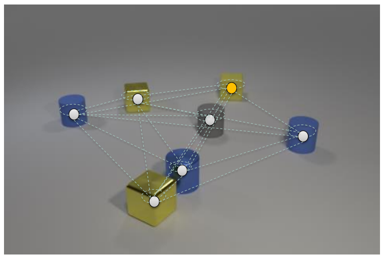
In case of CLEVR, we have five different types of questions, where we use sigmoid in all counting related tasks (*exist, count and compare numbers*) and softmax otherwise (*query attributes and compare attributes*). The prediction modules of softmax related tasks consist of a simple fully connected layer with the number of hidden units equal to the number of possible answers in the training set. This final layer is also using a softmax normalization over the answers present in the training set. In case of the sigmoid-based tasks, we make use of a greedy procedure (common in reinforcement learning) in combination with the cross entropy loss. For example, in case of *exist* question, we use the neuron with the (currently) highest activation of the nodes in the final step T in the loss as the prediction, while as the target we use 0 if the correct answer is no, otherwise we input a 1. During the test phase we predict a 'yes' if any of the nodes have a final confidence of over 0.5, otherwise the model outputs a 'no'. In case of *counting*, we use the cross entropy loss over the confidences in time step T as the prediction. For a ground truth answer k, we assign as the target a vector that has the same size as the number of nodes in the graph. In this vector we set a 1 if the particular node value is in the top-k highest confidence in the prediction (*i.e.* confidence in step T), otherwise the particular value in the target vector is set to 0. In the test phase, we count the number of activations that have a higher value than 0.5.

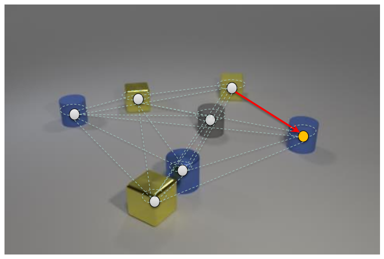
Optimization. Except for the weights of the pre-trained GloVe model in AI2D, all weights are initialized randomly following the Xavier initialization [4]. Biases are initialized with zeros. Network weights are optimized using ADAM [10] with an initial learning rate of 0.00025. For other parameters, we use default values of TensorFlow: 0.9 for the exponential decay rate for the first moment estimates β_1 and 0.999 for the second one β_2. The small initial learning rate is common for graph neural networks (*e.g.* [14]) as larger ones often cause convergence instabilities. The models are trained at most 30 epochs using early stopping with the validation set performance as an indicator. All models are implemented from scratch in TensorFlow.

Zero-Shot Setting. In our zero-shot setting, we train solely on one task *e.g. exist* questions and evaluate on a different task *e.g. counting* task. We make use of the different heads in the prediction module and switch them between the source and target task.

Computational Complexity. For each step, we have a computational complexity of $O(E)$ where E is the number of edges. Note that this is also the case for *e.g.* graph convolutions [11]. The training time for 30 epochs on a single GeForce GTX 1080 Ti is 1 hour for the AI2D and 20 hours for the CLEVR image datasets and 100 hours for the COG video dataset.

Best viewed in color.


Step 1 There is **a** tiny rubber thing **that is right of the matte cube**; are there any yellow cubes in front of it?


Step 2 There **is** a **tiny rubber thing** that is **right of the** matte cube; are there any yellow cubes in front of it**?**

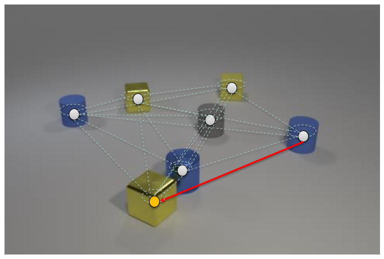
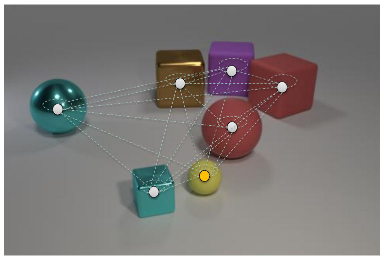
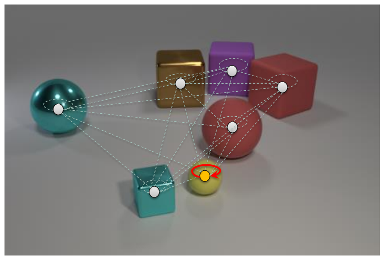

Step 3 There is a tiny rubber thing that is right of the matte cube; are there any yellow **cubes in front of it**? *Answer: yes* ✓

Figure 1: Existence question with a necessary reasoning chain length of three.


