Attention Models

Jaegul Choo (주재걸)
Korea University
https://sites.google.com/site/jaegulchoo/\
Slide made by Cheonbok Park

Contents

Image Captioning

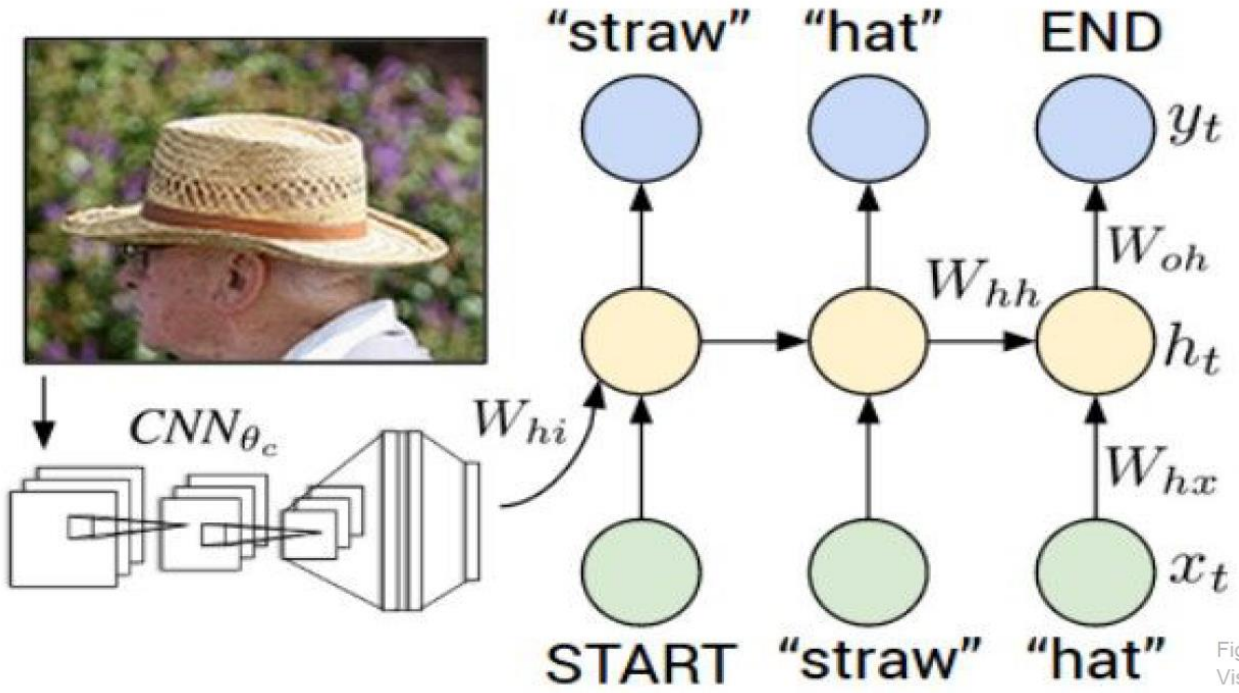
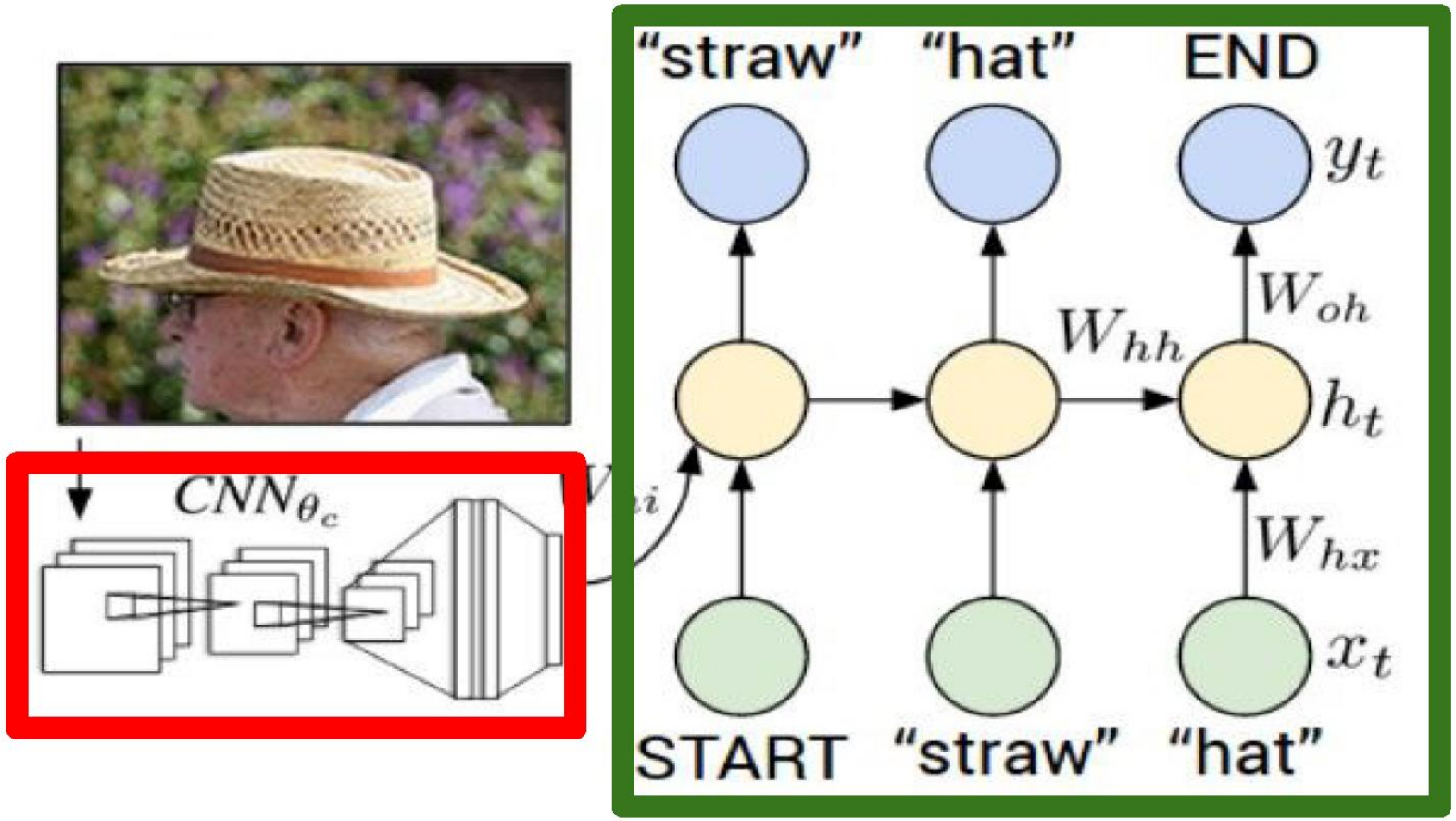


