Guessing State Tracking for Visual Dialogue

Wei Pang and Xiaojie Wang

Center for Intelligence Science and Technology, School of Computer Science,
Beijing University of Posts and Telecommunications
{pangweitf,xjwang}@bupt.edu.cn

Abstract. The Guesser is a task of visual grounding in GuessWhat?! like visual dialogue. It locates the target object in an image supposed by an Oracle oneself over a question-answer based dialogue between a Questioner and the Oracle. Most existing guessers make one and only one guess after receiving all question-answer pairs in a dialogue with the predefined number of rounds. This paper proposes a guessing state for the Guesser, and regards guess as a process with change of guessing state through a dialogue. A guessing state tracking based guess model is therefore proposed. The guessing state is defined as a distribution on objects in the image. With that in hand, two loss functions are defined as supervisions to guide the guessing state in model training. Early supervision brings supervision to Guesser at early rounds, and incremental supervision brings monotonicity to the guessing state. Experimental results on GuessWhat?! dataset show that our model significantly outperforms previous models, achieves new state-of-the-art, especially the success rate of guessing 83.3% is approaching the human-level accuracy of 84.4%.

Keywords: Visual Dialogue, Visual Grounding, Guessing State Tracking, GuessWhat?!

1 Introduction

Visual dialogue has received increasing attention in recent years. It involves both vision and language processing and interactions between them in a continuous conversation and brings some new challenging problems. Some different tasks of visual dialogue have been proposed, such as Visual Dialog [5], GuessWhat?! [21], GuessWhich [4], and MNIST Dialog [29, 15, 11]. Among them, GuessWhat?! is a typical object-guessing game played between a Questioner and an Oracle. Given an image including several objects, the goal of the Questioner is to locate the target object supposed by the Oracle oneself at the beginning of a game by asking a series of yes/no questions. The Questioner, therefore, has two sub-tasks: one is Question Generator (QGen) that asks questions to the Oracle, the other is Guesser that identifies the target object in the image based on the generated dialogue between the QGen and Oracle. The Oracle answers questions with yes or no. The left part of Fig.1 shows a game played by the QGen, Oracle, and Guesser. The Guesser, which makes the final decision, is the focus of this paper.

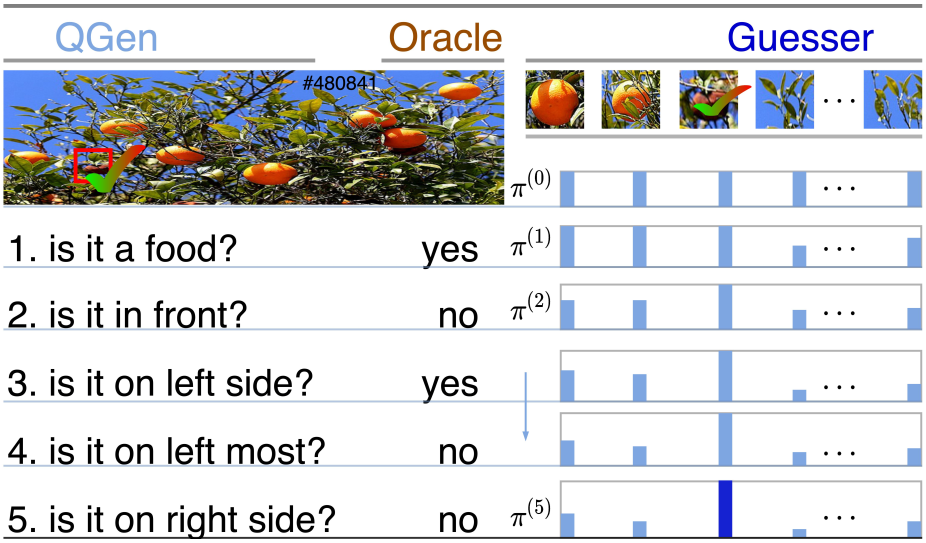


Fig. 1. The left part shows a game of GuessWhat?!. The right part illustrates the guess in Guesser as a process instead of a decision in a single point (the strips lineup denotes a probability distribution over objects, the arrowhead represents the tracking process).

Compared with QGen, relatively less work has been done on Guesser. It receives as input a sequence of question-answer (QA) pairs and a list of candidate objects in an image. The general architecture for Guesser introduced in [21, 20] that encodes the QA pairs into a dialogue representation and encodes each object into an embedding. Then, it compares the dialogue representation with any object embedding via a dot product and outputs a distribution of probabilities over objects, the object with higher probability is selected as the target. Most current work focuses on encoding and fusing multiple types of information, such as QA pairs, images, and visual objects. For example, some models [21, 20, 14, 19, 27, 17, 18, 3] convert the dialogue into a flat sequence of QA pair handled by a Long Short-Term Memory (LSTM)[8], some models [28, 2, 6, 24] introduce attention and memory mechanism to obtain a multi-modal representation of the dialogue.

Most of the existing Guesser models make a guess after fixed rounds of QA pairs, and this does not fully utilize the information from the sequence of QA pairs, we refer to that way as single-step guessing. Different games might need different rounds of QA pairs. Some work [17, 2] has therefore been done on choosing when to guess, i.e., make a guess after different rounds of question-answer for different games.

No matter the number of question-answer rounds is fixed or changed in different dialogues, existing Guesser models make one and only one guess after the final round of question-answer, i.e., Guesser is not activated until it reaches the final round of dialogue.

