Image Titles - Variations on Show, Attend and Tell

Vincent Sitzmann **Timon Ruban** **Robert Konrad**

Abstract

Inspired by recent advances in machine translation and object detection, we implement an image captioning pipeline, consisting of a Fully Convolutional Neural Network piping image features into an image-captioning LSTM, based on the popular Show, Attend, and Tell model. We implement the model in TensorFlow and recreate performance metrics reported in the paper. We identify and experiment with variations on the model, and evaluate them via a series of experiments on the MS COCO benchmark dataset.

1 Introduction

The concept of image captioning, generating a single sentence describing the content of a previously unseen image, is very near to the heart of scene understanding. The problem is very difficult at its core, as computer vision algorithms must first recognize objects within a scene and then understand and output the relationship between them in natural language. Machine learning, specifically deep learning, algorithms have made significant progress in mimicking the ability of humans to boil down huge amounts of salient information into natural language.

We propose modifications to the seminal model in deep image captioning, the "Show, Attend and Tell" [22] model. The original model is structured as an image-ingesting Fully Convolutional Neural Network (FCNN). This FCNN is used as a semantic feature extractor and outputs a feature map that is then fed into a caption-producing LSTM. At each timestep, the LSTM receives a weighted sum of the image features as input and can thus decide only to attend to the parts of the image that are relevant to the current word.

We implement this model ourselves in TensorFlow, reproducing baseline results, and experiment with a number of possible improvements. By replacing the VGG feature extractor with an Inception feature extractor, we were able to basically reproduce the soft-attention results of Show, Attend, and Tell (SAT) without having to resort to beam search to boost performance like they do in the paper. We also investigate constraining the feature space of the feature extractor to be the space of word vectors, expecting it to be more compatible with the image-captioning LSTM, which turns out not to be the case.

2 Related Work

Image captioning has been a long-standing problem at the interface of computer vision and NLP. However, models developed in the "pre-deep-learning" era struggled with poor generalization and their limited capability to express complex language in rule-based systems ([12], [2], [6]).

In recent years, deep learning has rapidly taken over the field of semantic image understanding. The seminal paper on successfully using convolutional neural networks (CNNs) for image understanding is the work by Krizhevsky et al. [11]. Since then, CNNs have achieved state-of-the-art performance on problems such as scene segmentation ([3], [14]), single-image depth estimation ([7]), and many other classic imaging problems.

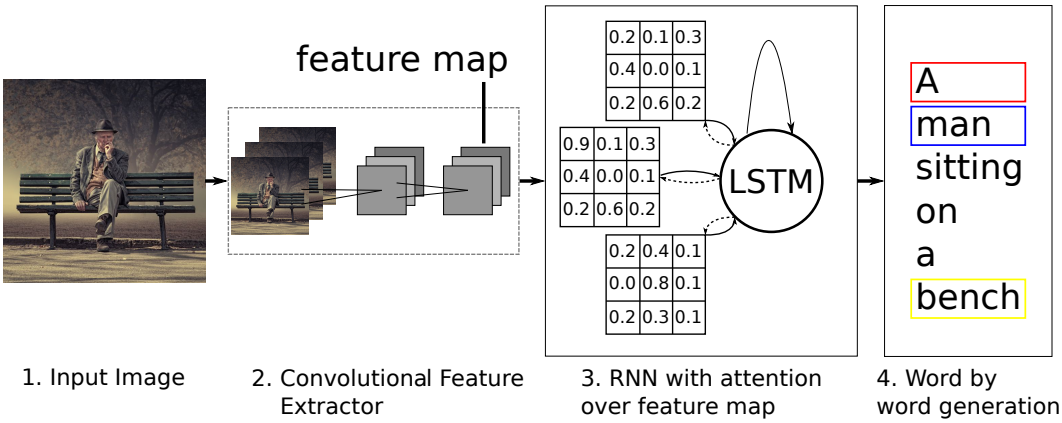


Figure 1: High level overview. An image is fed into a CNN which extracts word vectors pertinent to the image, and then uses an RNN with attention to generate a sentence.

