Show, Attend and Tell: Neural Image Caption Generation with visual Attention

2019.6.20

Paper

- Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015. (ICML 2015)

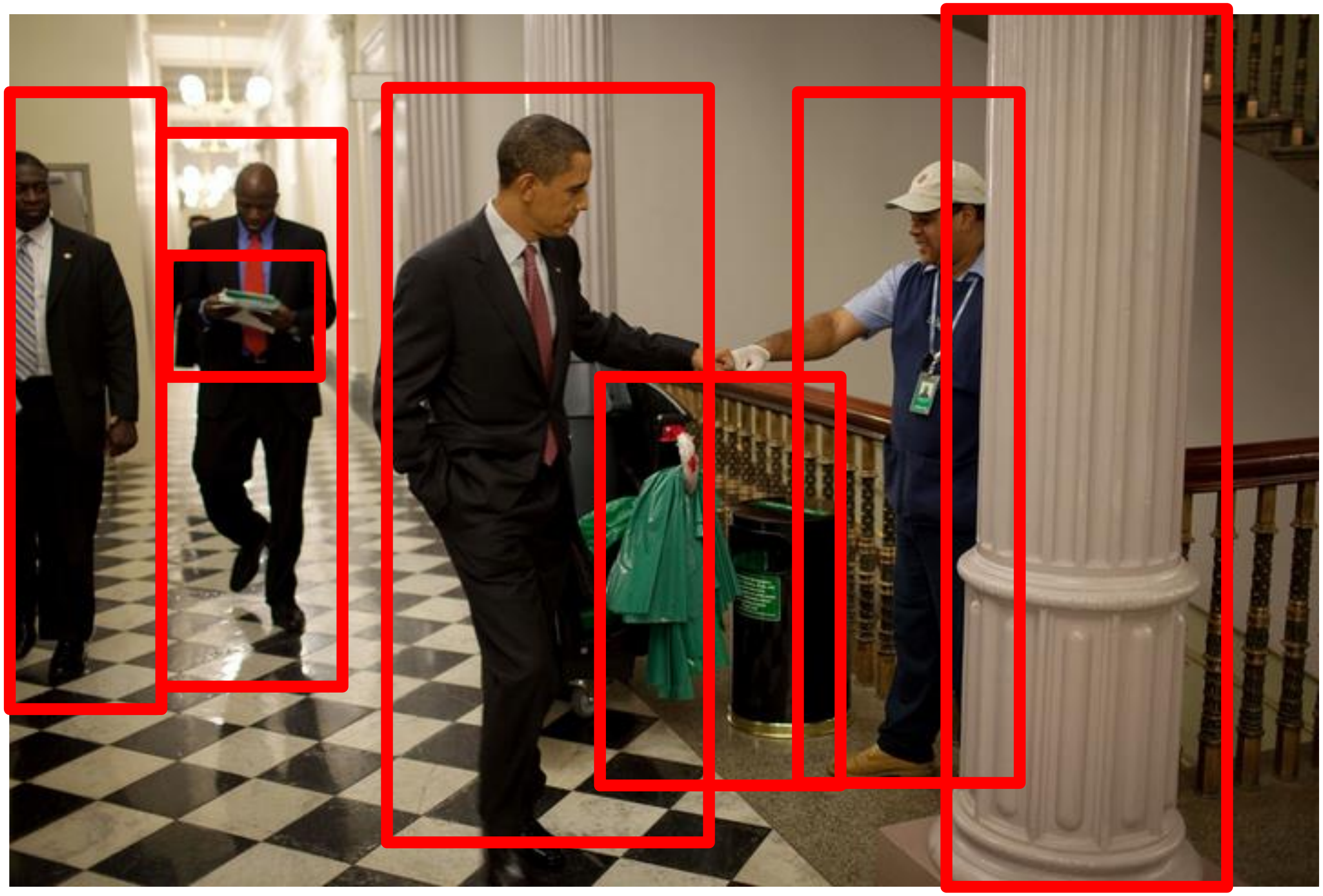
Image captioning

How do we create a caption for an image?

He is greeting the housekeeper.