Figure from Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015.
Reproduced for educational purposes.

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Recurrent Neural Network



Convolutional Neural Network

image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096
FC-1000
softmax

test image

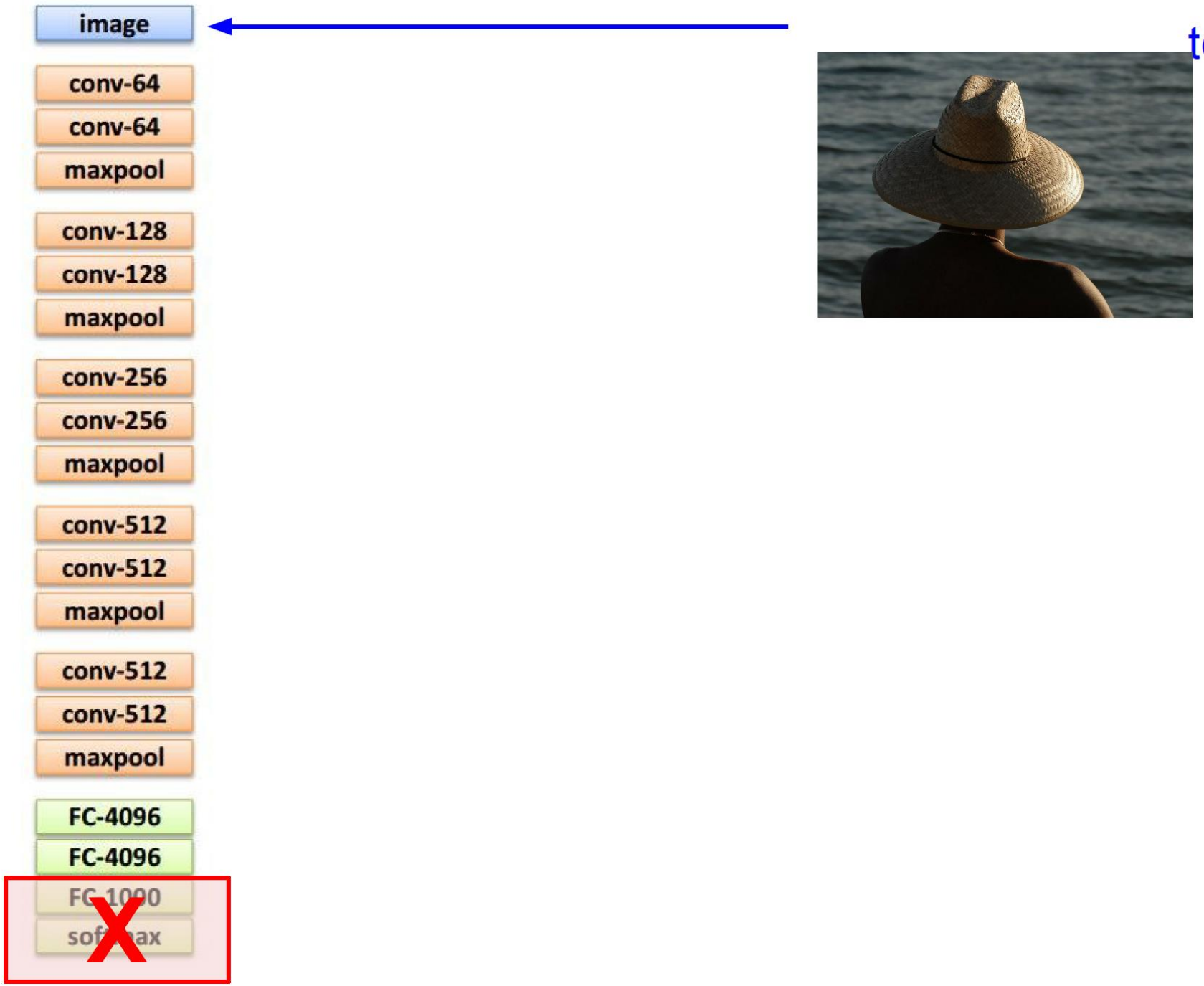


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
test image

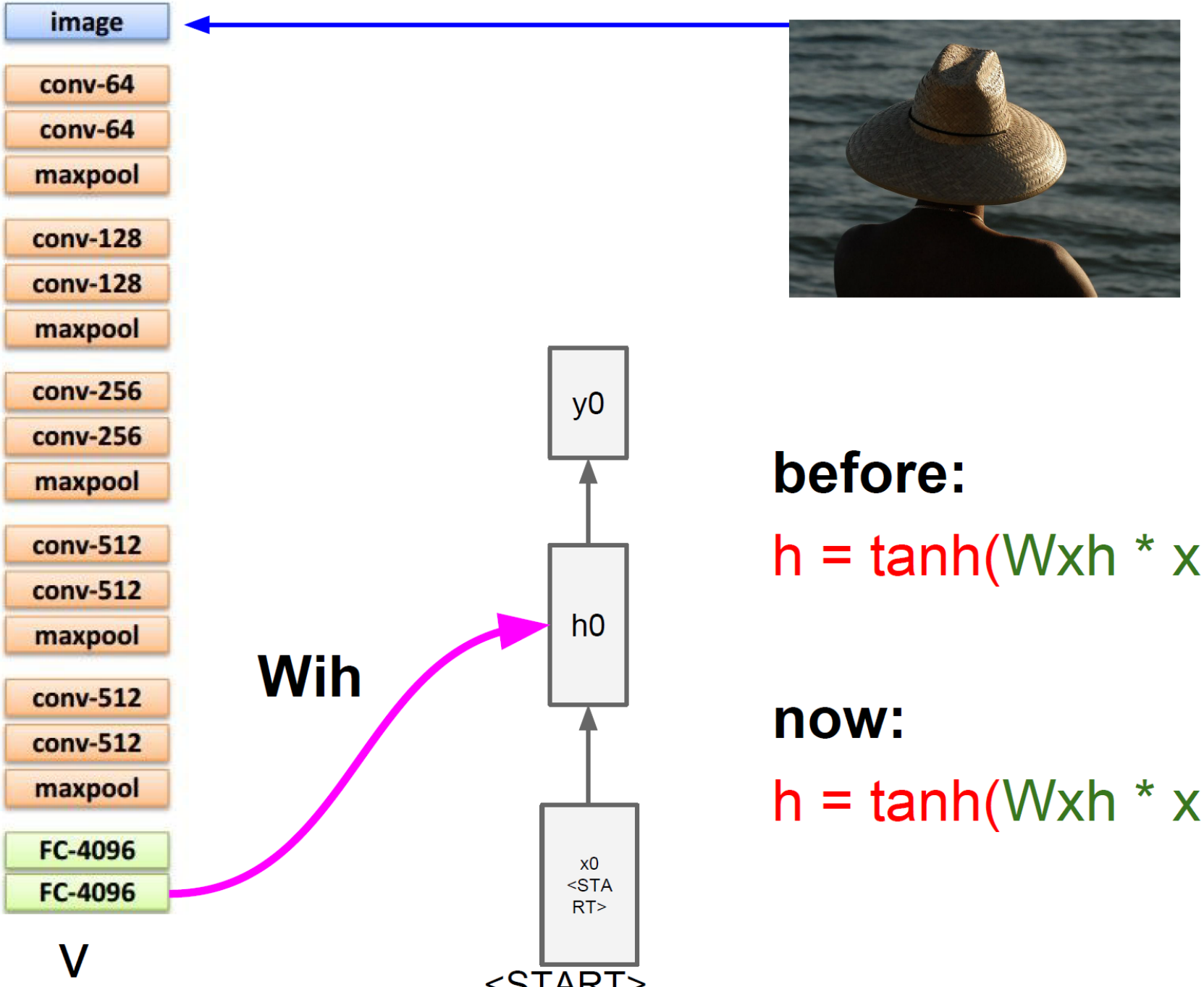


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
V
Wih
y0
h0
x0
<STA
RT>
<START>
test image
before:
h = tanh(Wxh * x + Whh * h)
now:
h = tanh(Wxh * x + Whh * h + Wih * v)

Image Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0 Public domain:
cat suitcase, cat tree, dog, bear,
surfers, tennis, giraffe, motorcycle

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

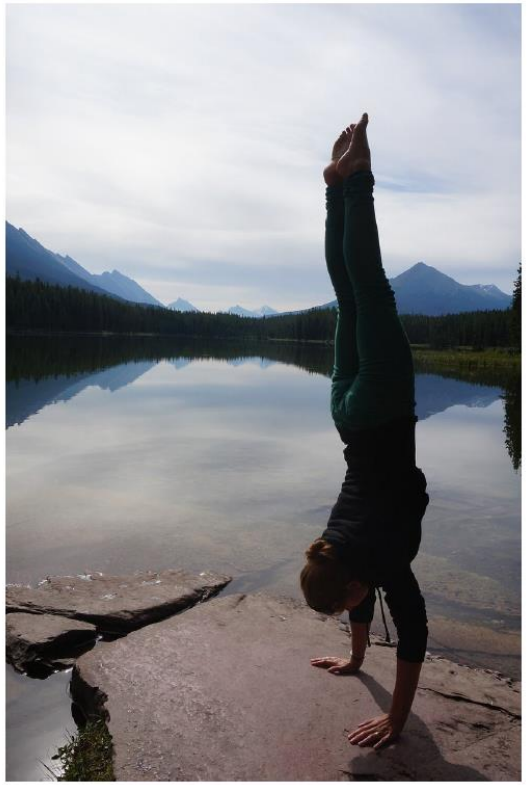
A man riding a dirt bike on a dirt track

Image Captioning: Failure Cases

Captions generated using neuraltalk2
All images are CC0 Public domain: fur coat, handstand, spider web, baseball

