Adaptively Aligned Image Captioning via Adaptive Attention Time

Lun Huang1 Wenmin Wang1,3* Yaxian Xia1 Jie Chen1,2
1School of Electronic and Computer Engineering, Peking University
2Peng Cheng Laboratory
3Macau University of Science and Technology
huanglun@pku.edu.cn, {wangwm@ece.pku.edu.cn, wmwang@must.edu.mo}
xiayaxian@pku.edu.cn, chenj@pcl.ac.cn

Abstract

Recent neural models for image captioning usually employ an encoder-decoder framework with an attention mechanism. However, the attention mechanism in such a framework aligns one single (attended) image feature vector to one caption word, assuming one-to-one mapping from source image regions and target caption words, which is never possible. In this paper, we propose a novel attention model, namely Adaptive Attention Time (AAT), to align the source and the target adaptively for image captioning. AAT allows the framework to learn how many attention steps to take to output a caption word at each decoding step. With AAT, an image region can be mapped to an arbitrary number of caption words while a caption word can also attend to an arbitrary number of image regions. AAT is deterministic and differentiable, and doesn’t introduce any noise to the parameter gradients. In this paper, we empirically show that AAT improves over state-of-the-art methods on the task of image captioning. Code is available at https://github.com/husthuaan/AAT.

1 Introduction

Image captioning aims to automatically describe the content of an image using natural language [13, 29, 16, 6]. It is regarded as a kind of “translation” task that translates the content in an image to a sequence of words in some language.

Inspired by the development of neural machine translation [22], recent approaches to image captioning that adopt an encoder-decoder framework with an attention mechanism have achieved great success. In such a framework, a CNN-based image encoder is used to extract feature vectors for a given image, while an RNN-based caption decoder to generate caption words recurrently. The attention mechanism selects one attended feature vector for each one decoding step.

Despite the success that the existing approaches have achieved, there are still limitations in the encoder-decoder framework. The attention model generates one single image feature vector at each decoding step, which subsequently provides the decoder information for making the prediction about a caption word. Obviously, one-to-one mapping from image regions to caption words is assumed in the paradigm, which, however, is never possible and may involve the following issues: 1) unnecessary or even misleading visual information is forcibly given to all caption words, no matter whether they require visual clues or not [15]; 2) visual information in the decoder accumulates over time; however, words at earlier decoding steps require more knowledge about the image, but they are actually provided less; 3) it’s hard to understand interactions between objects by analyzing one single

*Corresponding author

attended weighted averaged feature vector. To conclude, a decoder requires a different number of attention steps in different situations, which is what one-to-one mapping can’t provide.

To this end, we develop a novel attention model, namely Adaptive Attention Time (AAT), to realize adaptive alignment from image regions to caption words. At each decoding step, depending on its confidence, AAT decides whether to take an extra attention step, or directly output a caption word and move on to the next decoding step. If AAT does so, then at the subsequent attention step, AAT will take one more attended feature vector into the decoder, and again, AAT will make the decision. The attention process continues, until AAT is confident enough to output a word. With the techniques from Adaptive Computation Time (ACT) [8], we are able to make AAT deterministic and differentiable. Furthermore, we take advantage of the multi-head attention [23] to introduce fewer attention steps and make it easier to comprehend interactions among objects in an image.

We evaluate the effectiveness of AAT by comparing it with a base attention model, which takes one attending step as one decoding step, and a recurrent attention model, which takes a fixed number of attention steps for each decoding step. We show that those two models are special cases of AAT, and AAT is superior to them with empirical results. Experiments also show that the proposed AAT outperforms previously published image captioning models. And one single image captioning model can achieve a new state-of-the-art performance of 128.6 CIDEr-D [24] score on MS COCO dataset offline test split.

2 Related Work

Image Captioning. Early approaches to image captioning are rule/template-based [30, 21] which generate slotted caption templates and use the outputs of object detection, attribute prediction and scene recognition to fill in the slots. Recently, inspired by the great development of neural machine translation [22], neural-based encoder-decoder framework became the mainstream choice for image captioning. For instance, an end-to-end framework is proposed with a CNN encoding the image to feature vector and an LSTM decoding it to caption [25]. Further in [27], the spatial attention mechanisms on CNN feature map is used to incorporate visual context and is implicitly conditioned on the text generated so far. In [19], reinforcement learning algorithms are designed to directly optimize the non-differentiable evaluation metrics (e.g., BLEU [17] and CIDEr [24]). Later, [15] introduces an adaptive attention mechanism to decide when to activate the visual attention. More recently, semantic information such as objects, attributes and relationships are integrated to generate better descriptions [32, 2, 31, 28, 9].

