Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering

IEEE Conference on Computer Vision and Pattern Recognition
Date: JUNE. 2018
Page(s): 46047 - 46057

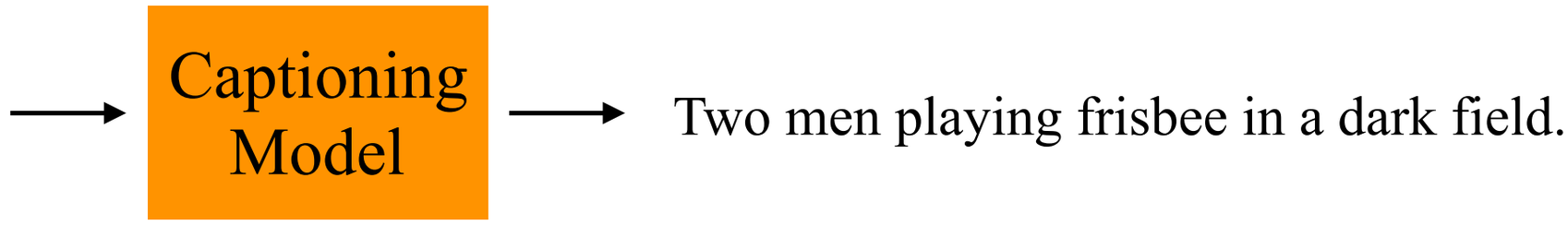
Authors: Peter Anderson, Xiaodong He, Chirs Buehler, Damien Teney, Mark Johnson, Stephen Gould, Lei Zhang.

Presenter: WenWei Kang

- Introduction
- Related work
- Captioning Model
- Visual Question Answering Model
- Evaluation
- Conclusion

Introduction(1/3)

- Image caption



- Visual Question Answering (VQA)

Question: What room are they in?

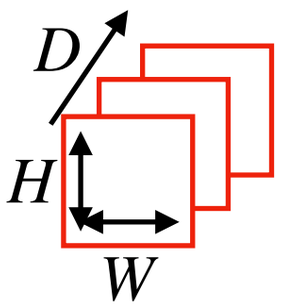
- Yes/No
- 1,2,3,…
- Object

Introduction(2/3)

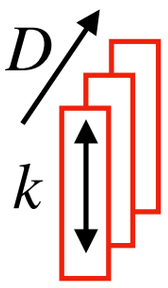
Image caption

Bottom-Up

- CNN



$V = \{m_1, m_2, \ldots, m_D\},\ m_i \in R^{H \times W}$



$V = \{v_1, v_2, \ldots, v_k\},\ v_i \in R^D$

Each location represents a image feature vector v_i.

Top-Down

Captioning Model

- RNN
- LSTM
- GRU

Sequence representation h_t.

Attention distribution over the feature vector V.

$\boldsymbol{\alpha_t} = \{\alpha_{1,t}, \alpha_{2,t}, \ldots, \alpha_{k,t}\},\ \alpha_{i,t} \in R$

Caption

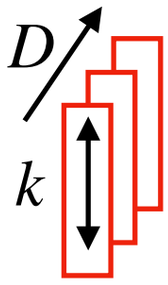
Introduction(3/3)

Visual Question Answering

Bottom-Up

CNN

$V = \{v_1, v_2, \ldots, v_k\},\ v_i \in R^D$



VQA Model

Top-Down

Question: What room are they in?

- RNN
- LSTM
- GRU

Question representation q

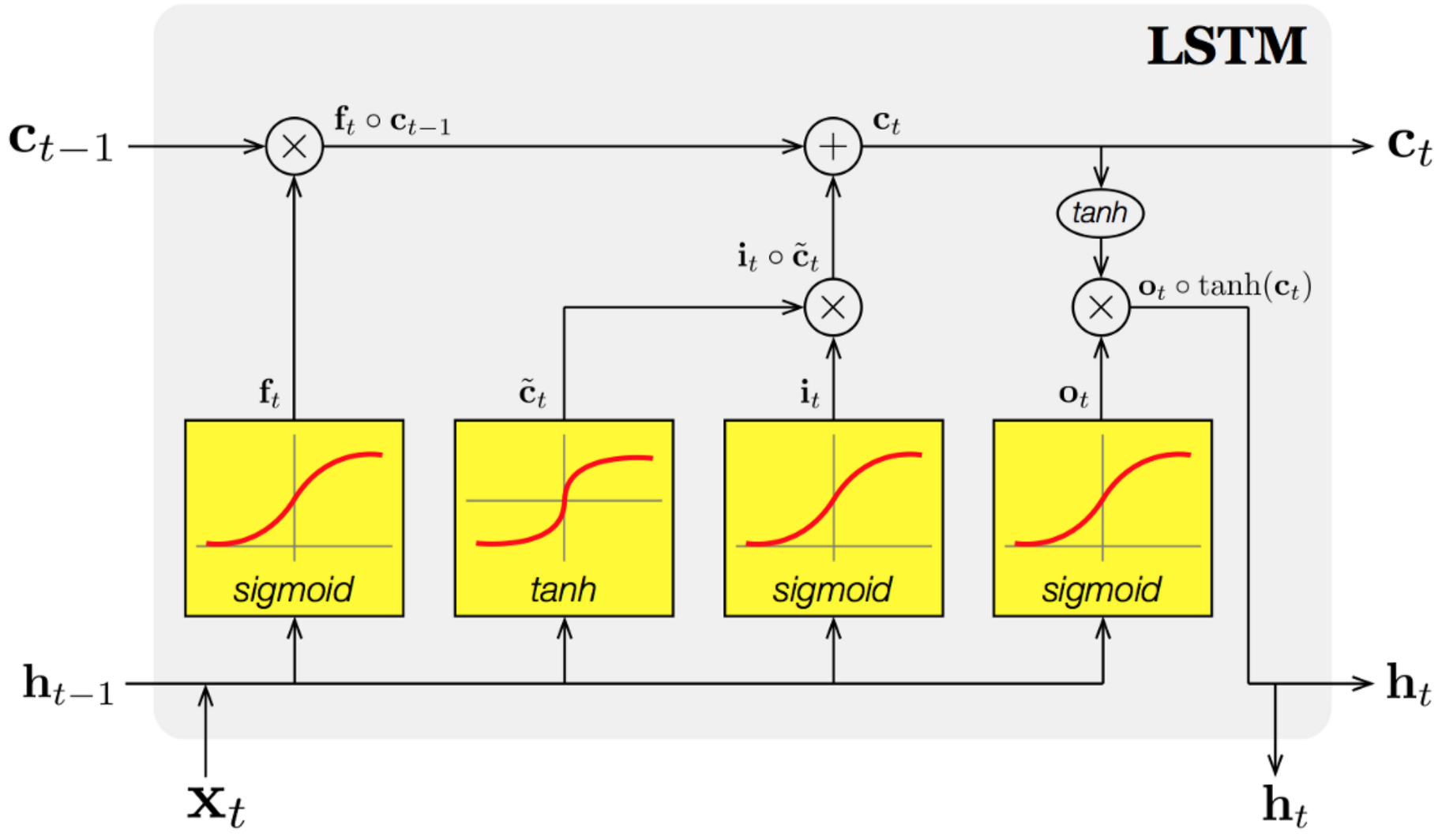
$\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\},\ \alpha \in R$