Step 1 Are there any small cyan objects on the **left** side **of the yellow object?**


Step 2 Are there any small cyan objects on the left side of **the** yellow object?

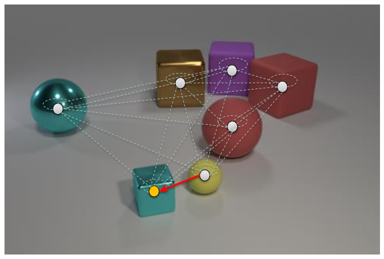
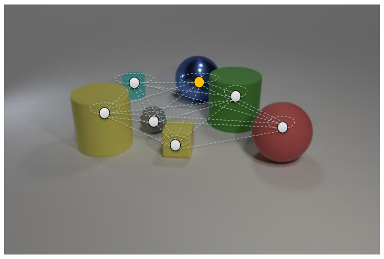
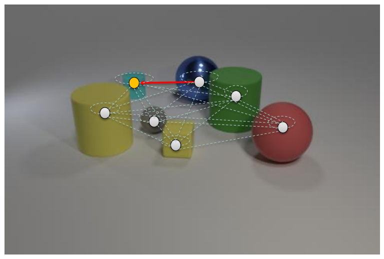

Step 3 Are there **any small cyan objects** on the **left** side **of** the yellow object? *Answer: yes* ✓

Figure 2: Existence question with a necessary reasoning chain length of two and a single found destination.


Step 1 Is there a big blue metal thing that is **behind the rubber object behind the blue shiny object?**


Step 2 Is there a big blue metal thing that is behind **the rubber object** behind the blue shiny object?

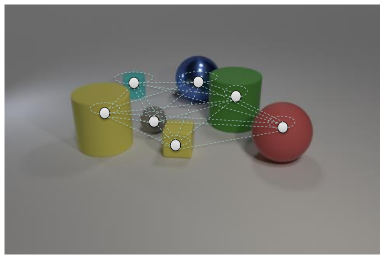
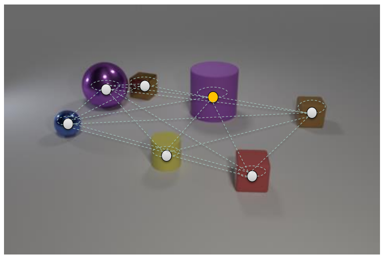
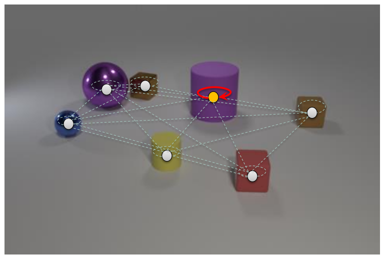

Step 3 Is there **a big blue metal thing** that is **behind** the rubber object behind the blue shiny object? *Answer: no* ✓

Figure 3: Example of question about the existence of an object where the correct answer is ‘no’.


Step 1 Are there any tiny objects in front **of the purple rubber object?**


Step 2 Are there any tiny objects in front of **the purple** rubber object?

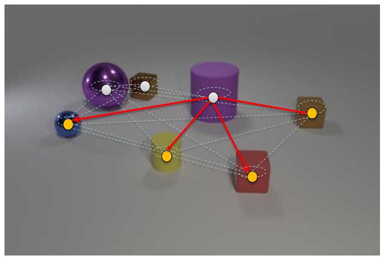
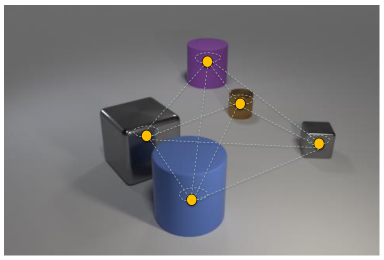
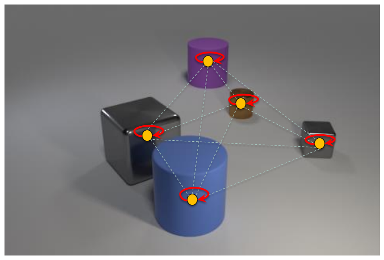

Step 3 Are there **any tiny objects in front of** the purple rubber object? *Answer: yes* ✓

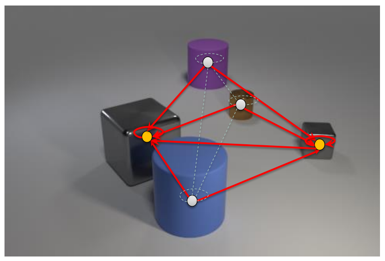
Figure 4: Question about existence with multiple found destinations.



