Neural Image Caption Generation with Visual Attention

Show, Attend and Tell

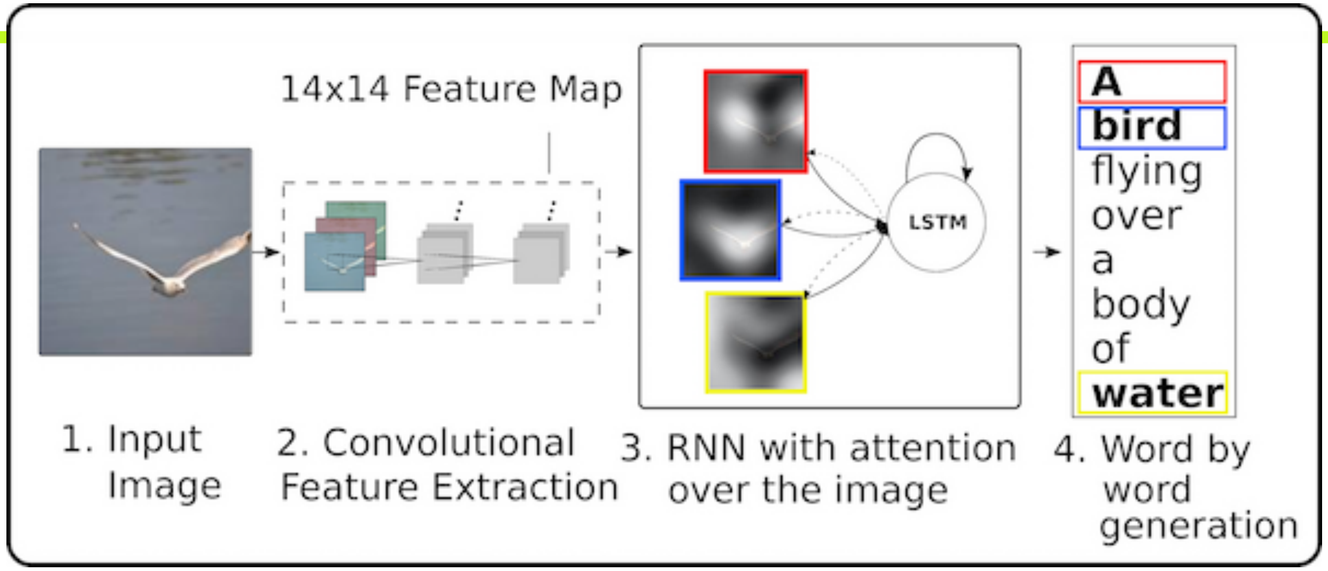
2020 8월 23일 발표 조은수

Show, Attend and Tell

- 목표
  - 이미지에 있는 사물을 알아보고 글로 된 caption을 만들어내는 것
- 내용
  - RNN(LSTM), CNN과 attention mechanism (hard, soft) 적용
- 의의
  - 이미지 인식과 caption 성능 개선

- Architecture Overview :
- CNN Encoder (paper uses VGG architecture)
  - Extracts 512 14x14 feature maps
  - Contrary to using vectors from a fully connected layer on a CNN
- Visual Features (a) | a_i in R^D | (total L x D matrix)
- Increased CNN capacity for Image Captioning : BLEU Score
- Generate Words : LSTM with Attention
- Predicting Next Word

Introduction



- Caption Generation
- CNN과 RNN 사용
- Loss function에서 사용하는 attention mechanism
  - Soft와 Hard attention
  - “Soft” attention은 deterministic machanism으로, standard back-propagation 방법으로 train할 수 있기 때문에 전체 모델이 end-to-end로 learning된다. Soft attention model은 hard attention model의 approximation model이라고 생각하면 된다.
  - “Hard” attention은 stochastic mechanism이며, reinforcement learning으로 train할 수 있다. Hard attention model은 매 iteration마다 데이터를 sampling을 해야하고, reinforcement learning과 neural network 부분이 분리되어있어 end-to-end learning이 아니라는 단점이 있다.