FCN

Answer: kitchen

Related work(1/3)

Long-Short Term Memory (LSTM)



Gating variables

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_t\right)$$
$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right)$$

Candidate (memory) cell state

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right)$$

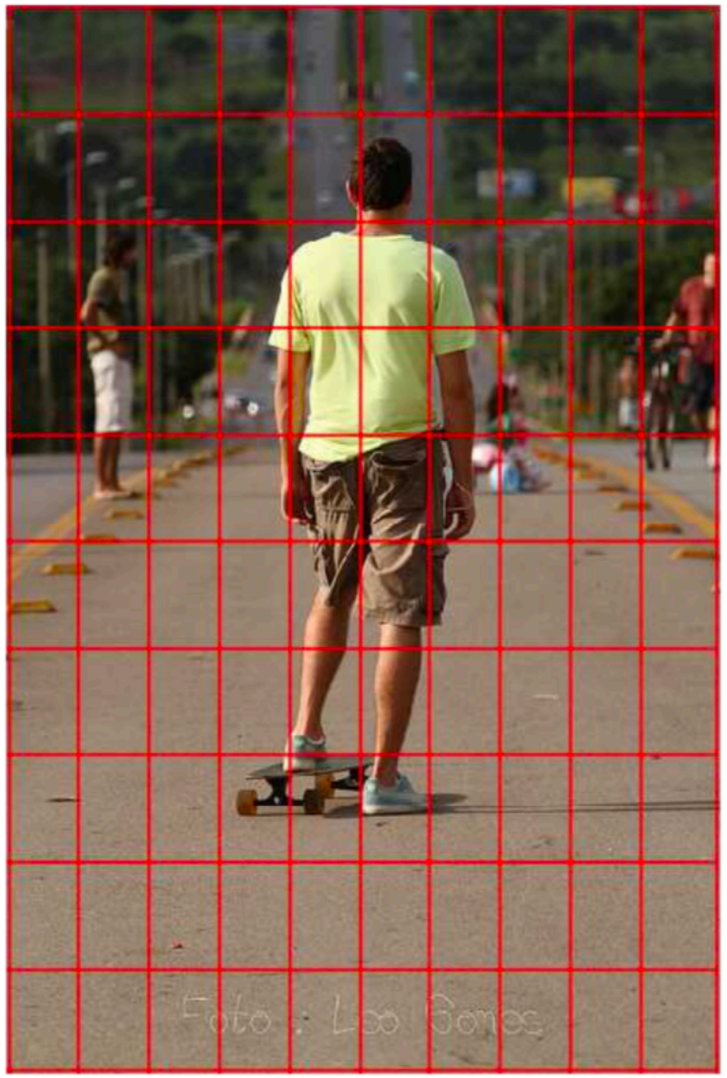
Cell & Hidden state

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

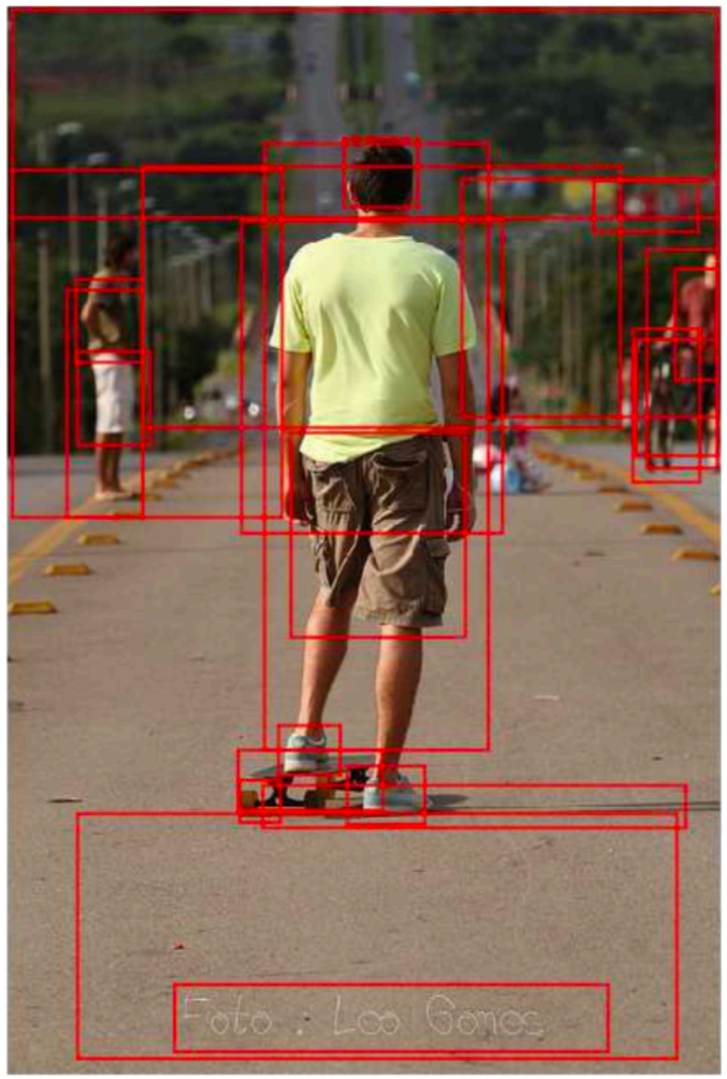
https://codeburst.io/generating-text-using-an-lstm-network-no-libraries-2dff88a3968

