DeepWhat?! End-to-end models for task-oriented visual dialogue with reinforcement

Charles Lu*
Stanford University
charleslu@stanford.edu

Elias Wu†
Stanford University
eliwu@stanford.edu

Abstract

Development of systems for dialogue-centered tasks such as visual question answering and chatbots have become an active area of research in deep learning, facilitated by the capabilities of recurrent architectures. However, most past research has framed such tasks as a supervised learning problem. This is not ideal, considering that there are often countless appropriate responses to an utterance, depending on the context. As a result, supervised learning approaches have only been successful in dialogue tasks with either very limited context (such as question answering) or with inconsequential conversation (chatbots). Furthermore, in the context of task-oriented dialogue, supervised learning techniques fail to encourage agents to achieve the relevant goal through expedient dialogue.

In our project, we implement systems to play the GuessWhat?! game, a collaborative game involving task-oriented visual dialogue. We first train models representing the two players of the game using a supervised dataset of 150,000 games. We then build upon recent research in using reinforcement learning to improve agents, initially trained using supervised techniques, to fine-tune one of the three models within the entire system, by fixing the two remaining models and using them as ground truth, in an unsupervised fashion.

1. Introduction

Language, especially as it relates to visual recognition, is perhaps the most important contributor to human development. It comes as no surprise, then, that systems with the ability to communicate naturally with humans have been one of the top goals of artificial intelligence research.

Most research thus far has tackled dialogue-related problems in a supervised learning setting—models are trained with a corpus of human-generated dialogues, and evaluated on their ability to regurgitate a similar response to the one in the dataset. This practice is commonplace when training machine translation systems, chatbots, visual question answering systems, and models for other tasks. However, this is not necessarily the ideal way to train, considering the vast space of possible natural language dialogues. For instance, in the chatbot setting, given some utterance, or even a complete conversation history, there are countless valid responses.

Many existing dialogue systems also lack context. Context, in human conversation, is especially important to determining the following dialogue. As such, supervised training of these systems limit the context in which these methods can respond, and as such real-world applications of such methods are delegated to menial tasks such as basic question answering, task management, or chatbots useful only for small talk.

The artificial intelligence research community has, in recent years, introduced various tasks which include context or a goal. For instance, the task of visual question answering [6], introduced in 2015, requires agents to correctly answer natural-language questions about a visual scene. This allows for more meaningful evaluation and supervised training, since the goal of correctly answering questions about an image vastly decreases the number of acceptable responses to a prompt. This trend has continued; Johnson et al. introduced CLEVR [18], a dataset which tests various aspect of visual reasoning in a natural language setting.

Notably, visual question answering tasks, such as those introduced by CLEVR, often involve simple responses, often just a few words or less, allowing them to be easily evaluated. The task of image captioning [9] [26] is an exception. In this setting, tasks are still grounded in the context of the input image, but outputs can be of arbitrary length. As such, evaluation of a model's performance on the image captioning task requires the use of contrived metrics such as the BLEU score [20] or CIDEr [24]. Research has found that

*Charles Lu is a student in CS 231N and CS 234. Both authors contributed equally apart from the reinforcement learning portion of the project, which Charles Lu contributed. A report with additional focus on visual aspects of the project has been submitted for CS 231N.

†Elias Wu is a student in CS 231N.

such evaluation metrics do not perfectly correlate with human evaluations. [8][5]

Another motivator for work on goal-driven dialogue tasks has been due to findings that previous successful visual question answering models have been able to exploit underlying biases in training data to correctly answer questions without reasoning or compositionality.[18][3][27][15] Agrawal *et al*. [4] introduced C-VQA, a compositional split of the original Visual Question Answering dataset[6], and found that many VQA models significantly decreased in performance when using this dataset.

2. Problem

In this project, we tackle GuessWhat?! [14], a collaborative game involving visual context and goal-driven dialogue. Though the task is fairly artificial, it provides an excellent test bed for research in the areas of visual understanding and reasoning, natural language dialogue generation, collaboration between agents, and reinforcement learning. Though systems for playing the entire game are complex and contain many moving parts, evaluation of a system is straightforward.

Well-performing agents on GuessWhat?! must solve several problems. Agents within the full system must be capable of understanding a visual scene, generating goal-driven questions from this visual context, correctly answer questions about an image, and ultimately determine a correct object within an image based on a question-answer dialogue.