Similarly, deep neural networks are the dominating architecture in the field of natural language processing (NLP). One notable architecture is the Long-Short Term Memory (LSTM) network [8]. LSTMs have been successfully employed in language translation [5], sentiment prediction [17], natural language inference [19], etc.

Image captioning is at the interface of these two disciplines. In this project, we build on the model proposed by Vinyals et al. [18]. This paper first paired a CNN with a LSTM to learn rich features for both images and language. This seminal work has essentially been preserved in following work and improved upon with visual attention models (Show, Attend and Tell, [22]) or dense captioning of several instances of interest in an image [9]. Seeing that this is still the dominant architecture in image captioning, we take the same approach.

Similar progress has been made in the field of visual question answering ([21], [20]). The current state-of-the-art model in image captioning still uses the same underlying architecture, but uses reinforcement learning to represent richer loss functions [15].

3 Dataset

We learn image captioning on the dataset from the popular MS COCO Image Captioning Challenge [13]. We use the same split as in [10], which leaves us with 113,287 images in the training set and 5,000 images in both the test and validation set and after constraining the annotations to be of maximum length 20 we get a total of 609,860 annotations (4-5 captions per image). Our vocabulary size is 10,203 (including special tokens like <UNK>, <START>, <END>and <NULL>) after replacing all words that occur five or fewer times in the annotations with <UNK>.

4 Approach

A high-level overview of the complete architecture can be seen in Figure 1.

4.1 Feature-extracting Convolutional Neural Network

Handing images directly to a language-modeling LSTM would not yield satisfactory results, as the LSTM cannot easily identify salient features in the image. Instead we need a feature extractor that embeds salient image features into some lower-dimensional, rich embedding that the LSTM can then ingest - a feat that CNNs readily accomplish. The original SAT implementation uses VGG net [16] for this task. To use this classification model as a feature extractor, we "cut off" the model before the fully connected layers. The resulting feature map has size 14x14x512.