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



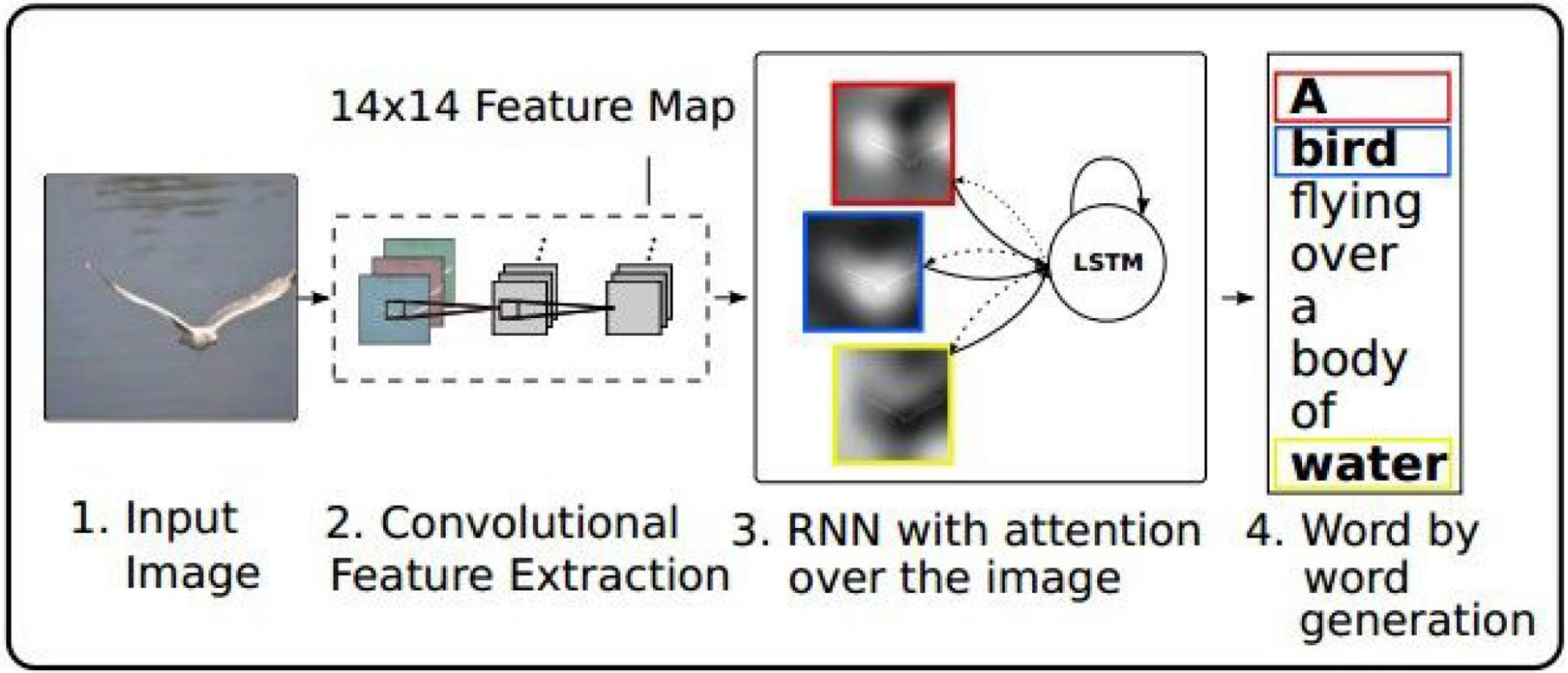
A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Image Captioning with Attention

RNN focuses its attention at a different spatial location when generating each word



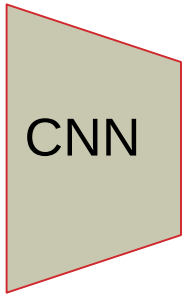
Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning

Image:
H x W x 3

Image Captioning

Image:
H x W x 3



Features:
D

Image Captioning

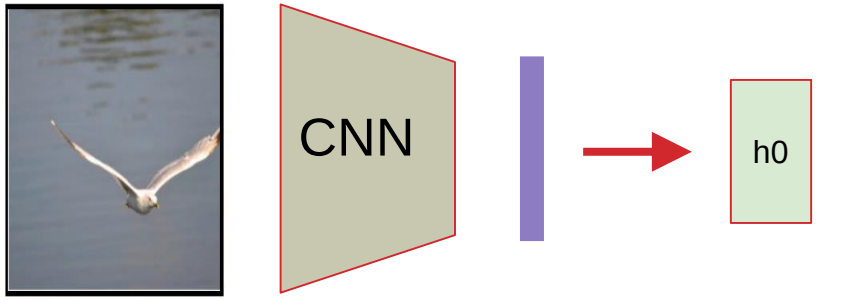


Image:
H x W x 3

Features:
D

Hidden
state: H

Image Captioning

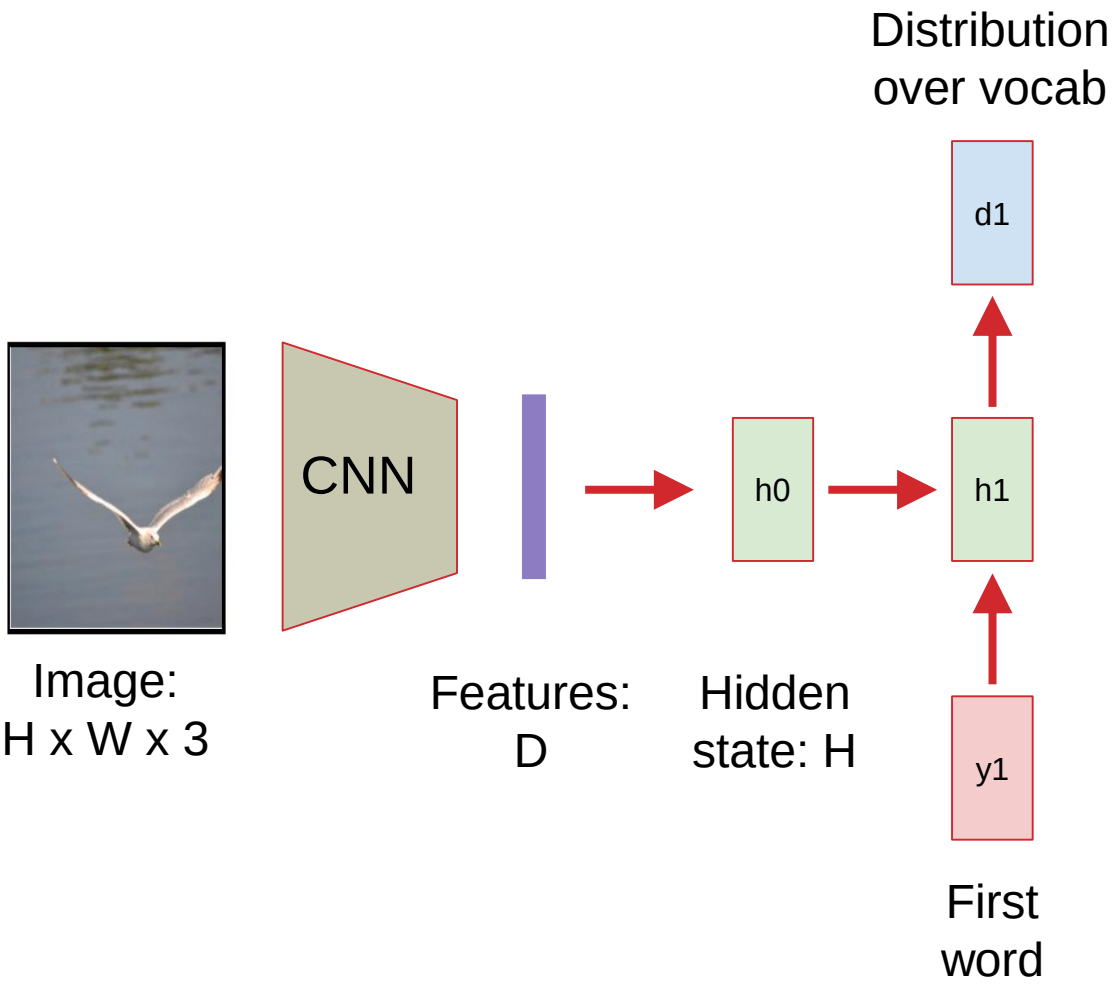


Image Captioning

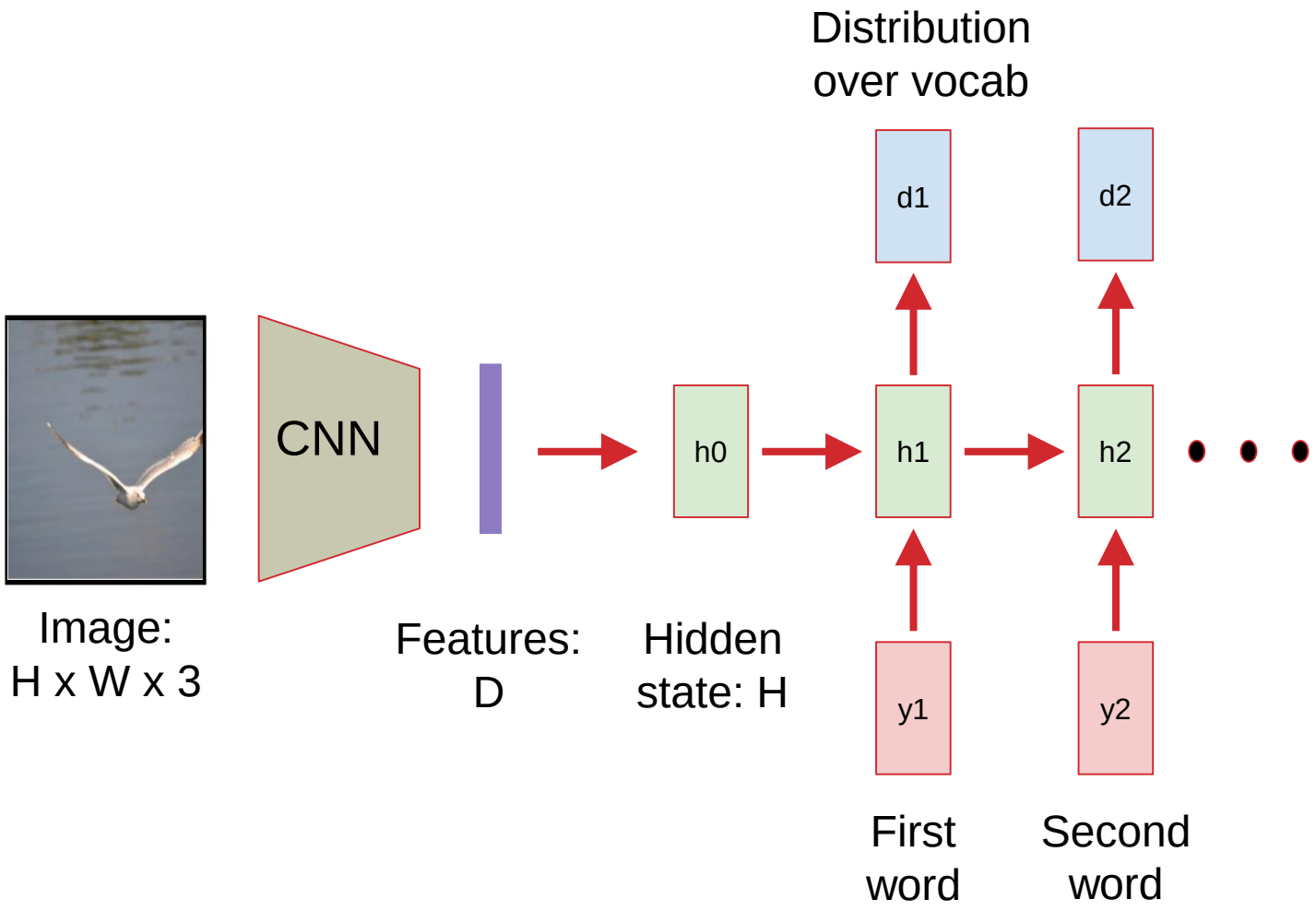


Image Captioning

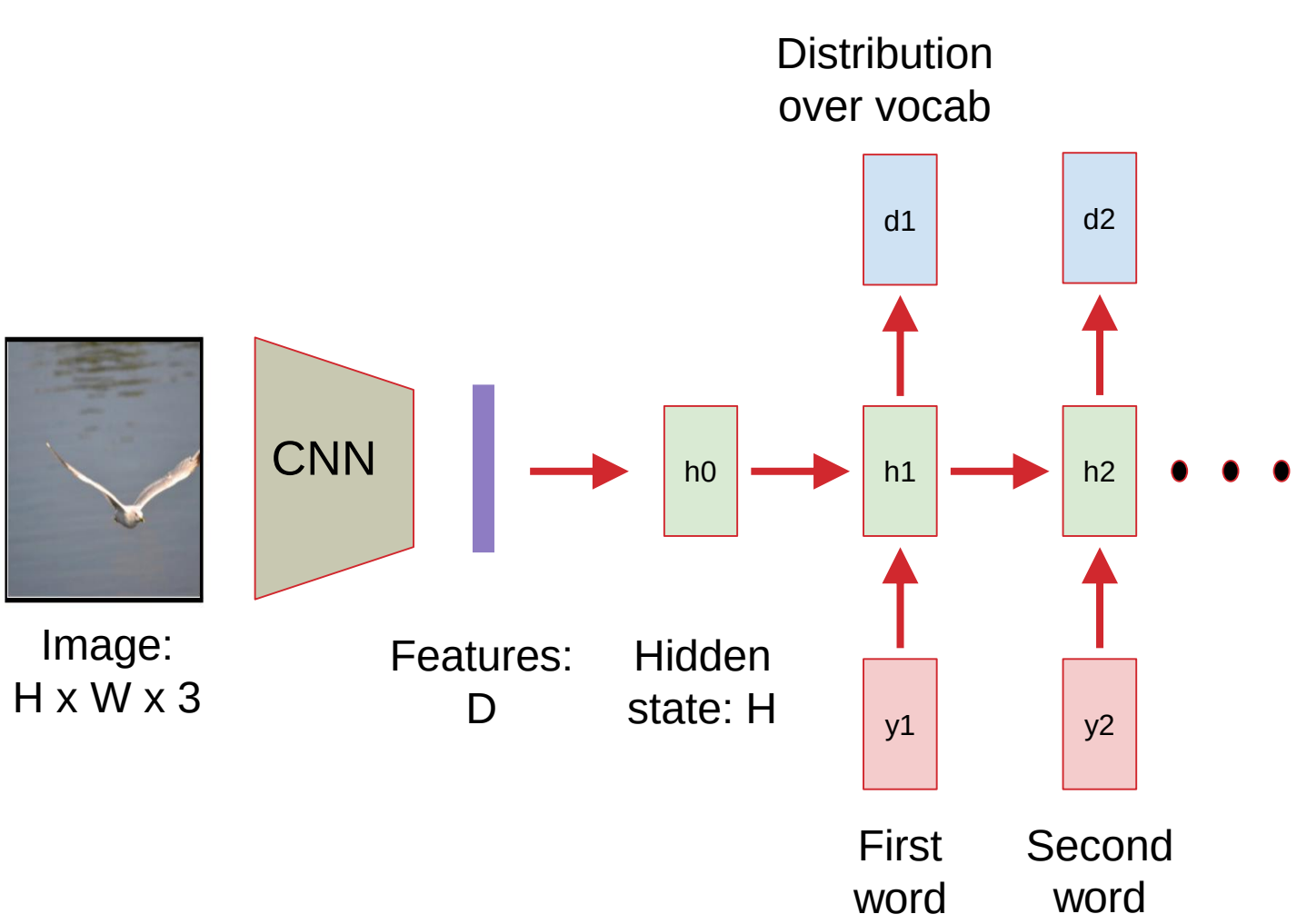