Image captioning

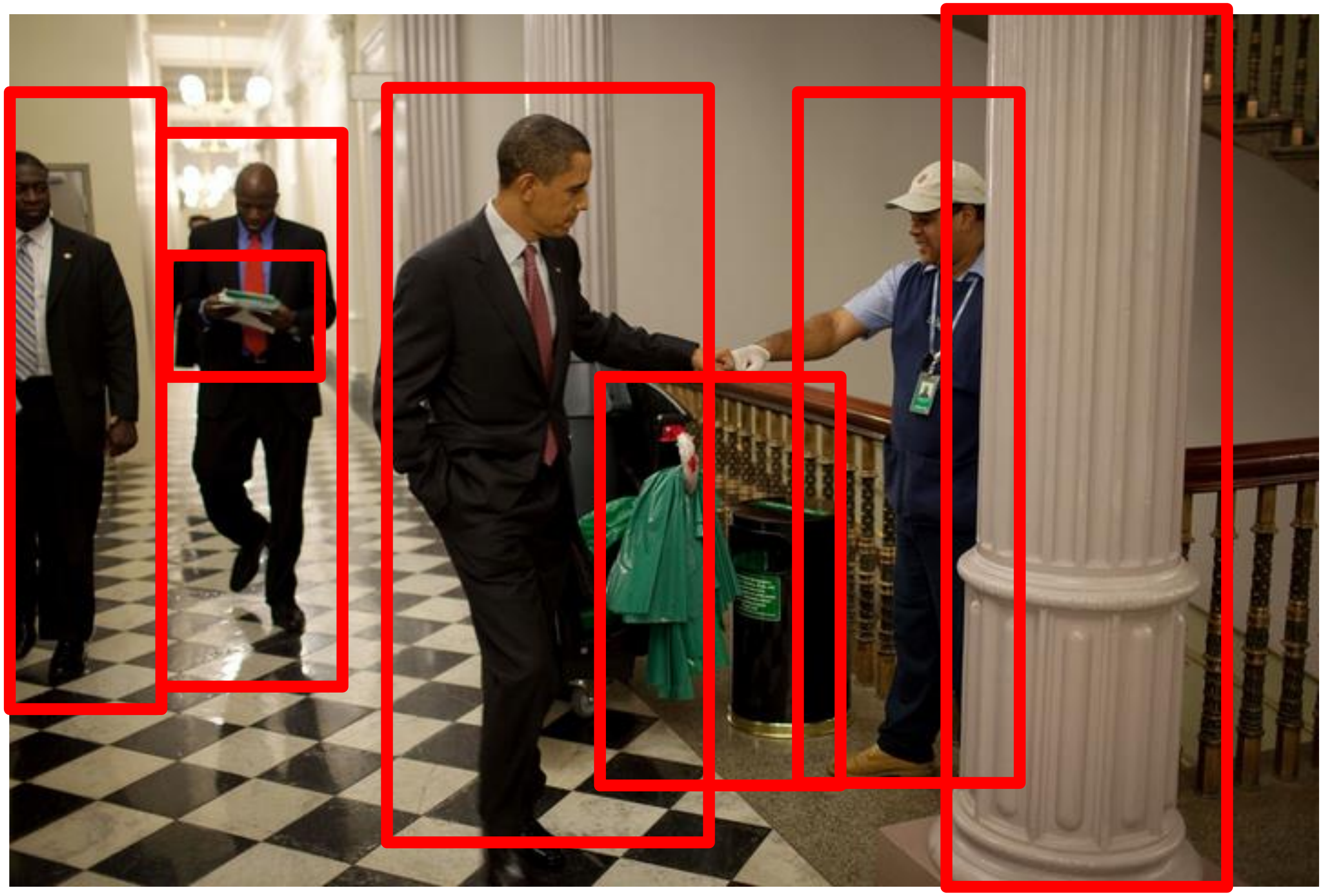
How does a machine create a caption for an image?



Object detection+segmentation

Image captioning

How does a machine create a caption for an image?



Event estimation

He is greeting the housekeeper.
A man is reading a book.

Caption generation

Image captioning

State-of-art based on neural net

- machine translation

Image(language)

Caption(language)

Encoder
(CNN)