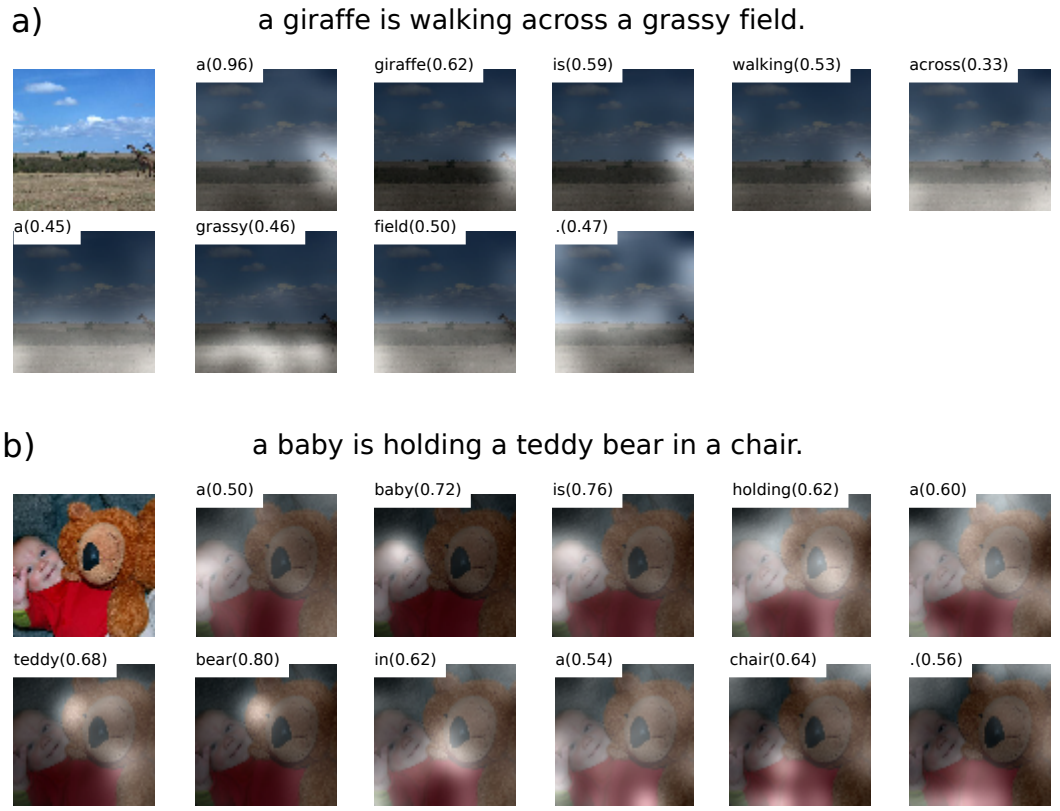


Figure 2: Visualization of soft-attention in images.

4.2 Soft attention model

Even though rich in information, the 14x14x512 feature embedding is still very high-dimensional. Such high-dimensional feature maps are suboptimal, because they can contain superfluous information that allows an upstream model to overfit through the unnecessary degrees of freedom. We thus seek a way to force the LSTM to ingest only a part of the whole feature map. This is achieved by soft attention. Soft attention collapses the 14x14x512 feature map to a single 1x1x512 feature vector, which is a weighted sum of the 14x14 feature map.

We thus have to predict 14x14 weights α_i that have to depend on their respective feature vectors in the CNN feature map and the current hidden state h of the LSTM. First, a function f_{att} maps the flattened hidden vector h as well as a single feature vector m_i to a score e_i:

$$e_i = f_{att}(m_i, h) \tag{1}$$

We parametrized f_{att} as a two-layer perceptron with "exponential linear unit" [4] nonlinearities, which have been recently proposed as an alternative to the popular ReLu units that address the vanishing gradients issue of very deep neural networks. These scores are then softmaxed to ensure they are summing up to one. This yields the weights α_i:

$$\alpha_i = \frac{\exp(e_i)}{\sum \exp e_i} \tag{2}$$

We can now compute the context vector c:

$$c = \sum_i \alpha_i * m_i \tag{3}$$

This special interface layer between the LSTM and the CNN forces the LSTM to attend to salient parts of the images. Another added benefit is interpretability, as the weights α_i can easily be projected back into the original image space, where they can provide insight as to what part of the image the LSTM deemed relevant for the sampling of the next word in the caption.

4.3 Language-Modeling Long-Short Term Recurrent Network

Finally, the caption-generating part of the architecture is parametrized as a LSTM, as discussed in class. As inputs to the input, forget and output gates as well as the new memory cell we use a learned affine transformation of the hidden state, context vector (from the soft attention mechanism) and previously predicted word. The initial hidden state of the LSTM, h_0, is generated by a multi-layer perceptron that takes as input the averaged 14x14 CNN feature map. The initial "previously" predicted word is always a special <START>" token. The output layer again uses the context vector, previously predicted word and current hidden state to predict the word probabilities of the next word. At training time, the LSTM is fed a *ground-truth* caption and the cross-entropy loss between the prediction and ground-truth is computed. At test time, the highest-scoring word is sampled and fed as input to the next time step.

4.4 Potential improvements

In this section, we identify a number of potential amendments to the original image captioning model:

Swap VVG-net with inception-resnet-v2 The original VGG net has long been overtaken by other architectures in terms of performance on the imagenet classification task and as a feature extractor. We will thus swap the VGG net architecture with the more potent inception-resnet-v2.

Shared feature space for image features The LSTM is directly trained on some feature map output by the CNN and is expected to adapt its weights to figure out a good embedding of these high-level image features. We propose to pre-train the CNN to embed the image into a feature space that is more accessible to the LSTM. We describe the exact architecture, which we dubbed "word CNN" in the next section 4.5.

End-to-end training In the original paper, the LSTM is trained on the vanilla CNN output - i.e., in a preprocessing step, the CNN is run on all images, and the extracted features are paired with their lists of captions. The CNN and the LSTM are never trained jointly, nor are the weights in the CNN fine-tuned for this specific usecase. We thus propose to instead train the whole architecture end-to-end, such that the LSTM can directly backpropagate error gradients into the CNN.