Adaptive Computation Time. Adaptive computation time (ACT) was proposed as an algorithm to allow recurrent neural networks to learn how many computational steps to take between receiving an input and emitting an output. Later, ACT is utilized to design neural networks of a dynamic number of layers or operations [26, 5]. In this paper, we implement our Adaptive Attention Time (AAT) model by using the techniques of ACT.

3 Method

Our image captioning model with Adaptive Attention Time (AAT) is developed upon the attention based encoder-decoder framework. In this section, we first describe the framework in Section 3.1, then show how we implement AAT to realize adaptive alignment in Section 3.2.

3.1 Model Framework

Image Encoder. The encoder in the attentive encoder-decoder framework is to extract a set of image feature vectors $A = \{a_1, a_2, \ldots, a_k\}$ of different image regions for the given image, where $k$ is the number of image regions. A typical choice is a pre-trained Faster-RCNN [18] model [2].

Caption Decoder. A two-layer LSTM decoder is popular in recent researches for image captioning [2]. We formalize a decoder with this kind of structure into three parts: an input module, an attention module and an output module.
The input module, which models the input word as well as the context information of the decoder, consists of an LSTM layer (LSTM1). The process of this layer is:

\[
(h^1_t, m^1_t) = \text{LSTM}_1([\Pi_t W_e, \tilde{a} + c_{t-1}], (h^1_{t-1}, m^1_{t-1}))
\]  

(1)

where \(h^1_t, m^1_t\) are the hidden state and memory cell of LSTM1, \(W_e\) is a word embedding matrix, and \(\Pi_t\) is one-hot encoding of the input word at time step \(t\), \(c_{t-1}\) is the previous context vector of the decoder, which is also the output of the attention module and input of the output module, and \(\tilde{a} = \frac{1}{T} \sum a_i\) is the mean-pooling of \(A\) and added to \(c_{t-1}\) to provide global information to the input module.

The attention module applies the proposed attention model on the image feature set \(A\) and generates a context vector \(c_t\), which is named as Adaptive Attention Time (AAT) and will be introduced in the following section together with other comparing attention models.

The output module passes \(c_t\) through a linear layer with softmax activation to predict the probability distribution of the vocabulary:

\[
p(y_t \mid y_{1:t-1}) = \text{softmax}(c_t W_p + b_p)
\]  

(2)

### 3.2 Adaptive Alignment via Adaptive Attention Time

To implement adaptive alignment, we allow the decoder to take arbitrary attention steps for every decoding step. We first introduce the base attention mechanism which takes one attention step for one decoding step, then the recurrent version which allows the decoder to take a fixed number of attention steps, finally the adaptive version Adaptive Attention Time (AAT), which adaptively adjusts the attending time.

#### 3.2.1 Base Attention Model

Common practice of applying attention mechanism to image captioning framework is to measure and normalize the attention score \(\alpha_i\) of every candidate feature vector \(a_i\) with the given query (i.e. \(h^1_t\)), resulting in one single weighted averaged vector \(\tilde{a}_t = \sum a_i \alpha_i\) over the whole feature vector set \(A\).

By applying an attention mechanism to the image captioning framework, we obtain the attended feature vector:

\[
\tilde{a}_t = f_{att}(h^1_t, A)
\]  

(3)

Next \(\tilde{a}_t\) together with \(h^1_t\) is fed into another LSTM layer (LSTM2):

\[
(h^2_t, m^2_t) = \text{LSTM}_2([\tilde{a}_t, h^1_t], (h^2_{t-1}, m^2_{t-1}))
\]  

(4)

We let the context vector to be the output hidden state \(c_t = h^2_t\).