2.1. GuessWhat?! game

GuessWhat?! is a cooperative, interactive two player game introduced by de Vries. *et al*. [14] The two places, the **questioner** and the **oracle**, are both given an image which contains a visual scene. However, only the oracle has knowledge of a previously chosen, "correct" image within the scene. The questioner must ask natural language questions, to which the oracle can only respond "Yes", "No", or "N/A". Once the questioner determines that it is ready to guess the object (or the questioner asks the maximum allowed number of questions), a set of possible objects in the scene is revealed to the questioner. In each image, the questioner must guess from a variable number of possible objects, ranging from 2 to 20.

The GuessWhat?! dataset is composed of 150K human-played games with a total of 800K visual question-answer pairs on 66K images. The images and object annotations are a subset of the Microsoft COCO dataset [19] which have a sufficient number of objects in the scene. Notably, the dataset was verified by [14] to not significantly skew the distribution of images in the original dataset.

Figure 1. A sample GuessWhat?! game. The correct object, known only to the oracle, is randomly selected from a set of between 2-20 possible objects within the image. In this case, the correct object is the bottom-most orange highlighted in the image.

2.2. Literature review

Due to the relatively new dataset, only one additional paper has been published using the GuessWhat?! dataset. Our project builds upon this follow-up research by the team which introduced the dataset.[22] They first independently train agents for each of three tasks—question generation, question answering (oracle), and guessing—with the supervised learning approach. Then, fixing the oracle and guesser models, they use a reinforcement learning approach to fine tune the question generation model.

Similar work has been done on a related task by Das *et al*., in which two agents are fine-tuned through RL techniques[13] to play an image guessing game introduced in their Visual Dialog dataset.[12].

3. Methods

3.1. Implementation

Our code is implemented in Python 3.5 with PyTorch 0.1.12.[1] We used no starter code; everything was implemented from scratch apart from a few code snippets for extending PyTorch functionality.[28][11] Our models were trained on Google Cloud virtual machines with Tesla K80 GPUs and Microsoft Azure NV6 virtual machines with Tesla M60 GPUs.

3.2. Supervised training

We first train models for the question generation, oracle, and guesser tasks through supervised learning with the examples in the GuessWhat?! dataset. Notably, the player in the questioner role is modeled by two separate models, one for question generation and one for guessing.

3.2.1 Questioner

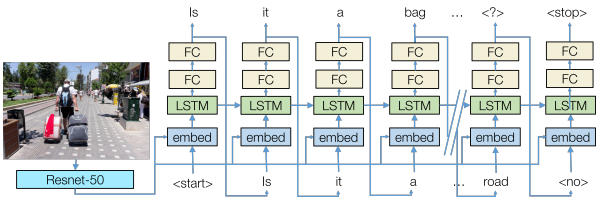

Figure 2. A well-performing architecture we experimented with for the questioner model.

The question generation model is trained to predict a new question $\boldsymbol{q}_{j+1}$ word-by-word given an input consisting of an image $\mathcal{I}$ and the previous questions and answers $(\boldsymbol{q}, a)_{1:j}$. Specifically, it models a probability distribution

$$p(w_i^j | w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}.$$

We implement our model as a sequence-to-sequence model.

Given an image $\mathcal{I}$, our model first obtains its ResNet-50 features[16] (a 2048-dimensional vector), whose weights are fixed (the model proposed in [14] uses VGG16 FC8 features[21], which is a 4096-dimensional vector). The ResNet-50 features are concatenated to the input to the model's recurrent cell at each time step.

A recurrent neural network using gated recurrent units (GRU)[10] or long short-term memory units (LSTM)[17] is then used to generate questions word by word. The input to the RNN is initialized to the start token `<start>` for the first time step. During training time, the inputs and outputs are entire dialogues from the training set (with the outputs shifted by one time step). The embedding for the ground truth word at the previous time step is used as input in future time steps. During evaluation, this embedding is simply the previously chosen word.

When the previous output is the `<?>` token representing the end of the previous question, the embedding corresponding to the answer is used. During training time, this is the answer given in the dataset; during evaluation, the answer returned by the oracle is used. Finally, the `<stop>` token represents that the questioner is done asking questions and is ready to guess the object.

The output of each recurrent cell is fed into a fully connected network followed by a softmax layer to form a probability distribution over possible tokens in the vocabulary.