4.5 The Word CNN

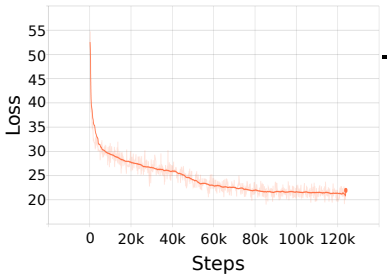
We hypothesize that the interface between the LSTM and the CNN is not ideal. We note that at some layer in this complex model, the image features have to be mapped into the same semantic space as the word vectors. In the original model, it is assumed that the LSTM will learn this mapping itself - yet, we note that this mapping has to take place in the input layer of the LSTM, which is a single fully connected layer.

We thus propose to extend the architecture of the CNN feature extractor such that it can directly map into the space of 300-dimensional word vectors. Specifically, we extend the CNN such that the last layer is a 8x8x300 feature map, where each of the 8x8 features is interpreted as a word vector. We pre-train this FCNN by feeding it images and directly supervising the last layer with the glove-embeddings of single words in the caption.

To achieve this feat, however, we have to solve a difficult problem: How do we map the words in the captions to word vectors in the 8x8 feature map? Importantly, the mapping must map the same word or words that are similar in meaning to the same word vector, so that single word vectors in the feature map can specialize to single semantic meanings. This rules out naive mappings, such as mapping the first word to the word vector at position (1,1), the second word vector to position (1,2) etc.

We solve this problem by leveraging a soft attention mechanism similar to the soft attention mechanism discussed above.

For each image, the CNN produces a map m of 8x8x300 features, where each of the 8x8 features models a word vector. Our model takes in a single word vector embedding w of a word in the caption,



	TF SAT with VGG		TF SAT with Inception		Original(Beam Search)	
	Val	Test	Val	Test	Val	Test
Bleu-1	66.1	65.6	66.4	68.5	-------	70.7
Bleu-2	45.3	44.6	48.4	47.8	-------	49.2
Bleu-3	31	30.4	34.2	33.3	-------	34.4
Bleu-4	21.8	21.1	24.3	23.5	-------	24.3
METEOR	21.7	21.6	22.3	22.9	-------	23.9
CIDER	68.3	65.7	77.6	74.8	-------	-------

Figure 3: Training loss and experiment results. The left panel shows the converging training loss of our final model. The right panel shows a summary of our experiments: TensorFlow implementation of SAT available online with VGG feature extractor, TensorFlow SAT with Inception feature extractor, original SAT paper results.

and uses this embedding to compute 8x8 weights α_i for each of the 8x8 features. As previously, this is achieved by the classic soft attention architecture (see Equation 2). For each word in the caption, we can now compute an estimated word vector $\hat{w}$, which is a weighted sum of the 8x8 features:

$$\hat{w} = \sum_i \alpha_i * m_i$$

We can now formulate a loss over all estimated word vectors $\hat{w}_j$ and all true word vectors w_j in the caption:

$$\mathbb{L} = \sum_j \|\hat{w}_j - w_j\|_2^2$$

This mechanism allows the CNN to specialize single features in its 8x8 feature maps to represent a specific semantic point in the space of word vectors.

Having thus pre-trained the CNN, we can then train the LSTM on this specialized 8x8 feature map.

5 Experiments and Results

We implement all models with TensorFlow [1]. To this end, we build on an open-source implementation of a CNN+LSTM image captioning model[1], which we extend as described in sections 3 and 4.

We parametrize the LSTM with 1800 hidden units. The feature-extracting CNN is the inception-resnet-v2. All experiments are conducted with RMSprop, a batch-size of 128, a learning rate of 0.001 and exponential learning rate decay by 0.1 every 50 epochs. All dropout layers have a keep probability of $p_{\text{keep}} = 0.5$.

5.1 Metrics

We measure and compare model performances using the Bleu-4 score.

