A Details on Model Training and Inference

A.1 Training Details

In order to optimize the training objective given by Equation (4), we use REINFORCE [37] to obtain the gradient approximation

$$\hat{g} = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)(\sum_{t'=t}^{T-1} R_{t'} \gamma^{t'-t} - B(S_t)), \tag{6}$$

where γ is the discount factor for the reward. The gradient of the weights are aggregated over multiple rollouts. To reduce the variance, we adopt a moving average baseline function $B(S_t)$. The baseline function is an approximation of the value of a state S_t. We could have employed more sophisticated methods such as advantage network or actor-critic algorithm. However, we find the current baseline works sufficiently well. Formally, the baseline function consists of a non-trainable variable b and a hyperparameter λ. The baseline is updated by $b_{t+1} = \lambda b_t + (1 - \lambda) r_t$ at each optimization step. Another technique that affects the training speed is the reward normalization. Concretely, the accumulated rewards at each time step for each rollout are collected and normalized after subtraction of the baseline value.

We introduce a regularization term on the entropy of the resulting probability distribution from the policy network $\pi_\theta(A_t|S_t)$, which enforces that the agent explores the SG. The regularization is controlled by a hyperparameter β. In addition, we apply exponential decay to β during training so that β converges to zero.

Moreover, we use the chain rule to calculate the gradients of the parameters of the graph encoder (GAT) θ_{GAT} and the question encoder (Transformer) $\theta_{Transformer}$. The weight updates can be performed via gradient ascent, $\theta \leftarrow \theta + \eta\hat{g}$ or more advanced optimization methods such as Adam [21].

A.2 Inference

Beam search is used to infer the answer to a given question. Our inference approach is based on evaluating how likely specific paths are appearing among all possible paths with a fixed length. More specifically, given an input question, the agent's initial location is given by the hub node. At each time step, the agent scores the next permissible actions based on the learned policy. The value of action represents the transition probability from the current node to a target node. Next, we keep the top k (also known as beam width) paths among all possible transitions and move the agent to the corresponding targets. This computation is iteratively performed until the maximum number of transitions is reached. In the end, we obtain multiple rollouts ranked by the path probabilities. The target node (i.e. the last node) of the path is regarded as an answer candidate. Unlike Monte Carlo sampling which does not consider path probabilities, beam search yields better answer candidates, as it always chooses the best choice within the search region. The algorithm for inference is summarized in table 2.

Algorithm 1: Training regime

Input: Question Q, Scene Graph $\mathcal{SG} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$
Model : Policy Network with $\theta := \{\theta_{GAT}, \theta_{Transformer}, \theta_{Agent}\}$, Baseline with b

for $i \leftarrow 0$ **to** N **do** `// Loop over epochs`
  Initialize Q and $\mathcal{SG}$ with GloVe embeddings;
  $Q \leftarrow \mathrm{GAT}(Q)$ `// Update the question with the question encoder`
  $\mathcal{SG} \leftarrow \mathrm{Transformer}(\mathcal{SG})$ `// Update the SG with the graph encoder`
  $C \leftarrow []$ `// Initialize the trajectory buffer`
  for $r \leftarrow 0$ **to** N **do** `// Loop over samples`
    $\tau \leftarrow []$ `// Initialize the trajectory`
    $E_0 \leftarrow$ hub `// Initialize the start position`
    $A_0 \leftarrow$ dummy `// Initialize the dummy start action`
    for $t \leftarrow 0$ **to** T **do** `// Loop over time steps`
      if $t\%\Delta == 0$ **then** `// Restart and prompt the agent to the hub node`
        `// so that the agent is aware of its own action`
        $E_{t+1} \leftarrow$ hub `// Set next nodes to the hub node`
        $A_{t+1} \leftarrow$ dummy `// Set next actions to the dummy return action`
      end
      Sample an action (A_t, E_{t+1}) from d_t τ.append(A_t, E_t) `// Extend the trajectory`
      $E_t \leftarrow E_{t+1}$ `// Move the agent to the next entity`
    end
    C.append(τ) `// Collect the trajectory`
  end
  $r \leftarrow R(C)$ `// Gather rewards`
  $g \leftarrow \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=t}^{T-1} r_{t'} \gamma^{t^{'}-t} - b(s_t))$ `// Approximate gradients`
  $\theta \leftarrow \theta + \eta g$ `// Update the policy network`
  $b \leftarrow b + (1-\lambda) r$ `// Update the baseline function`
end

Inference Complexity The inference of our method is computationally efficient. Unlike other methods that need to iterate through each candidate answer for a final prediction, we only need to run the inference once so that the score of each answer is obtained. Let d denote the embedding dimension of the words and entities. Analytically, the embedding stage has asymptotic complexity $\mathcal{O}(|\mathcal{E}| + |\mathcal{R}| + |Q|)$. For the GAT, the implementation of a single attention head and multi-head attention is similar. In particular, they have the same time complexity $\mathcal{O}(|\mathcal{E}|dd^{'} + |\mathcal{R}|d^{'})$. The computation of the question encoding is given by $\mathcal{O}(|\mathcal{Q}|^2 d)$. It is efficient as it only runs once for each question and is used for arbitrary times during random walks. Also, the length of the questions Q is usually short (less than 30 words). Finally, during the random walk sampling, the agents complxity is given by $\mathcal{O}(T(D^2 + |\mathcal{E}|d))$, where d is dominant. The inference time depends largely on the path length.

