Improved Image Captioning via Policy Gradient optimization of SPIDEr

Siqi Liu[*1], Zhenhai Zhu[2], Ning Ye[2], Sergio Guadarrama[2], and Kevin Murphy[2]

siqi.liu@cs.ox.ac.uk
{zhenhai,nye,sguada,kpmurphy}@google.com
[1]Department of Computer Science, University of Oxford
[2]Google

Abstract

Current image captioning methods are usually trained via (penalized) maximum likelihood estimation. However, the log-likelihood score of a caption does not correlate well with human assessments of quality. Standard syntactic evaluation metrics, such as BLEU, METEOR and ROUGE, are also not well correlated. The newer SPICE and CIDEr metrics are better correlated, but have traditionally been hard to optimize for. In this paper, we show how to use a policy gradient (PG) method to directly optimize a linear combination of SPICE and CIDEr (a combination we call SPIDEr): the SPICE score ensures our captions are semantically faithful to the image, while CIDEr score ensures our captions are syntactically fluent. The PG method we propose improves on the prior MIXER approach, by using Monte Carlo rollouts instead of mixing MLE training with PG. We show empirically that our algorithm leads to easier optimization and improved results compared to MIXER. Finally, we show that using our PG method we can optimize any of the metrics, including the proposed SPIDEr metric which results in image captions that are strongly preferred by human raters compared to captions generated by the same model but trained to optimize MLE or the COCO metrics.

1 Introduction

Image captioning is the task of describing the visual content of an image using one or more sentences. This has many applications, including text-based image retrieval, accessibility for blind users [25], and human-robot interaction [6].

Most methods for solving this task require training a statistical model on a dataset of (image, caption) pairs. The model is usually trained to maximize the log likelihood of the training set. After training, these models are usually

evaluated by computing a variety of different metrics on a test set, such as COCO [12]. Standard metrics from the machine translation community include BLEU [14], METEOR [3], and ROUGE [11]. More recently, the CIDEr metric [20] was proposed, specifically for the image captioning task. We shall call the combination of these four metrics "BCMR", for short. Unfortunately, none of these metrics correlate strongly with human measures of caption quality. In fact, humans score lower on these metrics than the methods that won the COCO 2015 challenge, despite the fact that humans are still much better at this task.

These results motivated Anderson et al. [1] to propose the SPICE metric. Rather than directly comparing a generated sentence to a set of reference sentences in terms of syntactic agreement, SPICE first parses each of the reference sentences, and then uses them to derive an abstract scene graph representation. The generated sentence is then also parsed, and compared to the graph; this allows for a comparison of the semantic similarity, without paying attention to syntactic factors (modulo the requirement that the generated sentence be parseable). [1] showed that SPICE is the only existing metric that has a strong correlation with human ratings, and ranks human captions above algorithms submitted to the COCO benchmark.

Given this result, it is natural to want to directly optimize SPICE. However, this is tricky, since it is not a differentiable objective. In this paper, we show that it is possible to use policy gradient (PG) methods [17] to optimize such objectives. The idea of using PG to optimize non differentiable objectives for image captioning was first proposed in the MIXER paper [15]. However, they only used it to optimize BLEU-4, which is not correlated with human quality. Furthermore, when we tried to use their method to optimize other metrics, such as CIDEr or SPICE, we got poor results, since their method (which involves an intricate incremental schedule which mixes from MLE training to full PG training) is not very robust and require careful tunning.

In this paper, we propose an improvement to MIXER, that uses Monte Carlo rollouts to get a better estimate of the

value function, and avoids the need to "mix in" the MLE objective. We show that this leads to faster convergence, and is significantly more robust to choice of learning rates and other hyper-parameters. We then show that we can use our new PG method to optimize the BCMR metrics, thus achieving state of the art results on the COCO leaderboard, despite using a very simple baseline model.

However, being on top of the COCO leaderboard is not our ultimate goal, since we know that the COCO metrics (based on BCMR) are only weakly correlated with human judgement [1]. So we decided to optimize SPICE with our algorithm. Unfortunately, optimizing SPICE produced long repetitive sentences which are not good results, based either on BMCR COCO metrics, or as judged by humans.

The reason optimizing SPICE gives poor captions is that SPICE ignores syntactic quality (see examples in Table 2) More generally, we argue that a good image captioning metric should satisfy two criteria: (1) captions that are considered good by humans should achieve high scores; and (2) captions that achieve high scores should be considered good by humans. SPICE satisfies criterion 1, but not criterion 2. We therefore propose a new metric, which is a linear combination of SPICE and CIDEr; we call this new metric SPIDEr. This metric automatically satisfies criterion 1, since both SPICE and CIDEr do. Also, when we optimize for SPIDEr, we show that the captions are judged by humans to be significantly better than the ones generated by training the same model using any of the other metrics. This shows that SPIDEr satisfies criterion 2 to a much greater degree than existing metrics.

In summary, we make the following contributions in this paper: (1) we identify criteria for a good image captioning metric, and proposed a new metric, SPIDEr, which meets both criteria; (2) we propose a new policy gradient method that can optimize arbitrary captioning metrics, and which is much faster and more stable than previous PG methods; (3) we show that using our new PG method to optimize existing BCMR metrics leads to state of the art results on COCO; (4) we show that using our new PG method to optimize our new SPIDEr metric results in much better human scores than optimizing for other metrics.

2 Related work

There is an extensive body of work on image captioning (see e.g., [5] for a recent review). Below we summarize some of the most relevant works.

2.1 Models

Most methods make use of an encoder-decoder style neural network, where the encoder is a convolutional neural network (CNN), and the decoder a recurrent neural network

(RNN). In this work, we use the encoder-decoder proposed in [22], known as "Show and Tell" (ST).

Numerous extensions to the basic encoder-decoder framework have been proposed. One line of work (e.g., the "Show, Attend and Tell" model of [26], and the "Review Network" model of [27]), leverages attention, which lets the decoder focus on specific parts of the input image when generating a word. Another line of work enriches the image encoding beyond just using a CNN that was trained for image classification. For example, [24, 28] use image taggers, [21] use object detectors, and [19] use face detection and landmark recognition. We stress that these extensions are orthogonal to the ideas in this paper.