#### 3.2.2 Recurrent Attention Model

Rather than limiting the attention module to accessing the feature vectors only once for one decoding step. We allow attending for multiple times with recurrent attention.
First, we need to make the attention output vary with attention time, and we construct the attention query \( q_{t,n} \) at attention time step \( n \) of decoding time step \( t \) by transforming \( h_t^1 \) and \( h_{t,n-1}^1 \) through a linear layer:

\[
q_{t,n} = [h_t^1, h_{t,n-1}^1]W_q + b_q
\]

where \( h_{t,0}^2 = h_{t-1}^2 \), and \( h_{t,n-1}^2 \) is the output of LSTM2 at attention time step \( n - 1 \) of decoding time step \( t \).

Then \( q_{t,n} \) is fed to the attention module and obtain \( \hat{a}_t = f_{att}(q_{t,n}, A) \), which is further fed to LSTM2:

\[
h_{t,n}^2, m_{t,n}^2 = \text{LSTM2}([a_{t,n}, h_t^1], (h_{t-1,j}^2, m_{t-1,j}^2))
\]

We set \( h_t^2 = h_{t,M_r}, m_t^2 = m_{t,M_r} \), where \( h_{t,M_r}, m_{t,M_r} \) are the hidden state and memory cell of the last attention step and \( M_r \) is the attention steps for each decoding step.

We let the context vector to be the output hidden state at the last attention step \( c_t = h_{t,M_r}^2 = h_t^2 \).

### 3.2.3 Adaptive Attention Time

We further allow the decoder attending to the image for arbitrary times with Adaptive Attention Time.

To determine how many times to perform attention, an extra confidence network is added to the output of LSTM2, since it’s used to predict probability distribution. We design the confidence network as a two-layer feed forward translation:

\[
p_{t,n} = \begin{cases} 
\sigma(\max(0, h_t^1 W_1 + b_1)) W_2 + b_2 & n = 0 \\
\sigma(\max(0, h_{t,n}^2 W_1 + b_1)) W_2 + b_2 & n > 0 
\end{cases}
\]

and the total attention steps is determined by:

\[
N(t) = \min\{n' : \prod_{n=0}^{n'} (1 - p_{t,n}) < \epsilon \}
\]

where \( \epsilon \) is a threshold which is a small value and slightly greater than 0 (\( \epsilon = 1e-4 \) in this paper).

The final hidden state and memory cell of LSTM2 are computed as:

\[
\begin{align*}
\begin{cases} 
\hat{h}_t^2 = \beta_{t,0} h_t^1 + \sum_{n=1}^{N(t)} \beta_{t,n} h_{t,n}^2, \\
\hat{m}_t^2 = \beta_{t,0} m_t^2 + \sum_{n=1}^{N(t)} \beta_{t,n} m_{t,n}^2 
\end{cases}
\end{align*}
\]

where

\[
\beta_{t,n} = \begin{cases} 
p_{t,0} & n = 0 \\
p_{t,n} \prod_{n'=0}^{n-1} (1 - p_{t,n'}) & n > 0 
\end{cases}
\]

and \( h_t^1 \) is added to show that we can obtain context directly from the input module without attending to image feature vectors. and \( c_{t-1}^2 \) is to show that when deciding not to attend image feature, the memory cell should maintain the previous state and not be updated.

**Normalization.** We normalize the hidden state and memory cell at each attention step: \( h_{t,n}^2 = \text{LayerNorm}_h(h_{t,n}^2) \), and \( m_{t,n}^2 = \text{LayerNorm}_m(m_{t,n}^2) \)

We also normalize the weight of each attention step to make the sum of them to 1: \( \beta_{t,n} \leftarrow \frac{\beta_{t,n}}{\sum_{n=0}^{N(t)} \beta_{t,n}} \).

**Time Cost Penalty.** We add a “attention time” loss to the training loss in order to penalize the time cost for attention steps:

\[
L_t^a = \lambda (N(t) + \sum_{n=0}^{N(t)} (n + 1)(1 - p_{t,n}))
\]

where \( \lambda \) is a hyper-parameter, \( (n + 1)(1 - p_{t,n}) \) encourages \( p_{t,n} \) to be larger so that the total attention steps can be reduced, and \( N(t) \) is added to indicate the attention time steps and doesn’t contribute to the parameter gradients.
Minimum and Maximum Attention Steps. We can set a minimum attention step \( M_{\text{min}} \) to make the attention module takes at least \( M_{\text{min}} \) attention steps for each decoding step, by simply setting \( p_{t,n} = 0 \) for \( 0 \leq n < M_{\text{min}} - 1 \).