Image Caption Generation with Attention Mechanism

3.1 Model Details

- 3.1.1. Encoder : Convolutional Features
- 3.1.2. Decoder : Long Short-Term Memory Network

Image Caption Generation with Attention Mechanism

3.1 Model Details

- 3.1.1. Encoder : Convolutional Features (CNN 사용)
- Caption Generation task 의 정의 :
- 모델의 input : 이미지
- 모델의 output : 적절한 caption vector y

$$y = \{\mathbf{y}_1, \ldots, \mathbf{y}_C\}, \ \mathbf{y}_i \in \mathbb{R}^K$$

C = (Caption 길이)

K = (단어 크기)

Image Caption Generation with Attention Mechanism

3.1 Model Details

- 3.1.1. Encoder : Convolutional Features
- CNN을 사용하여 feature vectors a들을 추출, annotation vectors라고 부른다.

$$a = \{\mathbf{a}_1, \ldots, \mathbf{a}_L\}, \ \mathbf{a}_i \in \mathbb{R}^D$$

L = (추출한 벡터의 개수) = (CNN 마직막 layer에 있는 필터 개수)

D = (filter마다 있는 D개의 neuron)

- Feature vectors와 2-D인 이미지 사이에서 정보를 주고 받아야 하기 때문에, lower convolutional layer에서 features를 추출한다.
- 그러면 Decoder에서는 이미지의 모든 부분이 아니라 특정 부분에서, 즉 feature vectors의 부분집합에 집중할 수 있다.

3. Image Caption Generation with Attention Mechanism

3.1 Model Details

- 3.1.2. Decoder : Long Short-Term Memory Network

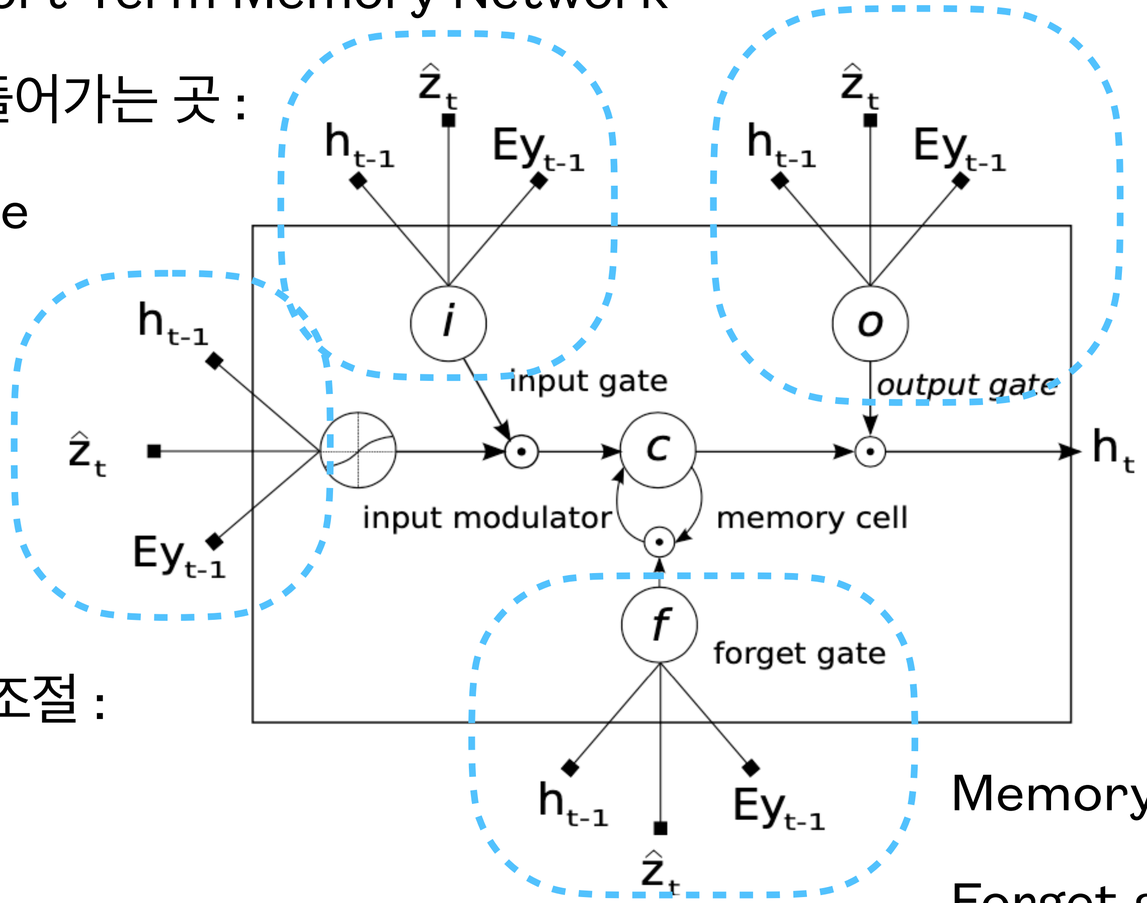
Input이 들어가는 곳 :