2.2 Metrics and objective functions

Most prior work uses maximum likelihood estimation (MLE) for training. That is, the model parameters $\boldsymbol{\theta}$ are trained to maximize

$$
\begin{aligned}
L(\theta) &= \frac{1}{N} \sum_{n=1}^{N} \log p(\mathbf{y}^n | \mathbf{x}^n, \boldsymbol{\theta}) \\
&= \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(y_t^n | y_{1:t-1}^n, \mathbf{x}^n, \boldsymbol{\theta})
\end{aligned}
$$

where $\mathbf{x}^n$ is the n'th image, $\mathbf{y}^n = (y_1^n, \dots, y_{T_n}^n)$ is the ground truth caption of the n'th image and N is the total number of labelled examples.

One problem with the MLE objective is that at training time, each prediction is conditioned on the previously observed words from the ground truth. At test time, however, the model will be fed with its own predictions, leading to quickly accumulating errors during inference, so the model will likely diverge from desired trajectories. This discrepancy is known as "exposure bias" [4]. One solution to exposure bias is "scheduled sampling" [4], although this method has been shown to be statistically inconsistent [9].

An alternative to maximizing likelihood is to try to maximize some other objective that is more closely related to the true metric of interest. This can be done using a policy gradient (PG) method [17] such as REINFORCE [23], by treating the score of a candidate sentence as analogous to a reward signal in a reinforcement learning setting. In such a framework, the RNN decoder acts like a stochastic policy, where choosing an action corresponds to generating the next word.

[15] use a modified form of REINFORCE to optimize the BLEU score for image captioning. We explain this approach in more detail in Section 2.3, since it is closely related to our method. [8] used REINFORCE to optimize sequence level reward such that generated captions are class discriminative. However, this is a different task than the one we consider.

More recently, there have been a variety of other papers on optimizing sequence level objective functions (see e.g., [13, 30, 2, 16]), but mostly in the context of machine translation.

2.3 MIXER

The most closely related work is the "MIXER" paper [15]. They also use the REINFORCE method, combined with a baseline reward estimator. However, they implicitly assume each intermediate action (word) in a partial sequence has the same reward as the sequence-level reward, which is not true in general. To compensate for this, they introduce a form of training that mixes together the MLE objective and the REINFORCE objective. Specifically, they evaluate the first M words using log-likelihood, and the remaining words using REINFORCE; they gradually decrease M from the maximum sentence length down to 0.

We have found that MIXER is very sensitive to the form of the annealing schedule, as well as other hyper-parameters, such as the learning rate or gradient scaling of the baseline estimator. We therefore propose a more robust alternative, in which we replace the constant reward assumption with an estimate of future rewards based on Monte Carlo rollouts (a similar idea is used in [30] for GAN training). In Section 4, we show that this change significantly improves convergence speed, as well as training stability. This lets us easily optimize a variety of different metrics, including our new metric, SPIDEr.

3 Methods

In this section, we explain our approach in more detail. First we discuss the policy gradient method, which can be used to robustly optimize any kind of reward function. Next we discuss which reward function to use. Finally, we discuss the model itself, which is a standard CNN-RNN.

3.1 Training using policy gradient

At time step t, we pick a discrete action, which corresponds to choosing a word $g_t \in \mathcal{V}$, using a stochastic policy or generator $\pi_\theta(g_t|s_t, \mathbf{x})$, where $s_t = g_{1:t-1}$ is the sequence of words chosen so far, $\mathbf{x}$ is the image and θ are the parameters of the model. Note that in our case the state transition function is deterministic: we simply append the word chosen at time t to get $s_{t+1} = s_t; g_t$ ($u; v$ is the concatenation of the strings u and v).

When we reach the end of the sequence (i.e., once the generator emits the end-of-sentence marker), we get a reward of $R(g_{1:T}|\mathbf{x}^n, \mathbf{y}^n)$, which is the score for producing caption $g_{1:T}$ given image $\mathbf{x}^n$ and ground truth caption (or set of
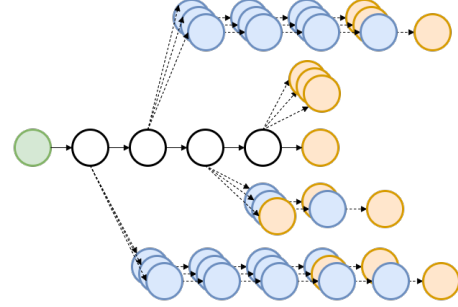

Figure 1: The value of each action is estimated as the average rewards received by its K rollout sequences (i.e. $K = 3$). Solid arrows indicate the sequence of actions being evaluated. The tokens in green and yellow are respectively BOS (beginning of sequence) and EOS (end of sequence) tokens. Sequences in blue are rollout sequences sampled from partial sequences. Note that rollout sequences do not always have the same length, as they are separately sampled from a stochastic policy.

captions) $\mathbf{y}^n$. This reward can be any function, such as BCMR or SPICE.

In typical reinforcement learning setting, an agent receives rewards at each intermediate step while future rewards are discounted to balance short-term and long-term gain. In our case, however, the agent receive zero reward during intermediate steps, observing a reward only at the end. To mitigate the lack of intermediate reward signal, we propose to estimate the value of intermediate states (partial sequence), via Monte-Carlo rollouts. This translate into significantly more robust credit assignment and efficient gradient estimation, as we show in Section 4. We define the value function of a partial sequence as its expected future reward:

$$V_\theta(g_{1:t}|\mathbf{x}^n, \mathbf{y}^n) = E_{g_{t+1:T}}[R(g_{1:t}; g_{t+1:T}|\mathbf{x}^n, \mathbf{y}^n)] \quad (1)$$

where the expectation is w.r.t. $g_{t+1:T} \sim \pi_\theta(\cdot|g_{1:t}, \mathbf{x}^n)$.