We can also set a maximum attention steps \( M_{\text{max}} \) to make sure the attention module takes at most \( M_{\text{max}} \) attention steps by modifying \( N(t) \):

\[
N(t) = \min\{M_{\text{max}}, \min\{n': \prod_{n=0}^{n'} (1 - p_{t,n}) < \epsilon\}\} \tag{12}
\]

As can be seen, the process of AAT is deterministic and can be optimized directly.

We let the context vector to be the weighted average over all hidden states at all attention steps

\[
c_t = \beta_{t,0} h_1^t + \sum_{n=1}^{N(t)} \beta_{t,n} h_{t,n}^2 = h_2^t.
\]

3.2.4 Connections between Different Attention Models

*Base* attention model is a special case of *recurrent* attention model when \( M_r = 1 \), and *recurrent* attention model is a special case of *adaptive* attention model when \( M_{\text{max}} = M_{\text{min}} = M_r \).

4 Experiments

4.1 Dataset, Settings, and Metrics

**Dataset.** We evaluate our proposed method on the popular MS COCO dataset [14]. MS COCO dataset contains 123,287 images labeled with at least 5 captions, including 82,783 for training and 40,504 for validation. MS COCO also provides 40,775 images as the test set for online evaluation. We use the “Karpathy” data split [11] for the performance comparisons, where 5,000 images are used for validation, 5,000 images for testing, and the rest for training.

**Settings.** We convert all sentences to lower case, drop the words that occur less than 5 times, and trim each caption to a maximum of 16 words, which results in a vocabulary of 10,369 words.

Identical to [2], we employ Faster-RCNN [18] pre-trained on ImageNet and Visual Genome to extract bottom-up feature vectors of images. The dimension of the original vectors is 2048, and we project them to the dimension of \( d = 1024 \), which is also the hidden size of the LSTM of the decoder and the size of word embedding.

As for the training process, we train our model under cross-entropy loss for 20 epochs with a mini-batch size of 10, and ADAM [12] optimizer is used with a learning rate initialized with 1e-4 and annealed by 0.8 every 2 epochs. We increase the probability of feeding back a sample of the word posterior by 0.05 every 3 epochs [4]. Then we use self-critical sequence training (SCST) [19] to optimize the CIDEr-D score with REINFORCE for another 20 epochs with an initial learning rate of 1e-5 and annealed by 0.5 when the CIDEr-D score on the validation split has not improved for some training steps.

**Metrics.** We use different metrics, including BLEU [17], METEOR [20], ROUGE-L [7], CIDEr-D [24] and SPICE [1], to evaluate the proposed method and compare with others. All the metrics are computed with the publicly released code.

4.2 Ablative Studies

**Attention Time Steps (Attention Model).** To show the effectiveness of adaptive alignment for image captioning, we compare the results of different attention models with different attention time steps, including *base*, which uses the conventional attentive encoder-decoder framework and forces to align one (attended) image region to one caption word; *recurrent*, which at each decoding step attends to a fixed length of image regions recurrently; and *adaptive*, the proposed method in this paper, which attends to an adaptive length of image regions at each decoding step, allowing adaptive alignment from image regions to caption words.

[^2]: https://github.com/tylin/coco-caption
Table 1: Ablative studies of attention time steps. We show the results of different attention models with different attention time steps, which are reported after the cross-entropy loss training stage and the self-critical loss training stage. We obtain the mean score and the standard deviation of a metric for a model by training it for 3 times, each with a different seed for random parameter initialization.

<table>
<thead>
<tr>
<th>Model</th>
<th>Attention Time Steps</th>
<th>Cross-Entropy Loss</th>
<th>Self-Critical Loss</th>
<th>METEOR</th>
<th>CIDEr-D</th>
<th>SPICE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min.</td>
<td>max.</td>
<td>avg.</td>
<td>METEOR</td>
<td>CIDEr-D</td>
<td>SPICE</td>
</tr>
<tr>
<td>Base</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>27.58 ± 0.03</td>
<td>113.37 ± 0.15</td>
<td>20.76 ± 0.03</td>
</tr>
<tr>
<td>Recurrent</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>27.66 ± 0.06</td>
<td>114.21 ± 0.35</td>
<td>20.84 ± 0.05</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>27.79 ± 0.01</td>
<td>114.72 ± 0.17</td>
<td>20.93 ± 0.08</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>27.75 ± 0.05</td>
<td>114.74 ± 0.23</td>
<td>20.93 ± 0.05</td>
</tr>
<tr>
<td>Adaptive</td>
<td>0</td>
<td>4</td>
<td>2.55 ± 0.01</td>
<td>27.82 ± 0.02</td>
<td>115.12 ± 0.38</td>
<td>20.94 ± 0.07</td>
</tr>
</tbody>
</table>