Input gate

Memory를 어떻게

내보낼지 결정 :

Output gate



memory로 들어가는 input 조절 :

Input modulator

Memory cell 지우기 :

Forget gate

Figure 4. LSTM cell

Image Caption Generation with Attention Mechanism

3.1 Model Details

- 3.1.2. Decoder : Long Short-Term Memory Network
- LSTM에서 context vector에서 설정한 time step t마다 한 단어의 caption(y_t)을 만들어낸다.
- Context vector는 이전의 hidden state(h_{t-1})와 이전에 만들어낸 단어(y_{t-1})로 만들어낸다.

- Affine transformation

$$T_{s,t} : \mathbb{R}^s \rightarrow \mathbb{R}^t$$

Image Caption Generation with Attention Mechanism

3.1 Model Details

- Affine transformation
- Affine 변환은 직선, 길이(거리)의 비, 평행성(parallelism)을 보존하는 변환이며 그 일반식은 다음과 같습니다.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

- 좀더 쉽게 말하면 Affine 변환은 회전, 평행이동, 스케일 뿐만 아니라 shearing, 반전(reflection)까지를 포함한 변환입니다.

Image Caption Generation with Attention Mechanism

3.1 Model Details

D+m+n 차원 -> n 차원으로 변환

Input
Forget
Output

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix} \quad (1)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \quad (3)$$

E = embedding matrix

바로 전 caption y_{t-1}을

K -> m 차원으로 바꿈

바로 전 hidden state

of LSTM

Context vector :

Attention model에 의해 결정됨

t = input이 들어오는 시간

Input

LSTM cell

Image Caption Generation with Attention Mechanism

3.1 Model Details

- 3.1.2. Decoder : Long Short-Term Memory Network

a: weight vector 중 weight value

-

Attention model

$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_{t-1}) \quad (4)$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})}. \quad (5)$$

(4) f_{att} attention model

(5) 각 Annotation vector의 weight

Context vector

$$\hat{\mathbf{z}}_t = \phi\left(\{\mathbf{a}_i\}, \{\alpha_i\}\right), \quad (6)$$

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

(6) Annotation vector $\{a_i\}$, 그 weight vector(5)를 함께 받아 결과를 하나의 벡터로 출력하는 함수

Image Caption Generation with Attention Mechanism

3.1 Model Details

- 3.1.2. Decoder : Long Short-Term Memory Network

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

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

첫 memory state와 hidden state :

두 MLP에 들어간 annotation vector의 평균

$$p(\mathbf{y}_t|\mathbf{a}, \mathbf{y}_1^{t-1}) \propto \exp(\mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h\mathbf{h}_t + \mathbf{L}_z\hat{\mathbf{z}}_t)) \quad (7)$$

Deep outer layer

-> 내보내는 단어의 확률은 LSTM state의

context vector와 이전에

출력한 단어와 관련된 확률로 계산한다.

$\mathbf{L}_o \in \mathbb{R}^{K\times m}$, $\mathbf{L}_h \in \mathbb{R}^{m\times n}$, $\mathbf{L}_z \in \mathbb{R}^{m\times D}$, and $\mathbf{E}$: train과정에서 learning하는 parameter

4. Learning Stochastic “Hard” vs Deterministic “Soft” Attention

Attention model은 어떻게 하는 게 좋을까?

- 선택지 두 가지:
  - 4.1 Stochastic “Hard” Attention
  - 4.2 Deterministic “Soft” Attention

4. Learning Stochastic “Hard” vs Deterministic “Soft” Attention

Attention model은 어떻게 하는 게 좋을까?