The question generation model is trained by minimizing the total cross-entropy loss for each token in each question:

$$\mathcal{L}_Q = -\log p(\boldsymbol{q}_{1:J} | a_{1:J}, \mathcal{I}) \quad (1)$$

$$= -\sum_{j=1}^{J} \sum_{i=1}^{I_j} \log p(w_i^j | w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}) \quad (2)$$

During evaluation, we implement random sampling as well as a greedy approach, which simply selects the token with the highest predicted probability. Beam search was found by [22] to lead to worse results than random sampling, though their architecture was simpler. Though the questioner is trained on entire dialogues during training time, during evaluation, the hidden state of the questioner when it terminates a question with `<?>` and we extract the question to be answered by the oracle.

3.2.2 Oracle

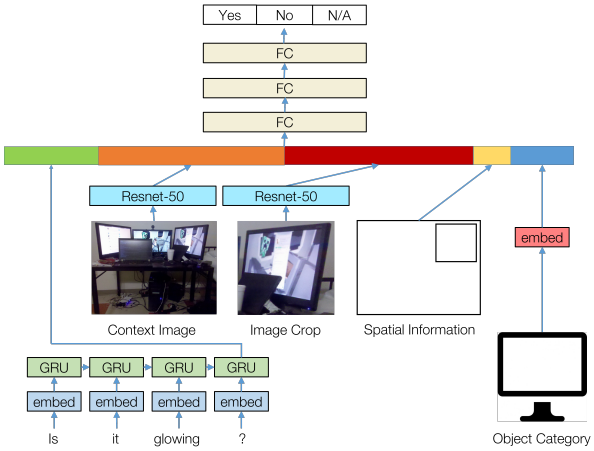

Figure 3. A well-performing architecture we experimented with for the oracle model.

The oracle is trained to predict one of the three answers ("Yes", "No", or "N/A") given an input consisting of an image, the bounding box and category of the correct object, and the question itself. In particular, the model models a distribution over the three possible answers:

$$p(a | \mathcal{I}, \boldsymbol{q}, c^*, x_{box}).$$

Note that we did not experiment with including past question-answer history as input to the oracle model, and instead only input the most recent question.

We implement the oracle model with a fully connected neural network with several hidden layers and softmax output which takes up to five concatenated inputs—the encoded question, the object category embedding, the

image features, the image features of the cropped object (using its bounding box annotation), and spatial information representing the object's bounding box. The question is encoded using a GRU-based recurrent network. We also use a trainable embedding for the object's category. As before, the image features are extracted from ResNet-50. Finally, the bounding box is a vector $[x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}] \in \mathbb{R}^8$, where the image coordinates are normalized to be in the range $[-1, 1]$.

de Vries *et al*. [14] did some work in experimenting with performance of the oracle given different subsets of the available information (the question, image, object category, object crop, and object spatial information). Their work did not include a model which included all five pieces of information (only models which used up to four). Notably, they found that excluding the object crop features from the input to oracle model resulted in better performance!? Though excluding the object crop reduces the model's ability to answer questions about the object's color and texture, they hypothesize that the better results are a result of imperfect feature extraction from the crop, which can often be from a very small bounding box.

For our oracle model, we implemented a model which uses all five pieces of information as input, in addition to an oracle "lite" model which does not use the object crop. Similar to the previous results, our oracle "lite" model performs on par with our full oracle model. Therefore, we use the oracle "lite" model for all future experiments for the task as a whole due to increased efficiency and decreased memory and image preprocessing requirements.

3.2.3 Guesser

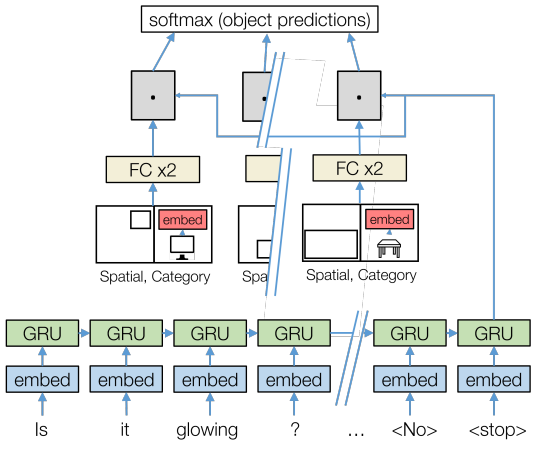

Figure 4. A well-performing architecture we experimented with for the guesser model. Our final model uses 2 GRU layers instead of the single layer shown.

