Adaptively Aligned Image Captioning via Adaptive Attention Time

Lun Huang[1] **Wenmin Wang**[1,3*] **Yaxian Xia**[1] **Jie Chen**[1,2]
[1]School of Electronic and Computer Engineering, Peking University
[2]Peng Cheng Laboratory
[3]Macau 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 employs an encoder-decoder framework with 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), which can adaptively align source to target 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, image regions and caption words can be aligned adaptively in the decoding process: an image region can be mapped to arbitrary number of caption words while a caption word can also attend to arbitrary number of image regions. AAT is deterministic and differentiable, and doesn't introduce any noise to the parameter gradients. AAT is also generic and can be employed by any sequence-to-sequence learning task. In this paper, we empirically show that AAT improves over state-of-the-art methods on the task of image captioning.

1 Introduction

Image captioning aims to automatically describe the content of an image using natural language [13, 29, 16, 5]. It is 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 attention mechanism have achieved great success. In such a framework, an image encoder which is based on convolutional neural network (CNN) is first used to extract region-level visual feature vectors for an given image, next a caption decoder which is based on recurrent neural network (RNN) is adopted to generate caption words recurrently, taking one attended feature vector for each one decoding step.

Despite the success that the existing approaches have achieved, there are still limitations in the framework. The attention model in the encoder-decoder framework generates one single image feature vector, which is next conditioned by the decoder to make predictions about caption words. Obviously, one-to-one mapping from image regions to caption words is assumed by the vanilla attention model, which, however, is never possible and may involve the following issues: **1)** unnecessary or even misleading visual information are forcibly given to caption words that require little visual clues[15]; **2)** words at early decoding steps have little access to image information while the accuracy of the predictions of them is more important for generating a caption of high quality than those at late

*Corresponding author

Preprint. Under review.

decoding steps; **3)** one single attended weighted averaged feature vector gives little knowledge that helps the decoder to understand complex interactions between objects in an image since there is only one operation between feature vectors in the attended vector: addition, which is definitely not enough.

To this end, we develope a novel attention model, named as *Adaptive Attention Time* (AAT), to realize the adaptive alignment from image regions to caption words. At each decoding step, depending on its confidence, AAT decides whether to take extra attention steps or directly output a caption word and move on to the next decoding step. If AAT decides so, then at the subsequent attention step, AAT takes another attended feature vector into the decoder; then again, AAT makes decision upon whether to take further attention step. The attention process continues, until the decoder itself is confident enough to output a word. With the techniques from *Adaptive Computation Time* (ACT) [7], we are able to make AAT deterministic and differentiable. Further more, to introduce less attention steps for a decoding step and more complex operations between objects in an image, we take advantage of the multi-head attention[23] rather than use the conventional attention mechanism which has only one single head.

We evaluate the impact of AAT against 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 that AAT is superior than the those models 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 128.2 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) is first 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 dynamic number of layers or operations [26, 4]. In this paper, we implement our *Adaptive Attention Time* (AAT) model with the techniques of ACT.

3 Method

The target of image captioning is to generate a natural sentence $\boldsymbol{S}$ to describe a given image $\boldsymbol{I}$. Our captioning model with *Adaptive Attention Time* (AAT) is developed upon the general attention based encoder-decoder framework, as is shown in Figure 1. We first describe the framework for image captioning in Section 3.1, then show how we implement AAT to realize adaptive alignment as well as other attention models 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 $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k\}$ of different image regions for the given image I, where k is the number of image regions. A CNN based image encoder (*e.g.* ResNet[8]) is a typical choice for image captioning. Recently, R-CNN based models (*e.g.* Faster RCNN[18]) are utilized to implement