This paper models the Guesser in a different way. We think the Guesser to be active throughout the conversation of QGen and Oracle, rather than just only guessing at the end of the conversation. It keeps on updating a guess distribution after each question-answer pair from the beginning and does not make a final guess until the dialogue reaches a predefined round or it can make a confident guess. For example, as shown in Figure 1, a guess distribution is initiated as uniform distribution, i.e., each object has the same probability as the target object at the beginning of the game. After receiving the first pair of QA, the guesser updates the guess distribution and continues to update the distribution in the following rounds of dialogue. It makes a final guess after predefined five rounds of dialogue.

We think that modeling the Guesser as a process instead of a decision in a single point provides more chances to not only make much more detailed use of dialogue history but also combine more information for making better guesses. One such information is monotonicity, i.e., a good enough guesser will never reduce the guessing probability on the target object by making proper use of each question-answer pair. A good guess either raises the probability of a target object in guess distribution when the pair contains new information about the target object or does not change the probability when the pair contains no new information.

This paper proposes a guessing state tracking (GST) based Guesser model for implementing the above idea. Guessing state (GS) is at first time introduced into the game. A GS is defined as a distribution on candidate objects. A GST mechanism, which includes three sub-modules, is proposed to update GS after each question-answer pair from the beginning. Update of Visual Representation (UoVR) module updates the representation of image objects according to the current guessing state, QAEncoder module encodes the QA pair, and Update of Guessing State (UoGS) module updates the guessing state by combining both information from the image and QA. GST brings a series of GS, i.e., let the Guesser make a series of guesses during the dialogue.

Two loss functions are designed on making better use of a series of GS, or the introduction of GS into visual dialogue makes the two new loss functions possible. One is called early supervision loss that tries to lead GS to the target object as early as possible, where ground-truth is used to guide the guesses after each round of QA, even the guess after the first round where a successful guess is impossible at that time. The other is called incremental supervision loss that tries to bring monotonicity mentioned above to the probability of target object in the series of GS.

Experimental results show that the proposed model achieves new state-of-the-art performances in all different settings on GuessWhat?!. To summarize, our contributions are mainly three-fold:

- We introduce guessing state into visual dialogue for the first time and propose a Guessing State Tracking (GST) based Guesser model, a novel mechanism that models the process of guessing state updating over question-answer pairs.

- We introduce two guessing states based supervision losses, early supervision loss, and incremental supervision loss, which are effective in model training.
- Our model performs significantly better than all previous models, and achieves new state-of-the-art in all different settings on GuessWhat?!. The guessing accuracy of 83.3% approaches the human's level of 84.4%.

2 Related Work

Visual Grounding is an essential language-to-vision problem of finding the most relevant object in an image by a natural language expression, which can be a phrase, a sentence, or a dialogue. It has attracted considerable attention in recent years[7, 26, 23, 6, 1, 12, 25], and has been studied in the Guesser task in the GuessWhat?![6]. This paper focuses on grounding a series of language descriptions (QA pair) in an image gradually by dialoguing.

Most previous work views Guesser as making a single-step guess based on a sequence of QA pairs. In [21, 20, 27, 19, 18, 17, 11, 14, 24, 3], all the multi-round QA pairs are considered as a flat sequence and encoded into a single representation using either an LSTM or an HRED[16] encoder, each object is represented as an embedding encoded from their object category embedding and 8-d spatial position embedding. A score is obtained by performing a dot product between the dialogue representation and each object embedding, then followed a softmax layer on the scores to output distribution of probabilities over objects, the object with higher probability is chosen as the most relevant object. As we can see, only one decision is made by the Guesser.

Most of the guesser models explored to encode the dialogue of multi-round QA pairs in an effective way. For example, in [28, 2], they integrate Memory and Attention into the Guesser architecture used in [20]. Where the memory is consist of some facts that are separately encoded from each QA pairs, the image feature vector is used as a key to attend the memory. In [6], an accumulated attention (ATT) mechanism is proposed. It fuses three types of information, i.e., dialogue, image, and objects, by three attention models. Similarly, [24] proposed a history-aware co-attention network (HACAN) to encode the QA pairs.

As we can see, the models, as mentioned above, all make a single-step guess at the time that the dialogue ended, these might be counterintuitive. Different from them, we consider the guess as a process, and explicitly track the guessing states after every dialogue round. Compared with prior works, we refer to the proposed GST as multi-step guessing.

Our GST based Guesser model is related to the VDST[14] based QGen model. [14] proposed a well-defined questioning state for the QGen and implemented a suitable tracking mechanism through the dialogue. The crucial difference in tracking state is that the QGen requires to track changes on the representations of objects because it needs more detailed information concerning the attended objects for asking more questions, while the Guesser does not need it.

3 Model: Guessing State Tracking

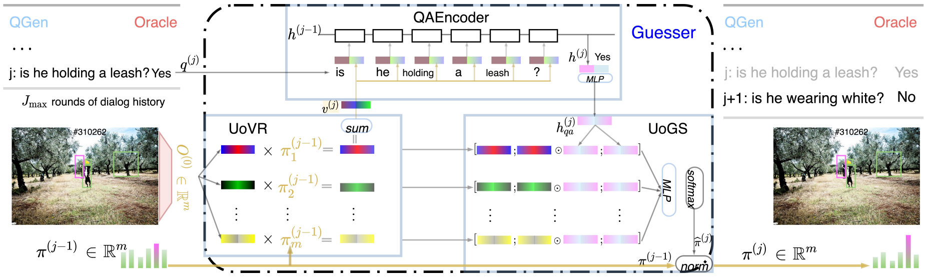


Fig. 2. Overview of the proposed Guesser model.

