An Image Caption Generation Model Using Encoder-Decoder Architecture

Zhouyao Xie
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
zhouyaox@andrew.cmu.edu

Qingrui Cao
Department of Physics
Carnegie Mellon University
Pittsburgh, PA 15213
qingruic@andrew.cmu.edu

Abstract

As an intersection of computer vision (CV) and natural language progressing (NLP), image caption generation has sparked the interest of the deep learning research community for its potential real-life applications. In this work, we implement a generative CNN-LSTM model with two major modifications: (i) scheduled sampling, and (ii) beam search. Experiments on the Microsoft Common Objects in Context (MSCOCO) dataset shows that scheduled sampling provides significant improvements in performance scores and generate sensible and accurate captions in a majority of cases, with our best-performing model improving BLEU-4 and GLEU by 9.7 and 6.7 points relative to the baseline model, respectively. Result analysis also demonstrates that beam search improves the quality and diversity of generated sequence.

1 Introduction

The task of image caption generation, whose goal is to automatically generate textual description of an image, has attracted increasing interest from the deep learning research community. Its many applications include aid to blind, self-driving cars, object recognition in computer vision, and so on. While it seems to be an easy task for humans beings, generating accurate and complete captions for images is not a trivial task for machines, as it requires a model that can both determine which objects are in an image and capture and express their relationships in a natural language. Such combination of computer vision and natural language progressing has long been viewed as a challenging problem.

In this work, we adopt the CNN-RNN architecture that was first proposed by [1] to develop a image captioning model. We use a pretrained ResNet50 model as the encoder and a one-layer LSTM as the decoder. Furthermore, we implement two improvements based on this baseline model. We apply scheduled sampling [2] with a teacher enforcing rate of 0.8 to shorten training convergence time while maintaining model's generalization ability. Secondly, when generating model outputs, we apply beam search strategies to enhance the quality and diverstiy of the generated sequences.

2 Data Preprocessing

We use the MSCOCO dataset [3] to train and test our captioning model. The dataset, first published in 2014, is a large-scale object detection, segmentation, and captioning dataset. In this study, we use its 2017 version, which contains 118,287 training images and 5,000 validation images. Most images come with 5 captions and a small number of images have 6 captions, resulting in a total of 591,753 captions in the train set and 25,014 captions in the validation set.

10-701 Introduction to Machine Learning (PhD). Carnegie Mellon University, Pittsburgh, PA, USA

2.1 Image

The MSCOCO dataset contains images of various sizes and black-and-white images. First, we standardize the input shape by resizing each image to $3 \times 256 \times 256$, where 3 is the RGB channels and 256 is the width and height. The pixel values are normalized to a range of $[0, 1]$. Next, to prepare these images as inputs to ResNet50, we further standardize them to having a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225] across each channel [4].

2.2 Caption

We apply a series of text preprocessing steps to transform captions into model inputs. First, we discard all non-alphanumeric characters, lowercase all letters, and tokenize each sentence. Then, after inspecting the distribution of caption lengths over the entire training set, we choose 18 as the maximum sequence length, as most captions are relatively short sequences that contain fewer than 18 tokens. We perform padding on captions with length shorter than 18 by appending a special '<pad>' token and cropping on sequences that are longer.

Words that appear in the training corpus few than or equal to 10 times are classified as rare words, and are replaced by a special '<UNK>' token. We append '<start>' and '<end>' tokens at the beginning and end of each captions, respectively. As a result, each training caption is 20 in length, and the total corpus contains 7103 unique words (including the four special tokens, '<start>', '<end>', '<UNK>', '<pad>'). Finally, each caption is converted to a 20×7103 vector using one-hot encoding.

3 Related Work

Despite the difficult nature of this task, there has been tons of research attention in attacking the image caption generation problem. Early works [5] mainly use template-based methods, which involve a two-step approach. In the first step, the model extracts a series of key features using classifiers such as SVM; then, it uses lexical models or other template-based models to convert the extracted feature data into captions. In recent years, with the rising popularity of of deep learning techniques, neural networks have been used extensively in image caption generation. In this section, we review two important lines of methods which have been implemented for automatic image caption generation.