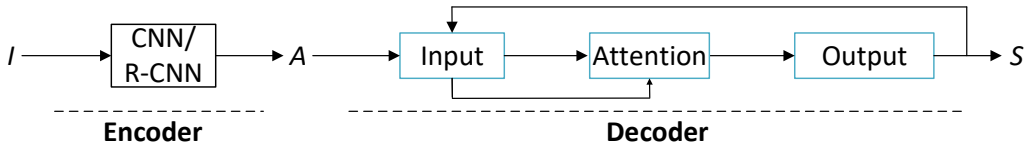
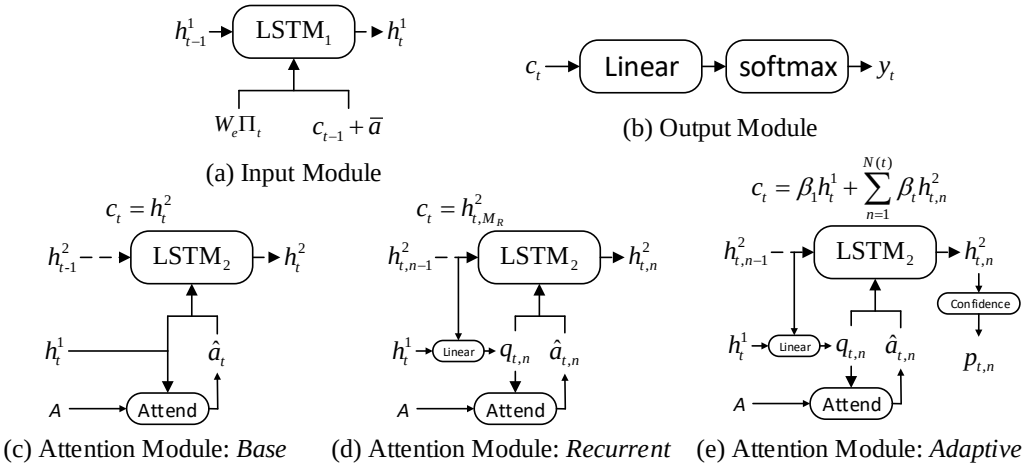


Figure 1: The encoder-decoder framework for image captioning. First the CNN/R-CNN based encoder extract a set of feature vectors A from the given image I, and then the decoder recurrently decodes the feature vector to a sentence S.



(a) Input Module (b) Output Module

(c) Attention Module: *Base* (d) Attention Module: *Recurrent* (e) Attention Module: *Adaptive*

Figure 2: Modules of the decoder. **(a)** is the *input* module; **(b)** is the *output* module; and **(c), (d), (e)** are three different attention models for the *attention* module.

bottom-up attention [2] and provides a better understanding of the objects in the image, boosting the performance of image captioning.

For a CNN based encoder, taking ResNet-101 for example, we can use it as the encoder by applying spatially adaptive max-pooling so that the output has a fixed size of 14x14x2048, and we will have a set of 196 feature vectors for an input image.

As for an R-CNN based encoder, taking Faster-RCNN as example, which detects objects in two stages. The first stage predicts region proposals, and the second stages predicts class labels and bounding box refinements, and non-maximum suppression is applied to remove duplicate regions. To obtain a feature vector set, following the practice in [2], we select all regions which any class detection probability exceeds a confidence threshold, and apply mean-pooling on the convolutional features for each of them, and the number of feature vectors are in the range from 10 to 100.

We project all the feature vector $\boldsymbol{a}_i$ in A to a size of d via a linear transformation.

Caption Decoder. We design the decoder with three modules: an *input* module, an *attention* module and an *output* module, as shown in Figure 2.

The *input* module, which models the input word as well as the context information of the decoder, consists of an LSTM layer (LSTM_1). The process of this layer is follows:

$$(\boldsymbol{h}_t^1, \boldsymbol{m}_t^1) = \text{LSTM}_1([W_e \Pi_t, \bar{\boldsymbol{a}} + \boldsymbol{c}_{t-1}], (\boldsymbol{h}_{t-1}^1, \boldsymbol{m}_{t-1}^1)) \tag{1}$$

where $\boldsymbol{h}_t^1, \boldsymbol{m}_t^1$ are the hidden state and memory cell of LSTM_1 with a hidden size of d, which is same to the feature vector size, $W_e \in \mathbb{R}^{E \times |\Sigma|}$ is a word embedding matrix for a vocabulary Σ and E is the embedding size, and Π_t is one-hot encoding of the input word at time step t, $\boldsymbol{c}_{t-1}$ is the previous context vector of the decoder, which is also the output of the *attention* module and input of the *output* module, and $\bar{\boldsymbol{a}} = \frac{1}{k} \sum_i \boldsymbol{a}_i$ is the mean-pooling of A and added to $\boldsymbol{c}_{t-1}$ to provide global information to the *input* module.