The framework of our guessing state tracking (GST) model is illustrated in Fig.2. Three modules are implemented in each round of guessing. There are Update of Visual Representation (UoVR), Question-Answer Encoder (QAEncoder), and Update of Guessing State (UoGS). Where, UoVR updates representation of an image for Guesser according to the previous round of guessing state, new visual representation is then combined into QAEncoder for synthesizing information from both visual and linguistic sides up to the current round of dialogue for the Guesser. Finally, UoGS is applied to update the guessing state of the guesser. We give details of each module in the following sub-sections.

3.1 Update of Visual Representation (UoVR)

Following previous work [21, 20], candidate objects in an image are represented by their category and spatial features as in Eq.1:

$$O^{(0)} = \{o_i^{(0)} | o_i^{(0)} = \text{MLP}([o_{cate}; o_{spat}])\}_{i=1}^{m}, \tag{1}$$

where $O^{(0)} \in \mathbb{R}^{m \times d}$ consists of m initial objects. For each object $o_i^{(0)}$, it is a concatenation of an 512-d category embedding o_{cate} and an 8-d vector o_{spat} of object location in an image. Where o_{cate} are learnable parameters, o_{spat} are coordinates $[x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}]$ as in [21], w_{box} and h_{box} denote width and height of an object, the coordinates range from -1 to 1 scaled by the image width and height. To map object and word embedding to the same dimension, the concatenation is passed through an MLP to obtain a d-dimensional vector.

Let $\pi^{(j)} \in \mathbb{R}^m$ be an accumulative probability distribution over m objects after jth round of dialogue. It is defined as the guessing state and will be updated with the guessing process. At the beginning of a game, $\pi^{(0)}$ is a uniform distribution. With the progress of guessing, the visual representation in guesser's mind would update accordingly. Two steps are designed. The first step is an update of representations of objects. Pang and Wang[14] use an effective representation update in the VDST model. We borrow it for our GST model, as written in Eq.2:

$$O^{(j)} = (\pi^{(j-1)})^T O^{(0)}, \tag{2}$$

where $O^{(j)} \in \mathbb{R}^{m \times d}$ is a set of m updated objects at round j. Second, the summed embedding of all objects in $O^{(j)}$ is used as new visual representation as shown in Eq.3,

$$v^{(j)} = \mathrm{sum}(O^{(j)}), \tag{3}$$

where $v^{(j)} \in \mathbb{R}^d$ denotes updated visual information for the guesser at round j.

3.2 Question-Answer Encoder (QAEncoder)

For encoding linguistic information in the current question with visual information in hand, we concatenate $v^{(j)}$ to each word embedding $w_i^{(j)}$ in jth turn question $q^{(j)}$, take the concatenation as input to a single-layer LSTM encoder one by one as shown in Eq.4,

$$h^{(j)} = \mathrm{LSTM}([w_i^{(j)}; v^{(j)}]_{i=1}^{N^{(j)}}, h^{(j-1)}), \tag{4}$$

where $N^{(j)}$ is the length of question $q^{(j)}$. The last hidden state of the LSTM, $h^{(j)}$, is used as representation of $q^{(j)}$, and $h^{(j-1)}$ is used as initial input of the LSTM as shown in Fig.2.

$h^{(j)}$ is then concatenated to $a^{(j)}$, which is the embedding of the answer to jth turn question, the result $[h^{(j)}; a^{(j)}]$ are passed through an MLP to obtain the representation of QA pair at round j, as written in Eq.5,

$$h_{qa}^{(j)} = \mathrm{MLP}([h^{(j)}; a^{(j)}]), \tag{5}$$

where $h_{qa}^{(j)} \in \mathbb{R}^d$ synthesizes information from both questions and answers up to jth round dialogue for the guesser. It will be used to update the guessing state in the next module.

3.3 Update of Guessing State (UoGS)

When a new QA pair is received from the QGen and the Oracle, the Guesser needs to make a decision on which object would be ruled out, or which one would be gained more confidence, then renews its guessing state over objects in the image. Three steps are designed for updating the guessing state.

First, to fuse two types of information from QA and visual objects, we perform an element-wise product of $h_{qa}^{(j)}$ and each object embedding in $O^{(j)}$ to generate a joint feature for any object, as shown in Eq.6,

$$O_{qa}^{(j)} = h_{qa}^{(j)} \odot O^{(j)}, \tag{6}$$

where $\odot$ denotes element-wise product, $O_{qa}^{(j)} \in \mathbb{R}^{m \times d}$ contains m joint feature objects.

Second, to measure how much the belief changes on ith object after jth dialog round, three feature vectors: the QA pair feature, joint feature of the ith object and updated representation of the ith object are concatenated and passed through a two-layer linear with a tanh activation, followed a softmax layer to produce change of belief as described in Eq.7,

$$\hat{\pi}_i^{(j)} = \text{softmax}(W_2^T(tanh(W_1^T([h_{qa}^{(j)}; (O_{qa}^{(j)})_i; (O^{(j)})_i])))), \tag{7}$$

where $i \in \{1, 2, \dots, m\}$, $W_1 \in \mathbb{R}^{1536 \times 128}$ and $W_2 \in \mathbb{R}^{128 \times 1}$ are learnable parameters, the bias b is omitted in the linear layer for simplicity. $\hat{\pi}_i^{(j)} \in [0, 1]$ means the belief changes on ith object after jth round. We find that this type of symmetric concatenation $[;;]$ in Eq.7, where language and visual information are in a symmetrical position, is an effective way to handle multimodal information, which is also used in [9].