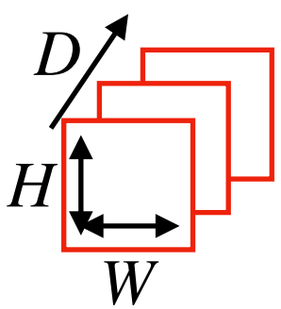
Related work(2/3)

Vanilla CNN



Object detection





$$V = \{m_1, m_2, \ldots, m_D\},\ m_i \in R^{H \times W}$$

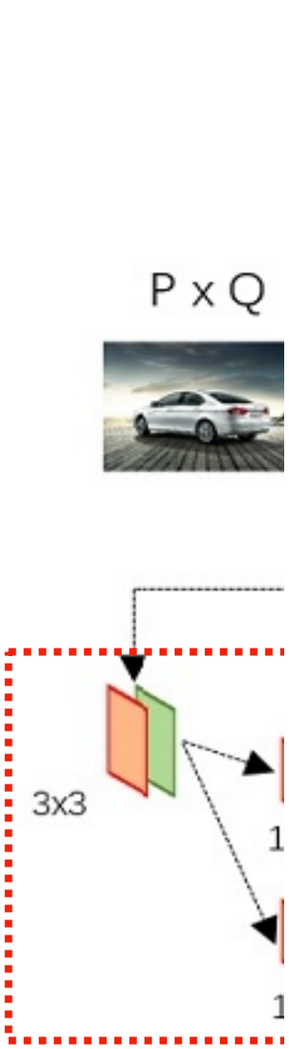
Related work(3/3)

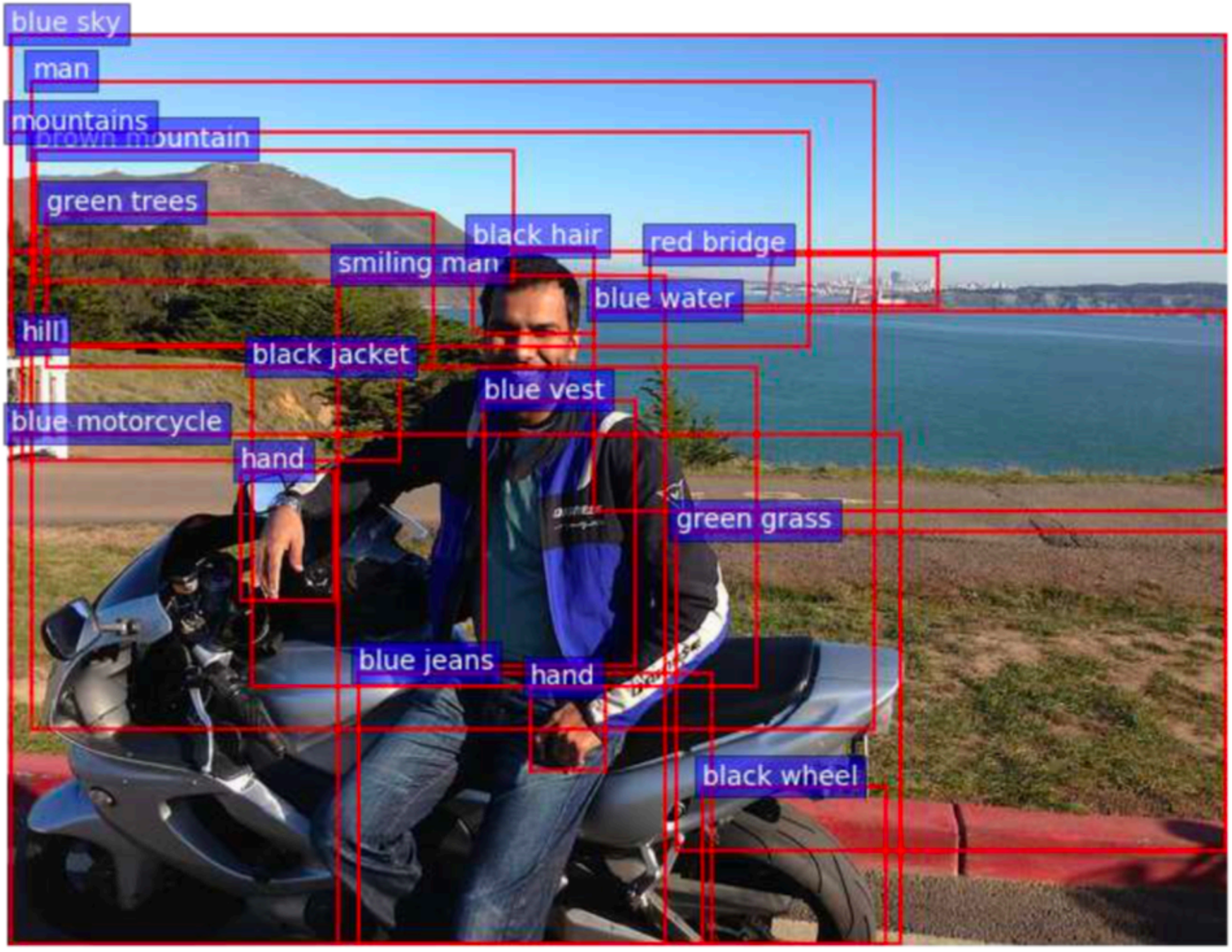
Faster R-CNN

(RPN)

Captioning Model(1/5)