Image Captioning

Image:
H x W x 3

CNN

Features:
D

h0

Distribution
over vocab

d1

h1

y1

First
word

d2

h2

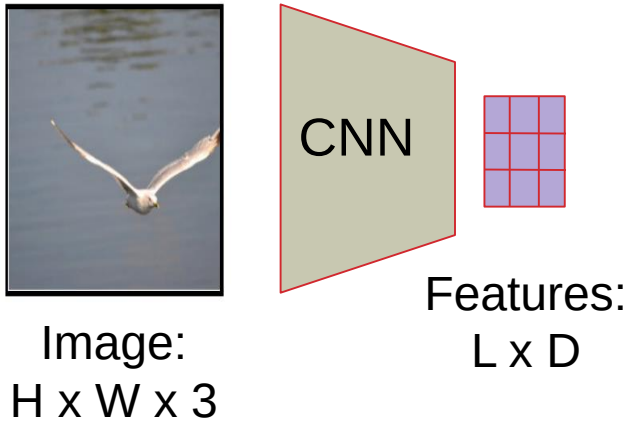
y2

Second
word

RNN only looks
at whole image,
once

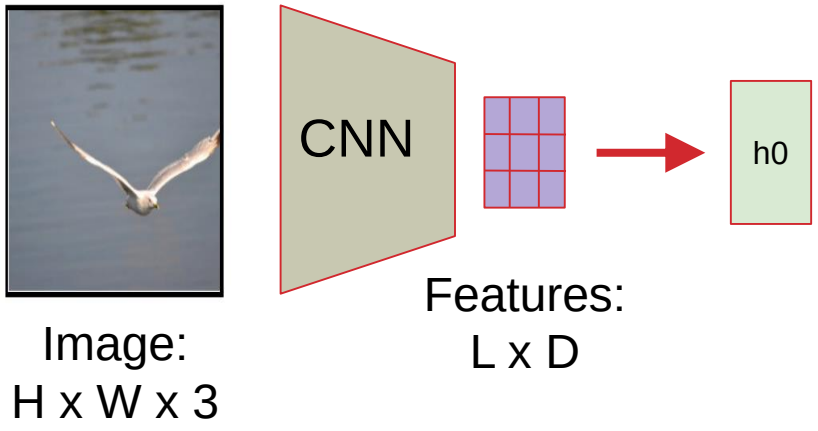
What if the
RNN looks at
different parts
of the image at
each timestep?