3.1 Top-down approaches

Inspired by the encoder-decoder architecture in neural translation, [1] first proposed using a similar architecture for image captioning generation, replacing the encoder RNN used in translation with pre-trained CNN. An encoder CNN reads the input image and transforms it into a rich fixed-length matrix representation, which in turn is used as the initial hidden state of a decoder RNN that generates the target sentence. More modern CNNs were used in [1] and [6] for encoding and replaced with recurrent neutral networks (RNN), in particular LSTM in [1]. The common theme of these works is that images are represented as high-dimensional embeddings through the encoder network and passed into the top layer of the decoder network, thereby producing models that were end-to-end trainable.

3.2 Bottom-up approaches

Another line of works employs a bottom-up approach to leverage the inter-modal correspondences between textual and imagery data. In this approach, the problem is divided into two simpler tasks that are resolved using an alignment model and a generative model. Firstly, a CNN and a bi-directional RNN are trained to map images and fragments of captions to the same multimodal embedding. Next, an RNN is trained to combine the inputs from a series of object fragments detected in the original image to form a caption. For example, [7] trains a CNN over image regions and a bi-RNN over sentences and utilizes a structured objective to align the two modalities for the alignment model. Then, it uses a multimodal RNN to use the learned alignments to generate image captions. Although this allowed the model to aggregate information on specific objects in the images and thus improved the model performance, one disadvantage is that these models were not end-to-end trainable.

One way to integrate the two approaches together is through the use of attention mechanism. Attention mechanism has been used extensively in deep learning tasks such as in machine translation [14],

handwriting generation [9], and image generation [10]. For instance, the Deep Recurrent Attentive Writer (DRAW) neural network architecture was introduced in [10] and demonstrated the ability to generate highly realistic natural images. [15] is the first work that applies attention to image captioning and achieves state-of-the-art performance. Following its success, numerous modifications and improvements to the attention mechanism has been proposed, such as [16, 17, 18].

4 Methods

Our model follows the encoder-decoder architecture introduced in [1] and uses a pre-trained CNN as encoder and an LSTM as decoder, as shown in figure 1.

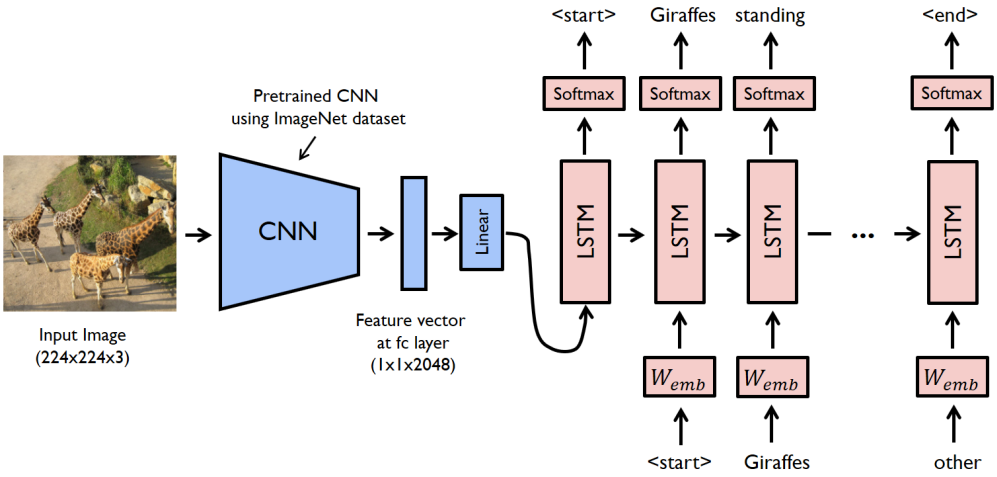

Figure 1: Schematic of CNN-LSTM architecture

4.1 Encoder

The encoder uses ResNet50, a CNN model trained on the ImageNet dataset that is 50 layers deep. The model has been shown to perform well in a series of computer vision tasks, including image classification and object detection.