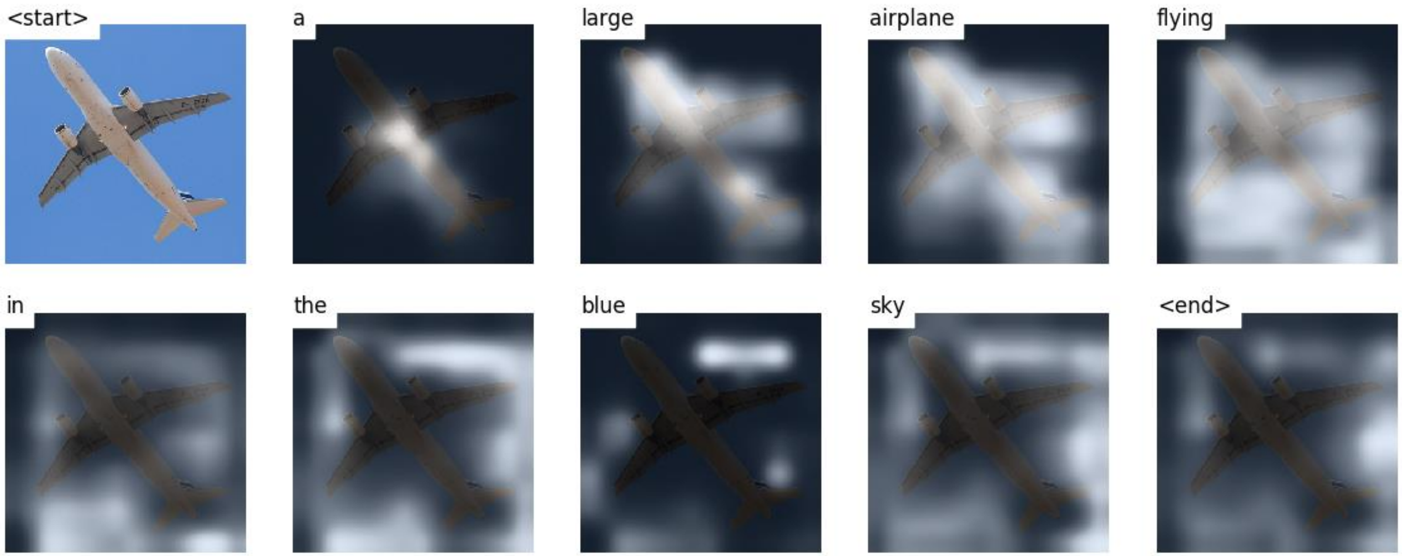
feature vector

Decoder
(RNN,LSTM)

Image captioning

Proposed model

- visual attention



Encoder
(CNN)

feature vector

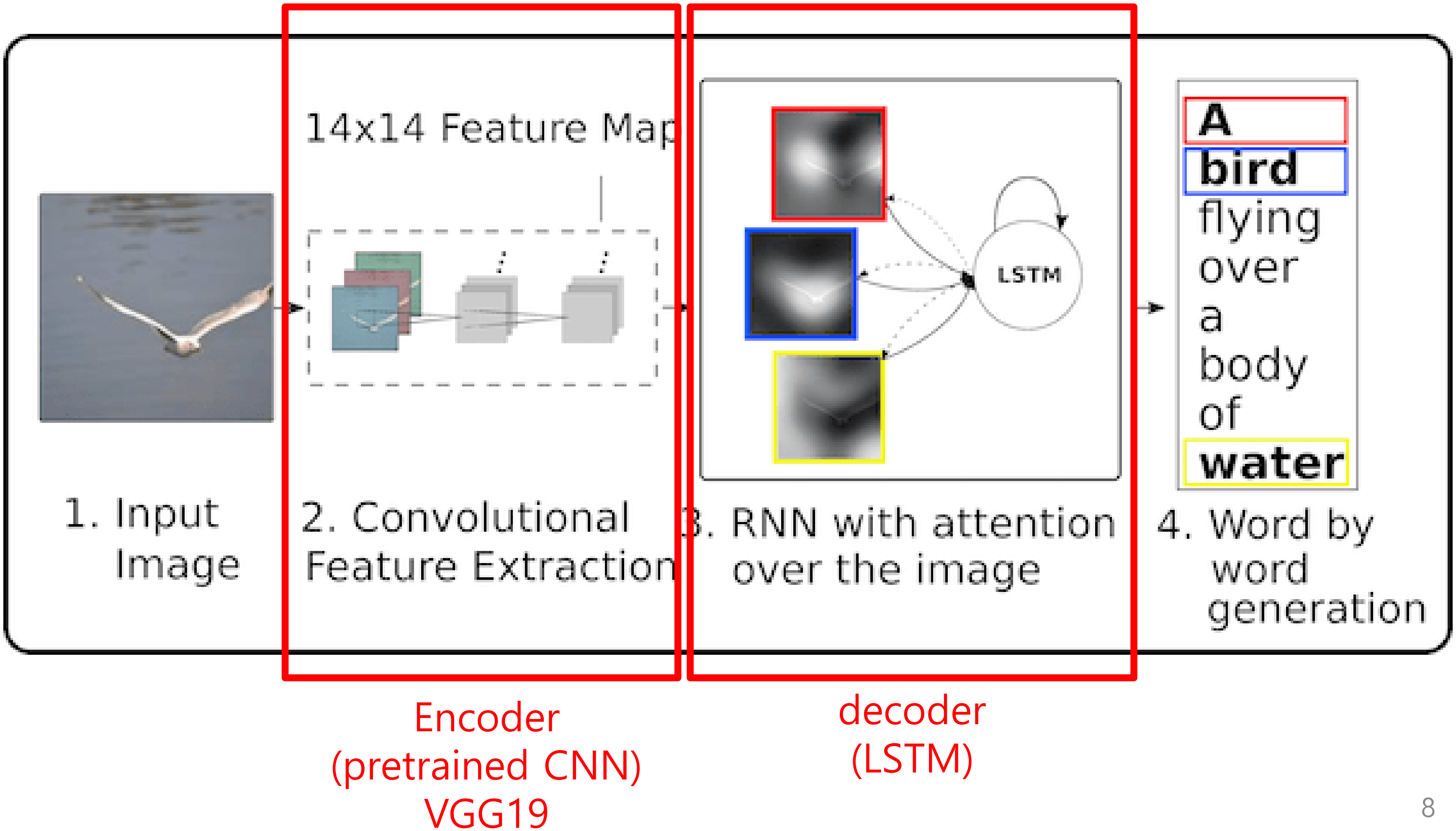
Decoder
(RNN,LSTM)