The guesser is trained to predict the correct object, given the input image, dialogue of questions and answers, and information about each possible object.

First, the question-answer sequence is encoded using a GRU-based recurrent network. we also compute an embedding for each of the possible objects. The embedding for the objects in the image are calculated from the spatial representation as mentioned in the oracle model, as well as new trainable embedding for the object's category. More precisely, this embedding is created by concatenating the trainable category embedding with the bounding box representation. Each concatenated vector is then input to a fully connected layer to generate the final embedding for the objects. Similarly to the model for the oracle, we do not include the object crops in the computation of the embedding.

We then compute the dot product between the output of the hidden state of encoder at the last time step and the embedding for each objects in the image. The result of this dot product for each object is fed into a softmax layer, which computes a probability distribution over the objects.

The structure of the guesser's task presented a difficult engineering challenge. Not only are the dialogues variable length like the oracle and questioner, but the guesser also has to deal with a variable number of possible objects in an image. Beyond implementing a dynamic computation graph, one way we addressed this issue while still allowing for the efficiency of batching was to implement a `CurriculumDataLoader`, which batched training examples in ascending order of number of possible objects. Another positive side effect of this implementation was to train the guesser with curriculum learning [7], first showing the guesser simpler problems with fewer possible objects before moving on to more difficult problems with up to 20 possible objects.

3.3. Fine tuning with reinforcement learning

After training the question generation, oracle, and guesser models, the parameters for the oracle and guesser models are fixed. The trained oracle and guesser are then used to tune the question generation model in a reinforcement learning setting.

3.3.1 Markov decision process formulation

The GuessWhat?! game can be framed as a Markov decision process.[22]

The state $\boldsymbol{s}_t$ at some point in the game is a tuple containing the image, all previous question-answer pairs, and all words already generated in the current question utterance, i.e. $\boldsymbol{s}_t = ((w_1^j, \ldots, w_i^j), (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I})$ where j is the current question.

Each action u_t is one of the words in the vocabulary, or the `<stop>` or `<?>` token.

The questioner therefore models a stochastic policy $\pi_\theta(u_t|\boldsymbol{s}_t)$ where θ refers to the parameters of the underlying neural network.

Each state-action pair is assigned a reward $r(\boldsymbol{s}_t, u_t)$. We use a zero-one reward function suggested by [22] which requires minimal prior knowledge:

$$r(\boldsymbol{s}_t, u_t) = \begin{cases} 1 & \text{if } \arg\max_o(Guesser(\boldsymbol{s}_t)) = o^* \text{ and } t = T \\ 0 & \text{otherwise} \end{cases}$$

where T is the length of the trajectory, i.e. the questioner receives a reward of 1 in the last question token it outputs if the guesser chooses the correct object.

3.3.2 Policy gradient fine tuning

As with [22], we elect to use policy gradient to fine tune the questioner due to the large action space of over 2000 possible words. We update the parameters of the questioner network θ to maximize the expected return

$$J(\theta) = E_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} r(\boldsymbol{s}_t, u_t)\right]$$

where $\gamma \in [0, 1]$ is the discount rate and the initial state $\boldsymbol{s}_t$ is drawn from some prior distribution over possible images and objects.

The gradient $\nabla J(\theta_h)$ at time step h is estimated from a batch of sample trajectories $\mathcal{T}_h$ sampled from the policy π_{θ_h}:[23]

$$\nabla J(\theta_h) = \left\langle \sum_{t=1}^{T} \sum_{u_t \in V} \nabla_{\theta_h} \log \pi_{\theta_h}(u_t|\boldsymbol{s}_t)(Q_{\pi_{\theta_h}}(\boldsymbol{s}_t, u_t) - b) \right\rangle_{\mathcal{T}_h}$$

We use the REINFORCE algorithm[25], which removes the inner sum over actions:

$$\begin{aligned} \nabla J(\theta_h) &= \left\langle \sum_{t=1}^{T} \nabla_{\theta_h} \log \pi_{\theta_h}(u_t|\boldsymbol{s}_t)(Q_{\pi_{\theta_h}}(\boldsymbol{s}_t, u_t) - b) \right\rangle_{\mathcal{T}_h} \\ &= \left\langle \sum_{j=1}^{J} \sum_{i=1}^{I_j} \nabla_{\theta_h} \log \pi_{\theta_h}(w_i^j | w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}) \right. \\ &\qquad \left. (Q_{\pi_{\theta_h}}((\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}), w_i^j) - b) \right\rangle_{\mathcal{T}_h}. \end{aligned} \tag{3}$$

Previous work [22] has used a one layer fully connected network to estimate the baseline b (for variance reduction). The network is trained with an L2 loss with respect to the discounted reward of the trajectory starting at each time step. In our case, we simply use a moving average to maintain the baseline, which seems to work well.