In our implementation, we replace the final pooling layer of ResNet50 with a linear layer and a 1-d batch normalization layer. During training, We freeze gradient updates for all previous layers while only updating the gradients for these last two layers. We use an embedding shape of 512, so the embedded images have shape *batch size* $\times$ 1 $\times$ 512.

4.2 Decoder

We use a one-layer LSTM for the decoder, with a hidden size of 512 and a dropout ratio of 0.3. At each step, the decoder takes in an input token and a hidden state generated from the previous step, and outputs an output token and a hidden state. The input token is initialized as the '<start>' token, while the hidden state is initialized as the embedded image, i.e. encoder output. Output from the LSTM is passed into a linear output layer to convert its shape to *batch size* $\times$ *vocab size*.

4.3 Scheduled Sampling

While during training, a RNN model maximizes the likelihood of each token in the sequence given the current state and the previous token, at inference, the previous token is replaced by a token generated by the model itself. As [2] notes, this discrepancy between training and inference leads to

the exposure bias problem. We adopt the scheduled sampling technique proposed by [2] to simulate the inference process during training, thereby bridging the gap between training and inference.

While more complicated scheduling techniques have been proposed, such as using confidence-aware schedules [11] and step-dependent schedules [12], we use the vanilla version scheduled sampling in our model. At each training step, the model randomly decides whether to use the previous output or the true label as the input to the decoder based on a fixed probability ratio. In our experiments, we use a teacher forcing ratio of 0.8.

4.4 Greedy search vs. Beam search

During inference, the fundamental question that our model tries to answer is "what is the most likely sequence of words given some input image?". Denote a sequence of words with length l as $\mathbf{w}$, the probability of $\mathbf{w}$ is given can be decomposed into a series of conditionals using the chain rule such that the probability of the next word depends on the words preceding it:

$$\begin{aligned} p(\mathbf{w}) &= p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2) \times \cdots \times p(w_l|w_1, w_2, \ldots, w_{l-1}) \\ &= \prod_{n=1}^{l} p(w_n|w_1, w_2, \ldots, w_{n-1}) \end{aligned}$$

Given a conditioning context $\mathbf{x}$, the goal of image captioning is to find the sequence of words with the greatest joint likelihood:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmax}}\, p(\mathbf{w}|\mathbf{x}) = \underset{\mathbf{w}}{\text{argmax}} \prod_{n=1}^{l} p(w_n|\mathbf{x}, w_1, w_2, \ldots, w_{n-1})$$

However, such exhaustive search is exponential in sequence length, leading to an intractable problem. In our baseline model, we uses the simplest solution, greedy search, to generate output sequences. Greedy search approximates finding $\mathbf{w}^*$ by iteratively finding the most likely word $\tilde{w}_i$ given the previous words such that:

$$\tilde{\mathbf{w}} = \tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_n$$

where

$$\tilde{w}_i = \underset{w_i}{\text{argmax}}\, p(w_i|\mathbf{x}, \tilde{w}_1, \ldots, \tilde{w}_{i-1})$$

While very straightforward to implement and intuitive to understand, this is obviously non-optimal due to its greedy nature. Therefore, we implement beam search algorithm [13], where instead of taking only the most likely word at each decoding iteration, we take the k most likely, where k is the beam width. The generation algorithm is as follows:

1. For every beam, predict a probability distribution over all words in the vocabulary of length v given the preceding word.
2. Find the k most likely sequences from all $k \times v$ potential candidates.
3. Update the beam sequences with the b most likely sequences and repeat until the stopping condition is satisfied (either an "end" token is predicted or the maximum length is reached).