Step 1 How many cubes are there?

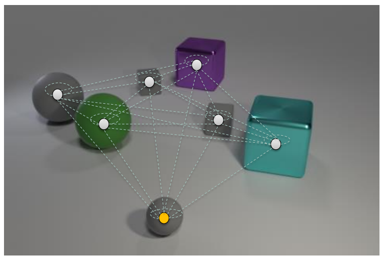


Step 2 How many **cubes are** there?

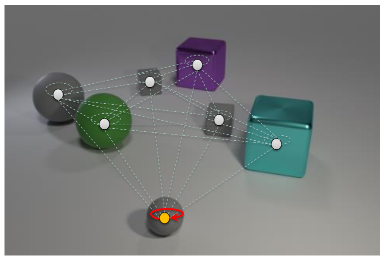


Step 3 How many **cubes are there**?
Answer: 2 ✓

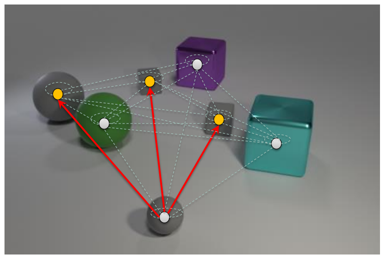
Figure 5: Counting task with a necessary reasoning chain length of one. The model handles the fixed maximum path length of $T = 3$ through self-loops and correctly finds two objects.



Step 1 What number of other things are the same color **as the small matte ball**?

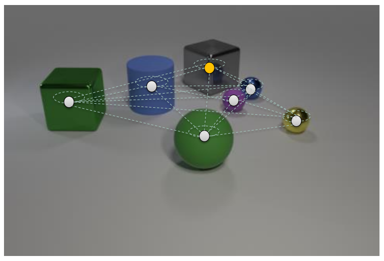


Step 2 What number of other **things are** the same **color as** the small matte ball?

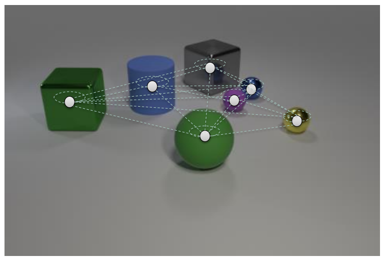


Step 3 What number of **other things are** the **same color as** the small matte ball?
Answer: 3 ✓

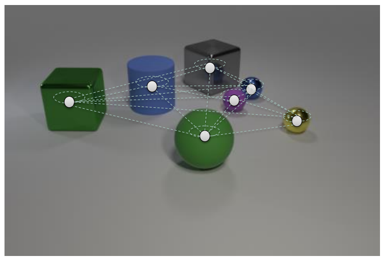
Figure 6: Counting question with a necessary path length of two. The model was able to find all three destinations.



Step 1 Are there any big green things that are in front **of the** large metal **object that is on the left side of the gray thing?**

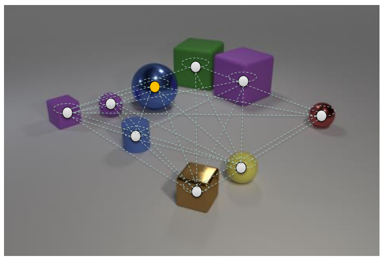


Step 2 Are there any big green things that are in front **of the large metal object** that is on the **left** side of the gray thing?

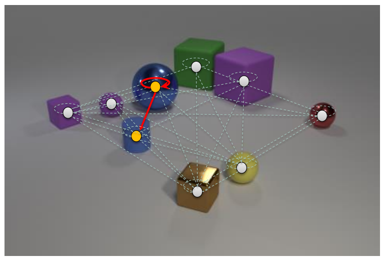


Step 3 Are there **any big green things** that are **in front of** the large metal object that is on the left side of the gray thing? Answer: no ✗

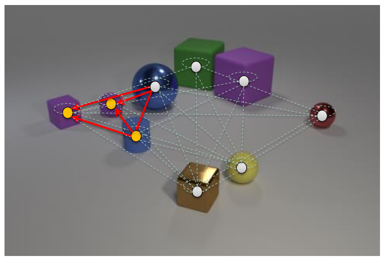
Figure 7: Example of an incorrectly answered question. Since the large metal object left of the starting node was not found in the second step (computed confidence of 0.3 below our threshold of 0.5), the model also did not find any destination nodes.