Attention mechanism

Model

Proposed model

- visual attention



Model

Encoder

• caption y

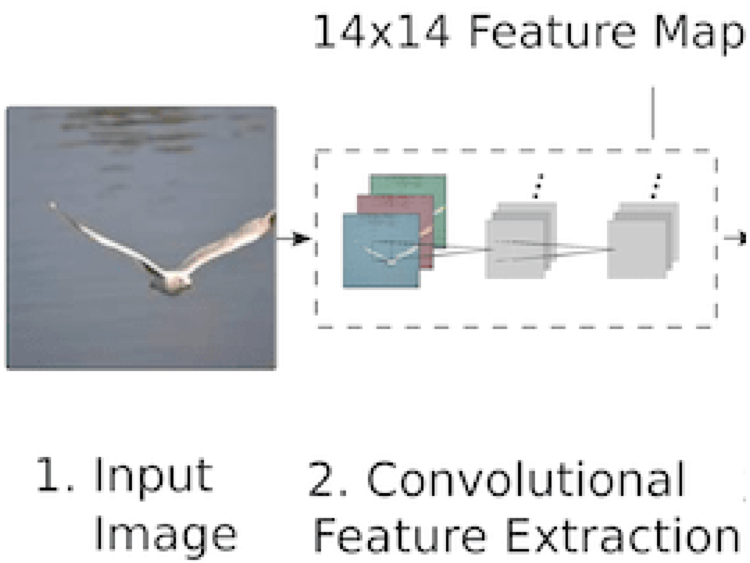
$$y = \{y_1, \dots, y_C\}, y_i \in R^k$$

C: Length of caption
K: size of the vocabulary

•the extractor produces L vectors, each of which is a D-dimensional representation corresponding to a part of the image.

$$a = \{a_1, \dots, a_L\}, a_i \in R^D$$

L: the number of last layer filter

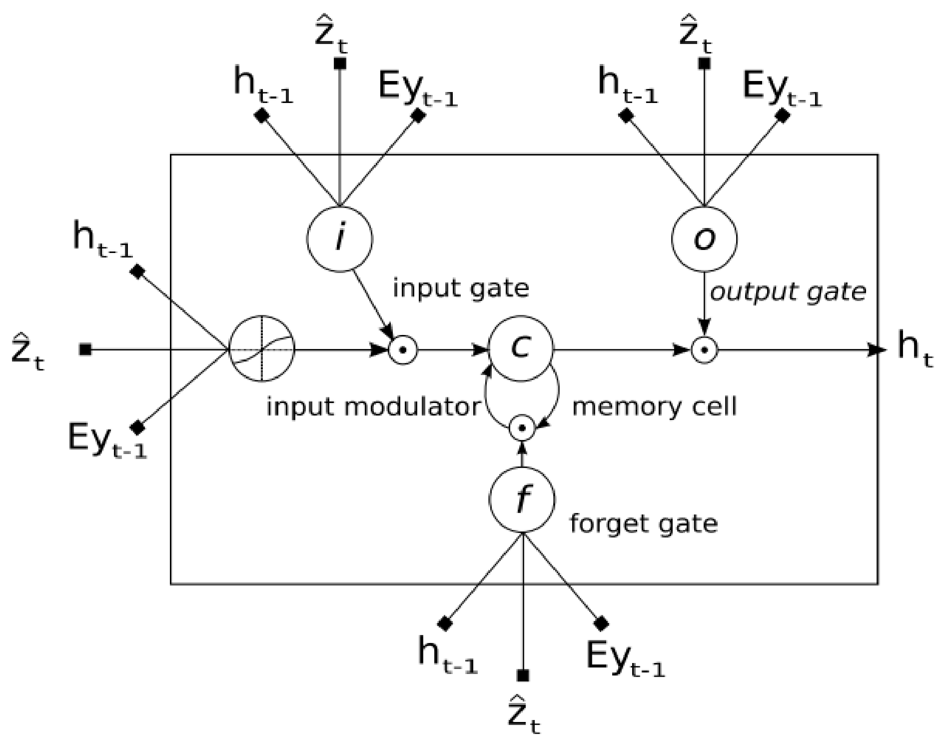


Feature vector a
(annotation vector)

Model

Decoder(LSTM)

- for each time step t, LSTM generates a element y_t of caption vector y.
- total unfolding time : C(length of caption)
- for each time step t,
    - Input : hidden state h_{t-1}, generated word y_{t-1} for time step t-1
    - Output: y_t(current time)



Model

Decoder(LSTM)