Algorithm 2: Inference with beam search

Input: Question Q, Scene graph $\mathcal{SG} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$
Output: Answer

Initialize Q and $\mathcal{SG}$ with GloVe embeddings;
$Q \leftarrow \text{GAT}(Q)$ `// Update the question with the question encoder`
$\mathcal{SG} \leftarrow \text{Transformer}(\mathcal{SG})$ `// Update the SG with the graph encoder.`
$P \leftarrow []$ `// Initialize the probability register`
$\tau \leftarrow []$ `// Initialize the trajectory`
$E_0 \leftarrow$ hub `// Initialize the start position`
$A_0 \leftarrow$ dummy `// Initialize the dummy start action`
for $t \leftarrow 0$ **to** T **do** `// Loop over time steps`
    for $r \leftarrow 0$ **to** N **do** `// Loop over rollouts`
        if $t\%\Delta == 0$ **then** `// Restart and prompt the agent to the hub node`
            `// so that the agent is aware of its own action`
            $E_{t+1} \leftarrow$ hub `// Set next nodes to the hub node`
            $A_{t+1} \leftarrow$ dummy `// Set next actions to the dummy return action`
        end
        Forward pass through the policy network to generate candidate actions $\{(A_t, E_t)\}$ along with their probabilities $\{p_i\}$ τ.append($\{(A_t, E_t)\}$) `// Extend the trajectory`
        P.append($\{p_i\}$) `// Store corresponding probabilities`
    end
    $indicies \leftarrow argmax(P, k)$ `// Filter indices of top k probabilities from P`
    $\tau \leftarrow \tau[indices]$ `// Choose top k paths ranked by their probabilities`
    $E_{t+1} \leftarrow e \in \tau$ `// Conduct corresponding transitions`
end
Prediction $\leftarrow \tau[0]$ `// Predict the end entity of the top path as the answer`

A.3 Complexity Analysis

For analyzing the complexity of our method, we provide all the parameters contained in the building blocks. Moreover, we present the number of operations of a forward pass - the complete run that derives the answer from a given Q and $\mathcal{SG}$. They are listed in the table 3.

Group	Name	No. Parameters	No. Operations
Word Embeddings*	Entity	$N_e \times d$	$\mathcal{O}(N)$
	Relation	$N_r \times D$	$\mathcal{O}(N)$
GAT	Conv layer weight	$d \times H_1$	$\mathcal{O}(BHN_e)$
	Conv layer attention	$d \times H_1$	$\mathcal{O}(BHN_e)$
	Conv layer bias	H_1	$\mathcal{O}(BHN_e)$
Transformer	Positional encoder	$d \times d$	$\mathcal{O}(N_e)$
	Layer self attention (qkv)	$H_t(512) \times d$	$3 \times H_t \times d$
	Self attn norm (W, b)	d	$2 \times d$
	Layer enc attn	$H_t(512) \times d$	$3 \times H_t \times d$
	Enc attn norm (W, b)	d	$2 \times d$
	Pos ffn 1 (W, b)	$d \times d + d$	$d \times d + d$
	Pos ffn 2 (W, b)	$d \times d + d$	$d \times d + d$
	Pos ffn norm(W, b)	$2d$	$2 \times d$
	Enc attn norm (W, b)	d	$2 \times d$
Agent-MLP	Dense 0	$4d \times 4d + 4d$	$(H \times 4d + 4d) \times T$
	Dense 1	$2d \times 4d + 2d$	$(H \times 2d + 2d) \times T$
Agent-LSTM	Lstm_cell W_{ih}	$4d \times 16d$	$(4d \times 16d) \times T$
	Lstm_cell $_{ih}$	$16d$	$(16d) \times T$
	Lstm_cell W_{hh}	$4d \times 16d$	$(4d \times 16d) \times T$
	Lstm_cell b_{hh}	$16d$	$(16d) \times T$

Table 3. An overview of the number parameters and the asymptotic number of operations for the individual modules. The batch size is indicated by B. T corresponds to the number of time steps. D and H denote the embedding size and hidden size, respectively. Blocks are marked with a "*" if their weights are not trainable.

B Additional Details on the Dataset GQA

In this section we describe various question category and their type. We list the question based on semantic and structural categories. We further grouped them based on their entity type like object, attribute, category etc. Table 4, describes the detailed list of question category.

Table 4. List of question examples in the GQA dataset.

Category	Type	Description	Example
Semantics	Object	Existence of object	Are there any doors that are not made of metal?
	Attribute	Property about an object	Does the soap dispenser that is to the right of the other soap dispenser have small size and white color?
	Category	Identify an object class	What kind of animal is standing?
	Relation	Relationship of object	What is the food that is to the left of the white object that is to the left of the chocolate called?
	Global	Overall scene property	Which place is it?
Structural	Query	Open-form question	What type of furniture is to the left of the silver device which is to the left of the helmet?
	Choose	Choose from alternatives	What are the floating people in the ocean doing, riding or swimming?
	Verify	Simple yes/no question	Are there statues above the brass clock that is on the building?
	Compare	Comparison of objects	Are the drawers made of the same material as the cages?
	Logical	And/or operators	Are both the giraffe near the building and the giraffe that is to the left of the tray standing?