Our goal is to maximize the average reward starting from the initial (empty) state s_0 defined as:

$$J(\theta) = \frac{1}{N} \sum_{n=1}^{N} V_\theta(s_0|\mathbf{x}^n, \mathbf{y}^n) \quad (2)$$

where N is the number of examples in the training set. We now discuss how to optimize Eqn. (2). For simplicity, we will consider a single example n, so we will drop the $\mathbf{x}^n$ and $\mathbf{y}^n$ notation. To compute the gradient of $J(\theta)$, we can use the policy gradient theorem from [17]. In the special case of deterministic transition functions, this theorem simplifies as

shown below (see [2] for a proof):

$$\nabla_\theta V_\theta(s_0) = E_{g_{1:T}} \left[\sum_{t=1}^{T} \sum_{g_t \in \mathcal{V}} \nabla_\theta \pi_\theta(g_t|g_{1:t-1}) Q_\theta(g_{1:t-1}, g_t) \right] \tag{3}$$

where we define the Q function for a state-action pair as follows:

$$Q_\theta(g_{1:t-1}, g_t) = E_{g_{t+1:T}}[R(g_{1:t-1}; g_t; g_{t+1:T})] \tag{4}$$

We can approximate the gradient of the value function with M sample paths, $g_{1:T}^m \sim \pi_\theta$, generated from our policy. This gives

$$\nabla_\theta V_\theta(s_0) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T} E_{g_t} \left[\nabla_\theta \log \pi_\theta(g_t|g_{1:t-1}^m) \right.$$
$$\left. \times Q_\theta(g_{1:t-1}^m, g_t) \right] \tag{5}$$

where the expectation is w.r.t. $g_t \sim \pi_\theta(g_t|g_{1:t-1}^m)$, and where we have exploited the fact that

$$\nabla_\theta \pi_\theta(a|s) = \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} = \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)$$

If we use $M = 1$, we can additionally replace the E_{g_t} with the value in the sample path, g_t^m, as in REINFORCE. In our experiment, we used $M = 1$ and we subsequently drop the superscript m in the rest of this paper for notational clarity.

The only remaining question is how to estimate the function $Q(s_t, g_t)$. For this, we will follow [30] and use Monte Carlo rollouts. In particular, we first sample K continuations of the sequence $s_t; g_t$ to get $g_{t+1:T}^k$. Then we compute the average

$$Q(g_{1:t-1}, g_t) \approx \frac{1}{K} \sum_{k=1}^{K} R(g_{1:t-1}; g_t; g_{t+1:T}^k) \tag{6}$$

We estimate how good a particular word choice g_t is by averaging over all complete sequences sampled according to the current policy, conditioned on the partial sequence $g_{1:t-1}$ sampled from the current policy so far. This process is illustrated in Figure 1. If we are in a terminal state, we define $Q(g_{1:T}, \texttt{EOS}) = R(g_{1:T})$.

The above gradient estimator is an unbiased but high variance estimator. One way to reduce its variance is to estimate the expected baseline reward $E_{g_t}[Q(g_{1:t-1}, g_t)]$ using a parametric function; we will denote this baseline as $B_\phi(g_{1:t-1})$. We then subtract this baseline from $Q_\theta(g_{1:t-1}, g_t)$ to get the following estimate for the gradient (using $M = 1$ sample paths):

$$\nabla_\theta V_\theta(s_0) \approx \sum_{t=1}^{T} \sum_{g_t} [\pi_\theta(g_t|s_t) \nabla_\theta \log \pi_\theta(g_t|s_t)$$
$$\times (Q_\theta(s_t, g_t) - B_\phi(s_t))] \tag{7}$$

where $s_t = g_{1:t-1}$. Subtracting the baseline does not affect the validity of the estimated gradient, but reduces its variance. Here, we simply refer to prior work ([31], [23]) for a full derivation of this property.

We train the parameters ϕ of the baseline estimator to minimize the following loss:

$$L_\phi = \sum_t E_{s_t} E_{g_t} (Q_\theta(s_t, g_t) - B_\phi(s_t))^2 \tag{8}$$

In our experiments, the baseline estimator is an MLP which takes as input the hidden state of the RNN at step t. To avoid creating a feedback loop, we do not back-propagate gradients through the hidden state from this loss.

In language generation settings, a major challenge facing PG methods is the large action space. This is the case in our task, where the action space corresponds to the entire vocabulary of 8,855 symbols. To help "warm start" the training, we pre-train the RNN decoder (stochastic policy) using MLE training, before switching to PG training. This prevents the agent from performing random walks through exponentially many possible paths at the beginning of the training.

The overall algorithm is summarized in Algorithm 1. Note that the Monte Carlo rollouts only require a forward pass through the RNN, which is much more efficient than the forward-backward pass needed for the CNN. Additionally the rollouts can be also be done in parallel for multiple sentences. Consequently, PG training is only about twice as slow as MLE training (in wall time).

Algorithm 1: PG training algorithm

1 Input: $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{y}^n) : n = 1 : N\}$;
2 Train $\pi_\theta(g_{1:T}|x)$ using MLE on $\mathcal{D}$;
3 Train B_ϕ using MC estimates of Q_θ on a small subset of $\mathcal{D}$;
4 **for** each epoch **do**
5   **for** example (x^n, y^n) **do**
6   Generate sequence $g_{1:T} \sim \pi_\theta(\cdot|x^n)$;
7   **for** $t = 1 : T$ **do**
8   Compute $Q(g_{1:t-1}, g_t)$ for g_t with K Monte Carlo rollouts, using (6);
9   Compute estimated baseline $B_\phi(g_{1:t-1})$;
10   Compute $\mathcal{G}_\theta = \nabla_\theta V_\theta(s_0)$ using (7);
11   Compute $\mathcal{G}_\phi = \nabla_\phi L_\phi$;
12   SGD update of θ using $\mathcal{G}_\theta$;
13   SGD update of ϕ using $\mathcal{G}_\phi$;

3.2 Reward functions for the policy gradient

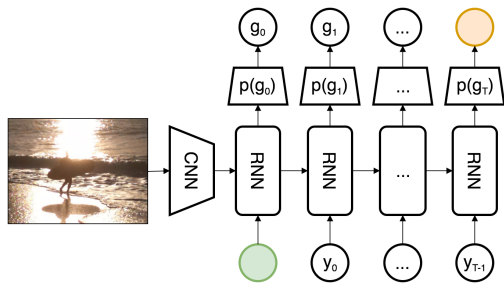We can use our PG method to optimize many different reward functions. Common choices include BLEU, CIDEr,

Figure 2: Model architecture of Show and Tell image captioning system [22]. The tokens in green and yellow are respectively BOS (beginning of sequence) and EOS (end of sequence) tokens. At testing time, output from previous time step g_{t-1} is used as input in lieu of y_{t-1}.