The key difference compared to the greedy search is that in step 2, we are instead finding:

$$\underset{w_i,\beta}{\text{kargmax}}\, p(w_i|\mathbf{x}, w_1^\beta, \ldots, , w_{i-1}^\beta) \times p(beam_\beta)$$

where kargmax is the argmax function exteneded to the top k arguments and $p(beam_\beta)$ is the probability of the sequences of words in beam β given by:

$$\begin{aligned} p(beam_\beta) &= p(w_1^\beta, w_2^\beta, \ldots, w_{i-1}^\beta|\mathbf{x}) \\ &= \prod_{n=1}^{i-1} p(w_n^\beta|\mathbf{x}, w_1^\beta, w_2^\beta, \ldots, w_{n-1}^\beta) \end{aligned}$$

where w_n^β is the n'th word in beam β.

5 Experiments and Results

When performing model training, we compute the cross entropy loss between decoder outputs and reference captions at each iteration, and use it as the objective function and minimize its value. The cross entropy function is given by:

$$\ell(x, y) = \sum_{n=1}^{N} l_n = -\sum_{n=1}^{N} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})}$$

where l_n is the loss at step n, C is *vocab size*, N is the maximum sequence length, $x_{n,c}$ is the generated probability of the c-th token at step n, and y_n is the true token at step n.

We optimize cross entropy loss using SGD optimizers with learning rate set to 0.05 and momentum set to 0.9. We use step schedulers with a gamma of 0.9 and a step size of 1 (i.e. reduce learning rate by 0.9 each epoch). The optimizers and schedulers have the same setup parameters for the encoder and the decoder. For the encoder, we use an encoding size of 512; for the LSTM decoder, the hidden size is also 512. We choose a *batch size* of 8.

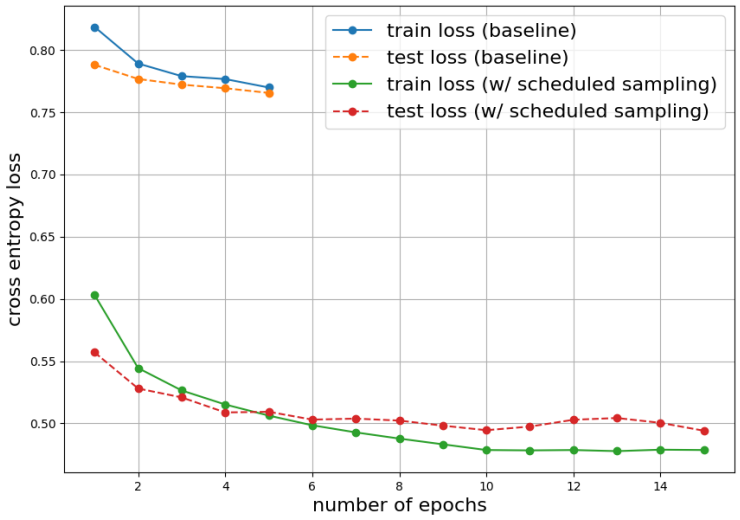

Figure 2: Train and test loss progression for baseline model and model with scheduled sampling for a teacher forcing ratio of 0.8.

We train the baseline model up to 5 epochs, and the improved model with scheduled sampling for more than 15 epochs. Figure 2 shows the progression of their train and test loss with the number of epochs. As expected, the cross entropy loss for our improved model is lower than the baseline model, decreasing by more than 50% at the fifth epoch. Also, The loss for improved model gradually reaches a plateau after the tenth epoch and thus converges.