The *attention* module applies the proposed attention model on the image feature set A and generate a context vector $\boldsymbol{c}_t$, 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 $\boldsymbol{c}_t$ through a linear layer with softmax activation to predict the probability distribution of the vocabulary:

$$p(y_t \mid y_{1:t-1}) = \text{softmax}(W_p \boldsymbol{c}_t + \boldsymbol{b}_p) \tag{2}$$

where $W_p \in \mathbb{R}^{\Sigma \times d}, b_p \in \mathbb{R}^{\Sigma}$ are transformation weights and bias. The distribution over complete output sequences is calculated as the product of conditional distributions:

$$p\left(y_{1:T}\right) = \prod_{t=1}^{T} p\left(y_t | y_{1:t-1}\right) \tag{3}$$

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" 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 α_i of every candidate feature vector $\boldsymbol{a}_i$ with the given query (i.e. h_t^1), resulting in one single weighted averaged vector $\hat{\boldsymbol{a}}_t = \sum_i^K \alpha_i \boldsymbol{a}_i$ over the whole feature vector set A.

There are two categories of methods for calculating α_i: *addictive* and *productive*. The former is usually adopted in previous image captioning models [15, 19, 2]. In this paper, we adopt the *productive* version instead, specifically, the *scaled dot-product attention*:

$$\boldsymbol{f}_a(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})V \tag{4}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n_q \times d}, \boldsymbol{K} \in \mathbb{R}^{n_k \times d}, \boldsymbol{V} \in \mathbb{R}^{n_v \times d}$ are the query, key and value set respectively. Further, the *multi-head attention* (MHA) based on it is formulated as:

$$\boldsymbol{f}_{mha}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(head_i, \ldots, head_h)W^o \tag{5}$$

where $head_i = \boldsymbol{f}_a(QW_i^Q, KW_i^K, VW_i^V)$, $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d_h}$ are the projection matrices of the i-th head, $W^o \in \mathbb{R}^{d \times d}$ is the output projection matrix. h is the number of attention heads, d_h is the hidden dimension for each head and $d = d_h \times h$.

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

$$\hat{\boldsymbol{a}}_t = \boldsymbol{f}_{mha}(\boldsymbol{h}_t^1, A, A) \tag{6}$$

where we omit the dimension expanding of h_t^1 and squeezing of $\hat{\boldsymbol{a}}_t$ for simplicity. Next $\hat{\boldsymbol{a}}_t$ together with $\boldsymbol{h}_t^1$ is fed into another LSTM layer (LSTM_2):

$$(\boldsymbol{h}_t^2, \boldsymbol{m}_t^2) = \text{LSTM}_2([\hat{\boldsymbol{a}}_t, \boldsymbol{h}_t^1], (\boldsymbol{h}_{t-1}^2, \boldsymbol{m}_{t-1}^2)) \tag{7}$$

We let the context vector to be the output hidden state $\boldsymbol{c}_t = \boldsymbol{h}_t^2$.

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 $\boldsymbol{q}_{t,n}$ at attention time step n of decoding time step t by transforming $\boldsymbol{h}_t^1$ and $\boldsymbol{h}_{t,n-1}^2$ through a linear layer:

$$\boldsymbol{q}_{t,n} = W_q[\boldsymbol{h}_t^1, \boldsymbol{h}_{t,n-1}^2] + \boldsymbol{b}_q \tag{8}$$

where $\boldsymbol{h}^2_{t,0} = \boldsymbol{h}^2_{t-1}$, $W_q \in \mathbb{R}^{d\times 2d}, b_q \in \mathbb{R}^d$ is the transformation weights and bias, $\boldsymbol{h}^2_{t,n-1}$ is the output of LSTM_2 at attention time step $n-1$ of decoding time step t.

