VISUAL REASONING USING RECURRENT ATTENTION

Zhiyao Yan
University of California, San Diego
zhy038@ucsd.edu

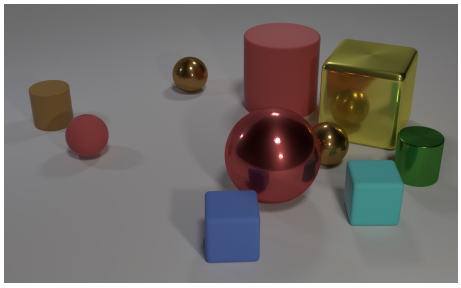
ABSTRACT

We look at recurrent attention-based approaches to visual reasoning. Visual reasoning tasks such as the CLEVR dataset involves engaging composition recognition and multiple reasoning steps, which previous visual question answering techniques have struggled with. We implement and examine 1) an LSTM with attention model and 2) Memory, Attention, and Composition (MAC) networks on the Sort-of-CLEVR dataset, a simpilified version of CLEVR, and evaluate their multi-step visual reasoning capabilities. We observe that the two models achieve subpar performance on the Sort-of-CLEVR dataset, achieving 78% and 66% test accuracy respectively.

1 INTRODUCTION

Visual reasoning differs from standard visual question answering tasks in that it requires consideration of relationships between objects and multi-step reasoning. These tasks typically present an image and natual language question regarding the image. However, Agrawal et al. (2016) reveal many previous techniques for visual question answering learn to exploit features of the image and questions and do not learn to properly "reason." The CLEVR (Johnson et al., 2016) dataset was specifically designed to test visual reasoning capabilities by presenting an image of various rendered 3D shapes and questions testing "attribute identification, counting, comparison, spatial relationships, and logical operations."

We follow the intuition that reasoning tasks like counting involves multiple "reasoning" steps and attribute identification requires attention, hence our interest in recurrent attention techniques. We examine two such techniques: 1) we propose and implement an LSTM with attention on image features and 2) Memory, Attention, and Composition (MAC) networks (Hudson & Manning, 2018). MAC networks have achieved a state-of-the-art accuracy of 98.9% accuracy on the CLEVR dataset. However, the CLEVR dataset is extremely large and training takes more time and resources than was available for this project, so we instead examine performance on the simpler Sort-of-CLEVR dataset originally proposed by Santoro et al. (2017).



Are there an equal number of large things and metal spheres?

Figure 1: CLEVR example.

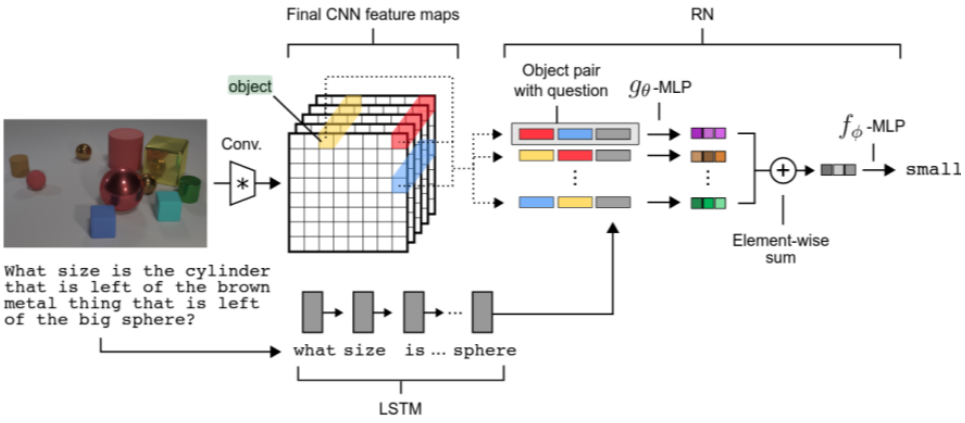


Figure 2: Relation Network architecture. Image from Santoro et al. (2017)

1.1 Relation Networks

Our project idea was originally inspired by the at-the-time state-of-the-art results achieved by Relation Networks (RN) on CLEVR (Santoro et al., 2017).