Bottom-Up Attention Model

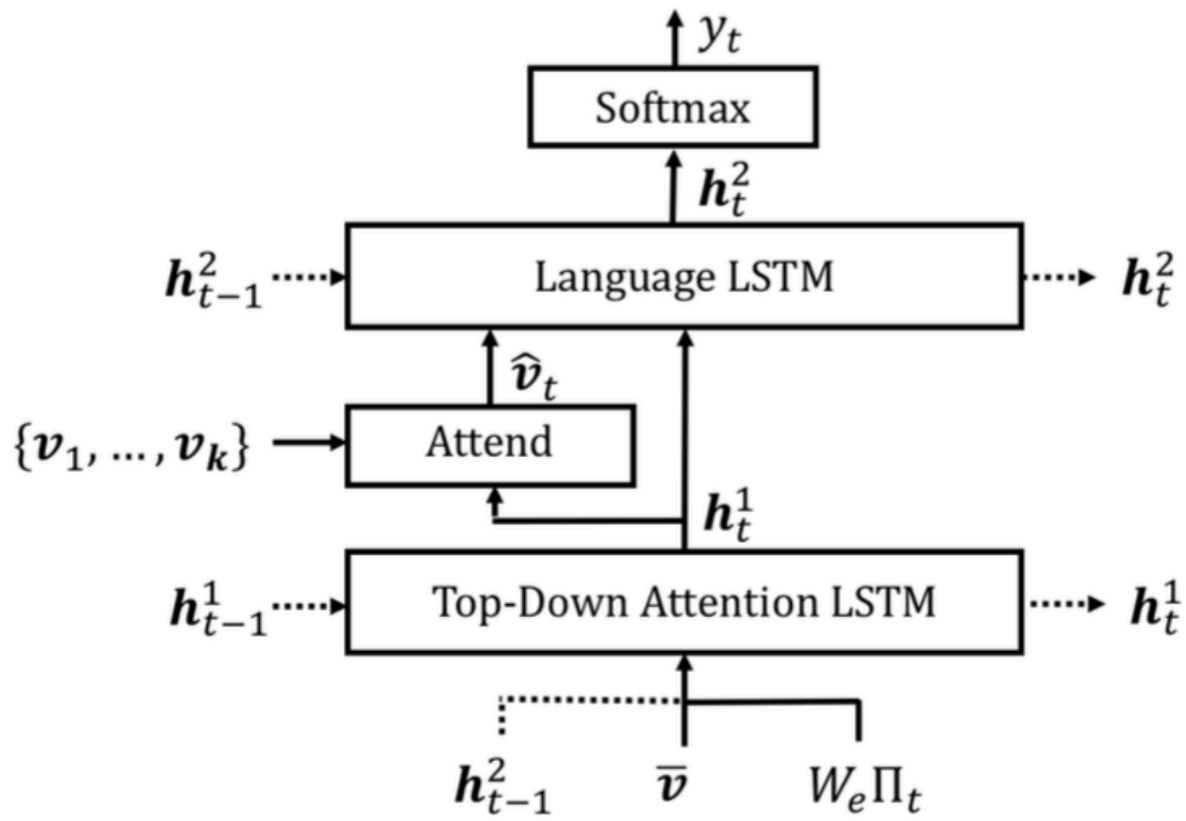




F

ibute classes

Captioning Model(2/5)



Top – Down Attention LSTM : Generate attention distribution $\boldsymbol{\alpha}$.

$$\boldsymbol{\alpha_t} = \{\alpha_{1,t}, \alpha_{2,t}, \ldots, \alpha_{k,t}\},\ \alpha_{i,t} \in R$$

Language LSTM : Generate caption y_t.

Captioning Model(3/5)

Top – Down Attention LSTM

1. Generate hidden state h_t^1 :

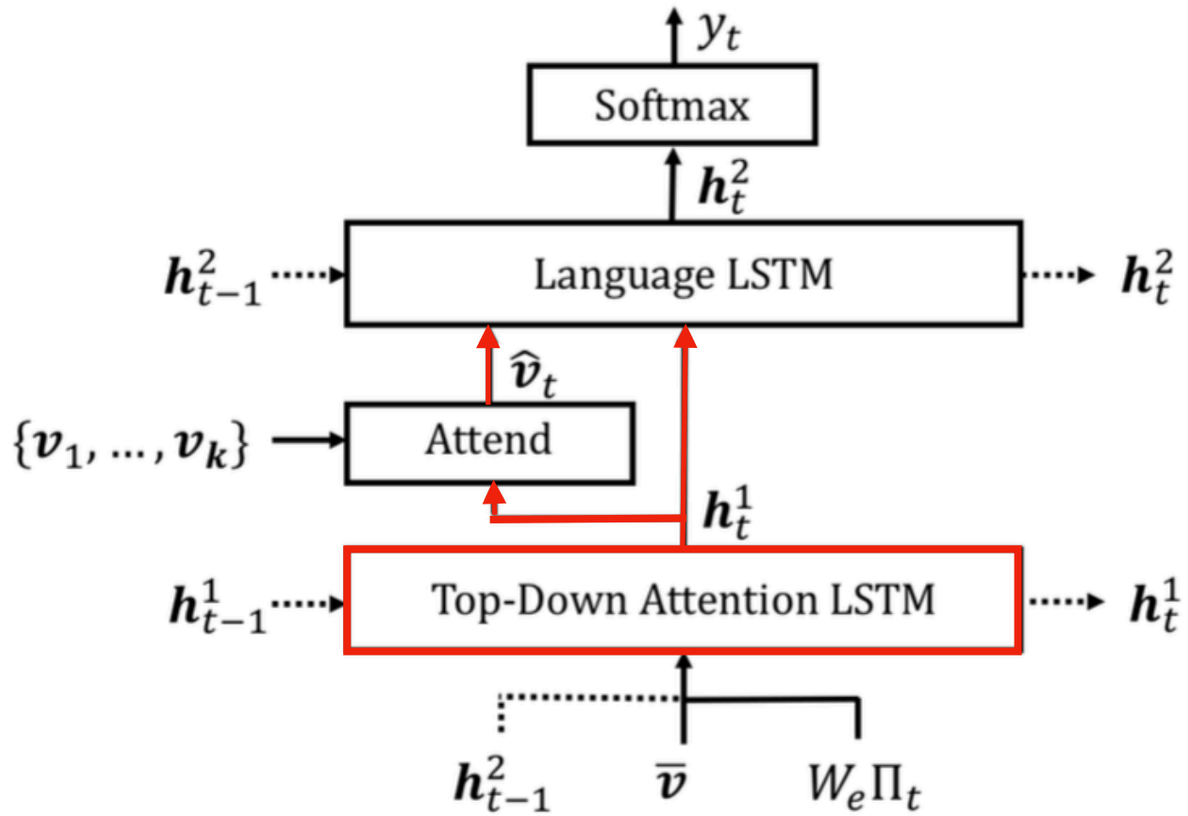
Input : $x_t^1 = [h_{t-1}^2, \bar{v}, W_e \Pi_t]$

h_{t-1}^2 : Language LSTM hidden state

$\bar{v}$: Mean – pooled feature, $\bar{v} = \frac{1}{k} \sum_i v_i$

W_e : Word embedding matrix

Π_t : Caption output generated (one – hot vector)