Then $\boldsymbol{q}_{t,n}$ is fed to the attention module and obtain $\hat{\boldsymbol{a}_t} = \boldsymbol{f}_{mha}(\boldsymbol{q}_{t,n}, A, A)$, which is further fed to LSTM_2:

$$\boldsymbol{h}^2_{t,n}, \boldsymbol{m}^2_{t,n} = \text{LSTM}_2([\hat{\boldsymbol{a}_{t,n}}, \boldsymbol{h}^1_t], \boldsymbol{h}^2_{t-1,j}, \boldsymbol{m}^2_{t-1,j}) \tag{9}$$

We set $\boldsymbol{h}^2_t = \boldsymbol{h}^2_{t,M_r}, \boldsymbol{m}^2_t = \boldsymbol{m}^2_{t,M_r}$, where $\boldsymbol{h}^2_{t,M_r}, \boldsymbol{m}^2_{t,M_r}$ 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 $\boldsymbol{c}_t = \boldsymbol{h}^2_{t,M_r} = \boldsymbol{h}^2_t$.

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 LSTM_2, since it's used to predict probability distribution. We design the confidence network as a two-layer feed forward translation:

$$p_{t,n} = \sigma\left(W_2 \max\left(0, W_1 \boldsymbol{x}_{t,n} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) \tag{10}$$

where $\boldsymbol{x}_{t,0} = \boldsymbol{h}^1_t, \boldsymbol{x}_{t,n} = \boldsymbol{h}^2_{t,n} (n > 0)$, the total attention steps is determined by:

$$N(t) = \min\{n' : \prod_{n=0}^{n'} (1 - p_{t,n}) < \epsilon\} \tag{11}$$

where ϵ is a threshold which is a small value and slightly greater than 0.

The final hidden state and memory cell of LSTM_2 are computed as:

$$\boldsymbol{h}^2_t = \beta_{t,0}\boldsymbol{h}^1_t + \sum_{n=1}^{N(t)} \beta_{t,n}\boldsymbol{h}^2_{t,n} \tag{12}$$

$$\boldsymbol{m}^2_t = \beta_{t,0}\boldsymbol{m}^2_{t-1} + \sum_{n=1}^{N(t)} \beta_{t,n}\boldsymbol{m}^2_{t,n} \tag{13}$$

where $\beta_{t,n} = p_{t,n}\prod_{n'=0}^{n-1}(1 - p_{t,n'})$ (n>0) and $\beta_{t,0} = p_{t,0}$. $\boldsymbol{h}^1_t$ is added to show that we can obtain context directly from the *input* module without attending to image feature vectors. and $\boldsymbol{c}^2_{t-1}$ 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: $\boldsymbol{h}^2_{t,n} = \text{LayerNorm}_h(\boldsymbol{h}^2_{t,n})$, and $\boldsymbol{m}^2_{t,n} = \text{LayerNorm}_m(\boldsymbol{m}^2_{t,n})$

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

Time cost penalty. A loss is added to penalize the time cost for attention steps:

$$L^a_t = N(t) + \sum_{n=0}^{N(t)} n(1 - p_{t,n}) \tag{14}$$

where $\sum_{n=0}^{N(t)} n(1 - p_{t,n})$ encourages $p_{t,n}$ at early steps to be larger so that total attention steps can be reduced, and $N(t)$ indicates the attention time steps.

Minimum and maximum attention steps. We can set a minimum attention step M_{min} to make sure the attention module takes at least M_{min} attention steps, we simply set $p_{t,n} = 0$, for $0 <= n <= M_{min} - 1$.

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

$$N(t) = \min\{M_{max}, \min\{n' : \prod_{n=0}^{n'} (1 - p_{t,n}) < \epsilon\}\} \tag{15}$$

As can be seen, the process of "Adaptive Attention Time" 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 $\boldsymbol{c}_t = \beta_{t,0}\boldsymbol{h}_t^1 + \sum_{n=1}^{N(t)} \beta_{t,n}\boldsymbol{h}_{t,n}^2 = \boldsymbol{h}_t^2$.

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 $M_{max} = M_{min} = M_r$.