5.2 Changing the CNN architecture

Without any changes to the open-source baseline implementation (using the original VGG net as a feature extractor) we get a Bleu-4 score of 21.8. We swap out the CNN with the inception-resnet-v4 architecture and retrain with the hyperparameters as detailed above. We train for 26h on a Maxwell Titan X GPU. This gives us a boost in performance and leaves us with a Bleu-4 score of 24.3 on the validation set. A critical regularization step towards achieving a good performance on the validation set and preventing overfitting on the training set was to use early-stopping (after about 60,000 steps)

[1] https://github.com/yunjey/show-attend-and-tell/tree/master/core

Ground Truth:
a child is playing with a teddy bear
a baby is laying under a stuffed bear
a baby smiling and lying next to a teddy bear

a baby is holding a teddy bear in a chair

Ground Truth:
two sinks are shown on a bathroom counter
his and her bathroom sinks underneath a large mirror
a bathroom with a sink and a mirror

a bathroom with a sink and a mirror

Ground Truth:
there are many people outside having some fun
two people on skis pose for a picture on the slope
two people are seen on skis on a snow covered path

a man is skiing down a snowy hill

Ground Truth:
a group of young men playing on a soccer team
a boy tries to make a goal in a game of soccer
a childrens soccer game being played in a park

a group of young men playing a game of soccer

Ground Truth:
a tray holds an arrangement of various foods
a plate that has food on a table
a plate holds french fries a steak cole slaw and sauces

a plate of food with french fries and a sandwich

Ground Truth:
a couple of men sitting next to each other
two men sit together and pose for a picture
there are two men sitting at a table eating food

a man and a woman are posing for a picture

Ground Truth:
a child stands with a bat at a base
a young baseball player about to take a swing
a boy holding a bat while standing next to a baseball base

a little boy holding a baseball bat on a field

Ground Truth:
a person hitting a ball with a bat on a field
a black and white photo of a batter swing the bat
a person playing baseball in front of a small crowd

a man riding a snowboard down a snow covered slope

Figure 4: Model caption predictions. Ground truth annotations (italicized) and our model's predictions (bold) are shown above for various images.

before the training loss converged to its final value. A comparison of the performances on the test sets can be found in Figure 3 In a further experiment, we test with inserting / leaving out a batch-norm-layer between the CNN features and the attention layer. We find that performance without the batch-norm-layer is significantly worse.

5.3 Learning a word-vector embedding of images

We train and evaluate the word CNN architecture proposed in section 4.5. We use the cut-off inception-resnet-v2 architecture for this experiment. We parametrize all fully connected layers in the attention mechanism as two-layer perceptrons with ReLu nonlinearities. We find that our hypothesis regarding the specializiation of features in the 8x8 feature map specializing to single semantic meanings is true - for instance, the word vector at position (1,1) learns embeddings of articles such as "the" or "a". We observe other features in the map specializing to 'surfboard' and 'ski' or 'person'. We train this architecture for $\sim$30 epochs. We then use this pre-trained network as a feature extractor and extract the features of all 80.000 images in the training set. We subsequently train the LSTM with these features. To our surprise, the model could not outperform the baseline architecture - it reached a Bleu-4 score of a mere 14.7, staying far behind the baseline architecture. This demonstrates that though well-motivated, the space of word vectors is not a feasible embedding for the image features. This is interesting, as it seems to suggest that the original CNN features capture much more information than just the semantic concepts appearing in the scene as they can be capture in word vectors. This also seems to partially invalidate our hypothesis that the input layer of the LSTM (effectively a single fully connected layer) is insufficient to map the image features into a joint image feature space.

5.4 End-To-End training

We directly join the inception-resnet-v2 architecture with the LSTM. The inception-resnet-v2 was initialized with the weights pre-trained on Imagenet, while we initiliazed the LSTM with the best-performing weights on a previous run. During training, we observed that any significant learning rate led only to oscillations of the training error - no convergence to a lower training loss than before could be observed. This, together with mild overfitting, led to no performance boost from the end-to-end trained model.