Model	Train	Val	Test
1 GRU, 2 FC	78.94%	73.63%	73.62%
1 GRU, 2 FC+dropout	65.28%	64.55%	64.65%
1 GRU, 3 FC	79.95%	75.5%	75.23%
2 GRU, 3 FC	80.02%	75.78%	**75.44%**
Dominant class	52.6%	53.8%	49.1%
Previous best	82.8%	78.9%	**78.5%**

Table 1. Oracle model accuracy. Accuracy figures for "dominant class" and "previous best" are from [14].

Model	Train	Val	Test
1 GRU, 2 FC	64.21%	51.44%	51.58%
2 GRU, 2 FC	64.04%	58.69%	**58.84%**
2 GRU, 2 FC+dropout	58.31%	48.72%	50.26%
3 GRU, 2 FC	52.47%	48.03%	49.47%
1 LSTM, 2 FC	68.54%	56.43%	51.32%
2 LSTM, 2 FC	60.28%	51.71%	45.53%
Random	17.1%	17.1%	17.1%
Human	91.0%	90.8%	90.8%
Previous best	72.1%	62.1%	**61.3%**

Table 2. Guesser model accuracy. Accuracy figures for "previous best" are from [14], with "random" and "human" inferred from the dataset.

4. Experiments and results

In training the oracle model, we determined that using two GRU layers as well as three fully connected layers produced best validation results. Accuracy was determined by how many questions the model answered correctly. We achieved within 5% of state-of-the-art results.

· A full table of our results on small changes to the oracle model is presented in table 4.

In training the guesser model, we determined that using two GRU layers for the question/answer pairs and two fully connected layers after concatenating the category embedding and spatial information produced the best validation results. Accuracy was determined by the number of times the guesser guessed the correct object, given human-generated dialogue in the GuessWhat?! dataset. A table of our results for the guesser model is presented in table 4.

In training the questioner model, we found that an architecture with 1 LSTM layer followed by a two-layer fully connected net for each output performed well. We also found that fine-tuning with reinforcement learning significantly improves results.

The questioner is difficult to evaluate, since it cannot be accurately evaluated in a supervised fashion. For instance, simply attempting to match human-generated questions in the GuessWhat?! dataset is a poor metric of a questioner's performance. Therefore, we evaluate the questioner by fixing our best oracle and guesser models, and reporting the accuracy the guesser achieves using questions generated by

Model	Seen image, seen object	Seen image, new object	Unseen image
1 LSTM, 2 FC, greedy	26.1%	22.4%	20.1%
1 LSTM, 2 FC, sample	20.3%	22.6%	20.7%
1 LSTM, 2 FC, sample, fine-tuned	28.7%	25.2%	**27.0%**
Previous best	N/A	N/A	**34.0%**
Random	N/A	N/A	17.1%

Table 3. Questioner model accuracy. No figures are included for work in [22] since the authors compare questioner performance to an unknown human performance baseline.

the questioner. A table of our results for the questioner model is presented in table 4.

5. Discussion

Our results indicate that it is possible to train well-performing models for all tasks required to play GuessWhat?!. In particular, we achieve results within 5% of the state-of-the-art for both the oracle and guesser tasks. This is especially impressive given that our models were trained for at most 20 epochs within a few hours each.

Our best questioner model generated questions which allowed our best guesser model to guess the correct object over 20% of the time. This is particularly notable, as this required not only a trained questioner model, but also a trained oracle and guesser model to work together in tandem. As such, evaluating the questioner model requires evaluating not only the questioner by itself but the system containing all three models as a whole.