3.2.5 Training Objectives

Training with Cross Entropy Loss. The typical way of training a captioning model is to optimize cross entropy loss L_{XE}, and we add the attention time loss for *Adaptive Attention Time*. Given the sequence $y_{1:T}^*$ of a target ground truth and the parameters θ of the captioning model, the loss can be expressed as:

$$L_{XE}(\theta) = -\sum_{t=1}^{T} \log(p_\theta(y_t^* \mid y_{1:t-1}^*)) + \lambda_{xe} \sum_{t=1}^{T} L_t^a \tag{16}$$

where the second item is the attention time loss, and λ_{xe} is the factor for it.

Self-Critical Sequence Training. Following the approach described as Self-Critical Sequence Training [19] (SCST), we can directly optimize the NLP metrics which are used at test time. The loss can be approximated as the negtive expected score:

$$L_{RL}(\theta) = -\mathbf{E}_{y_{1:T} \sim p_\theta}[r(\boldsymbol{y}_{1:T})] + \lambda_{rl} \sum_{t=1}^{T} L_t^a \tag{17}$$

The gradient of the first item can be approximated as:

$$\nabla_\theta L_{RL}^1(\theta) \approx -(r(\boldsymbol{y}^s{}_{1:T}) - r(\hat{\boldsymbol{y}}_{1:T}))\nabla_\theta \log p_\theta(y_{1:T}^s) \tag{18}$$

where $\boldsymbol{y}^s{}_{1:T}$ is a sampled caption and $r(\hat{\boldsymbol{y}}_{1:T})$ defines the baseline score obtained by greedily decoding the current model.

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 training images and 40,504 validation images. MS COCO provides 40775 images as test set for online evaluation as well. The "Karpathy" data split [11] is used 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 projects 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. We use multi-head attention with the number of attention heads of 4. We set the epsilon $\epsilon = 0.0001$, the factor of time cost penalty $\lambda_{xe} = \lambda_{rl} = 0.0001$, and the minimum and maximum attending steps are set to $M_{min} = 0, M_{max} = 4$.

As for training process, we train our model under XE loss for 25 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 [3]. Then we optimize the CIDEr-D score with REINFORCE for another 15 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 [6], CIDEr-D [24] and SPICE [1], to evaluate the proposed method and compare with other methods. All the metrics are computed with the publicly released code[2].

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 cross entropy loss training stage and self critical sequence training stage. Results are obtained by beam search with a beam size of 2. All values are reported as percentage (%). (In the tables that follows, B@n is short for BLEU-n, M is short for METEOR, R for ROUGE-L, C for CIDEr and S is short for SPICE.)

Attention Model	**Attention Time Steps**			**Cross-Entropy Loss**			**Self-Critical Loss**		
	min.	max.	avg.	M	C	S	M	C	S
Base	1	1	1	28.0	116.6	21.1	28.3	123.5	21.9
Recurrent	2	2	2	28.0	116.8	21.1	28.4	123.8	21.9
	4	4	4	28.1	117.3	21.1	28.5	124.9	22.1
	8	8	8	28.0	117.1	21.2	**28.6**	125.2	22.3
Adaptive	0	4	2.2	**28.2**	**117.7**	**21.4**	28.5	**128.2**	**22.4**

Table 2: Tradeoff for time cost penalty. We show the average attention time steps as well as the performance with different values of λ_{xe}. The results are reported after cross entropy training stage.

λ_{xe}	avg. steps	B@1	B@2	B@3	B@4	R	C	M	S
1e-3	1.8	76.9	60.9	47.1	36.3	56.9	114.1	27.8	20.7
1e-4	2.2	**77.7**	**61.8**	**47.8**	**37.2**	**57.3**	**117.7**	**28.2**	**21.4**
0	4.0	77.3	61.5	47.8	36.9	**57.3**	117.0	28.1	21.3

4.2 Ablative Studies

Attention Time Steps (Attention Model) To show the effictiveness of adaptive alignment for image captioning, we compare the results of different attention models with different attention time steps. *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 image regions recurrently; *Adaptive:* which is the proposed method in this papers, and at each decoding step attends to an adaptive length of image regions, which can adaptively aligns image to caption.