Finally, the previous rounds of guessing state $\pi^{(j-1)}$ are updated by multiplying $\hat{\pi}^{(j)} \in \mathbb{R}^m$ as follows: $\pi^{(j)} = norm(\pi^{(j-1)} \odot \hat{\pi}^{(j)})$. Where $\pi^{(j)} \in \mathbb{R}^m$ is the accumulated guessing state till round j, *norm* is a sum-normalization method to make it a valid probability distribution, e.g., by dividing the sum of it.

3.4 Early and Incremental Supervision

The introduction of guessing states provides useful information for model training. Because the guessing states are tracked from the beginning of a dialogue, supervision of correct guess can be employed from an early stage, which is called early supervision. Because the guessing states are tracked at each round of dialogue, the change of guessing state can also be supervised to ensure that the guessing is alone in the right way. We call this kind of supervision, incremental supervision. Two supervision functions are introduced as follows.

Early Supervision Early supervision (ES) tries to maximize the probability of the right object from the beginning of a dialogue and keeps on using up to the penultimate round of the dialogue. It is defined as the summary of a series of cross-entropy between the guessing state and the ground-truth. That is:

$$L_{ES} = \frac{1}{J_{max} - 1} \sum_{j=1}^{J_{max}-1} \text{CrossEntropy}(\pi^{(j)}, y^{GT}), \tag{8}$$

where y^{GT} is a one-hot vector with 1 in the position of the ground-truth object, J_{max} is the maximum number of rounds. The cross-entropy at the final round, i.e. $CrossEntropy(\pi^{(J_{max})}, y^{GT})$, we refer to as **plain supervision** loss (L_{PS} in briefly).

Incremental Supervision Incremental supervision (IS) tries to keep the probability of the target object in guessing state increasing or nondecreasing as written in:

$$L_{IS} = -\sum_{j=1}^{J_{max}} \log(\pi_{target}^{(j)} - \pi_{target}^{(j-1)} + c), \tag{9}$$

where $\pi^{(j)}_{target}$ denotes the target probability at round j. IS is defined as the change in probability to the ground-truth object before and after a round of dialog. Besides the log function that served as an extra layer of smooth, IS is somewhat similar to the progressive reward used in [27] that is from Guesser but as a reward for training QGen model. c is a parameter that ensures the input to log be positive.

3.5 Training

Our model is trained in two stages, including supervised and reinforcement learning. For supervised learning, the guesser network is trained by minimizing the following objective, $L_{SL}(\theta) = \alpha(L_{ES} + L_{PS}) + (1 - \alpha)L_{IS}$, where α is a balancing parameter. For reinforcement learning, the guesser network is refined by maximizing the reward given in $L_{RL}(\theta) = -E_{\pi_\theta}[\alpha(L_{ES} + L_{PS}) + (1 - \alpha)L_{IS})]$, where π_θ denotes a policy parameterized by θ which associates guessing state over actions, e.g., an action corresponds to select an object over m candidate objects. Following [28], we use the REINFORCE algorithm[22] without baseline that updates policy parameters θ.

4 Experiments and Analysis

4.1 Experimental Setup

Dataset GuessWhat?! dataset containing 66k images, about 800k question-answer pairs in 150K games. It is split at random by 70%, 15%, 15% of the games into the training, validation, and test set [21, 20].

Baseline models A GuessWhat?! game involves Oracle, QGen, and Guesser. Almost all existing work uses the same Oracle model [21, 20], which will be used in all our experiments. Two different QGen models are used for validating our guesser model. One is the often used model in previous work [20], the other is a new QGen model which achieves new state-of-the-art [14]. Several different existing Guesser models are compared with our model. They are guesser [21, 20, 1], guesser(MN) [28, 2], GDSE [18, 17], ATT [6] and HACAN [24]. The models are first trained in a supervised way on the training set, and then, one Guesser and one QGen model are jointly refined by reinforcement learning or cooperative learning from self-play with the Oracle model fixed.

Implementation Details The maximum round J_{max} is set to 5 or 8 as in [27, 19, 14]. The balancing parameter in loss and reward objectives are set to 0.7, because we observe that our model achieves the minimum error rate on validation and test set when $\alpha = 0.7$. The parameter c in Eq.9 is set as 1.1. The size of word embedding and LSTM hidden unit number are set to 512. Early stopping is used on the validation set.

We use success rate of guessing for evaluation. Following previous work [21, 20, 14], both success rates on NewObject and NewGame are reported. Results by three inference methods described in [2], including Sampling (S), Greedy (G) and Beam-search (BS, beam size is set to 20) are used on both NewObject and NewGame. Following [28, 2], during joint reinforcement learning of Guesser and QGen models, only the generated successful games are used to tune the Guesser model, while all the generated games are used to optimize the QGen.

Supervised Learning (SL) We separately train the Guesser and Oracle model for 20 epochs, the QGen for 50 epochs using Adam optimizer [10] with a learning rate of 3e-4 and a batch size of 64.

Reinforcement Learning (RL) We use momentum stochastic gradient descent with a batch size of 64 with 500 epochs and learning rate annealing. The base learning rate is 1e-3 and decayed every 25 epochs with exponential rate 0.99. The momentum parameter is set to 0.9.

4.2 Comparison with the state-of-the-art

Task Success Rate Table 1 reports the success rate of guessing with different combinations of QGen and Guesser models with the same Oracle model used in [21, 20] for the GuessWhat?! game.

In the first part of table 1, all models are trained in SL way. We can see that no matter which QGen models are used, qgen [20] or VDST [14], our guesser model GST significantly outperforms other guesser models in both 5 and 8 rounds dialogue at all different settings. Specifically, GST achieves a new state-of-the-art of 54.10% and 50.97% on NewObject and NewGame in Greedy way by SL.