2. Generate attention distribution $\boldsymbol{\alpha_t}$:

$\alpha_{i,t} = w_a^\top \tanh(W_{v\alpha} v_i + W_{h\alpha} h_t^1),\ W_{v\alpha} \in R^{H \times V},\ W_{h\alpha} \in R^{H \times M},\ w_a \in R^H$

$\boldsymbol{\alpha}_t = \text{softmax}(\alpha_t), \alpha_t = \{\alpha_{1,t}, \alpha_{2,t}, \ldots, \alpha_{k,t}\}$ $\quad \text{softmax}(x) : \frac{e^{x_i}}{\sum_{i=1}^k e^{x_i}}$

3. Calculate a context vector $\hat{v}_t$:

$\hat{v}_t = \sum_{i=1}^{k} \alpha_{i,t} v_i,\ \hat{v}_t \in R^V$

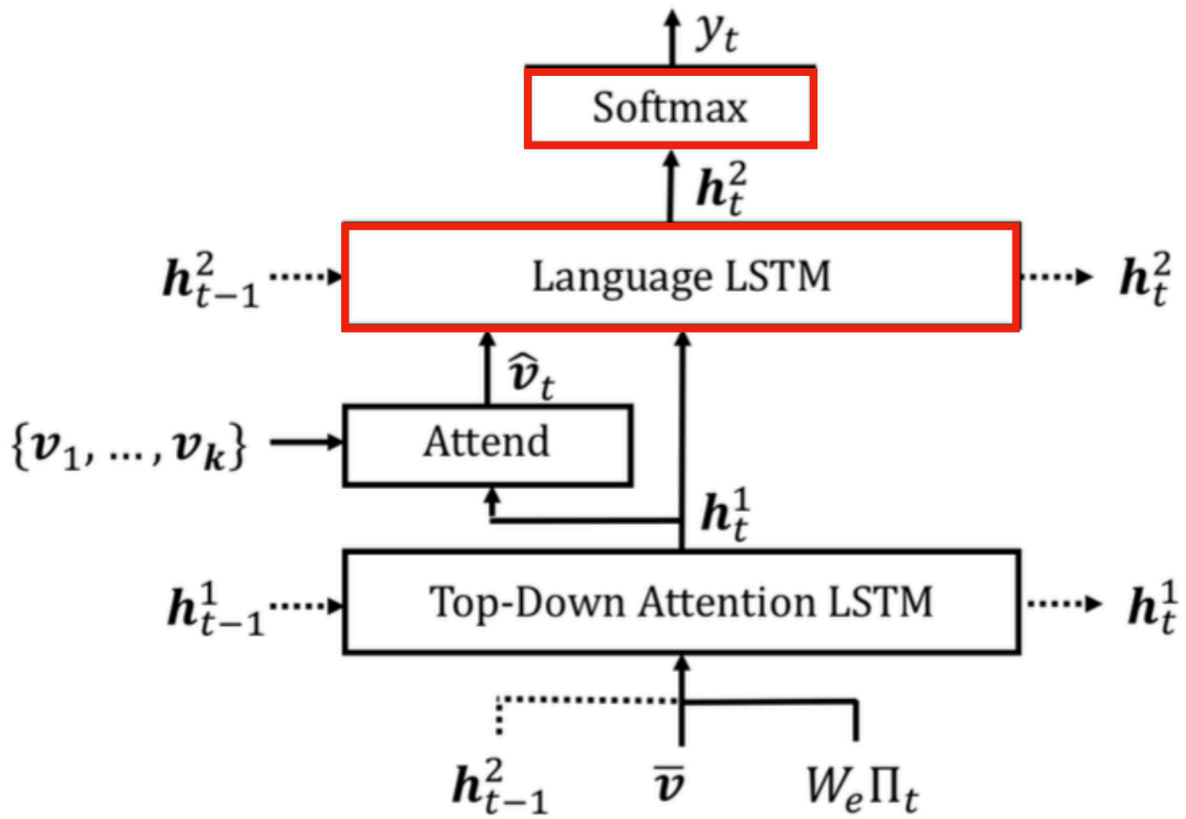
Captioning Model(4/5)

Language LSTM

Generate hidden state h_t^1 :

Input : $x_t^2 = [\hat{v}_t, h_t^1]$

$h_t^2 = \mathrm{LSTM}^2(x_t^2, h_{t-1}^2)$



Softmax

Generate caption y_t :

Output : $p(y_t \mid y_{1:t-1}) = \mathrm{softmax}(W_p h_t^2 + b_p)$

$W_p \in R^{|\Sigma| \times M}, b_p \in R^{|\Sigma|}$

Captioning Model(5/5)

Objective(loss function)

- Cross entropy

$$L_{XE}(\theta) = -\sum_{t=1}^{T} \log(p_\theta(y_t^* \mid y^*_{1:t-1}))$$

$Loss \downarrow \Rightarrow$ Classification Accuracy ↑

$Loss \downarrow \Rightarrow$ Caption Accuracy(BLEU, ROUGE, METEOR, CIDEr . . .) ?