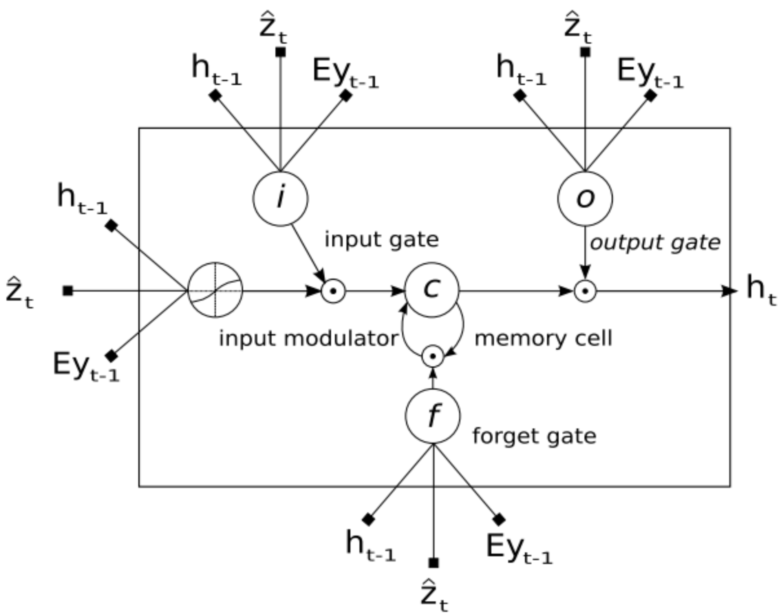
- $T_{s,t}$: $R^s \rightarrow R^t$ (simple affine transformation, $T_{n,m}(x) = Wx + b$))

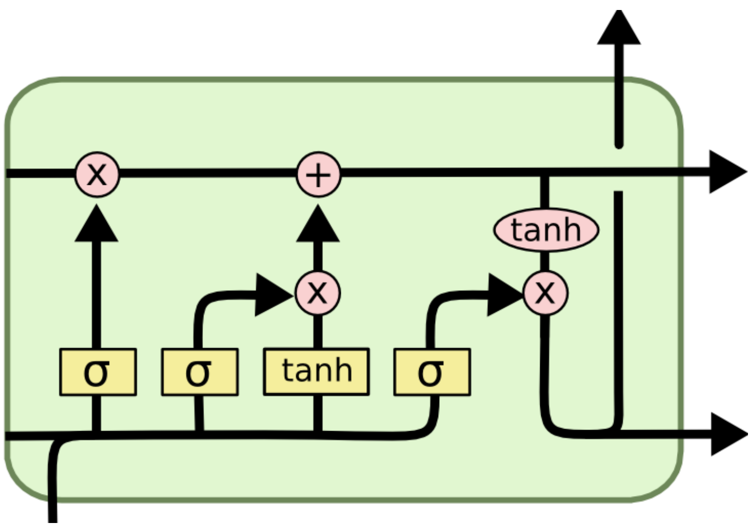
Input
Forgot
output

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} Ey_{t-1} \\ h_{t-1} \\ \hat{z}_{t-1} \end{pmatrix}$$

Memory
hidden

$$c_t = f_t \bigcirc c_{t-1} + i_t \bigcirc g_t$$
$$h_t = o_t \bigcirc \tanh(c_t)$$





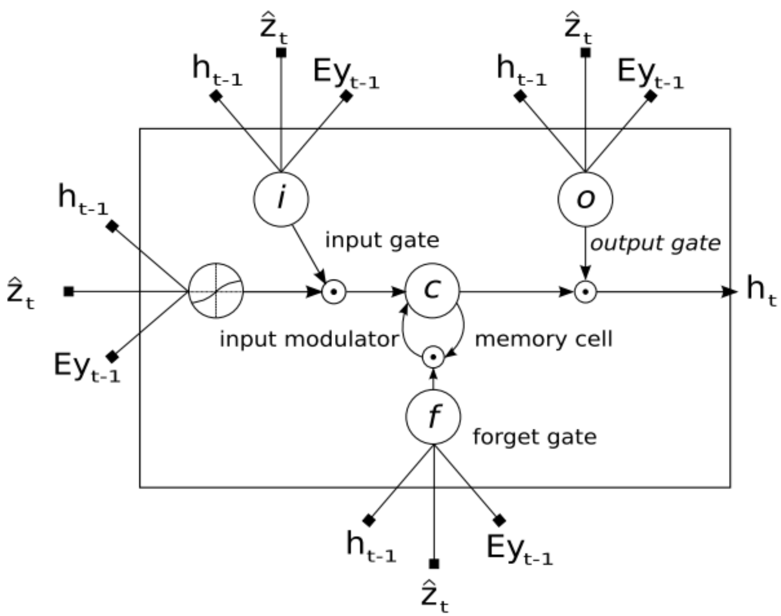
Model

Decoder(LSTM)