METEOR and ROUGE. Code for all of these metrics is available as part of the COCO evaluation toolkit.[1] We decided to use a weighted combination of all of these. Since these metrics are not on the same scale, we chose in our experiments the set of weights such that all metrics have approximately the same magnitude. More precisely, we choose the following weighted combination: 0.5*BLEU-1 + 0.5*BLEU-2 + 1.0*BLEU-3 + 1.0*BLEU-4 + 1.0*CIDEr + 5.0*METEOR + 2.0*ROUGE. Optimizing this weighted combination of BCMR gives state-of-the-art results on the COCO test set, as we discuss in Section 4.2.

One problem with the BCMR metrics is that individually they are not well correlated with human judgment [1]. We therefore also tried optimizing the recently introduced SPICE metric [1], which better reflects human estimates of quality. We use the open source release of the SPICE code[2] to evaluate the metric.

Interestingly, we have found that just optimizing SPICE tended to result in captions which are very detailed, but which often had many repeated phrases, as we show in Section 4. This is because SPICE measures semantic similarity (in terms of a scene graph) between sets of sentences, but does not pay attention to syntactical factors (modulo the requirement that the generated sentence be parseable). We therefore combined SPICE with the CIDEr metric (considered the best of the standard automatic metrics for COCO), a combination we call SPIDEr for short. Based on initial experiments, we decided to use an equal weighting for both.

3.3 Encoder-decoder architecture

We use a CNN-RNN architecture similar to the one proposed in the original Show-Tell paper [22]. A high-level diagram is shown in Figure 2. Each symbol in the vocabulary is

embedded as a 512 dimensional dense word embedding vector, whose values are initialized randomly.

The encoder CNN is implemented as an Inception-V3 [18] network pretrained on ImageNet[3].The RNN decoder is a one-layer LSTM with a state size of 512 units, initialized randomly. Each image is encoded by Inception-V3 as a dense feature vector of dimension $2,048$ which is then projected to 512 dimension with a linear layer and used as the initial state of RNN decoder.

At training time, we always feed in the ground truth symbol to the RNN decoder; at inference time we use just greedy decoding, where the sampled output is fed to the RNN as the next input symbol.

4 Results

4.1 Experimental protocol

We report results obtained by different methods on the COCO dataset. This has 82,081 training images, and 40,137 validation images, each with at least 5 ground truth captions. Following standard practice for methods that evaluate on the COCO test server, we hold out a small subset of 1,665 validation images for hyper-parameter tuning, and use the remaining combined training and validation set for training.

We preprocess the text data by lower casing, and replacing words which occur less than 4 times in the 82k training set with UNK; this results in a vocabulary size of 8,855 (identical to the one used in [22]). At training time, we keep all captions to their maximum lengths. At testing time, the generated sequences are truncated to 30 symbols in all experiments.

We use the Show-Tell model from [22] for all methods. We "pre-train" this model with MLE, and then optionally "fine tune" it with other methods, as we discuss below.

4.2 Automatic evaluation using BCMR metrics

In this section, we quantitatively evaluate various methods using the standard BCMR metrics on the COCO test set. Since the ground truth is not available for the test set, we submitted our results to the COCO online evaluation server[4] on Nov 2016. Table 1 shows the results of the top 5 methods (at the time of submission) on the official C-5 leaderboard, along with the results of our experiments. In particular, we tried training the Show-Tell model with the following methods: MLE, PG-BLEU-4, PG-CIDEr,

---

Submissions	CIDEr-D	Meteor	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4
MSM@MSRA [28]	0.984	0.256	0.542	0.739	0.575	0.436	0.330
Review Net [27]	0.965	0.256	0.533	0.720	0.550	0.414	0.313
ATT [29]	0.943	0.250	0.535	0.731	0.565	0.424	0.316
Google [22]	0.943	0.254	0.530	0.713	0.542	0.407	0.309
Berkeley LRCN [7]	0.921	0.247	0.528	0.718	0.548	0.409	0.306
MLE	0.947	0.251	0.531	0.724	0.552	0.405	0.294
PG-BLEU-4	0.966	0.249	0.550	0.737	0.587	**0.455**	**0.346**
PG-CIDEr	0.995	0.249	0.548	0.737	0.581	0.442	0.333
MIXER-BCMR	0.924	0.245	0.532	0.729	0.559	0.415	0.306
MIXER-BCMR-A	0.991	**0.258**	0.545	0.747	0.579	0.431	0.317
PG-BCMR	**1.013**	**0.257**	**0.55**	**0.754**	**0.591**	0.445	0.332
PG-SPIDEr	1.000	0.251	0.544	0.743	0.578	0.433	0.322

Table 1: Automatic evaluation on the official COCO C-5 test split, as of November 2016. Methods below the line are contributions from this paper.

PG-BCMR, PG-SPIDEr (with equal weight on SPICE and on CIDEr), and MIXER.

Our MLE method gets similar results to the scheduled sampling method of [22]. Not surprisingly, PG-BCMR significantly outperforms MLE training, since it directly optimizes for BCMR. Similarly, PG-BCMR outperforms PG-SPIDEr. We also showed that PG-BLEU-4 and PG-CIDEr specifically improves on the metric optimized for, compared to MLE baseline. In particular, PG-BLEU-4 successfully achieved the highest score on BLEU metrics, while largely neglected others. This demonstrates the general applicability of our optimization method to target any specific metric of interests. However, as shown below optimizing for the current COCO metrics does not translate into better captions.

We also see that our PG-BCMR method significantly outperforms all the top 5 methods, even the ones which use more sophisticated models, such as those based on attention (Montreal/Toronto, ATT, Review Net), those that use more complex decoders (Berkeley LRCN), and those that use high-level visual attributes (MSM@MSRA, ATT).

Our PG-BCMR method also outperforms the MIXER algorithm; we discuss this in more detail in Section 4.4.