In the second part of table 1, two combinations trained in cooperative learning (CL) way are given. Our model is not trained in this way. So we do not have a comparison in CL case with the performance of these models are lower than those in the RL part.

In the third part of table 1, all QGen and Guesser models are trained by RL. We can see that our GST Guesser model combined with the VSDT QGen model achieves the best performance in both 5 and 8 rounds dialogue at all different settings. It significantly outperforms other models. For example, it outperforms the best previous model at Sampling (S) setting on NewObject (i.e. guesser(MN)[28] + ISM [1] with 72.1%) by nearly 9 percent, outperforms the best previous model at Greedy (G) setting on NewObject (i.e. guesser(MN) [28] + TPG [28] with 74.3%) by more than 9 percent, outperforms the best previous model in NewObject at Beam-search (BS) setting on NewObject (i.e. guesser [20] + VDST [14] with 71.03%) by more than 12 percent. The same thing happens on NewGame case. That is to say, our model consistently outperforms previous models in all different settings on both NewObject and NewGame. Especially, GST achieves 83.32% success rate on NewObject in Greedy way, which is approaching human performance 84.4%. Fig.3(a) shows the learning curve for joint training of GST Guesser and VDST QGen with 500 epochs in Sampling way, it shows superior accuracy compared to the Guesser model[20] trained with VDST QGen.

Specifically, with same QGen (no matter which QGen models are used, qgen used in [20] or VDST used in [14]), our guesser model GST significantly outperforms other guesser models in both 5 and 8 rounds dialogue at all different settings. It demonstrates that GST is more able to ground a multi-round QA

Table 1. Success rates of guessing (%) with same Oracle (higher is better).

	Questioner		Max Q's	New Object			New Game		
	Guesser	QGen		S	G	BS	S	G	BS
pretrained in SL	guesser[21]	qgen[21]	5	41.6	43.5	47.1	39.2	40.8	44.6
	guesser(MN)[28]	TPG[28]	8	-	48.77	-	-	-	-
	guesser[20]	qgen[20]	8	-	44.6	-	-	-	-
	GST(ours)		8	41.73	**44.89**	-	39.97	41.36	-
	guesser[20]	VDST[14]	5	45.02	49.49	-	42.92	45.94	-
			8	46.70	48.01	-	44.24	45.03	-
	GST(ours)		5	**49.55**	**53.35**	**53.17**	**46.95**	**50.58**	**50.71**
			8	**52.71**	**54.10**	**54.32**	**50.19**	**50.97**	**50.99**
SL	GDSE-SL[18]	GDSE-SL[18]	5	-	-	-	-	47.8	-
			8	-	-	-	-	49.7	-
CL	GDSE-CL[18]	GDSE-CL[18]	5	-	-	-	-	53.7	-
			8	-	-	-	-	58.4	-
AQM	guesser[11]	randQ[11]	5	-	-	-	-	42.48	-
		countQ[11]	5	-	-	-	-	61.64	-
trained by RL	guesser(MN)[28]	TPG[28]	5	62.6	-	-	-	-	-
			8	-	-	-	-	74.3	-
		ISM[1]	-	74.4	-	-	72.1	-	-
		TPG[28]	8	-	74.3	-	-	-	-
		ISD[2]	5	68.3	69.2	-	66.3	67.1	-
	guesser[20]	VQG[27]	5	63.2	63.6	63.9	59.8	60.7	60.8
		ISM[1]	-	-	64.2	-	-	62.1	-
		ISD[2]	5	61.4	62.1	63.6	59.0	59.8	60.6
		RIG(rewards)[19]	8	65.20	63.00	63.08	64.06	59.0	60.21
		RIG(loss)[19]	8	67.19	63.19	62.57	65.79	61.18	59.79
	guesser[20]	qgen[20]	5	58.5	60.3	60.2	56.5	58.4	58.4
			8	62.8	58.2	53.9	60.8	56.3	52.0
	guesser(MN)[28]		5	59.41	60.78	60.28	56.49	58.84	58.10
			8	62.05	62.73	-	59.04	59.50	-
	GST(ours)		5	**64.78**	**67.06**	**67.01**	**61.77**	**64.13**	**64.26**
	guesser[20]	VDST[14]	5	66.22	67.07	67.81	63.85	64.36	64.44
			8	69.51	70.55	71.03	66.76	67.73	67.52
	GST(ours)		5	**77.38**	**77.30**	**77.23**	**75.11**	**75.20**	**75.13**
			8	**83.22**	**83.32**	**83.46**	**81.50**	**81.55**	**81.62**
	Human[20]	-	-	-	84.4	-	-	84.4	-

pairs dialogue in the image compared to previous single-step guessing models for the GuessWhat?! game.

Error Rate For a fair comparison of Guesser models alone, we follow the previous work[6, 24] by measuring error rate on training, validation, and test set. In Table 2, we can see that GST trained in SL is comparable to more complex attention algorithms, such as ATT[6] and HACAN[24]. After reinforcement learning, GST model achieves a lower error rate than the compared models in

Table 2. Error rate (%) on the GuessWhat?! dataset (lower is better).