5.5 Qualitative Results

In Figure 4 we present image captions generated by our model. We can see that for most images in this subset the model captions the images correctly, however there are some failure cases. For example, in the images of the two men, our model incorrectly captions the image stating that a man and woman pose for the picture, but it still concludes that there were two people posing for a picture. The last two images, of the baseball players exposes an interesting error. Our model accurately captions the image of the boy swinging the baseball bat, however it (very) incorrectly captions the black and white image of the woman swinging the bat as "a man riding a snowboard down a snow covered slope". We believe this error to stem from the image being black and white. Having not seen many black and white images, the model incorrectly assumed that an image with so much white must be in a snowy setting. Figure 2 illustrates the model correctly attending to words in the input image.

6 Conclusion

We successfully reproduce results from the original Show, Attend, and Tell paper in TensorFlow by modifying an available implementation. We propose certain improvements on the model by replacing the VGG feature extractor with the Inception feature extractor. In doing so we were able to basically reproduce original paper results without having to resort to beam-search to boost performance. We also investigate a fundamental variation to the Show, Attend, and Tell model by constraining the output feature space of Inception to be the word vector space, and therefore more compatible with the image captioning LSTM. Interestingly, this modification did not result in performance improvements, which we conclude to be the result of the LSTM being sufficiently robust to generate a mapping from the image feature space to captions (in the original paper).

7 Contributions

All teammates contributed equally to the project.

References

[1] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., ET AL. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] AKER, A., AND GAIZAUSKAS, R. Generating image descriptions using dependency relational patterns. In *ACL* (2010), Association for Computational Linguistics, pp. 1250–1258.

[3] CHEN, L.-C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* (2016).

[4] CLEVERT, D.-A., UNTERTHINER, T., AND HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

[5] ET AL., J. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* (2016).

[6] ET AL., M. Midge: Generating image descriptions from computer vision detections. In *EACL* (2012), Association for Computational Linguistics, pp. 747–756.

[7] GARG, R., CARNEIRO, G., AND REID, I. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV* (2016), Springer, pp. 740–756.

[8] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[9] JOHNSON, J., KARPATHY, A., AND FEI-FEI, L. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR* (2016), pp. 4565–4574.

[10] KARPATHY, A., AND LI, F. Deep visual-semantic alignments for generating image descriptions. *CoRR abs/1412.2306* (2014).

[11] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[12] KUZNETSOVA, P., ORDONEZ, V., BERG, A. C., BERG, T. L., AND CHOI, Y. Collective generation of natural image descriptions. In *ACL: Long Papers-Volume 1* (2012), Association for Computational Linguistics, pp. 359–368.

[13] LIN, T., MAIRE, M., BELONGIE, S. J., BOURDEV, L. D., GIRSHICK, R. B., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. Microsoft COCO: common objects in context. *CoRR abs/1405.0312* (2014).

[14] LONG, J., SHELHAMER, E., AND DARRELL, T. Fully convolutional networks for semantic segmentation. In *CVPR* (2015), pp. 3431–3440.

[15] RENNIE, S. J., MARCHERET, E., MROUEH, Y., ROSS, J., AND GOEL, V. Self-critical sequence training for image captioning. *CoRR abs/1612.00563* (2016).

[16] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[17] SOCHER, R., PENNINGTON, J., HUANG, E. H., NG, A. Y., AND MANNING, C. D. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP* (2011), Association for Computational Linguistics, pp. 151–161.

[18] VINYALS, O., TOSHEV, A., BENGIO, S., AND ERHAN, D. Show and tell: A neural image caption generator. In *CVPR* (2015), pp. 3156–3164.

[19] WANG, S., AND JIANG, J. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849* (2015).

[20] XIONG, C., MERITY, S., AND SOCHER, R. Dynamic memory networks for visual and textual question answering. *arXiv 1603* (2016).

[21] XU, H., AND SAENKO, K. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV* (2016), Springer, pp. 451–466.

[22] XU, K., BA, J., KIROS, R., CHO, K., COURVILLE, A. C., SALAKHUTDINOV, R., ZEMEL, R. S., AND BENGIO, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML* (2015), vol. 14, pp. 77–81.