From Table 1, we observe that: 1) recurrent attention model (slightly) improves base attention model for all metrics, especially for those under self-critical loss training stage. 2) Focusing on the Cider-D score under self-critical loss, greatening the attention time steps for recurrent attention improves the performance. 3) adaptive attention model (AAT) further outperforms recurrent attention while requiring an average attention time steps of 2.2, which is quite small comparing to 4 and 8 attention time steps of the recurrent attention model. It shows that: firstly, incorporating more attention steps by adopting recurrent attention model helps to obtain better performance, however increasing the computation cost linearly with the number of the recurrent steps; secondly, adaptively aligning image feature vectors to caption words via Adaptive Attention Time requires less computation cost meanwhile leading to further better performance. This indicates Adaptive Attention Time to be a general solution to such sequence to sequence learning tasks as image captioning.

Table 2: Tradeoff for time cost penalty. We show the average attention time steps as well as the performance with different values of $\lambda$. The results are reported by a single model after self-critical training stage.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>avg. steps</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>ROUGE</th>
<th>CIDEr-D</th>
<th>METEOR</th>
<th>SPICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e-1</td>
<td>0.35</td>
<td>78.1</td>
<td>62.1</td>
<td>47.4</td>
<td>35.8</td>
<td>56.9</td>
<td>118.7</td>
<td>27.2</td>
<td>20.5</td>
</tr>
<tr>
<td>1e-3</td>
<td>1.03</td>
<td>80.0</td>
<td>64.3</td>
<td>49.7</td>
<td>37.8</td>
<td>58.0</td>
<td>125.7</td>
<td>28.2</td>
<td>21.6</td>
</tr>
<tr>
<td>1e-4</td>
<td>2.54</td>
<td>80.1</td>
<td>64.6</td>
<td>50.1</td>
<td>38.2</td>
<td>58.2</td>
<td>126.7</td>
<td>28.3</td>
<td>21.9</td>
</tr>
<tr>
<td>1e-5</td>
<td>3.77</td>
<td>80.0</td>
<td>64.6</td>
<td>50.0</td>
<td>38.2</td>
<td>58.2</td>
<td>126.5</td>
<td>28.3</td>
<td>21.8</td>
</tr>
<tr>
<td>0</td>
<td>4.00</td>
<td>80.0</td>
<td>64.5</td>
<td>50.1</td>
<td>38.2</td>
<td>58.2</td>
<td>126.7</td>
<td>28.3</td>
<td>21.8</td>
</tr>
</tbody>
</table>

From Table 1, we observe that: 1) recurrent attention model (slightly) improves base attention model for all metrics, especially for those under self-critical loss training stage. 2) Focusing on the Cider-D score under self-critical loss, greatening the attention time steps for recurrent attention improves the performance. 3) adaptive attention model (AAT) further outperforms recurrent attention while requiring an average attention time steps of 2.2, which is quite small comparing to 4 and 8 attention time steps of the recurrent attention model. It shows that: firstly, incorporating more attention steps by adopting recurrent attention model helps to obtain better performance, however increasing the computation cost linearly with the number of the recurrent steps; secondly, adaptively aligning image feature vectors to caption words via Adaptive Attention Time requires less computation cost meanwhile leading to further better performance. This indicates Adaptive Attention Time to be a general solution to such sequence to sequence learning tasks as image captioning.

Tradeoff for Time Cost Penalty. We show the effect of $\lambda$, the penalty factor for attention time cost, in Table 2. We assign different values to $\lambda$ and obtain the results of average attention time as well as performance for each value after the cross-entropy loss training stage. From Table 2, we observe that: 1) smaller value of $\lambda$ leads to more attention time and relatively higher performance; 2) the increment of performance gain will stop at some value of $\lambda$ but the attention time won’t as $\lambda$ decreases. We find 1e-4 to be a perfect value for $\lambda$ with the best performance while involving relatively small attention time steps.