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 with Adaptive Attention Time 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 λ, the penalty factor for attention time cost. We assign different values to λ_{xe} and obtain the results of average attention time as well as performance for each value after cross-entropy loss training stage. From Table 2, we observe that: **1).** smaller value of λ_{xe} leads to more attention time and relatively higher performance; **2).** the increment of performance gain will stop at some value of λ_{xe} but the attention time won't as λ_{xe} decreases. We

[2]https://github.com/tylin/coco-caption

Table 3: Single model performance of other state-of-the-art methods as well as ours on the MS-COCO 'Karpathy' test split.

Method	Cross-Entropy Loss					Self-Critical Loss				
	B@4	M	R	C	S	B@4	M	R	C	S
LSTM [25]	29.6	25.2	52.6	94.0	-	31.9	25.5	54.3	106.3	-
ADP-ATT [15]	33.2	26.6	-	108.5	-	-	-	-	-	-
SCST [19]	30.0	25.9	53.4	99.4	-	34.2	26.7	55.7	114.0	-
Up-Down [2]	36.2	27.0	56.4	113.5	20.3	36.3	27.7	56.9	120.1	21.4
RFNet [10]	35.8	27.4	56.8	112.5	20.5	36.5	27.7	57.3	121.9	21.2
GCN-LSTM [31]	36.8	27.9	57.0	116.3	20.9	38.2	**28.5**	58.3	127.6	22.0
SGAE [28]	-	-	-	-	-	**38.4**	28.4	**58.6**	127.8	22.1
AAT (Ours)	**37.2**	**28.2**	**57.3**	**117.7**	**21.4**	**38.4**	**28.5**	**58.6**	**128.2**	**22.4**

finds that 1e-4 to be a perfect value for λ_{xe} with high performance while relatively small attention time steps. Thus we set $\lambda_{xe} = \lambda_{rl} = 1e-4$ for our model.

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 developes a visual sentinel to decide how much to attend to images; SCST [19], which employs a modified visual attention 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 single model of these methods under both cross entropy loss training stage and self critical loss training stage. It can be seen that our single model can achieve the highest score among all compared methods in terms of all metrics under both stages.

Comparing to Up-Down, 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 2.8%, 4.4%, 1.6%, 3.7%, 5.4% respectively for cross-entropy loss training stage and by 5.8%, 2.9%, 3.0%, 6.7%, 4.7% respectively for cross-entropy loss training stage, which indicates a significant margin and shows that *Adaptive Attention Time* is superior to the vanilla attention model.

4.4 Qualitative Analysis

To gain a qualitative understanding of our proposed attention model, *Adaptive Attention Time* (AAT), for image captioning, we visualize the caption generation process of AAT in Figure 3 of two examples. 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 is before the attention steps), which corresponds to $\boldsymbol{h}_t^1$ 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 the 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 in 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.