Image Captioning with Attention



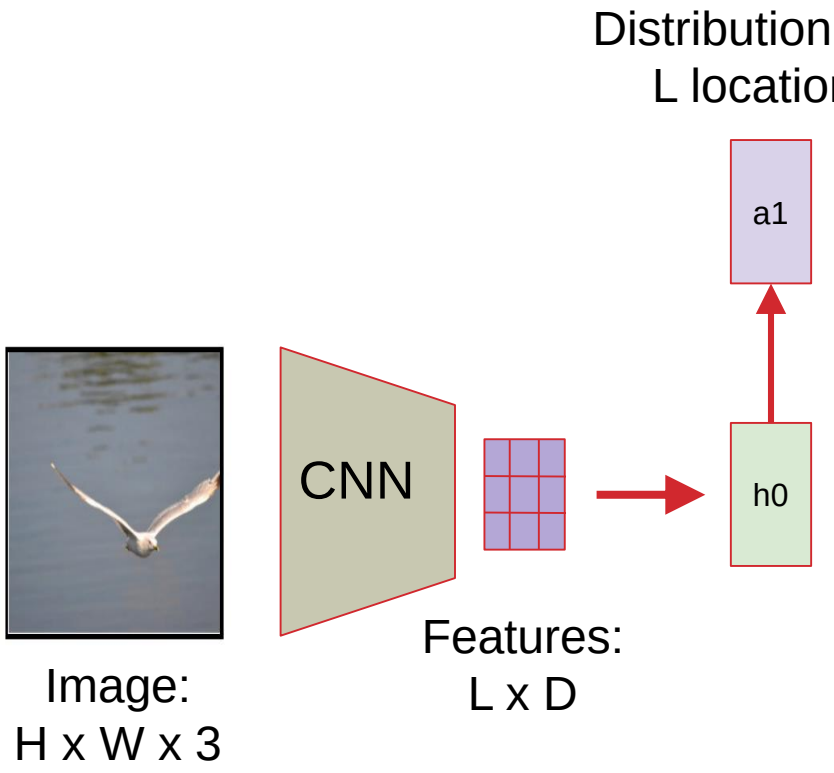
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image Captioning with Attention



Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image Captioning with Attention



Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image Captioning with Attention