• the initial memory state and hidden state of the LSTM are predicted by an average of the annotation vectors fed through two separate MLPs

$$\mathbf{c}_0 = f_{\text{init,c}}\left(\frac{1}{L}\sum_i^L \mathbf{a}_i\right)$$

$$\mathbf{h}_0 = f_{\text{init,h}}\left(\frac{1}{L}\sum_i^L \mathbf{a}_i\right)$$



$Ey_{t-1} = y_{t-1}$ is embedded by embedding metric $\mathrm{E} \in R^{mXK}$,
m-dimensional vector
(trainable parameter, random initialization)

$=$ *context vector*, determined by Attention model

Model

Decoder(LSTM)

•$\hat{z} \in R^D$=$context\ vector$, determined by Attention model

$$\hat{z}_t = \phi(a, \alpha_t),\ \text{where}\ \alpha_{ti} = \frac{\exp(f_{att}(a_i, h_{t-1}))}{\sum_{k=1}^{L} \exp(f_{att}(a_k, h_{t-1}))}$$

α_t:a의 weight vector, determine where is attend.
Element-wise summation=1

f_{att}: attention model to calculate weight vector α
Using a and h_{t-1}

ϕ: calculate $\hat{z}$ using a and α_t

•In this work, we use a deep output layer to compute the output word probability given the LSTM state, the context vector and the previous word:

$$\mathbf{L}_o \in \mathbb{R}^{K \times m},\ \mathbf{L}_h \in \mathbb{R}^{m \times n},\ \mathbf{L}_z \in \mathbb{R}^{m \times D}$$

$$p(\mathbf{y}_t | \mathbf{a}, \mathbf{y}_1^{t-1}) \propto \exp(\mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h \mathbf{h}_t + \mathbf{L}_z \hat{\mathbf{z}}_t))$$

Stochastic Hard Attention

s_t : location variable as where the model decides to focus attention when Generating the t^{th} word. The part that we want to focus is set 1, if not 0.
-> latent variable-> parameterize multimoulli distribution using α_t

α_{ti}: For time step t, Probability(to be 1) of ith element of s_t (s_{ti})

$$p(s_{t,i} = 1 \mid s_{j<t}, \mathbf{a}) = \alpha_{t,i}$$

$$\hat{\mathbf{z}}_t = \sum_i s_{t,i}\mathbf{a}_i.$$

Random variable

Stochastic Hard Attention

Our goal is select the most likely caption y for a given feature vector a.
-> calculate maximum log likelihood $max_y \log p(y|a)$
-> using attention location s_t to calculate lower bound for log likelihood.

$$\begin{aligned} L_s &= \sum_s p(s \mid \mathbf{a}) \log p(\mathbf{y} \mid s, \mathbf{a}) \\ &\leq \log \sum_s p(s \mid \mathbf{a}) p(\mathbf{y} \mid s, \mathbf{a}) \\ &= \log p(\mathbf{y} \mid \mathbf{a}) \end{aligned}$$

$$\frac{\partial L_s}{\partial W} = \sum_s p(s \mid \mathbf{a}) \Big[\frac{\partial \log p(\mathbf{y} \mid s, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid s, \mathbf{a}) \frac{\partial \log p(s \mid \mathbf{a})}{\partial W} \Big]$$

To calculate gradient, we should consider all attention location s
-> too slow

Stochastic Hard Attention

Using Mote Carlo based sampling (more fast)
-> total value(expected value of some random variable) can be approximated using independent sample mean.

$$\tilde{s}_t \sim \text{Multinoulli}_L(\{\alpha_i\})$$

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\partial \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a}) \frac{\partial \log p(\tilde{s}^n \mid \mathbf{a})}{\partial W} \right]$$

To lower gradient variance
Using moving average,
Entropy term H[s]
0.5 prob -> $\tilde{s} \rightarrow \alpha$

Accumulated sum of the previous log-likehood with exponential decay

$$b_k = 0.9 \times b_{k-1} + 0.1 \times \log p(\mathbf{y} \mid \tilde{s}_k, \mathbf{a})$$

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\partial \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})}{\partial W} + \lambda_r (\log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a}) - b) \frac{\partial \log p(\tilde{s}^n \mid \mathbf{a})}{\partial W} + \lambda_e \frac{\partial H[\tilde{s}^n]}{\partial W} \right]$$

Stochastic Hard Attention

Hard attention model.
-> reinforcement learning update rule
-> for each time step, calculate $\tilde{z}$ though hard choice(sampling a_i)

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\partial \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})}{\partial W} + \lambda_r (\log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a}) - b) \frac{\partial \log p(\tilde{s}^n \mid \mathbf{a})}{\partial W} + \lambda_e \frac{\partial H[\tilde{s}^n]}{\partial W} \right]$$

Action : select location of attention
Reward: log-likelihood lower bond

Deterministic Soft Attention

Instead of stochastic sampling for each time, calculate context vector $\hat{z}_t$