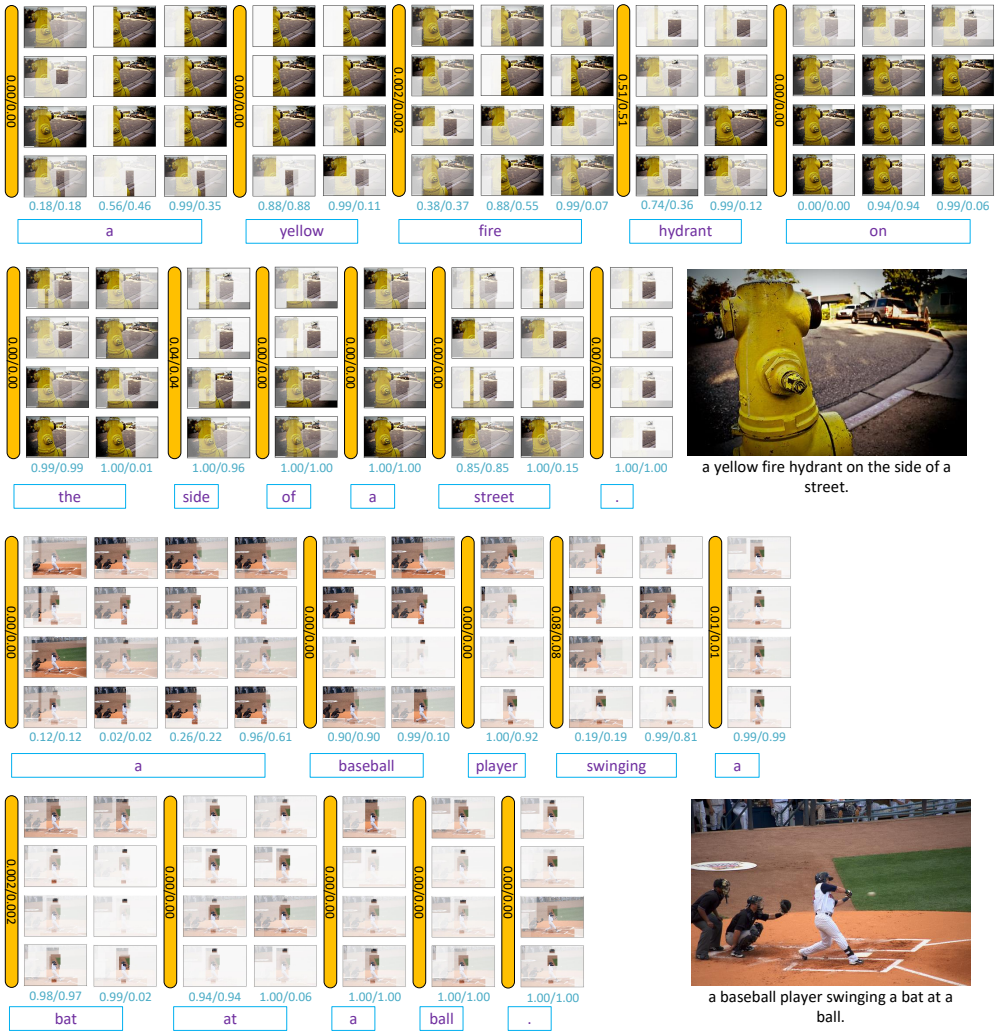


Figure 3: Qualitative examples for the caption generation process of AAT. We show the attention steps taken at each decoding step, with the visualized attention regions, the confidence and the weights of each attention step (*confidence/weight* is shown below the attention regions for each step).

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 attention mechanism.

Acknowledgment

This project was supported by Shenzhen Key Laboratory for Intelligent Multimedia and Virtual Reality (ZDSYS201703031405467), National Natural Science Foundation of China (NSFC, No.U1613209, 61872256, 61972217), and National Engineering Laboratory for Video Technology - Shenzhen Division. We would also like to thank the anonymous reviewers for their insightful comments.

References

[1] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *ECCV*, 2016.

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018.

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, 2015.

[4] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *ICML*, 2019.

[5] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *CVPR*, 2015.

[6] Carlos Flick. Rouge: A package for automatic evaluation of summaries. In *The Workshop on Text Summarization Branches Out*, 2004.

[7] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv: Neural and Evolutionary Computing*, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[9] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. In *ICCV*, 2019.

[10] Wenhao Jiang, Lin Ma, Yu-Gang Jiang, Wei Liu, and Tong Zhang. Recurrent fusion network for image captioning. In *ECCV*, 2018.

[11] Andrej Karpathy and Fei Fei Li. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[13] Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2891–2903, 2013.

[14] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[15] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, 2017.

[16] Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé III. Midge: Generating image descriptions from computer vision detections. In *ECACL*, 2012.

[17] Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[18] S. Ren, K. He, R Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1137–1149, 2015.

[19] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *CVPR*, 2017.

[20] Banerjee Satanjeev. Meteor : An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL*, 2005.

[21] Richard Socher and Li Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *CVPR*, 2010.

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, pages 3104–3112, 2014.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[24] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.

[25] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164, 2015.

[26] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018.

[27] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[28] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *CVPR*, June 2019.

[29] Yezhou Yang, Ching Lik Teo, Hal Daumé III, and Yiannis Aloimonos. Corpus-guided sentence generation of natural images. In *EMNLP*, 2011.

[30] Benjamin Z Yao, Xiong Yang, Liang Lin, Mun Wai Lee, and Song-Chun Zhu. I2t: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485–1508, 2010.

[31] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *ECCV*, 2018.

[32] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting image captioning with attributes. In *ICCV*, 2017.