RN first extracts image features using a simple 4 layer convolutional network with 24 kernels each resulting in a $d \times d \times 24$ feature map, which is then interpreted as d^2 object vectors o of dimension 24. Questions are embedded as vectors q using an LSTM. Each object-pair is then fed into a simple MLP $g_\theta(o_i, o_j, q)$ to calculate a "relationship" between all objects. The resulting vectors are then summed together and fed into another MLP $a = f_\phi(\sum_{i,j} g_\theta(o_i, o_j, q))$ resulting in the logits used for softmax prediction of the answer. Essentially, RN considers the relationship between every pair of objects and their alignment with the question.

Unfortunately, RN has the drawback of having to consider every object-pair relationship, akin to a "brute force" solution. This prompted us to study the possibility of using attention to select only relevant object-pairs and then perhaps operating on these object-pair proposals over multiple time steps, which brings us to the LSTM with attention model.

However, after multiple iterations of the LSTM with attention model we were still not achieving desirable accuracy. We then found MAC networks which is also a recurrent attention model but achieves the current state-of-the-art performance on CLEVR. We examine both the LSTM with attention model and MAC networks and compare their performance against Relation Networks.

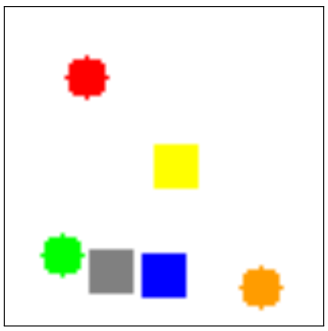


Figure 3: Sort-of-CLEVR example.

Table 1: Feature extractor for LSTM with attention

Bilinear resize	$64 \times 64 \times 24$
Conv2D, 24 kernels, size 5, stride 2	$32 \times 32 \times 24$
Conv2D, 24 kernels, size 3, stride 2	$16 \times 16 \times 24$
Conv2D, 24 kernels, size 3, stride 2	$8 \times 8 \times 24$

2 METHODS

2.1 SORT-OF-CLEVR

Due to time and resource constraints, we experiment with the simple Sort-of-CLEVR dataset instead of CLEVR. We use the implementation of the data generator found at `https://github.com/kimhc6028/relational-networks` (Heecheol). The dataset generates two different types of questions: relational and non-relational questions. We only use relational questions in our experiments. Within relational questions, there are 3 subtypes asking for the:

1. Shape of the object which is closest to the certain colored object
   e.g. *What is the shape of the object closest to the red object? — square*
2. Shape of the object which is furthest to the certain colored object
   e.g. *What is the shape of the object furthest to the orange object? — circle*
3. Number of objects which have the same shape with the certain colored object
   e.g. *How many objects have same shape with the blue object? — 3*

Please refer to Figure 3 for the above questions.

Questions are encoded as binary vectors $q \in \mathbb{R}^{11}$ with 6 positions as the one-hot embedding for a color, 2 positions as the one-hot embedding indicating relational or non-relational question, and 3 positions as the one-hot embedding of the subtype. Similarly, answer vectors are encoded as $y \in \mathbb{R}^{10}$ one-hot embedding representing color or shape.

We generate 9,800 image samples with 10 relational questions per image for the training set, resulting in a total of 98,000 training samples. Similarly, we generate 400 image samples for the test set, resulting in 4,000 test samples. Images are 75×75 and RGB values are normalized to [0,1].

2.2 LSTM WITH ATTENTION

Inspired by the success of recurrent attention models for image descriptions (Xu et al., 2015), we attempt to design and train LSTM with image feature attention. We also note that our architecture is also similar to that presented by Yang et al. (2015). We extract image features k using a 4 convolutional layers with ReLU activation as described in Table 1. These image features are then fed to a sequence of LSTM with attention (AttLSTM) cells.