Number of Attention Heads. There are two kinds of commonly used attention functions: additive attention [3] and dot-product attention. The former is popular among image captioning models while the latter has also shown its success by employing multiple attention heads [23]. We compare the two attention functions and explore the impact of using different numbers of attention heads. From Table 3, we find that: 1) when using one single head, additive attention outperforms dot-product attention with higher scores for all metrics and fewer attention steps. 2) as the number of attention increases, the number of attention steps reduces and the performance first goes up then down. 4 and 8 bring better performance than other choices.

<table>
<thead>
<tr>
<th>Model</th>
<th>Attention Time Steps</th>
<th>Cross-Entropy Loss</th>
<th>Self-Critical Loss</th>
<th>METEOR</th>
<th>CIDEr-D</th>
<th>SPICE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min.</td>
<td>max.</td>
<td>avg.</td>
<td>METEOR</td>
<td>CIDEr-D</td>
<td>SPICE</td>
</tr>
<tr>
<td>Base</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>27.58 ± 0.03</td>
<td>113.37 ± 0.15</td>
<td>20.76 ± 0.03</td>
</tr>
<tr>
<td>Recurrent</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>27.66 ± 0.06</td>
<td>114.21 ± 0.35</td>
<td>20.84 ± 0.05</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>27.79 ± 0.01</td>
<td>114.72 ± 0.17</td>
<td>20.93 ± 0.08</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>27.75 ± 0.05</td>
<td>114.74 ± 0.23</td>
<td>20.93 ± 0.05</td>
</tr>
<tr>
<td>Adaptive</td>
<td>0</td>
<td>4</td>
<td>2.55 ± 0.01</td>
<td>27.82 ± 0.02</td>
<td>115.12 ± 0.38</td>
<td>20.94 ± 0.07</td>
</tr>
</tbody>
</table>

4.3 Comparisons with State-of-the-arts

Comparing Methods. The compared methods include: LSTM [25], which encodes the image using a CNN and decode it using an LSTM; ADP-ATT [15], which develops a visual sentinel to decide how much to attend to images; SCST [19], which employs a modified visual attention
Table 3: Ablation study of attention type and number of attention heads. For all models, we set $\lambda = 1e^{-4}$. The results are reported by a single model after self-critical training stage.

<table>
<thead>
<tr>
<th>Type</th>
<th>Head(s)</th>
<th>avg. steps</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>ROUGE</th>
<th>CIDEr-D</th>
<th>METEOR</th>
<th>SPICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addictive</td>
<td>1</td>
<td>2.54</td>
<td>80.1</td>
<td>64.6</td>
<td>50.1</td>
<td>38.2</td>
<td>58.2</td>
<td>126.7</td>
<td>28.3</td>
<td>21.9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.78</td>
<td>80.0</td>
<td>64.6</td>
<td>50.1</td>
<td>38.3</td>
<td>58.2</td>
<td>126.8</td>
<td>28.2</td>
<td>21.9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.75</td>
<td>80.3</td>
<td>65.0</td>
<td>50.5</td>
<td>38.6</td>
<td>58.4</td>
<td>127.7</td>
<td>28.4</td>
<td>22.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2.66</td>
<td>80.1</td>
<td>64.9</td>
<td>50.5</td>
<td>38.7</td>
<td>58.5</td>
<td>128.6</td>
<td>28.6</td>
<td>22.2</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2.64</td>
<td>80.1</td>
<td>64.7</td>
<td>50.2</td>
<td>38.4</td>
<td>58.3</td>
<td>128.0</td>
<td>28.4</td>
<td>22.1</td>
</tr>
</tbody>
</table>