- Hard Attention vs Soft Attention 은?
- Soft : weight가 이미지에 골고루 배정된다.
  - 장점 : model이 원활하게 돌아간다.
  - 단점 : 이미지가 크면 효율적이지 못하다
- Hard : 한 번에 이미지의 한 부분만 고려한다.
  - 장점 : 계산 시간의 단축
  - 단점 : model에서 큰 차이가 없으며 강화 학습 등 별도로 복잡한 기술을 사용해야 한다.

4. Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4.1 Stochastic “Hard” Attention

- S_t : location variable (t 번째 단어 생성 중에)

주어진 시간 t에서의 s_t의 i번째 element

$$p(s_{t,i} = 1 \mid s_{j<t}, \mathbf{a}) = \alpha_{t,i} \quad (8)$$

$$\hat{\mathbf{z}}_t = \sum_i s_{t,i}\mathbf{a}_i. \quad (9)$$

Multinoulli distribution

st,i가 1이 될 확률 (alpha를 모두 더하면 1)

(9) 위 값을 이용해 random variable을 새롭게 정의

4. Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4.1 Stochastic “Hard” Attention

$$
\begin{aligned}
L_s &= \sum_s p(s \mid \mathbf{a}) \log p(\mathbf{y} \mid s, \mathbf{a}) \\
&\leq \log \sum_s p(s \mid \mathbf{a}) p(\mathbf{y} \mid s, \mathbf{a}) \\
&= \log p(\mathbf{y} \mid \mathbf{a}) \qquad (10)
\end{aligned}
$$

Ls = log likelihood의 lower bound

최적화: Ls의 maximum

y = sequence of words

a = image features

$$
\frac{\partial L_s}{\partial W} = \sum_s p(s \mid \mathbf{a}) \left[\frac{\partial \log p(\mathbf{y} \mid s, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid s, \mathbf{a}) \frac{\partial \log p(s \mid \mathbf{a})}{\partial W} \right]. \qquad (11)
$$

미분값 W에 대한 learning algorithm

4. Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4.1 Stochastic “Hard” Attention

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N}\sum_{n=1}^{N}\left[\frac{\partial \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})}{\partial W} + \log p(\mathbf{y} \mid \tilde{s}^n, \mathbf{a})\frac{\partial \log p(\tilde{s}^n \mid \mathbf{a})}{\partial W}\right] \quad (12)$$

Monto Carlo based sampling 으로 gradient를 근사함

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

Moving avg baseline

4. Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4.1 Stochastic “Hard” Attention

- Final Learning Algorithm : 강화학습 update rule과 같다
- H[s] = (entropy term on multinoulli distribution)
- $\tilde{s}$: attention location
- λr and λe are two hyper-parameters set by cross- validation

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

Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4. 2 Deterministic “Soft” Attention

- Hard attention은 timestamp t마다 attention location을 sampling 해줘야 한다.

  Soft attention에서 context vector 예측값 직접 구하기 :

$$\mathbb{E}_{p(s_t|a)}[\hat{\mathbf{z}}_t] = \sum_{i=1}^{L} \alpha_{t,i}\mathbf{a}_i \qquad (13)$$

  전체 모델이 smooth해지고, differentiable해지기 때문에 back-propagation을 사용해서 end-to-end learning이 가능해진다.

Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4. 2 Deterministic “Soft” Attention

- k번째 단어 y prediction을 위한 Normalized Weighted Geometric Mean

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

Likelihood

$$= \frac{\exp(\mathbb{E}_{p(s_t|a)}[n_{t,k}])}{\sum_j \exp(\mathbb{E}_{p(s_t|a)}[n_{t,j}])}$$

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

$n_{t,i}$는 a_i와 random variable $\hat{z}_t$를 이용해 구한 n_t 값

Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4.2 Deterministic “Soft” Attention

4.2.1 Doubly Stochastic Attention

- $\sum_i \alpha_{ti} = 1$ 이라는 정의에 $\sum_t \alpha_{ti} \approx 1$ 조건 추가 :
  - 모델이 모든 부분을 다 보는 것을 방해하기 때문에 더 focused 된 결과가 나와 더욱 정확하다