A diagram of the AttLSTM cell can be seen in Figure 4. The idea behind our design is to use the current hidden state of the LSTM and the question vector q to generate a attention map to be applied

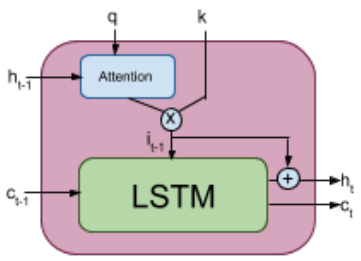


Figure 4: An LSTM with attention cell.

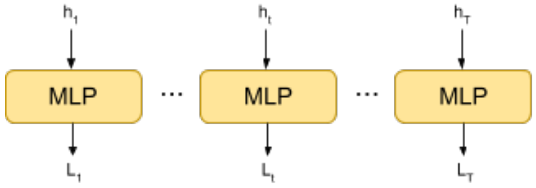


Figure 5: AttLSTM loss calculation

to the image features k. Borrowing terms from Hudson & Manning (2018), we hope that the LSTM hidden and cell state can be used as part of a "controller" and "reader" to attend to certain features in k.

We define attention as $a_t = f(h_{t-1}, q) \in \mathbb{R}^{d^2}$ which we then reshape to $a \in \mathbb{R}^{d \times d}$ to apply to k. f consists of 3 fully connected layers with 256, 128, and 64 units using ReLU activations for layers 1 and 2 and softmax for the last layer.

The attended image feature is calculated as $i_t = g(\sum_{i,j} a_{t,i,j} k_{i,j})$ and fed as input to the LSTM where g is a fully connected layer with ReLU activation. g has d units where d is the number of hidden units in the LSTM.

The output of the AttLSTM cell is then calculated as $h_t = h'_t + i_t$ where h'_t is the hidden state of the LSTM. h_0 is simply a zero vector.

Each time step output is then fed to 3 fully connected layers p of 128, 128, and 10 units with ReLU activations and a final softmax activation to generate the logits for prediction $\hat{y}_t = p(h_t)$. p shares weights across all time steps. The total loss is $L = \sum_t c_t L_t$ where c_t is some hyperparameter constant and L_t is the time step cross entropy loss $L_t = -\sum_i y_i log \hat{y}_i$. Unrolling the network, AttLSTM networks can be thought of as a deeply-supervised network (Lee et al., 2014) with weight sharing. In our experiments we use $c = [0.8, 0.8, 0.5, 0.5]$.

Dropout is applied to g, the LSTM, and p.

We train using Adam (Kingma & Ba, 2014) optimization with default hyperparameters and learning rate 0.001. We clip gradient magnitudes to 10. We train using batch size of 100. All weights are initialized with MSRA initialization (He et al., 2015a). We train using 4 time steps.

2.3 Memory, Attention, and Composition (MAC) network

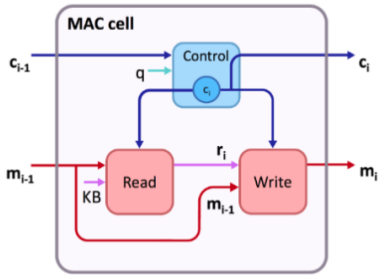


Figure 6: **The MAC cell architecture**.The MAC recurrent cell consists of a control unit, read unit, and write unit, that operate over dual control and memory hidden states. The **control unit** successively attends to different parts of the task description (question), updating the control state to represent at each timestep the reasoning operation the cell intends to perform. The **read unit** extracts information out of a knowledge base (here, image), guided by the control state. The **write unit** integrates the retrieved information into the memory state, yielding the new intermediate result that follows from applying the current reasoning operation. *Image and description from Hudson & Manning (2018)*

Similar to our proposed model, Hudson & Manning (2018) propose MAC cells which are capable of attention on both image and questions sequences. MAC cells consist of a controller, read, and write unit and can be stacked together to form chains of "reasoning." MAC networks currently achieve state-of-the-art performance on CLEVR. Please see the paper for more specific implementation details.

Some key differences between our AttLSTM cell and MAC cells are

1. MAC cells do not use LSTM. Instead uses their own read and write units which control memory updates.
2. No intermediate time step losses. The loss is only calculated using the output of the final time step.