- Approach described as Self-Critical Sequence Training (SCST)

1. Reinforcement learning expected loss function

$$L_R(\theta) = -E_{y_{1:T} \sim p_\theta}[r(y_{1:T})], r(\,.\,) : \text{score function(CIDEr)}$$

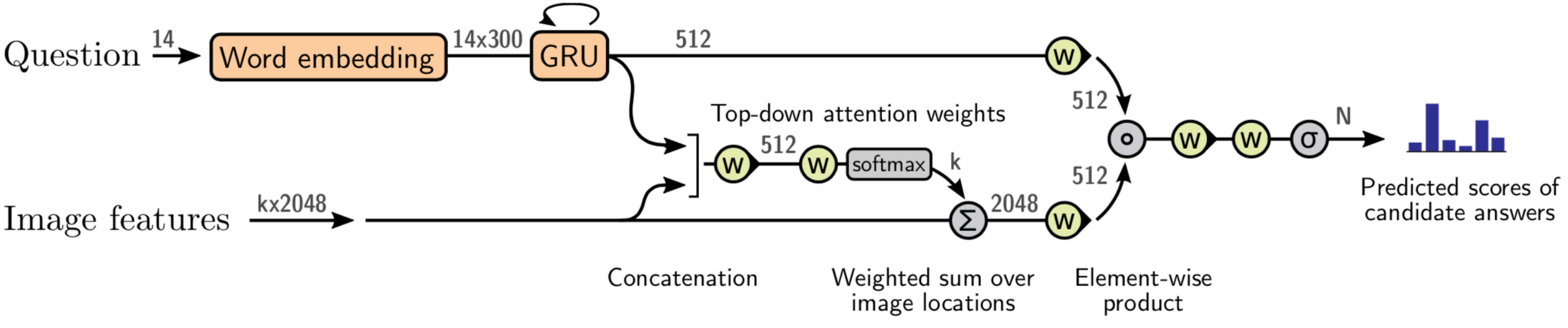
2. Gradient

$$\nabla_\theta L_R(\theta) \approx -(r(y^s_{1:T}) - r(\hat{y}_{1:T}))\nabla_\theta \log p_\theta(y^s_{1:T})$$

$y^s_{1:T}$: sampled caption, $r(\hat{y}_{1:T})$: baseline score

Visual Question Answering Model(1/3)

Tips and Tricks for Visual Question Answering: Learnings from the 2017 Challenge



W : linear layer .

W : nonlinear layer(gated tanh) . $f : x \in R^m \rightarrow y \in R^n$

$\tilde{y} = \tanh(Wx + b) \leftarrow$ candidate output

$g = \sigma(W'x + b') \leftarrow$ gate

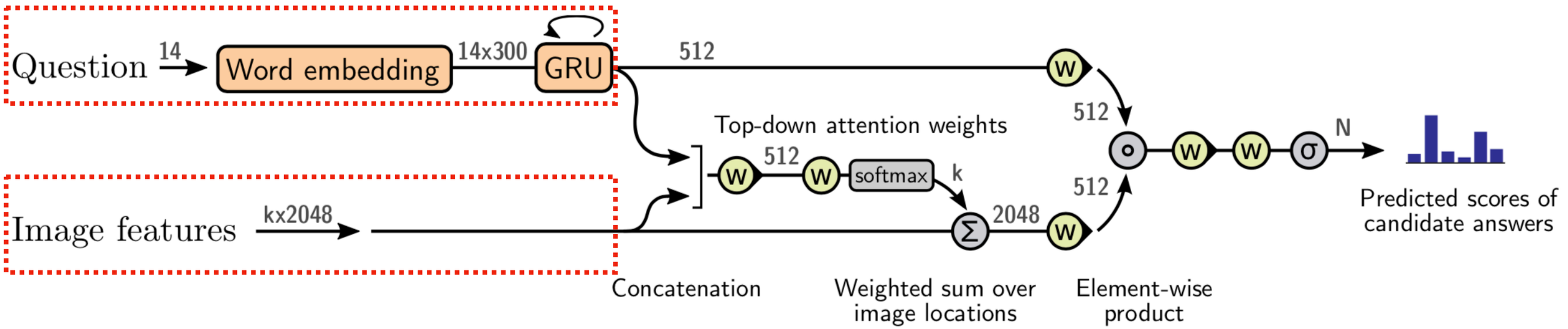
$y = \tilde{y} \circ g \leftarrow$ current output

o : element − wise multiplication .

σ : sigmoid function .

Visual Question Answering Model(2/3)

[Tips and Tricks for Visual Question Answering: Learnings from the 2017 Challenge]



Top-Down

- Question

  $X = \{x_1, x_2, \ldots, x_{14}\}$

  Word embedding

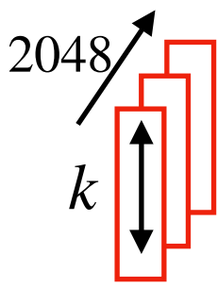
  $W \in R^{N \times 14}$

  GRU hidden state

  $q = \mathrm{GRU}(XW, h_t),\ q \in R^{512}$

Bottom-Up

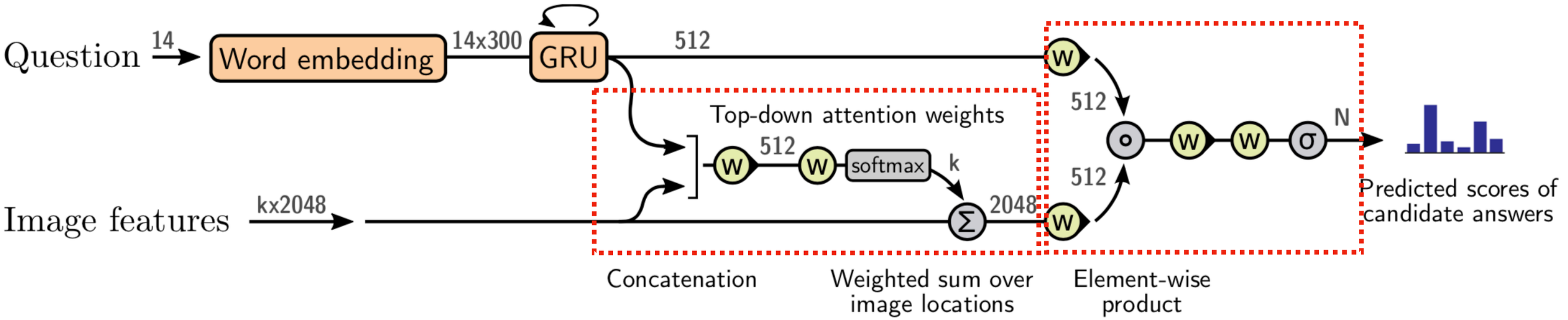
- Image feature



$V = \{v_1, v_2, \ldots, v_k\},\ v_i \in R^{2048}$

Visual Question Answering Model(3/3)