4.3 Human evaluation

Table 2 shows some example captions generated by 6 different methods: MLE, PG-SPICE, MIXER-BCMR, MIXER-BCMR-A, PG-BCMR, and PG-SPIDEr. We see that PG-SPICE tends to generate ungrammatical sentences, with a lot of repeated phrases. This is because SPICE measures how well the scene graph induced by a sentence matches the ground truth scene graph, but is relatively insensitive to syntactic quality. However, when we combine
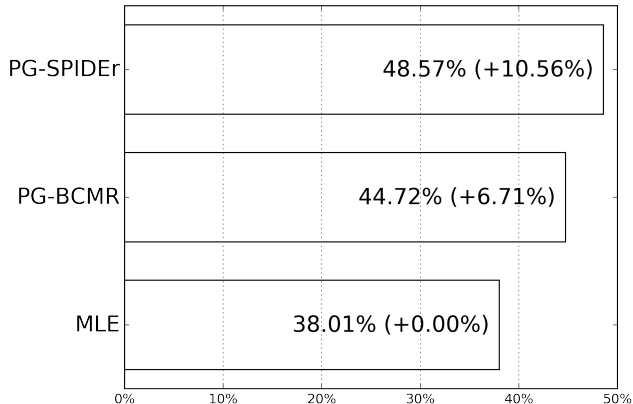

Figure 3: Results of human evaluation on 492 images randomly sampled from the COCO test set. We report the difference in percentage of "not bad" captions for each method compared to baseline 38% of MLE model.

SPICE with CIDEr, we get much better results. We therefore ignore pure SPICE in the rest of this paper.

We also see that PG-SPIDEr tends to generate more reasonable captions that PG-BCMR, even though it did worse on the COCO metrics. Also, both methods seem to be better than MLE. (See for example the third row in Table 2.)

To quantify this, we turn to a user study. In particular, we use a crowd sourcing platform, using raters who have prior experience with evaluating image captioning and other computer vision models. We showed each image-caption pair to 3 different raters, and asked them to evaluate it on a 4 point scale, depending on whether the caption is "bad", "okay", "good" or "excellent".[5] We then take the majority

---

[5] The definitions of these terms, which we gave to raters, is as follows.

Images	Ground Truth Captions	Generated Captions
	1. a red and yellow fire truck and some buildings 2. An overhead view shows a fire engine in the street. 3. A red and yellow fire truck with ladders on top 4. A firetruck is parked in the street in between stop lights. 5. A fire truck (ladder truck) drives down a street in the city.	• MLE: a red and white bus is driving down the street • PG-SPICE: a red double decker bus on a city street on a street with a bus on the street with a bus on the street in front of a bus on • MIXER-BCMR: a yellow bus driving down a city street . • MIXER-BCMR-A: a red fire truck driving down a city street . • PG-BCMR: a red bus driving down a city street . • PG-SPIDEr: a red fire truck is on a city street.
	1. A woman walking on a city street in a red coat. 2. A group of people that are standing on the side of a street. 3. A woman in a red jacket crossing the street 4. a street light some people and a woman wearing a red jacket 5. A blonde woman in a red coat crosses the street with her friend.	• MLE: a woman walking down a street while holding an umbrella . • PG-SPICE: a group of people walking down a street with a man on a street holding a traffic light and a traffic light on a city street with a city street • MIXER-BCMR: a group of people walking down a street . • MIXER-BCMR-A: a group of people walking down a street . • PG-BCMR: a group of people walking down a city street . • PG-SPIDEr: a group of people walking down a street with a traffic light .
	1. A group of people converse in an office setting. 2. A group of people playing a game with remote controllers. 3. Four young people have crowded into a small office. 4. A group of people standing next to each other in a room. 5. a group of people standing next to each other with some of them holding video game controllers	• MLE: a group of people standing around a living room . • PG-SPICE: a group of people in a room with a man in a chair holding a nintendo wii remote in a living room with a man in a chair holding a • MIXER-BCMR: a group of people standing in a living room . • MIXER-BCMR-A: a group of people standing in a living room playing a video game . • PG-BCMR: a group of people standing in a room . • PG-SPIDEr: a group of people playing a video game in a living room .
	1. A man looking through a book on top of a table. 2. A man sitting on a bed looking at a book 3. a man is flipping through a book on a bed 4. A man sitting on a bed flipping through pages of a book. 5. A man in a black jacket is flipping through a large book.	• MLE: a man sitting in front of a laptop computer . • PG-SPICE: a man sitting in front of a book and a laptop on a table with a laptop computer on top of a table with a laptop computer on top of • MIXER-BCMR: a man sitting in a chair with a book . • MIXER-BCMR-A: a man sitting at a table with a book . • PG-BCMR: a man sitting in front of a book . • PG-SPIDEr: a man sitting at a table with a book .

Table 2: Example captions from different models on COCO hold-out validation images.

p-value(X $>$Y)	PG-BCMR	MLE
PG-SPIDEr	**0.014**	**$<$0.001**
PG-BCMR	-	**0.003**

Table 3: p-values derived from a pairwise sign test applied to human ratings on 492 images from COCO test set. Statistically significant comparisons (at the 0.05 level) are shown in bold. X and Y correspond to rows and columns respectively.

vote to get the final rating. If no majority is found, the rating is considered unknown, and this image is excluded from the analysis.

Since current captioning systems are far from perfect, our main goal is to develop a captioning system that does not make "embarrassing" errors, we focus on measuring the fraction of captions that are classified as "not bad", which we interpret as the union of "okay", "good" and "excellent".

As "quality control", we first evaluated 505 ground truth captions from the COCO validation set. Humans said that 87% of these captions were "not bad". Some of the 13% of ground truth captions that were labeled "bad" do indeed contain errors[6], due to the fact that COCO captions were generated by AMT workers who are not perfect. On the other hand, some captions seem reasonable to us, but did not meet the strict quality criteria our raters were looking for. In any case, 87% is an approximate upper bound on performance we can hope to achieve on the COCO test set.

We then randomly sampled 492 images from the test set (for which we do not have access to the ground truth captions), and generated captions from all of them using our 3 systems, and sent them for human evaluation. Figure 3 shows the fraction of captions that are "not bad" compared to the MLE baseline of 38%. We draw the following conclusions:

- All methods are far below the human ceiling of 87%.
- All PG methods outperform MLE training by a significant margin (see Table 3 for pairwise p-value analysis). This is because the PG methods optimize metrics that are much more closely related to caption quality than the likelihood score.

Excellent: "The caption correctly, specifically and completely describes the foreground/main objects/events/theme of the image." Good: "The caption correctly and specifically describes most of the foreground/main objects/events/theme of the image, but has minor mistakes in some minor aspects." Okay: "The caption correctly describes some of the foreground/main objects/events/theme of the image, but is not specific to the image and has minor mistakes in some minor aspects." Bad: "The caption misses the foreground/main objects/events/theme of the image or the caption contains obviously hallucinated objects/activities/relationships."