Soft attention $$\phi(\{a_i\},\{\alpha_i\}) = \sum_i^L \alpha_i a_i.$$

More smooth
More differentiable

$$\mathbb{E}_{p(s_t|a)} p[\hat{z}_t] = \sum_{i=1}^{L} \alpha_{ti} a_i.$$

It can use approximation

$$\mathbf{n}_t = \mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h \mathbf{h}_t + \mathbf{L}_z \hat{\mathbf{z}}_t)$$

$(p(y_t|a, y_1^{t-1})$ approximation

h_t is a linear projection of the stochastic context vector z^t followed by tanh non-linearity

Deterministic Soft Attention

n_{ti} : n_t calculated by random variable zt
NWGM (Normalized Weighted Geometric Mean) For Kth word prediction

$$NWGM[p(y_t = k \mid \mathbf{a})] = \frac{\prod_i \exp(n_{t,k,i})^{p(s_{t,i}=1|a)}}{\sum_j \prod_i \exp(n_{t,j,i})^{p(s_{t,i}=1|a)}}$$
$$= \frac{\exp(\mathbb{E}_{p(s_t|a)}[n_{t,k}])}{\sum_j \exp(\mathbb{E}_{p(s_t|a)}[n_{t,j}])}$$

$$\mathbb{E}[n_t] = L_o(Ey_{t-1} + L_h h_t + L_z \hat{z}_t)$$

NWGM for caption prediction is
approximated by expected context vector

Doubly stochastic attention

$$\sum_i \alpha_{ti} = 1$$

$$\sum_t \alpha_{ti} \approx 1$$

Add new constraint

Better performance
-> the model is more focused attention
because it prevent all parts of every
image from seeing for the whole time

Negative log-likelihood minimize-> end to end learning

$$L_d = -\log(p(y|x)) + \lambda \sum_i^L \left(1 - \sum_t^C \alpha_{ti}\right)^2.$$

Experiment

Dataset: MSCOCO, Flickr8k, Frlickr30k

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

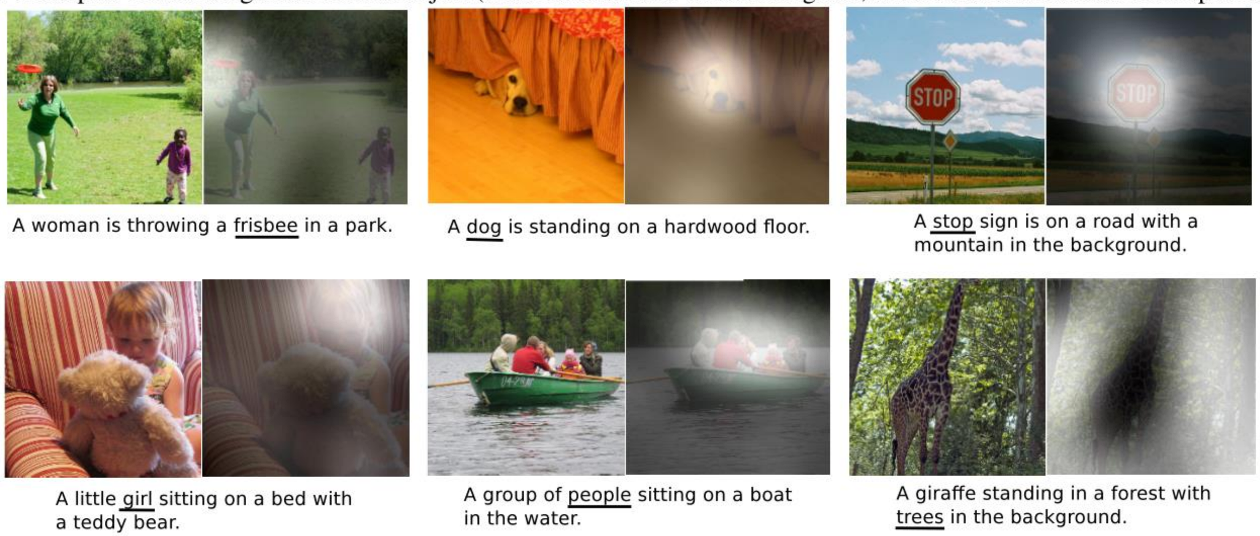
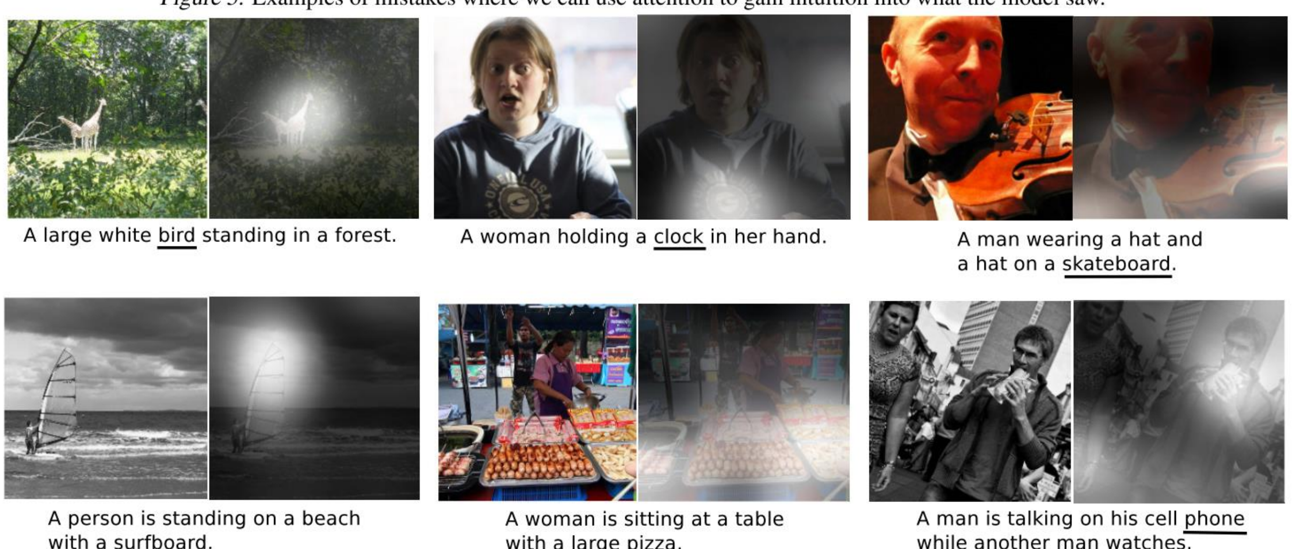


Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.



Experiment

Dataset: MSCOCO, Flickr8k, Frlickr30k

Dataset	Model	BLEU				
		BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC(Vinyals et al., 2014)$^{\dagger\Sigma}$	63	41	27	—	—
	Log Bilinear (Kiros et al., 2014a)$^{\circ}$	65.6	42.4	27.7	17.7	17.31
	Soft-Attention	**67**	44.8	29.9	19.5	18.93
	Hard-Attention	**67**	**45.7**	**31.4**	**21.3**	**20.30**
Flickr30k	Google NIC$^{\dagger\circ\Sigma}$	66.3	42.3	27.7	18.3	—
	Log Bilinear	60.0	38	25.4	17.1	16.88
	Soft-Attention	66.7	43.4	28.8	19.1	**18.49**
	Hard-Attention	**66.9**	**43.9**	**29.6**	**19.9**	18.46
COCO	CMU/MS Research (Chen & Zitnick, 2014)^{a}	—	—	—	—	20.41
	MS Research (Fang et al., 2014)$^{\dagger a}$	—	—	—	—	20.71
	BRNN (Karpathy & Li, 2014)$^{\circ}$	64.2	45.1	30.4	20.3	—
	Google NIC$^{\dagger\circ\Sigma}$	66.6	46.1	32.9	24.6	—
	Log Bilinear$^{\circ}$	70.8	48.9	34.4	24.3	20.03
	Soft-Attention	70.7	49.2	34.4	24.3	**23.90**
	Hard-Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04

Conclusion

- Image caption 문제를 풀기 위해 encoder-decoder concept 사용
- 본 논문에서는 decoder에 attention이라는 개념을 추가
- Encoder는 CNN(VGG), Decoder는 LSTM을 사용하여 구현,
- 바로 전 state h, 바로 전 caption word y, 그리고 attention model을 통해 생성되는 context vector z가 LSTM cell의 input.
- Context vector z는 hard attention과 soft attention 두 가지 방법 중에 한 가지 방법을 선택하여 생성
- Hard attention은 먼저 location variable s를 정의하고, 이것을 사용해 log-likelihood의 lower bound Ls를 계산. Ls를 optimization하기 위해 gradient를 구할 때, 계산의 편의를 위해 Monte Carlo based sampling approximation을 사용해 문제를 해결. 이 update rule은 reinforcement learning의 update rule과 일치.
- Soft attention은 매 iteration마다 sampling을 하는 대신, s의 확률 alpha를 직접 사용하여 z를 계산.
- 제안하는 Attention based caption generation model은 기존 image caption generation 모델들에 비해 훨씬 좋은 성능을 얻음.

Reference

- https://ko.wikipedia.org/wiki/%EB%AA%AC%ED%85%8C%EC%B9%B4%EB%A5%BC%EB%A1%9C_%EB%B0%A9%EB%B2%95
- https://m.blog.naver.com/PostView.nhn?blogId=jinohpark79&logNo=221166026859&categoryNo=1&proxyReferer=https%3A%2F%2Fwww.google.com%2F
- http://sanghyukchun.github.io/93/

Thank You!

Do you have any question?