[Tips and Tricks for Visual Question Answering: Learnings from the 2017 Challenge]



Top-Down

- Attention weights

$\alpha_i = w_a^T f_a([v_i, q])$, [.] : Concatenation

$\boldsymbol{\alpha} = \text{softmax}(\alpha), \alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$

- Weighted sum over image locations named context vector

$$\hat{v} = \sum_i^k \alpha_i v_i$$

Output

$$h = f_q(q) \circ f_v(\hat{v})$$

$$p(y) = \sigma(W_o f_o(h)),\ W_o \in R^{|\Sigma| \times N}$$

$|\Sigma|$: vocabulary size

Evaluation(1/4)

Bottom-Up attention model

1. Initialization Faster R-CNN with ResNet-101(**ImageNet**).
2. Use the **Visual Genome dataset** to **pre-train** bottom-up attention model.

VG dataset: Scene graphs containing objects, attributes and relationships.

Pre-train: We **manually** remove abstract classes that exhibit poor detection performance in **initial experiment**.

[2,000 object classes and 500 attribute classes] → [1,600 object classes and 400 attribute classes]

Image Caption

1. Microsoft COCO 2014 captions dataset.
2. Lower case, tokenizing on white space and filtering words.

≤ 5 times

Visual Question Answering

1. VQA v2.0 dataset.
2. Questions are trimmed to a maximum of 14 words and filtering answers.

≤ 7 times

Evaluation(2/4)

Bottom-Up: ResNet-101, Faster R-CNN(Up-Down)

	Cross-Entropy Loss						CIDEr Optimization					
	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE
SCST:Att2in [33]	-	31.3	26.0	54.3	101.3	-	-	33.3	26.3	55.3	111.4	-
SCST:Att2all [33]	-	30.0	25.9	53.4	99.4	-	-	34.2	26.7	55.7	114.0	-
Ours: ResNet	74.5	33.4	26.1	54.4	105.4	19.2	76.6	34.0	26.5	54.9	111.1	20.2
Ours: Up-Down	**77.2**	**36.2**	**27.0**	**56.4**	**113.5**	**20.3**	**79.8**	**36.3**	**27.7**	**56.9**	**120.1**	**21.4**
Relative Improvement	4%	8%	3%	4%	8%	6%	4%	7%	5%	4%	8%	6%

Relative Improvement : $\frac{r_{Up-Down} - r_{ResNet}}{r_{ResNet}}(\%)$

- SCST results are from the best of four initializations.
- Ours results are from a single initializations.

SPICE: Semantic Propositional Image Caption Evaluation

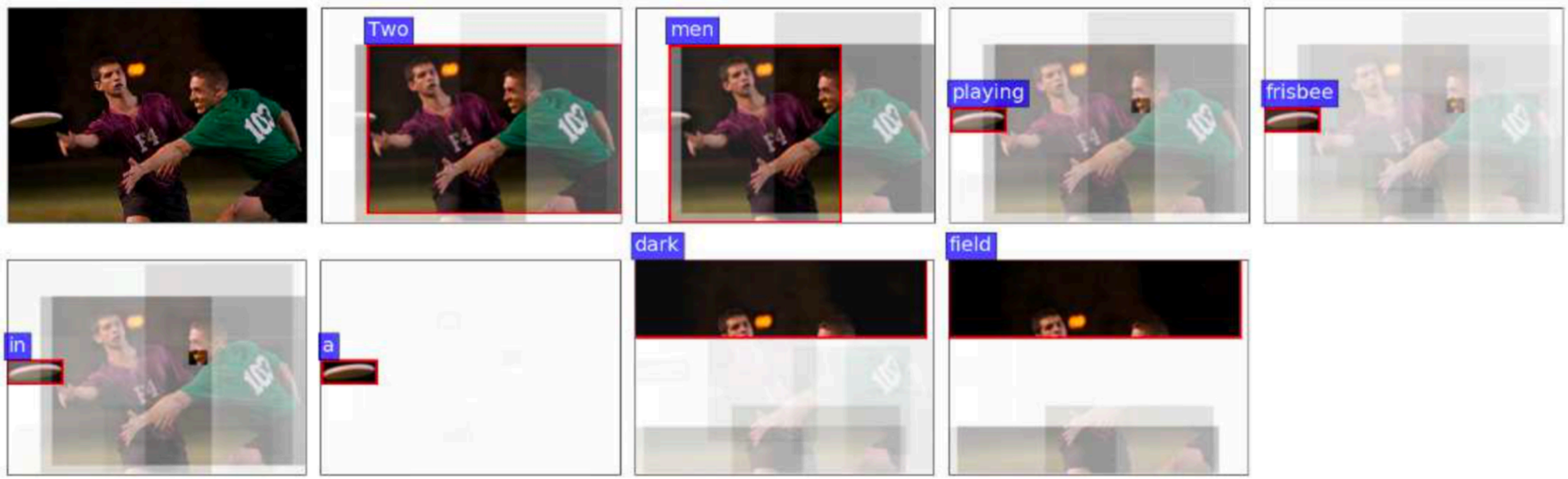
	Cross-Entropy Loss							CIDEr Optimization						
	SPICE	Objects	Attributes	Relations	Color	Count	Size	SPICE	Objects	Attributes	Relations	Color	Count	Size
Ours: ResNet	19.2	35.4	8.6	5.3	12.2	4.1	3.9	20.2	37.0	9.2	6.1	10.6	12.0	**4.3**
Ours: Up-Down	**20.3**	**37.1**	**9.2**	**5.8**	**12.7**	**6.5**	**4.5**	**21.4**	**39.1**	**10.0**	**6.5**	**11.4**	**18.4**	3.2

Evaluation(3/4)