Model	Train err	Val err	Test err	Max Q's
Random[21]	82.9	82.9	82.9	-
LSTM[21]	27.9	37.9	38.7	-
HRED[21]	32.6	38.2	39.0	-
Guesser[20]	-	-	36.2	-
LSTM+VGG[21]	26.1	38.5	39.5	-
HRED+VGG[21]	27.4	38.4	39.6	-
ATT-r2[6]	29.3	35.7	36.5	-
ATT-r3[6]	30.5	35.1	35.8	-
ATT-r4[6]	29.8	35.3	36.3	-
ATT-r3(w2v)[6]	26.7	33.7	34.2	-
Guesser[19]	-	-	35.8	-
HACAN[24]	26.1	32.3	33.2	-
GST(ours, trained in SL)	24.7	33.7	34.3	-
GST(ours, trained in RL)	**22.7**	**23.1**	**24.7**	5
GST(ours, trained in RL)	**16.7**	**16.9**	**18.4**	8
Human[21]	9.0	9.2	9.2	

Table 3. Comparison of success rate with different supervisions in SL.

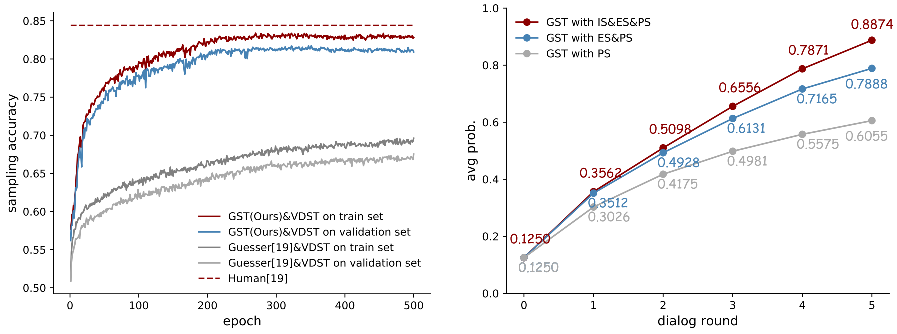
#	Model	New Object	
		S	G
1	GST with ES&PS and IS (full)	52.71	54.10
2	−ES&PS	37.10	42.58
3	−IS	48.96	53.49
		New Game	
1	GST with ES&PS and IS (full)	50.19	50.97
2	−ES&PS	34.41	39.48
3	−IS	46.10	50.33

both 5 and 8 rounds, especially at 8 rounds, it obtains error rate of 16.7%, 16.9%, and 18.4%, respectively.

4.3 Ablation Study

Effect of Individual Supervision In this section, we conduct ablation studies to separate contribution of supervisions: Plain Supervision (PS), Early Supervision (ES) and Incremental Supervision (IS).

Table 3 reports the success rate of guessing after supervised learning. Removing ES&PS from the full model, the game success rate significantly drops 11.52 (from 54.10% to 42.58%) and 11.49 (from 50.97% to 39.48%) points on



(a) (b)

Fig. 3. a, Sampling accuracy of reinforcement learning on training and validation set, our GST outperforms guesser[20] by a large margin. **b**, Average belief of the ground-truth object at each round, changes with an increase in the number of dialogue rounds.

NewObject and NewGame on Greedy case. Removing IS, the success rate drops 0.61 (from 54.10% to 53.49%) and 0.64 (from 50.97% to 50.33%), respectively. It shows that early supervision pair with ES&PS contributes more than incremental supervision.

We then analyze the impact of supervision losses to guessing state. We train three GST models with RL using three different loss functions, i.e. PS, PS&ES, and PS&ES&IS respectively and then count the averaged probability of the ground-truth object based on all the successful games in test set at each round. Fig.3(b) shows three curves of averaged belief changing with rounds of dialogue. As is observed, we have three notes.

First, guess probability is progressively increasing in all three different losses. It demonstrates our core idea: thinking of the guess as a process instead of a single decision is an effective practical way to ground a multi-round dialogue in an image. Because GST based Guesser makes use of more detailed information in the series of guessing states (GS), i.e. the two losses.

Second, average probability in the blue line, trained with ES&PS, is higher than that in the gray line (trained in PS alone), it demonstrates the effectiveness of early supervision loss.

Third, average probability in the red line, trained with IS&ES&PS, is better than that in the blue line, it shows incremental supervision gives further improvement to guess.

Overall, these results demonstrate the effectiveness of early supervision and incremental supervision. It is the combination of these supervisions that train GST based guesser model efficiently.

Effect of Symmetric Concatenation In table 4, compared with symmetric concatenation appears in Eq.7, average error rate increases 2.9 points on all three sets if $[h_{qa}^{(j)}; O^{(j)}]$ used and increases 1.9 points if $[h_{qa}^{(j)} \odot O^{(j)}]$ used. It indicates that symmetric concatenation serves as a valuable part in Eq.7.

Table 4. Comparison of error rate (%) for three types of concatenation during SL.

Concat	Train err	Val err	Test err
$[h_{qa}^{(j)}; h_{qa}^{(j)} \odot O^{(j)}; O^{(j)}]$	**24.8**	**33.7**	**34.4**
$[h_{qa}^{(j)} \odot O^{(j)}]$	26.3	35.7	36.7
$[h_{qa}^{(j)}; O^{(j)}]$	27.3	36.5	37.8

Table 5. Error rate of different c in Eq.9 during SL.

c	Train	Val	Test
c=1.1	24.7	**33.7**	**34.3**
c=1.5	26.5	34.1	34.8
c=2.0	**23.3**	34.0	34.8