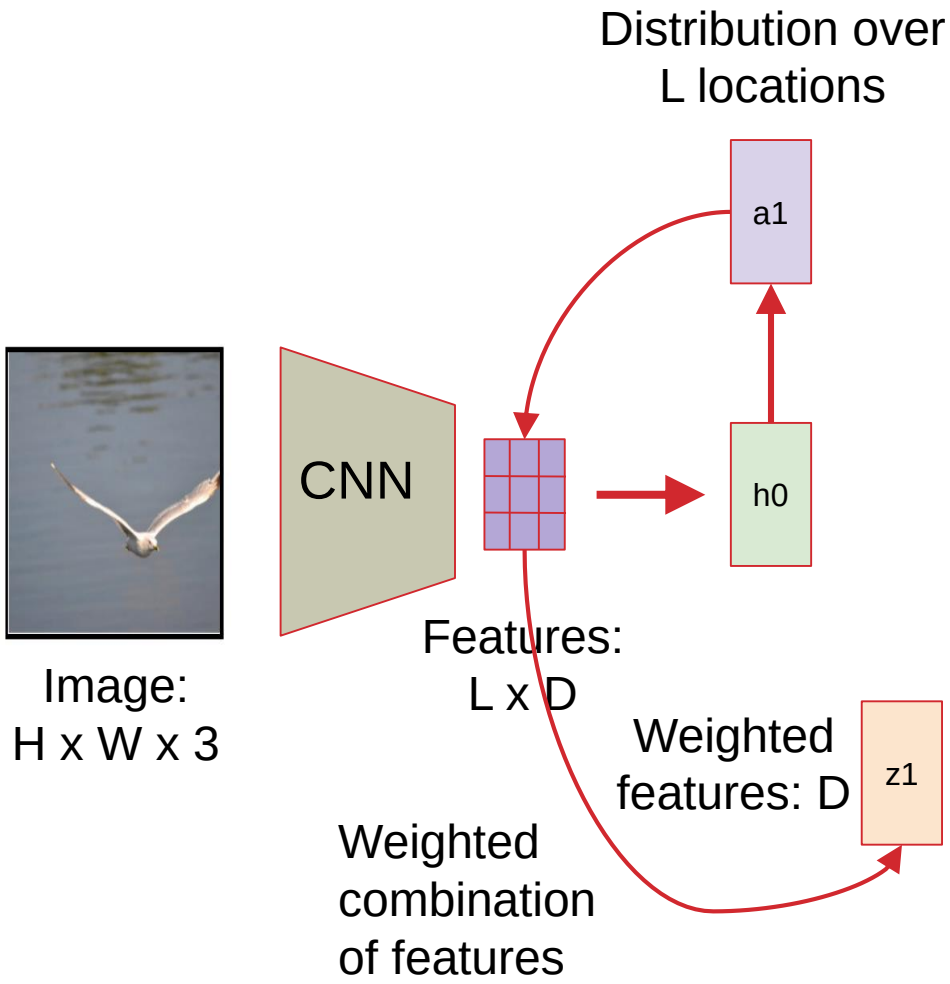


Image Captioning with Attention

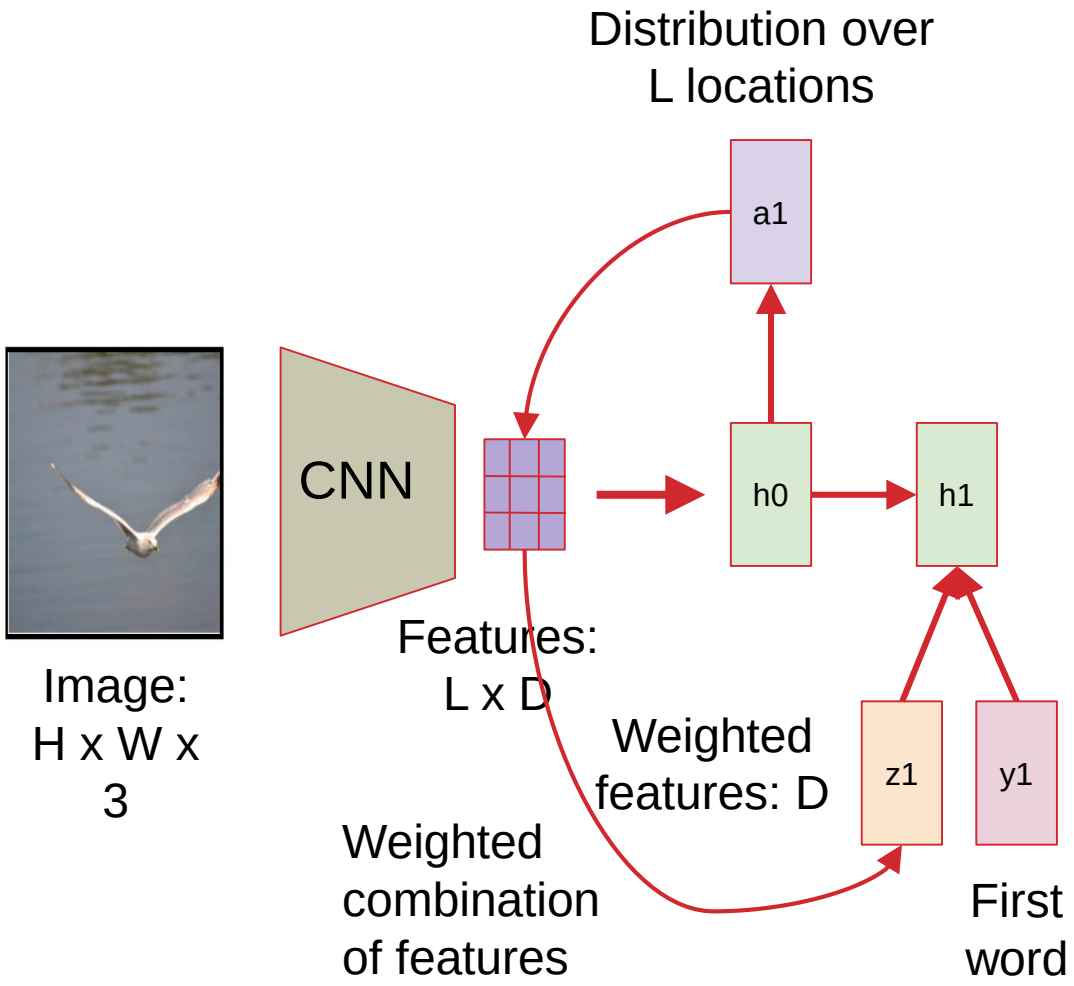