Microsoft COCO test server

	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr		SPICE	
	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40	c5	c40
Review Net [47]	72.0	90.0	55.0	81.2	41.4	70.5	31.3	59.7	25.6	34.7	53.3	68.6	96.5	96.9	18.5	64.9
Adaptive [27]	74.8	92.0	58.4	84.5	44.4	74.4	33.6	63.7	26.4	35.9	55.0	70.5	104.2	105.9	19.7	67.3
PG-BCMR [24]	75.4	-	59.1	-	44.5	-	33.2	-	25.7	-	55	-	101.3	-	-	-
SCST:Att2all [33]	78.1	93.7	61.9	86.0	47.0	75.9	35.2	64.5	27.0	35.5	56.3	70.7	114.7	116.7	20.7	68.9
LSTM-A_3 [48]	78.7	93.7	62.7	86.7	47.6	76.5	35.6	65.2	27	35.4	56.4	70.5	116	118	-	-
Ours: Up-Down	**80.2**	**95.2**	**64.1**	**88.8**	**49.1**	**79.4**	**36.9**	**68.5**	**27.6**	**36.7**	**57.1**	**72.4**	**117.9**	**120.5**	**21.5**	**71.5**

$$\boldsymbol{\alpha}_t = \text{softmax}(\alpha_t), \alpha_t = \{\alpha_{1,t}, \alpha_{2,t}, \ldots, \alpha_{k,t}\}$$



Two men playing frisbee in a dark field.

Evaluation(4/4)

VQA v2.0 validation

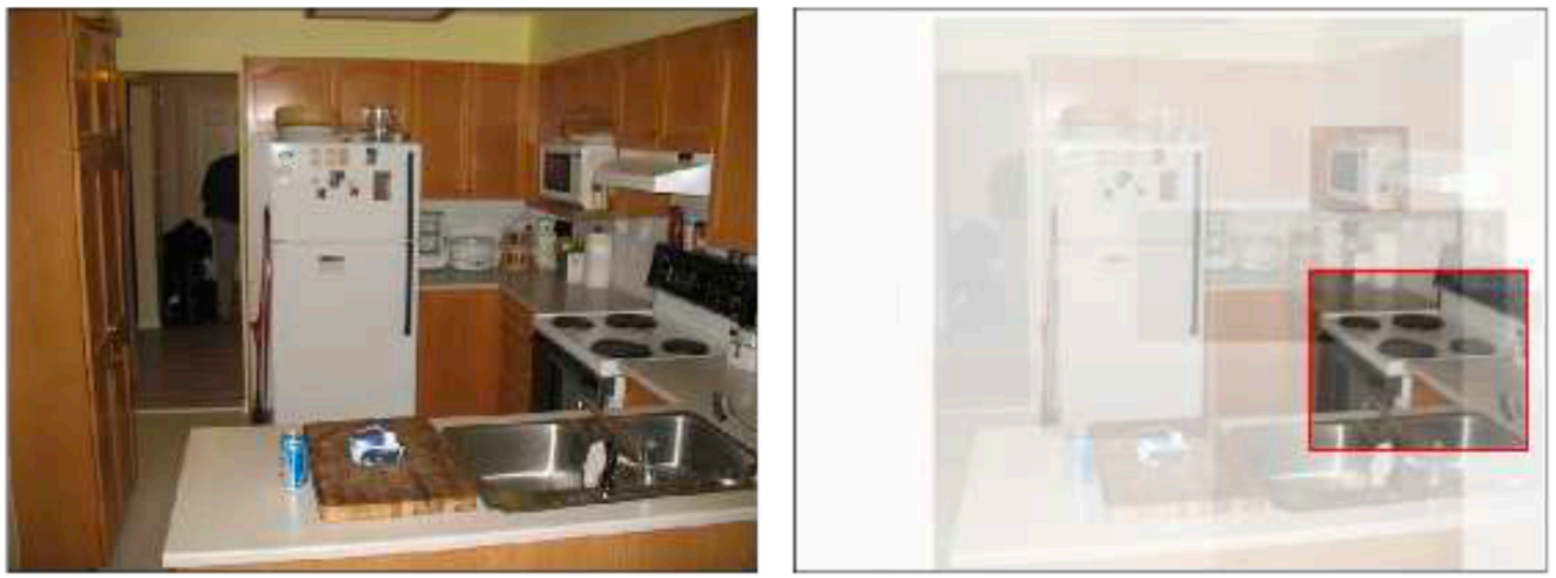
	Yes/No	Number	Other	Overall
Ours: ResNet (1×1)	76.0	36.5	46.8	56.3
Ours: ResNet (14×14)	76.6	36.2	49.5	57.9
Ours: ResNet (7×7)	77.6	37.7	51.5	59.4
Ours: Up-Down	**80.3**	**42.8**	**55.8**	**63.2**
Relative Improvement	3%	14%	8%	6%

Table 4. Single-model performance on the VQA v2.0 validation set. The use of bottom-up attention in the Up-Down model provides a significant improvement over the best ResNet baseline across all question types, even though the ResNet baselines use almost twice as many convolutional layers.

VQA v2.0 test server

	Yes/No	Number	Other	Overall
d-LSTM+n-I [26, 12]	73.46	35.18	41.83	54.22
MCB [11, 12]	78.82	38.28	53.36	62.27
UPMC-LIP6	82.07	41.06	57.12	65.71
Athena	82.50	44.19	59.97	67.59
HDU-USYD-UNCC	84.50	45.39	59.01	68.09
Ours: Up-Down	**86.60**	**48.64**	**61.15**	**70.34**

Table 5. VQA v2.0 test-standard server accuracy as at 8 August 2017, ranking our submission against published and unpublished work for each question type. Our approach, an ensemble of 30 models, outperforms all other leaderboard entries.



Question: What room are they in? Answer: kitchen

Conclusion

- This Faster R-CNN based Bottom-Up attention model can be considered to be a '**hard attention**' and attend to more salient regions.

- The number of attributes in each image is 36(k), it is increasable.

$$V = \{v_1, v_2, \ldots, v_k\},\ v_i \in R^{2048}$$

- The image feature from object detection can be used to apply on other 'image - text' task, like video caption.