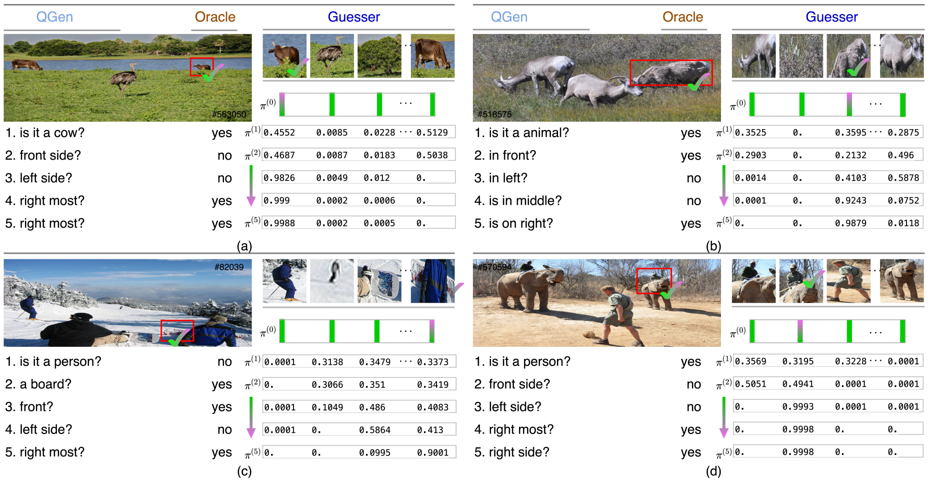

QGen
Oracle
Guesser
#563050
$\pi^{(0)}$
1. is it a cow? yes $\pi^{(1)}$ 0.4552 0.0085 0.0228 ··· 0.5129
2. front side? no $\pi^{(2)}$ 0.4687 0.0087 0.0183 0.5038
3. left side? no 0.9826 0.0049 0.012 0.
4. right most? yes 0.999 0.0002 0.0006 0.
5. right most? yes $\pi^{(5)}$ 0.9988 0.0002 0.0005 0.
(a)
QGen
Oracle
Guesser
#518575
$\pi^{(0)}$
1. is it a animal? yes $\pi^{(1)}$ 0.3525 0. 0.3595 ··· 0.2875
2. in front? yes $\pi^{(2)}$ 0.2903 0. 0.2132 0.496
3. in left? no 0.0014 0. 0.4103 0.5878
4. is in middle? no 0.0001 0. 0.9243 0.0752
5. is on right? yes $\pi^{(5)}$ 0. 0. 0.9879 0.0118
(b)
#82039
$\pi^{(0)}$
1. is it a person? no $\pi^{(1)}$ 0.0001 0.3138 0.3479 ··· 0.3373
2. a board? yes $\pi^{(2)}$ 0. 0.3066 0.351 0.3419
3. front? yes 0.0001 0.1049 0.486 0.4083
4. left side? no 0.0001 0. 0.5864 0.413
5. right most? yes $\pi^{(5)}$ 0. 0. 0.0995 0.9001
(c)
#570594
$\pi^{(0)}$
1. is it a person? yes $\pi^{(1)}$ 0.3569 0.3195 0.3228 ··· 0.0001
2. front side? no $\pi^{(2)}$ 0.5051 0.4941 0.0001 0.0001
3. left side? no 0. 0.9993 0.0001 0.0001
4. right most? yes 0. 0.9998 0. 0.
5. right side? yes $\pi^{(5)}$ 0. 0.9998 0. 0.
(d)

Fig. 4. Four successful games show the process of tracking guessing state.

Effect of c in Eq.9 Table 5 shows error rate of different c in Eq.9 trained with SL on three dataset. As is observed, c is insensitive to the error rate. We set c to 1.1 as it obtains a lower error rate on Val err and Test err.

4.4 Qualitative Evaluation

In Fig.4, we show four successful dialogues to visualize the process in guessing. We plot 4 candidate objects for simplicity, $\pi^{(0)}$ represents a uniform distribution of initial guessing state, $\pi^{(1)}$ to $\pi^{(5)}$ show the process of tracking GS. Taking Fig.4(a) as an example. Guesser has an initial uniform guess on all candidates, i.e. $\pi^{(0)}$. QGen starts a dialogue by asking “is it a cow?”, Oracle answer “yes”, then Guesser renews its $\pi^{(0)}$ to $\pi^{(1)}$. Specifically, the probabilities on the ostrich and tree approaches go down to close to 0, the cow on both sides increases to 0.45 and 0.51 respectively. At last, all the probabilities are concentrated on the cow on the right with high confidence of 0.9988, which is the guessed object. In 4(b) to 4(d), three more success cases are shown.

4.5 Discussion on Stop Questioning

When to stop questioning is also a problem in GuessWhat?! like visual dialogue. Most of the previous work chooses a simple policy, i.e., a QGen model stops questioning after a predefined number of dialogue rounds, and the guessing model selects an object as the guess.

Our model can implement this policy by making use of $\pi^{(j)}$, the guessing state after the jth round dialogue. If K is the predefined number, the guesser model will keep on updating $\pi^{(j)}$ till $j = K$. The object with the highest probability in $\pi^{(K)}$ will be then selected as the guess.

A same number of questions are asked for any game under this policy, no matter how different the different games are. The problem of the policy is obvious. On the one hand, the guesser model does not select any object even if it is confident enough about a guess and make a QGen model keep on asking till K questions are asked. On the other hand, the QGen model cannot ask more questions when K questions are asked even if the guesser model is not confident about any guess at that time. The guesser model must give a guess.

Our model can provide a chance to adopt some other policies for stopping questioning. A simple way is to predefine a threshold of confidence. Once the biggest probability in a guessing state is equal to or bigger than the threshold, question answering is stopped, and the guesser model output the object with the biggest probability as the guess. Another way involves the gain of guessing state. Once the information gain from the jth state to the j+1th state is less than a threshold, the guesser model outputs the object with the biggest probability as the guess.