We also evaluate these models using several commonly used metrics, including BLEU-1, BLEU-4, GLEU, ROUGE-L, and NIST. As shown in table 1, scheduled sampling leads to significant improvements in all metrics comparing with the baseline model. Specifically, it leads to a 4.0 increase in BLEU-1, 7.7 increase in BLEU-4, 5.1 improvement in GLEU, 0.5 increase in ROUGE-L, and a 0.454 improvement in 5-gram NIST. This performance boost is more apparent in the converged scheduled sampling model, which trained for three extra epochs. We observe a 9.7 increase in BLEU-1, 9.7 increase in BLEU-4, 6.7 improvement in GLEU, 1.6 increase in ROUGE-L, and a 0.508 improvement in 5-gram NIST.

models	epochs	BLEU-1	BLEU-4	GLEU	ROUGE-L	NIST (n=5)
baseline	5	57.6	11.0	17.6	32.8	1.096
w/ scheduled sampling	5	61.6	18.7	22.7	33.3	1.550
	8	67.3	20.7	24.3	34.4	1.604

Table 1: Performance of baseline model and model with scheduled sampling. Models with scheduled sampling perform better than baseline model in all metrics.

6 Discussion and Analysis

6.1 Human Evaluation

Figure 3 displays a few example outputs provided by our scheduled sampling model. In both figures, the model is able to correctly recognize and classify objects contained in those images.

One interesting observation is that the model sometimes generate low-level descriptions of images that enumerate all the objects detected and express their relationships, as in figure 3**(a)**-**(b)**, while other times it is capable of generating more "abstract" descriptions. Instead of outputting the name of each individual object, the model summarizes an image using high-level descriptions. For instance, in figure 3**(c)**, the church, buildings, and the cars are encapsulated in the word "city". In figure 3**(d)**, the model accurately describes the image as "A brown box filled with lots of different foods", instead of trying to enumerate the bread, carrots, muffin, etc. This shows that with the pre-trained CNN encoder fine tuned on the training set, the model is capable of performing different levels of image understandings.

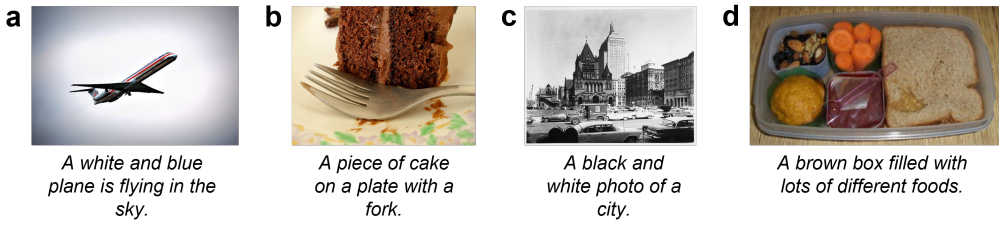

Figure 3: Examples **(a)**-**(d)** with correct descriptions

6.2 Error analysis

An in-depth error analysis of our model outputs reveal several weaknesses of the model that require further attention, including wrong object detection, hallucination, and poor robustness for low-light images. Figure 4 displays several example outputs that showcase each type of error respectively. In figure 4**(a)**-**(b)**, the model mistakenly recognizes an object as another closely related object (Lego man -> man in 4**(a)** and zebra -> giraffe in 4**(b)**). In figure 4**(c)**, the model describes the bedroom as having a "a bed and a window", while a window is clearly not present in the image, indicating the issue of hallucination, which is quite common among sophisticated NLP models. Lastly, our model struggles with generating complete captions for images that are dark and blurry. As shown in figure 4**(d)**, the caption generated by the model has a high precision but a low recall, as it completely ignores the cat in the front.

6.3 Metrics vs. Human Evaluation

Note that we select all the instances above based on human evaluations. During human evaluations, we notice that sometimes there is a discrepancy between the metrics scores and the actual quality of the outputs. For some instances, it is clear that our improved model performs better than the baseline model both in terms of grammatical correctness and logical soundness, but their metric scores are extremely close.