[6] For example, some captions contain repeated words, e.g., "this is an image image of a modern kitchen". Others contain typos, e.g., "a blue and white truck with pants in it's flat bed". And some do not make semantic sense, e.g., "A crowd of people parked near a double decker bus".

- PG-SPIDEr outperforms PG-BCMR by a 4% margin, despite the fact that PG-BCMR outperforms PG-SPIDEr on all the COCO metrics. This is because SPIDEr captures both fluency and semantic properties of the caption, both of which human raters are told to pay attention to, whereas BCMR is a more syntactic measure.

4.4 Comparison with MIXER

We now compare our method with [15] in more detail. When using MLE training, they get a BLEU-4 score of 0.278, and when they used MIXER to optimize BLEU-4, they get 0.292, which is about 0.1 better. (They do not report any other results besides BLEU-4.) Note, however, that their numbers are not comparable to the numbers in Table 1, since they are evaluated on 5000 images from the COCO validation set, and not the official test server.

In order to do a fair comparison with our work (and other publications), we reimplemented their algorithm. When we used their train/test split[7], and used our implementation of MIXER to optimize BLEU-4, we were able to reproduce their BLEU-4 result. We then used MIXER to optimize BCMR, using the standard train/test split. Once again, we see that the BLEU-4 score is about 0.1 better than MLE, but the other metrics are sometimes worse, whereas our PG method was significantly better than MLE across all metrics.

Upon digging into their open source code[8], we noticed that they used a different optimization algorithm: we use Adam [10], whereas they used vanilla stochastic gradient descent with a hard-coded learning rate decay. When we reran MIXER-BCMR with Adam (a combination we call MIXER-BCMR-A) with well tuned parameters, we saw significantly improved results, which come closer to our best results obtained with PG-BCMR in almost all metrics (and even slightly beating us in Meteor).

However, the final results are not the entire story. We have found that MIXER is much slower to converge, and much less stable during training, than our PG-Rollout method. This is illustrated in Figure 4, where we plot the BCMR metrics on the validation set for three methods: PG-BCMR (blue), MIXER-BCMR-A (red), MIXER-BCMR (green). We show the best of 10 runs for MIXER, and a single run for PG. (Repeated runs of PG give similar performance.)

We see that MIXER-BCMR with vanilla SGD does poorly. Using Adam helps a lot, but performance is still very unstable; in particular, we had to try the method 10 times, with different learning rates and gradient multipliers for the baseline estimator, to get their best result shown in Figure 4.

Note that the plateau in MIXER's performance curve for the first 500k steps is because MIXER starts with just MLE

---

[7] We thank Marc'Aurelio Ranzato for sharing their code and data split.
[8] https://github.com/facebookresearch/MIXER

(a) BLEU-4
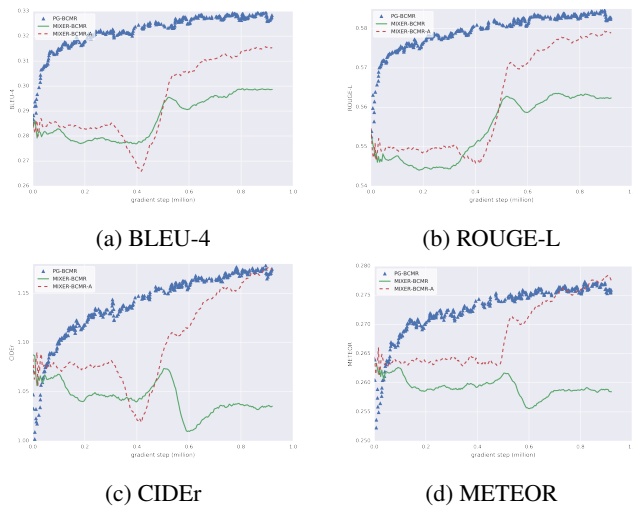
(b) ROUGE-L

(c) CIDEr

(d) METEOR

Figure 4: Performance of PG-BCMR (blue-triangle), MIXER-BCMR-A (red-dashed) and MIXER-BCMR (green-solid) on the validation set during the first 1 million gradient steps.

training; no progress is made during this time, since we initialize the models with a model that was already trained with MLE. At the 500k epoch, MIXER uses MLE for the first 6 words in the caption, and REINFORCE for the rest. At the 600k epoch, MIXER switches to REINFORCE for the entire sequence. Over time, the baseline estimator is learned, and this can compensate for the inaccurate estimate of future rewards. Hence eventually, MIXER-BCMR-A can catch up with our PG-Rollout method.

5 Conclusion

In this paper, we have proposed a robust and efficient policy gradient method, and successfully applied it to optimize a variety of captioning metrics. By optimizing the standard COCO metrics, we show that we can achieve state of the art results, according to the leaderboard. However, we also show that these metrics do not correlate well with human judgement. We therefore also proposed a new metric, SPIDEr, and show that optimizing this with our new algorithm produces qualitatively superior results, as judged by human raters.

References

[1] P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In ECCV, 2016. 1, 2, 5

[2] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An Actor-Critic algorithm for sequence prediction. Arxiv, 24 July 2016. 3, 12

[3] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In ACL Workshop on MT, volume 29, pages 65–72, 2005. 1

[4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In NIPS, 2015. 2

[5] R. Bernardi, R. Cakici, D. Elliott, A. Erdem, E. Erdem, N. Ikizler-Cinbis, F. Keller, A. Muscat, and B. Plank. Automatic description generation from images: A survey of models, datasets, and evaluation measures. J. of AI Research, 55:409–442, 2016. 2

[6] 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. 1

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In CVPR, 2015. 6

[8] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell. Generating visual explanations. ECCV, 2016. 2

[9] F. Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? Arxiv, 16 Nov. 2015. 2

[10] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In ICLR, 2015. 8

[11] C.-Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In NAACL, pages 71–78, 2003. 1

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In ECCV, 1 May 2014. 1

[13] M. Norouzi, S. Bengio, Z. Chen, N. Jaitly, M. Schuster, Y. Wu, and D. Schuurmans. Reward augmented maximum likelihood for neural structured prediction. In NIPS, 2016. 3