5 Conclusion

The paper proposes a novel guessing state tracking (GST) based model for the Guesser, which models guess as a process with change of guessing state, instead of making one and only one guess, i.e. a single decision, over the dialogue history in all the previous work. To make full use of the guessing state, two losses, i.e., early supervision loss and incremental supervision loss, are introduced. Experiments show that our GST based guesser significantly outperforms all of the existing methods, and achieves new strong state-of-the-art accuracy that closes the gap to humans, the success rate of guessing 83.3% is approaching the human-level accuracy of 84.4%.

Acknowledgements

We thank the reviewers for their comments and suggestions. This paper is supported by NSFC (No. 61906018), Huawei Noah's Ark Lab and MoE-CMCC "Artificial Intelligence" Project (No. MCM20190701).

References

1. Abbasnejad, E., Wu, Q., Abbasnejad, I., Shi, J., van den Hengel, A.: An active information seeking model for goal-oriented vision-and-language tasks. arXiv preprint arXiv:1812.06398 (2018)
2. Abbasnejad, E., Wu, Q., Shi, J., van den Hengel, A.: What's to know? uncertainty as a guide to asking goal-oriented questions. In: CVPR (2019)
3. Bani, G., Belli, D., Dagan, G., Geenen, A., Skliar, A., Venkatesh, A., Baumgartner, T., Bruni, E., Fernandez, R.: Adding object detection skills to visual dialogue agents. In: ECCV (2018)
4. Chattopadhyay, P., Yadav, D., Prabhu, V., Chandrasekaran, A., Das, A., Lee, S., Batra, D., Parikh, D.: Evaluating visual conversational agents via cooperative human-ai games. In: HCOMP (2017)
5. Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J.M., Parikh, D., Batra, D.: Visual dialog. In: CVPR (2017)
6. Deng, C., Wu, Q., Wu, Q., Hu, F., Lyu, F., Tan, M.: Visual grounding via accumulated attention. In: CVPR (2018)
7. Fukui, A., Park, D.H., Yang, D., Rohrbach, A., Darrell, T., Rohrbach, M.: Multimodal compact bilinear pooling for visual question answering and visual grounding. arXiv preprint arXiv:1606.01847 (2016)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
9. Kim, H., Tan, H., Bansal, M.: Modality-balanced models for visual dialogue. In: AAAI (2020)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
11. Lee, S.W., Heo, Y.J., Zhang, B.T.: Answerer in questioner's mind: Information theoretic approach to goal-oriented visual dialog. In: NeurIPS (2018)
12. Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A.L., Murphy, K.: Generation and comprehension of unambiguous object descriptions. In: CVPR. pp. 11–20 (2016)
13. Niu, Y., Zhang, H., Zhang, M., Zhang, J., Lu, Z., Wen, J.R.: Recursive visual attention in visual dialog. In: CVPR (2019)
14. Pang, W., Wang, X.: Visual dialogue state tracking for question generation. In: AAAI (2020)
15. Seo, P.H., Lehrmann, A., Han, B., Sigal, L.: Visual reference resolution using attention memory for visual dialog. In: NeurIPS (2017)
16. Serban, I., Sordoni, A., Bengio, Y., Courville, A., Pineau, J.: Hierarchical neural network generative models for movie dialogues. In: arXiv preprint arXiv:1507.04808 (2015)
17. Shekhar, R., Venkatesh, A., Baumgärtner, T., Bruni, E., Plank, B., Bernardi, R., Fernández, R.: Ask no more: Deciding when to guess in referential visual dialogue. In: COLING (2018)
18. Shekhar, R., Venkatesh, A., Baumgärtner, T., Bruni, E., Plank, B., Bernardi, R., Fernández, R.: Beyond task success: A closer look at jointly learning to see, ask, and guesswhat. In: NAACL (2019)
19. Shukla, P., Elmadjian, C., Sharan, R., Kulkarni, V., Wang, W.Y., Turk, M.: What should i ask? using conversationally informative rewards for goal-oriented visual dialogue. In: ACL (2019)

20. Strub, F., de Vries, H., Mary, J., Piot, B., Courville, A., Pietquin, O.: End-to-end optimization of goal-driven and visually grounded dialogue systems. In: IJCAI (2017)
21. de Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., Courville, A.C.: Guesswhat?! visual object discovery through multi-modal dialogue. In: CVPR (2017)
22. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning (1992)
23. Xiao, F., Sigal, L., Lee, Y.J.: Weakly-supervised visual grounding of phrases with linguistic structures. In: CVPR (2017)
24. Yang, T., Zha, Z.J., Zhang, H.: Making history matter: History-advantage sequence training for visual dialog. In: ICCV (2019)
25. Yu, L., Poirson, P., Yang, S., Berg, A.C., Berg, T.L.: Modeling context in referring expressions. In: ECCV (2016)
26. Yu, L., Tan, H., Bansal, M., Berg, T.L.: A joint speaker-listener-reinforcer model for referring expressions. In: CVPR (2017)
27. Zhang, J., Wu, Q., Shen, C., Zhang, J., Lu, J., van den Hengel, A.: Asking the difficult questions: Goal-oriented visual question generation via intermediate rewards. In: ECCV (2018)
28. Zhao, R., Tresp, V.: Improving goal-oriented visual dialog agents via advanced recurrent nets with tempered policy gradient. In: IJCAI (2018)
29. Zhao, R., Tresp, V.: Efficient visual dialog policy learning via positive memory retention. In: NeurIPS (2018)