Some other differences include more prominent use of input concatenations and fully connected layers with linear activations. Another feature of MAC is the ability to attend to individual words in the question, rather than only on a sentence-level embedding.

Hudson & Manning (2018) extract CLEVR image features k using *conv4* features of ResNet101 (He et al., 2015b) pretrained on ImageNet. However this is not necessary for the much simpler Sort-of-CLEVR dataset so we instead use 4 convolutional layers as described in Table 2. d is a hyperparameter.

We test using our own implementation of MAC cells, closely following the descriptions of the paper and public implementation[1]. Because Sort-of-CLEVR questions are already encoded as binary vectors q, we can not use the sentence word attention portion of the MAC controller. Instead, we implement our controller as a single d unit fully connected layer with linear activation.

$$q' = tanh(W^{d\times 11}q + b^d)$$
$$c_i = W^{d\times d}q' + b^d$$

The weights W used to calculate q' are not shared across MAC cells.

Our read unit follows the same implementation described by Hudson & Manning (2018). $\odot$ is element-wise multiplication.

$$I_{i,h,w} = [W_m^{d\times d}m_{i-1} + b_m^d] \odot [W_k^{d\times d}k_{h,w} + b_k^d]$$
$$I'_{i,h,w} = W^{d\times 2d}[I_{i,h,w}, k_{h,w}] + b^d$$
$$ra_{i,h,w} = W^{d\times d}(c_i \odot I'_{i,h,w}) + b^d$$
$$rv_{i,h,w} = softmax(ra_{i,h,w})$$
$$r_i = \sum_{h,w,=1,1}^{H,W} rv_{i,h,w} \odot k_{h,w}$$

Our write unit does not use the optional self-attention and memory gate steps described in Hudson & Manning (2018).

$$m_i^{info} = W^{d\times 2d}[r_i, m_{i-1}] + b^d$$

[1] `https://github.com/stanfordnlp/mac-network`

Table 2: Feature extractor for MAC network

Bilinear resize	$64 \times 64 \times 24$
Conv2D, 24 kernels, size 5, stride 2	$32 \times 32 \times 24$
Conv2D, d kernels, size 3, stride 2	$16 \times 16 \times d$
Conv2D, d kernels, size 3, stride 2	$8 \times 8 \times d$

$$m_i = W^{d \times 3d}[m_{i-1}, m_i^{info}, m_{i-1} \odot m_i^{info}] + b^d$$

The concatenation used to calculate m_i is not mentioned in the paper but is used in the authors' public implementation.

Following Hudson & Manning (2018), we can learn the initial memory state m_0 through standard backpropagation.

The logits are generated by $f(q, m_p)$ where m_p is the final memory state and f is 2 fully connected layers where the first layer has 128 units with ELU activation, and the second layer has 10 units with softmax activation. We calculate the loss using multiclass cross entropy.

Dropout is applied throughout the cell and output MLP.

We follow the implementation and train using Adam (Kingma & Ba, 2014) optimization with default hyperparameters and learning rate 0.0001. We clip gradient magnitudes to 7. We train using batch size of 100. All weights are initialized with MSRA initialization (He et al., 2015a). We train using 5 time steps.

3 RESULTS

Table 3: Relational question test results

Image Size	Model	Test Accuracy
75×75	CNN+MLP baseline	0.66*
	Relation Network	0.89*
	AttLSTM $d = 64$	0.462
	AttLSTM $d = 128$	0.780
	MAC Network $d = 64$	0.096
	MAC Network $d = 128$	0.664
128×128	AttLSTM $d = 64$	0.834
	AttLSTM $d = 128$	0.835
	MAC Network $d = 64$	0.745
	MAC Network $d = 128$	0.457

*Result taken from Heecheol

We found that neither of the two recurrent models are able to beat Relation Network's accuracy.