Table 4: Single model performance of other state-of-the-art methods as well as ours on the MS-COCO “Karpathy” test split. For our AAT model, we use multi-head attention [23] with the number of attention heads to be 8 and $\lambda = 1e^{-4}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cross-Entropy Loss</th>
<th>Self-Critical Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLEU-4</td>
<td>ROUGE</td>
</tr>
<tr>
<td>LSTM [25]</td>
<td>29.6</td>
<td>25.2</td>
</tr>
<tr>
<td>ADP-ATT [15]</td>
<td>33.2</td>
<td>26.6</td>
</tr>
<tr>
<td>SCST [19]</td>
<td>30.0</td>
<td>25.9</td>
</tr>
<tr>
<td>Up-Down [2]</td>
<td>36.2</td>
<td>27.0</td>
</tr>
<tr>
<td>RFNet [10]</td>
<td>35.8</td>
<td>27.4</td>
</tr>
<tr>
<td>GCN-LSTM [31]</td>
<td>36.8</td>
<td>27.9</td>
</tr>
<tr>
<td>SGAE [28]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AAT (Ours)</td>
<td>37.0</td>
<td>28.1</td>
</tr>
</tbody>
</table>

and is the first to use Self Critical Sequence Training (SCST) to directly optimize the evaluation metrics; Up-Down [2], which employs a two-layer LSTM model with bottom-up features extracted from Faster-RCNN; RFNet [10], which fused encoded results from multiple CNN networks; GCN-LSTM [31], which predicts visual relationships between any two entities in the image and encode the relationship information into feature vectors; and SGAE [28], which introduces language inductive bias into its model and applies auto-encoding scene graphs.

**Analysis.** We show the performance of one single model of these methods under both cross-entropy loss training stage and self-critical loss training stage. It can be seen that: for the cross-entropy stage, AAT achieves the highest score among all compared methods for all metrics (BLEU-4, METEOR, ROUGE-L, CIDEr-D, and SPICE); and for the self-critical stage, AAT reports the highest score for most metrics, with the METEOR score slightly lower than the highest (58.5 vs. 58.6).

Comparing to Up-Down [2], which is the previous state-of-the-art model and is the most similar to our AAT model in terms of the model framework, our single AAT model improves the scores on BLEU-4, METEOR, ROUGE-L, CIDEr-D, and SPICE by 2.2%, 4.1%, 1.6%, 3.3%, and 4.4% respectively for cross-entropy loss training stage by 6.6%, 3.2%, 2.8%, 7.1%, and 3.1% respectively for self-critical loss training stage, which indicates a significant margin and shows that AAT is superior to the vanilla attention model.

### 4.4 Qualitative Analysis

To gain a qualitative understanding of our proposed method, we use two examples and visualize the caption generation process of a model that employs AAT and a 4-head attention in Figure 2. For each example, we show the attention steps taken at each decoding step, with the visualized attention regions for all the 4 attention heads, the confidence for the output at the current attention step, and the corresponding weight. We also show the confidence/weight for the non-visual step (colored in orange and before the attention steps), which corresponds to $h^t_1$ of the input module in the decoder and contains the previous decoding information and is used to avoid attention steps.

We observe that: 1) the attention regions of each attention heads are different from others, which indicates that every attention head has its own concern; 2) the confidence increases with the attention steps, which is like human attention: more observing leads to better comprehension and higher confidence; 3) the number of attention steps required at different decoding steps is different, more
steps are taken at the beginning of the caption or a phase, such as “on the side” and “at a ball”, which indicates that adaptive alignment is effective and has been realized by AAT.

Figure 2: Qualitative examples for the caption generation process of AAT. We show the attention steps taken at each decoding step, with the visualized attention regions, the confidence and the weights of each attention step (confidence/weight is shown below the attention regions for each step).

5 Conclusion

In this paper, we propose a novel attention model, namely Adaptive Attention Time (AAT), which can adaptively align image regions to caption words for image captioning. AAT allows the framework to learn how many attention steps to take to output a caption word at each decoding step. AAT is also generic and can be employed by any sequence-to-sequence learning task. On the task of image captioning, we empirically show that AAT improves over state-of-the-art methods. In the future, it will be interesting to apply our model to more tasks in computer vision such as video captioning and those in natural language processing such as machine translation and text summarization, as well as any model that can be modeled under the encoder-decoder framework with an attention mechanism.
Acknowledgment

This project was supported by National Engineering Laboratory for Video Technology - Shenzhen Division, National Natural Science Foundation of China (NSFC, 61872256, 61972217), and The Science and Technology Development Fund of Macau (FDCT, 0016/2019/A1). We would also like to thank the anonymous reviewers for their insightful comments.

References