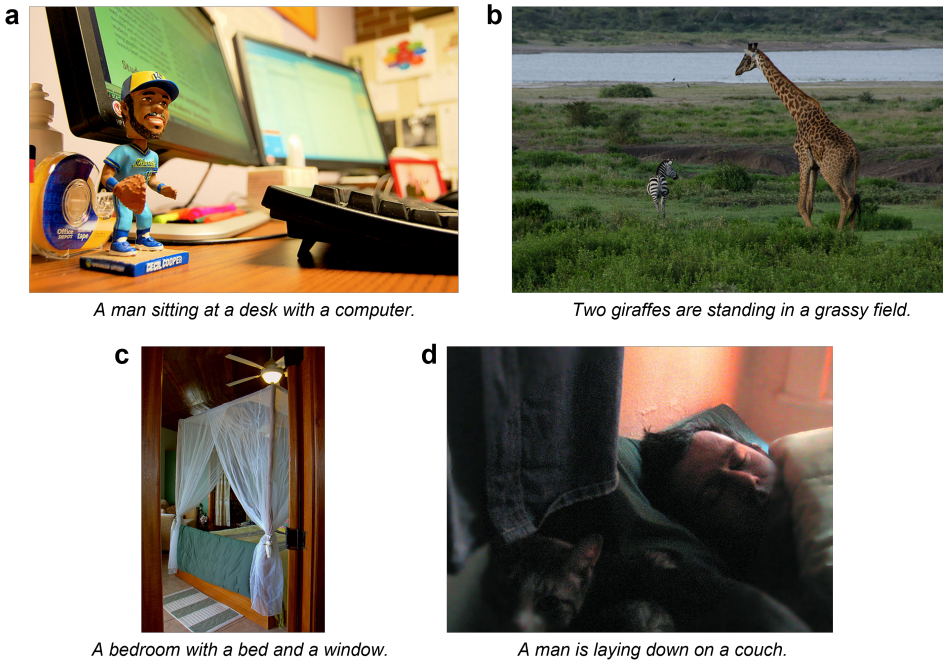

Figure 4: Examples **(a)**-**(d)** with minor mistakes

Closer inspection of these discrepancies reveals that existing evaluation metrics are far from perfect and all have various limitations. For one thing, most of these metrics are good indicators of model performance when applied on large-scale datasets, but their values are often inaccurate and biased on individual sentences. Moreover, metrics like BLEU and GLEU do not evaluate semantic correctness and only measure the number of exact n-grams matches of the generated output sequences. This means that an output sentence that correctly captures the meaning of an image can still receive a low score. However, despite these limitations, these metrics are still helpful tools for model selection and testing purposes.

6.4 Beam search

In figure 5, we show the results of beam search implementation compared with greedy search (i.e. beam width equals 1). The major improvements are twofold: firstly, beam search enhances the overall quality of the generated outputs by improving them coherence and grammatical correctness; at the same time, it enriches model outputs by introducing a higher level of diversity.

For example, in figure 5**(a)**, the output generated using greedy search only produces the phrase "a group of people" and fails to express other information in the image. The output is not only meaningless but also grammatically incorrect. This is because the model is prone to assigning larger output values to high-frequency words ("a" in this case), whose effect might overshadow the effect of the features obtained from the input image, leading to broken output sentences. This problem is particularly apparent when a greedy approach is used. Fortunately, we find that such results can be significantly improved by implementing beam search strategies. With a beam width of 3, the model is able to correctly identify other objects ("boat", "water" or "oars") in this image.

Moreover, we also notice that beam search can generate captions of different styles and various levels of specificity, thereby increasing the diversity of model outputs. In figure 5**(b)**, both models correctly identify the main object in the image as a parked motorcycle. However, the greedy search approach only mentions "A motorcycle parked on the side of a road", whereas the three outputs from beam search give other information like "grass" and "other vehicles".

a

Beam width	Predicted caption(s)
1	**A group of people in a in a a a.**
3	**A group of people in the boat.**
	A group of people in the water.
	A group of people are on the boat on their body with the oars in them on their backs as.

b

Beam width	Predicted caption(s)
1	**A motorcycle parked on the side of a road.**
3	**A motorcycle parked on the grass.**
	A motorcycle parked on the side of the street.
	A motorcycle is parked on the road side of a bike is in a grassy spot with other vehicles behind.