We notice that the circles in the original Sort-of-CLEVR dataset look very similar squares, and hypothesis that it might be making it harder for the feature extractor to distinguish between. To get around this, we generate a second set of data with the same number of training and test data but with 128×128 dimension images. We replace the Bilinear resize in the AttLSTM and MAC features extractors with a Conv2D, 24 kernels, size 5, stride 2 layer.

We train AttLSTM networks for 30,000 iterations and MAC Networks for 60,000 iterations.

We train AttLSTM and MAC networks both with $d = 64$ and $d = 128$. We noticed that both AttLSTM and MAC Networks sometimes will converge early on and never achieve good performance, depending on the dimension of the weights used. Some variants of MAC networks will converge early on even with multiple restarts, and can be seen in the plots in Figure 7. We hypothesize that training these recurrent networks can be unstable and some random seeds are better than others, much like in reinforcement learning. However, proper tuning of regularization like dropout may stabilize training.

We also note that prediction loss does not improve with increasing timestep in our AttLSTM model (Figure 8). This implies that the prediction of the first time step is just as good as the fourth time step.

We note that in both AttLSTM and MAC networks the test accuracy saturates early on and the network begins overfitting in the training data.

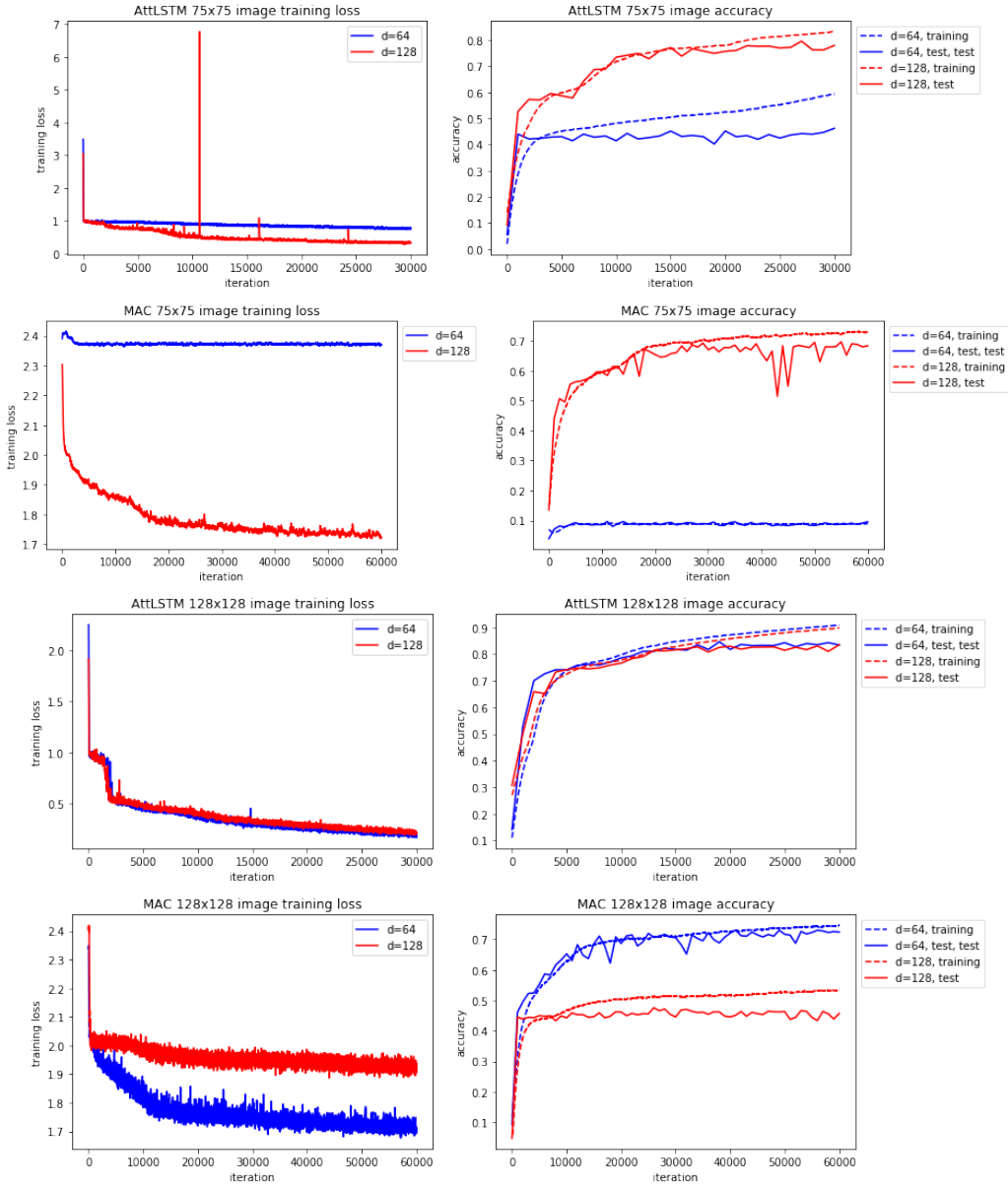


Figure 7: Losses and accuracy plots. We plot using smoothed losses and accuracy.

4 Discussion and Conclusion

There are several discrepancies in test accuracy that should be addressed. While MAC networks report the current state-of-the-art results on CLEVR, our emperical results show that their performance is rather poor Sort-of-CLEVR. This could be due to muliple reasons. 1) Sort-of-CLEVR is not representative of visual reasoning tasks. We see that Relation Networks also perform worse on Sort-of-CLEVR than on CLEVR (89% vs 95%) despite CLEVR being a much more difficult dataset. 2) Our implementation of MAC networks is not properly tuned, and/or has implementation flaws. 3) We do not use enough weights in our implementation. The public implementation of MAC networks use $d = 512$ while we only try $d = 64$ and $d = 128$ due to computational and time constraints. 4) We train our own feature extractor rather than use a pretrained ImageNet classifier. 5) We do not use all techniques described by Hudson & Manning (2018) such as sentence word attention or memory gating.