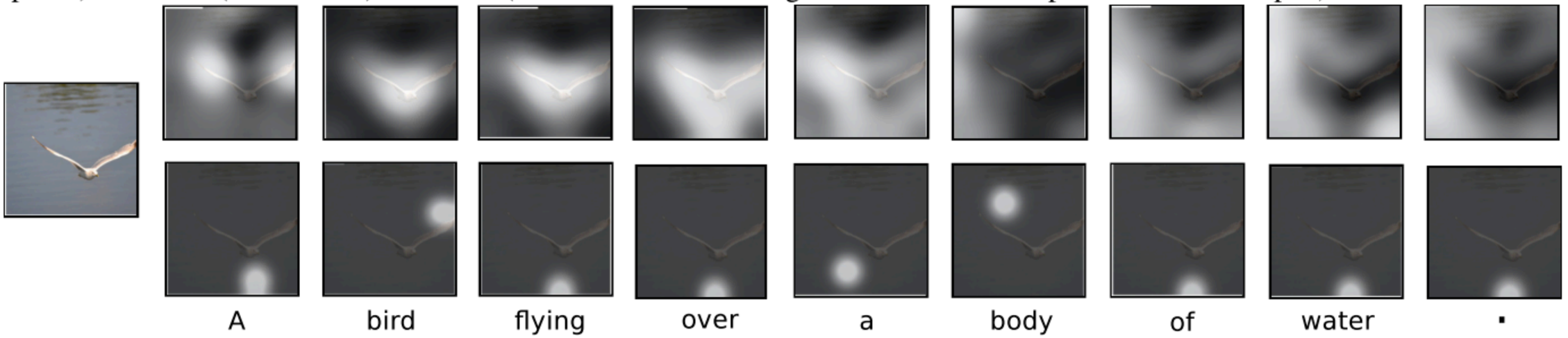
$$L_d = -\log(P(\mathbf{y}|\mathbf{x})) + \lambda \sum_i^L (1 - \sum_t^C \alpha_{ti})^2 \qquad (14)$$

이 negative log-likelihood를 minimize하여 end-to-end learning을 한다.

Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4.2 Experiment

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. “soft” (top row) vs “hard” (bottom row) attention. (Note that both models generated the same captions in this example.)



Learning Stochastic “Hard” vs Deterministic “Soft” Attention

4.3 Training Procedure

Table 1. BLEU-1,2,3,4/METEOR metrics compared to other methods, † indicates a different split, (—) indicates an unknown metric, ∘ indicates the authors kindly provided missing metrics by personal communication, Σ indicates an ensemble, a indicates using AlexNet

		BLEU				
Dataset	Model	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 caption에 encoder-decoder concept를 사용한다.
- encoder는 cnn, decoder는 rnn의 LSTM을 사용한다. LSTM cell의 input은 context vector z이다.
- Hard attention(stochastic)과 soft attention(deterministic) 두 가지 attention mechanism을 제시한다.
- Hard attention에서는 이미지의 일부분인 location variable s를 정의하고, 이것을 사용해 log-likelihood의 lower bound Ls를 계산한다. Ls를 optimization하기 위해 gradient를 구해야하는데, 이 값을 정확하게 구하는 것이 까다롭기 때문에 Monte Carlo based sampling approximation을 사용해 문제를 해결하게 된다. 이 update rule은 reinforcement learning의 update rule과 일치한다.
- Soft attention은 매 iteration마다 sampling을 하는 대신, s의 확률 alpha를 직접 사용하여 z를 계산한다.
- Attention based caption generation model은 기존 image caption generation 모델들에 비해 훨씬 좋은 성능을 보인다.

- Sunwoo Park, Medium, “Show, Attend, and Tell with Pytorch” https://medium.com/@sunwoopark/show-attend-and-tell-with-pytorch-e45b1600a749
- 논문에 대한 설명이 잘 나와있습니다 : http://sanghyukchun.github.io/93/
- 더 자세한 gif 파일을 볼 수 있는 곳 : http://kelvinxu.github.io/projects/capgen.html
- 더 많은 자료 : https://ahjeong.tistory.com/8
- 코드 :
- 모두의연구소 기술블로그 , “Tensorflow로 Show, Attend and Tell 구현하기” https://www.whydsp.org/299
- 모끼의 딥러닝 공부, “Show, Attend and Tell : Image Captioning에서 Soft Attention, Hard Attention” https://ahjeong.tistory.com/8

- Lil'Log, "Attention? Attention!", https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

- Henry AI Labs, "Show, Attend and Tell", https://www.youtube.com/watch?v=bBMxSg3c_6M