Step 1 How many matte things are on the **left side of the blue thing on the left** side of the large blue metal thing?



Step 2 How many **matte things** are on the **left** side of the blue thing on the left side of the large blue metal thing?



Step 3 How many **matte things** are **on** the **left** side of **the** blue thing on the left side of the large blue metal thing? Answer: 3 ✗

Figure 8: Due to an incorrect self-loop in the second step, presumably retained because the object satisfies one of the conditions of the next step (it is blue), the answer for the counting question was higher than required (correct answer is 2).

2. Additional Qualitative Results

We analyze the soft paths produced by our model in case of correct and incorrect answers. Figures 1-8 provide examples for maximum path length $T = 3$, with each column illustrating the state at step t. White circle markings depict the nodes with *low* confidence of being visited by the traveler in the current step t. Orange markings depict *high* probability nodes and the arrows mark a high transition confidence (> 0.5). Underneath each image, we highlight the question words which received high attention values (> 0.5) of the visual guide for the current path section.

Correct predictions. Figures 1-4 show a variety of the *existence* task queries, which were correctly handled. In the first example (Figure 1), the number of required reasoning steps corresponds to the maximum path length T and our model easily finds the path leading to the correct destination. The next example (Figure 2), on the other hand, only requires two reasoning steps. This is not an issue in our model for $T > 2$, as self-loops are permitted. The self-loop is present in the second time step on only a single node (the *yellow* sphere). This confirms, that the model leverages object attributes for traversal, as only edges with a *yellow* target have high transition confidence.

In case the query addresses a non-existent object (*i.e.* the correct answer is 'no'), our model does not have any high confidence destination nodes, as illustrated in the Figure 3. The traveler followed the correct path from the *blue shiny object* to the *rubber object behind* it. However, there are no other objects behind this rubber object, thus, there are no final high confidence destination nodes. In comparison, Figure 4 correctly found four destinations that are in front of the purple rubber object as multiple objects fit the query.

Figures 5 and 6 visualize the paths produced for *counting*. Both examples necessitate shorter reasoning chains than our maximal path length (*i.e.* 'How many cubes are there?' only requires a single reasoning step). The traveler handles this successfully by visiting some nodes more than once. Surprisingly, we observe different strategies for different lengths of the required reasoning chain. In case of the path of length 1 (Figure 5) the traveler visits all nodes twice and chooses the nodes of the type cube in the last step. For a reasoning chain with two steps (Figure 6) a single source node (the small matte ball) is selected in the first two steps.

Analysis of incorrect predictions. In Figures 7 and 8 we analyze queries which were not answered correctly. In the *existence* task example (Figure 7), the traveler was not able to find the large metal object on the left side of the source node, as it produced a confidence of 0.3 (our threshold was 0.5). Due to this deficit in an intermediate step, the traveler did not reach the correct destination node (the big green object) and is predicting the incorrect answer 'no'.

Figure 8 is an example of *counting*, where our model found one object more than necessary. First, the traveler correctly chose the blue sphere as its starting point. Then, the traveler should have moved to the blue object left of it, as stated in the query. While our model did this transition, it also wrongfully retained a self-loop to an object from the previous step. We suppose, this relation was kept, as this source object met one of the two conditions for the next step: it is blue (but *not* left of itself, which is the second condition). Finally, our model considers destinations left of both, the small cylinder and the incorrectly visited blue sphere, resulting in three counted objects, instead of two.

References

[1] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv*, 2018.

[2] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *arXiv*, 2016.

[3] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

[4] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[5] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[7] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, pages 7310–7311, 2017.

[8] A. Kembhavi, M. Salvato, E. Kolve, M. Seo, H. Hajishirzi, and A. Farhadi. A diagram is worth a dozen images. In *ECCV*, 2016.

[9] D. Kim, Y. Yoo, J. Kim, S. Lee, and N. Kwak. Dynamic graph generation network: Generating relational knowledge from diagrams. *CVPR*, 2017.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016.

[13] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[14] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017.

[15] G. R. Yang, I. Ganichev, X.-J. Wang, J. Shlens, and D. Sussillo. A dataset and architecture for visual reasoning with a working memory. *ECCV*, 2018.