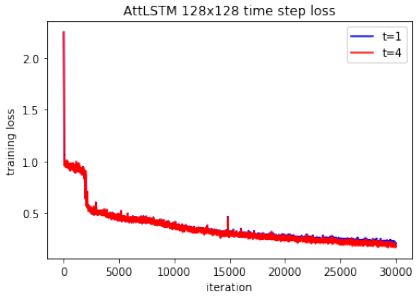


Figure 8: Loss at $t = 1$ vs $t = 4$

It should also be noted that while AttLSTM performs better than MAC networks in our experiments, we did not evaluate the performance of AttLSTM on the full CLEVR dataset, so conclusions can not be drawn about its ability to perform relational reasoning. In fact, our experiments show that AttLSTM prediction quality does not actually improve with increasing number of time steps, suggesting our network is not performing multi-step reasoning. Futher experiments will also be needed training with different number of time steps for both AttLSTM and MAC networks.

Overall we see that recurrent attention models did not perform as well as Relation Networks in our experiments.

REFERENCES

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *CoRR*, abs/1606.07356, 2016. URL http://arxiv.org/abs/1606.07356.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015a. URL http://arxiv.org/abs/1502.01852.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015b. URL http://arxiv.org/abs/1512.03385.

Kim Heecheol. Pytorch implementation of "a simple neural network module for relational reasoning" (relational networks). URL https://github.com/kimhc6028/relational-networks.

Drew A. Hudson and Christopher D. Manning. Compositional attention networks for machine reasoning. *CoRR*, abs/1803.03067, 2018. URL http://arxiv.org/abs/1803.03067.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. URL http://arxiv.org/abs/1612.06890.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. 2014. URL https://arxiv.org/abs/1409.5185.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. *CoRR*, abs/1706.01427, 2017. URL http://arxiv.org/abs/1706.01427.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation

with visual attention. *CoRR*, abs/1502.03044, 2015. URL http://arxiv.org/abs/1502.03044.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. Stacked attention networks for image question answering. *CoRR*, abs/1511.02274, 2015. URL http://arxiv.org/abs/1511.02274.