[14] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. In Proc. ACL, pages 311–318, 2002. 1

[15] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. Arxiv, 2015. 1, 2, 3, 8

[16] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Minimum risk training for neural machine translation. In Proc. ACL, 2016. 3

[17] R. S. Sutton, D. Mc Allester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In NIPS, 1999. 1, 2, 3

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567, 2015. 5

[19] K. Tran, X. He, L. Zhang, J. Sun, C. Carapcea, C. Thrasher, C. Buehler, and C. Sienkiewicz. Rich image captioning in the wild. In CVPR, 2016. 2

[20] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In CVPR, pages 4566–4575, 2015. 1

[21] S. Venugopalan, L. A. Hendricks, M. Rohrbach, R. Mooney, T. Darrell, and K. Saenko. Captioning images with diverse objects. Arxiv, 24 June 2016. 2

[22] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In CVPR, 2015. 2, 5, 6

[23] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning J., 8(3-4):229–256, 1 May 1992. 2, 4, 12

[24] Q. Wu, C. Shen, A. van den Hengel, L. Liu, and A. Dick. What value high level concepts in vision to language problems? In CVPR, 2016. 2

[25] S. Wu, J. Wieland, O. Farivar, and J. Schiller. Automatic alt-text: Computer-generated image descriptions for blind users on a social network service. In Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, pages 1180–1192. ACM, 25 Feb. 2017. 1

[26] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In ICML, 2015. 2

[27] Z. Yang, Y. Yuan, Y. Wu, R. Salakhutdinov, and W. W. Cohen. Review networks for caption generation. In NIPS, 2016. 2, 6

[28] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei. Boosting image captioning with attributes. In OpenReview, 2016. 2, 6

[29] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. arXiv preprint arXiv:1603.03925, 2016. 6

[30] L. Yu, W. Zhang, J. Wang, and Y. Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. Arxiv, 18 Sept. 2016. 3, 4

[31] W. Zaremba and I. Sutskever. Reinforcement learning neural turing machines-revised. arXiv preprint arXiv:1505.00521, 2015. 4, 12

A Full derivation of policy gradient

Given an image $\mathbf{x}^n$ and the ground-truth captions $\mathbf{y}^n$ associated with $\mathbf{x}^n$, the reward for a generated caption $\bar{g} = g_{1:T} = \{g_1, g_2, ..., g_T\}$ is defined as

$$R(\bar{g}|\mathbf{x}^n, \mathbf{y}^n) = \sum_{t=1}^{T} r(g_t|g_{1:(t-1)}, \mathbf{x}^n, \mathbf{y}^n), \qquad (9)$$

where $r(g_t|g_{1:(t-1)}, \mathbf{x}^n, \mathbf{y}^n)$ is the incremental reward for word g_t, given previously generated words $g_{1:(t-1)}$ and $(\mathbf{x}^n, \mathbf{y}^n)$. We define

$$r(g_1|g_{1:0}, \mathbf{x}^n, \mathbf{y}^n) = r(g_1|\mathbf{x}^n, \mathbf{y}^n) \qquad (10)$$

for notation consistency. Caption $\bar{g}$ is generated by sampling one word at each time step from the given vocabulary $\mathcal{V}$ using the conditional probability or policy $\pi_\theta(\bar{g}|\mathbf{x}^n)$ parameterized by θ. Here θ is the set of weights in the CNN-RNN neural network covered in section 3.3. The expected reward for the captions generated by following the policy $\pi_\theta(\bar{g}|\mathbf{x}^n)$ is

$$V_\theta(\mathbf{x}^n) = E_{\bar{g}}[R(\bar{g}|\mathbf{x}^n, \mathbf{y}^n)] = \sum_{\bar{g}} \pi_\theta(\bar{g}|\mathbf{x}^n)R(\bar{g}|\mathbf{x}^n, \mathbf{y}^n). \qquad (11)$$

The goal is to maximize the objective function

$$J(\theta) = \frac{1}{N} \sum_{n=1}^{N} V_\theta(\mathbf{x}^n) \qquad (12)$$

where N is the number of images in a given training data set.

In this paper we use the gradient-based methods to find the optimal solution θ^*. In view of equation (11), to compute the gradient

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta V_\theta(\mathbf{x}^n), \qquad (13)$$

we just need to focus on the term

$$\nabla_\theta V_\theta(\mathbf{x}^n) = \sum_{\bar{g}} \nabla_\theta \pi_\theta(\bar{g}|\mathbf{x}^n)R(\bar{g}|\mathbf{x}^n, \mathbf{y}^n) \qquad (14)$$

$$= \sum_{\bar{g}} \pi_\theta(\bar{g}|\mathbf{x}^n)\nabla_\theta \log(\pi_\theta(\bar{g}|\mathbf{x}^n))R(\bar{g}|\mathbf{x}^n, \mathbf{y}^n)$$

$$= E_{\bar{g}}[\nabla_\theta \log(\pi_\theta(\bar{g}|\mathbf{x}^n))R(\bar{g}|\mathbf{x}^n, \mathbf{y}^n)]. \qquad (15)$$

In view of the chain rule applied to the joint conditional probability

$$\pi_\theta(\bar{g}|\mathbf{x}^n) = \prod_{t=1}^{T} \pi_\theta(g_t|g_{1:(t-1)}, \mathbf{x}^n), \qquad (16)$$

we have

$$\nabla_\theta \log(\pi_\theta(\bar{g}|\mathbf{x}^n)) = \sum_{t=1}^{T} \nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}, \mathbf{x}^n)). \qquad (17)$$

Similar to equation (10), we define

$$\pi_\theta(g_1|g_{1:0}, \mathbf{x}^n) = \pi_\theta(g_1|\mathbf{x}^n). \qquad (18)$$

Equations (15) and (17) suggest a straight forward sequence-level Monte Carlo sampling algorithm. But we will show next that this can be further improved.