Even more notably, our results corroborate the hypothesis that models trained in a supervised fashion can be used to further fine-tune the system's overall performance in an unsupervised reinforcement learning setting. For instance, by fixing the oracle and guesser models and treating them as ground truth (even though they are far from ground truth in practice) allowed us to improve the questioner such that the overall system performance increased by almost 7%. The questioner training and fine-tuning process was also limited to less than three hours; furthermore, the fine-tuning process was limited by a PyTorch memory leak issue which significantly decreased the pace of fine-tuning. Our minimal training time indicates that there is still room for further improvement, even with the same models and techniques.

5.1. Future work

Future work which would likely immediately lead to improved results is training and fine-tuning for extended periods of time. In addition to training for longer, another obvious avenue for future work would be to increase model capacities. For instance, when preprocessing the vocabulary, we only use the top $\tilde{2}700$ most common words, whereas previous work [14] used over 5000. We also hope to design more complex models which allow additional information, such as object crops for the guesser and previous question-answer tuples for the oracle, to be input without decreasing performance due to noise. Perhaps most worrying is the fact that our current guesser model (as well as the guessers introduced in previous work) do not include image information at all; though previous work found this lack of image information to not be significantly detrimental, we believe it is low-hanging fruit.

Beyond simply swapping out and plugging in new "Lego blocks" in our model architectures, we hope to further address more fundamental design challenges. One particular avenue we plan to explore is the use of Gumbel-softmax as opposed to traditional softmax, which may allow a better formulation of the loss for the questioner. Furthermore, in the reinforcement learning setting, we hope to consider better approaches for addressing the scalability of the approach and sparsity of the reward. In our case, with an action space of size less than 3000, our current policy gradient approach is still viable; however, for problems with even larger action spaces, scalability is a significant problem.

We use PyTorch as it provides numerous advantages over alternative frameworks such as TensorFlow [2], such as dynamic computation graphs which significantly simplify code and increase speed of development. Nevertheless, PyTorch is still very new, and as such, lacks certain capabilities. Most notably, in our case, fine-tuning the questioner with REINFORCE gave rise to a nondeterministic internal cuDNN memory leak, and our workarounds significantly decreased the pace of fine-tuning. We also needed to implement other functionality like TensorFlow's `gather_nd`, a masked cross-entropy loss for our sequence-to-sequence model for the questioner [11], and a method for extracting final hidden states from PyTorch `PackedSequence` objects [28]. In many of these cases, relevant issues were last updated within the last week. We hope that as PyTorch quickly develops and addresses these issues, our implementations can be improved.

Acknowledgments

The authors would like to thank the teaching staff of both courses—CS 231N and CS 234—for offering a captivating and organized course. We especially enjoyed the poster sessions and appreciate the feedback we received. Finally, we

appreciate the generous donation of computation resources by Microsoft Azure and Google Cloud, without which our models could not have been trained in time.

References

[1] Pytorch, 2017.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] A. Agrawal, D. Batra, and D. Parikh. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*, 2016.

[4] A. Agrawal, A. Kembhavi, D. Batra, and D. Parikh. C-vqa: A compositional split of the visual question answering (vqa) v1.0 dataset. *arXiv preprint arXiv:1704.08243*, 2017.

[5] P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer, 2016.

[6] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[8] C. Callison-Burch, M. Osborne, and P. Koehn. Re-evaluation the role of bleu in machine translation research. In *EACL*, volume 6, pages 249–256, 2006.

[9] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[10] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[11] J. Choi. Pytorch workaround for masking cross entropy loss. `https://gist.github.com/jihunchoi/f1434a77df9db1bb337417854b398df1`, 2017.

[12] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra. Visual dialog. *arXiv preprint arXiv:1611.08669*, 2016.

[13] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra. Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*, 2017.

[14] H. de Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville. Guesswhat?! visual object discovery through multi-modal dialogue. *arXiv preprint arXiv:1611.08481*, 2016.

[15] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *arXiv preprint arXiv:1612.00837*, 2016.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *arXiv preprint arXiv:1612.06890*, 2016.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[20] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] F. Strub, H. de Vries, J. Mary, B. Piot, A. Courville, and O. Pietquin. End-to-end optimization of goal-driven and visually grounded dialogue systems. *arXiv preprint arXiv:1703.05423*, 2017.

[23] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[24] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

[25] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[26] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[27] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh. Yin and yang: Balancing and answering binary visual questions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5014–5022, 2016.

[28] H. Zheng. add functions in nn.utils.rnn to get the last step of a packedsequence object - pytorch pull request #1375. `https://github.com/pytorch/pytorch/pull/1375/files`, 2017.