Figure 5: Examples **(a)** and **(b)** using beam search algorithm, with bold text in tables indicating the model output given a certain beam width (the most likely result among top $k = 3$ choices).

7 Conclusions and Future Work

In this work, we implement an image caption generation model with encoder-decoder architecture using pre-trained CNN as encoder and one-layer LSTM as decoder. We show that scheduled sampling could significantly improve model performance both measured by various popular metrics and evaluated by humans. Despite a few limitations of the model which can lead to problems like misclassification of entities and hallucination, our model is capable of producing accurate and complete descriptions of images in most cases. Further more, we improves our output generation method using the beam search strategy. We show that beam search technique not only improves the overall quality of the generated captions, in particular in terms of grammatical correctness and coherence, but also leads to more diversified outputs.

There are various directions for future work. Firstly, the current caption generation system has a limit on the maximum length of input. Future works could be done to alleviate this constraint to allow for inputs of variable lengths. A second direction for further improvements is to experiment with different scheduled sampling strategies, such as the ones proposed in [11] and [12]. Using more fine-grained scheduled sampling techniques could ideally further decrease training speed while producing models with higher robustness. Lastly, while we experiment with using beam search to generate output sequences during inference, we have not incorporate it into training due to the increase in training time this would bring. Thus, further studies could be done to incorporate beam search into training while reducing the time complexity of such training procedures.

References

[1] Vinyals, O., Toshev, A., Bengio, S. & Erhan, D. (2014) Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

[2] Bengio, S., Vinyals, O., Jaitly, N. & Shazeer, N. (2015) Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099.

[3] Lin, T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L. & Dollár, P. (2014) Microsoft coco: Common objects in context. *CoRR*, abs/1405.0312.

[4] He, K., Zhang, X., Ren, S. & Sun, J. (2015) Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[5] Farhadi, A., Hejrati, M., Sadeghi, M.A., Young, P., Rashtchian, C., Hockenmaier, J. & Forsyth, D. (2010) Every picture tells a story: Generating sentences from images. *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 15–29, Berlin, Heidelberg: Springer-Verlag.

[6] Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. & Darrell, T. (2014) Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389.

[7] Karpathy, A. & Li, F. (2014) Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306.

[8] Li, S., Kulkarni, G., Berg, T., Berg, A. & Choi, Y. (2011) Composing simple image descriptions using web-scale n-grams. *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 220-228, Portland, Oregon, USA: Association for Computational Linguistics.

[9] Graves, A. (2013) Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

[10] Gregor, K., Danihelka, I., Graves, A. & Wierstra, D. (2015) DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623.

[11] Liu, Y., Meng, F., Chen, Y., Xu, J. & Zhou, J. (2021) Confidence-aware scheduled sampling for neural machine translation. *CoRR*, abs/2107.10427.

[12] Liu, Y., Meng, F., Chen, Y., Xu, J. & Zhou, J. (2021) Scheduled sampling based on decoding steps for neural machine translation. *arXiv preprint*, arXiv:2108.12963.

[13] Young, T., Hazarika, D., Poria, S. & Cambria, E. (2017) Recent Trends in Deep Learning Based Natural Language Processing. *CoRR*, abs/1708.02709.

[14] Bahdanau, D., Cho, K., & Bengio, Y. (2014) Neural machine translation by jointly learning to align and translate. *arXiv preprint*, arXiv:1409.0473.

[15] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. & Bengio, Y. (2015) Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044.

[16] Huang, L., Wang, W., Chen, J. & Wei, X. (2019) Attention on attention for image captioning. *CoRR*, abs/1908.06954.

[17] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S. & Zhang, L. (2017) Bottom-up and top-down attention for image captioning and visual question answering. *CoRR*, abs/1707.07998.

[18] Lu, J., Xiong, C., Parikh, D. & Socher, R. (2016) Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. *CoRR*, abs/1612.01887.