Taking the derivative on both sides of the equation (16) leads to

$$\nabla_\theta \pi_\theta(\bar{g}|\mathbf{x}^n) = \nabla_\theta \pi_\theta(g_1|\mathbf{x}^n) \prod_{t=2}^{T} \pi_\theta(g_t|g_{1:(t-1)}, \mathbf{x}^n)$$

$$+ \pi_\theta(g_1|\mathbf{x}^n)\nabla_\theta \pi_\theta(g_2|g_1, \mathbf{x}^n) \prod_{t=3}^{T} \pi_\theta(g_t|g_{1:(t-1)}, \mathbf{x}^n)$$

$$+ \dots$$

$$+ \prod_{t=1}^{T-1} \pi_\theta(g_t|g_{1:(t-1)}, \mathbf{x}^n)\nabla_\theta \pi_\theta(g_T|g_{1:(T-1)}, \mathbf{x}^n)$$

$$= \sum_{t=1}^{T} \pi_\theta(g_{1:(t-1)}|\mathbf{x}^n)\nabla_\theta \pi_\theta(g_t|g_{1:(t-1)}, \mathbf{x}^n)$$

$$\times \pi_\theta(g_{t+1:T}|g_{1:t}, \mathbf{x}^n), \qquad (19)$$

where the first equality is due to simple calculus rules, and the second equality is due to the chain rule in equation (16). Here we define

$$\pi_\theta(g_{1:0}|\mathbf{x}^n) = 1. \qquad (20)$$

$$\pi_\theta(g_{(T+1):T}|g_{1:T}, \mathbf{x}^n) = 1. \qquad (21)$$

for notation consistency in equation (19).

To simplify the notation and unless confusion arises, we will drop the conditional terms $(\cdot|\mathbf{x}^n, \mathbf{y}^n)$ and $(\cdot|\mathbf{x}^n)$ in the following derivation.

Substituting equation (19) into equation (14) and changing the order of summation, we obtain

$$\nabla_\theta V_\theta(\mathbf{x}^n) = \sum_{t=1}^{T} \sum_{g_{1:(t-1)}} \pi_\theta(g_{1:(t-1)}) \sum_{g_t} \nabla_\theta \pi_\theta(g_t|g_{1:(t-1)})$$

$$\times \sum_{g_{(t+1):T}} \pi_\theta(g_{t+1:T}|g_{1:t})R(\bar{g})$$

$$= \sum_{t=1}^{T} E_{g_{1:(t-1)}}[E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))E_{g_{(t+1):T}}[R(\bar{g})]]]. \qquad (22)$$

Note that In view of equation (9), we can partition the reward $R(\bar{g})$ in equation (22) into two parts

$$R(\bar{g}) = R(g_{1:(t-1)}) + R(g_{1:(t-1)}, g_{t:T}), \qquad (23)$$

where

$$R(g_{1:(t-1)}, g_{t:T}) = \sum_{i=t}^{T} r(g_i|g_{1:(i-1)}). \quad (24)$$

Here we want to show that the contribution of $R(g_{1:(t-1)})$ to the summation in equation (22) is zero. Specifically, the inner summation for $R(g_{1:(t-1)})$ becomes

$$E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))E_{g_{(t+1):T}}[R(g_{1:(t-1)})]]$$
$$= R(g_{1:(t-1)})E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))E_{g_{(t+1):T}}[1]]$$
$$= R(g_{1:(t-1)})\sum_{g_t}[\nabla_\theta \pi_\theta(g_t|g_{1:(t-1)})]$$
$$= R(g_{1:(t-1)})\nabla_\theta E_{g_t}[1] = 0, \quad (25)$$

where the first equality is due to the fact that $R(g_{1:(t-1)})$ is independent of g_t and $g_{(t+1):T}$, the second equality is due to the definition of expectation, and the third equality is due to the exchange of sum and derivative. The contribution of $R(g_{1:(t-1)}, g_{t:T})$ to the summation in equation (22) can be simplified as well. Specifically, the inner most summation becomes

$$Q_\theta(g_{1:(t-1)}, g_t) = E_{g_{(t+1):T}}[R(g_{1:(t-1)}, g_{t:T})]$$
$$= E_{g_{(t+1):T}}[r(g_t|g_{1:(t-1)}) + R(g_{1:t}, g_{(t+1):T})]$$
$$= r(g_t|g_{1:(t-1)})E_{g_{(t+1):T}}[1] + E_{g_{(t+1):T}}[R(g_{1:t}, g_{(t+1):T})]$$
$$= r(g_t|g_{1:(t-1)}) + E_{g_{(t+1):T}}[R(g_{1:t}, g_{(t+1):T})], \quad (26)$$

where the second equality is due to the definition in equation (24), and the third equality is due to the fact that $r(g_t|g_{1:(t-1)})$ is independent of $g_{(t+1):T}$. Hence equation (22) becomes

$$\nabla_\theta V_\theta(\mathbf{x}^n) = \sum_{t=1}^{T} E_{g_{1:(t-1)}}[E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))$$
$$\times Q_\theta(g_{1:(t-1)}, g_t)]] \quad (27)$$
$$= \sum_{t=1}^{T} E_{g_{1:T}}[E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))Q_\theta(g_{1:(t-1)}, g_t)]]$$
$$= E_{g_{1:T}}[\sum_{t=1}^{T} \sum_{g_t \in \mathcal{V}}[\nabla_\theta \pi_\theta(g_t|g_{1:(t-1)})Q_\theta(g_{1:(t-1)}, g_t)]], \quad (28)$$

where the second equality is due to the fact that the term $E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))Q_\theta(g_{1:(t-1)}, g_t)]$ is independent of $g_{t:T}$. Equation (28) is also shown in [2].

B Variance reduction

The Monte Carlo sampling used to compute the expectation in equations (26) and (27) typically suffers from high

variance problem due to very high dimensionality of the sample space. For example, the vocabulary of the COCO data set is about nine thousand. If we set caption length $T = 30$, then the discrete sample space has 9000^{30} grid points. One commonly used solution is to subtract a constant from $Q_\theta(g_{1:(t-1)}, g_t)$ in equation (27). The inner summation in equation (27) now becomes

$$E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))(Q_\theta(g_{1:(t-1)}, g_t) - B)]. \quad (29)$$

Since the second term is

$$E_{g_t}[\nabla_\theta \log(\pi_\theta(g_t|g_{1:(t-1)}))B] = \nabla_\theta E_{g_t}[B] = 0, \quad (30)$$

this manipulation does not theorectically change the gradient in equation (27). Prior work in [31, 23] show that

$$B = E_{g_t}[Q_\theta(g_{1:(t-1)}, g_t)] \quad (31)$$

is effective in reducing the variance due to small sample size.