Image Captioning with Attention

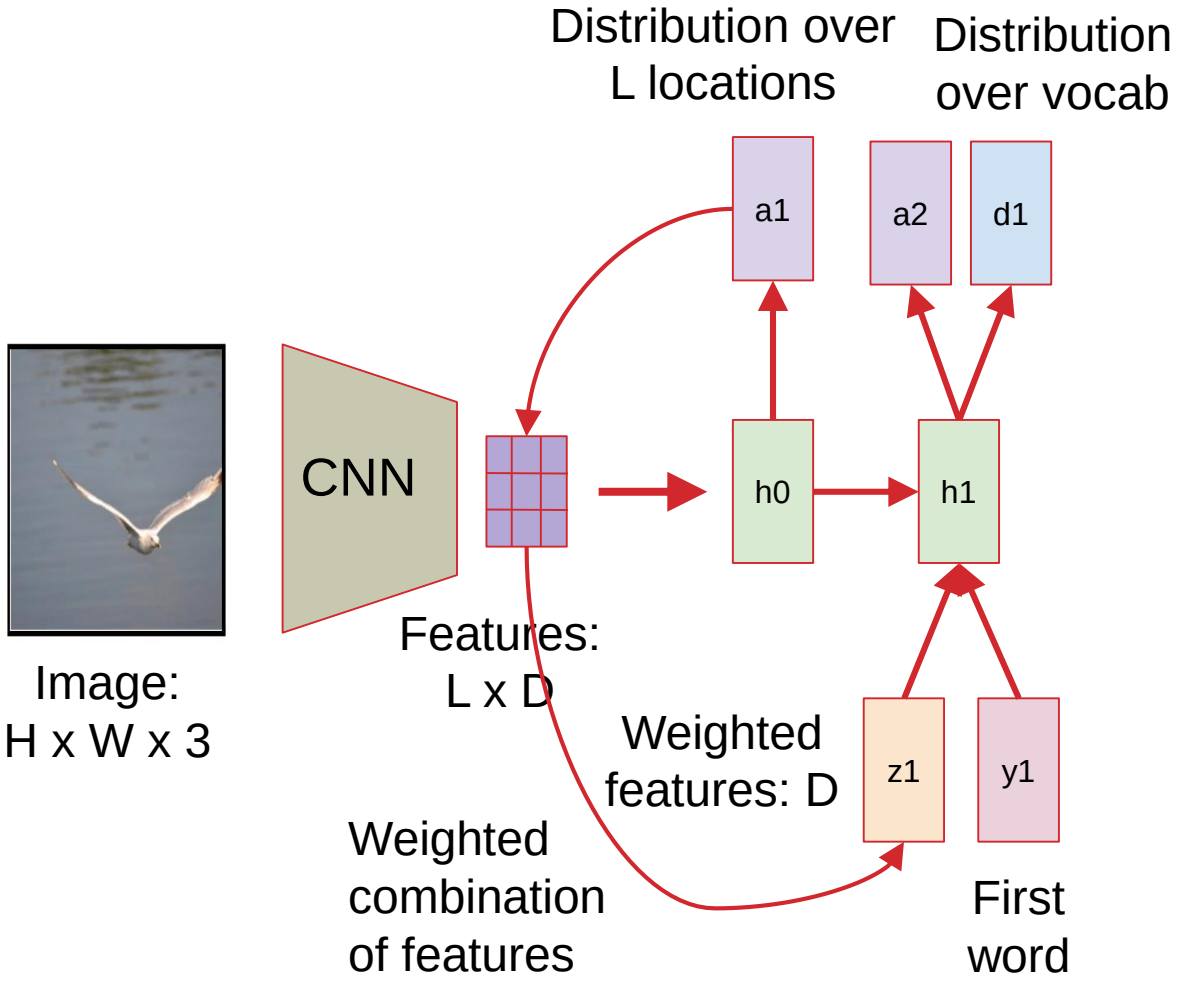


Image Captioning with Attention

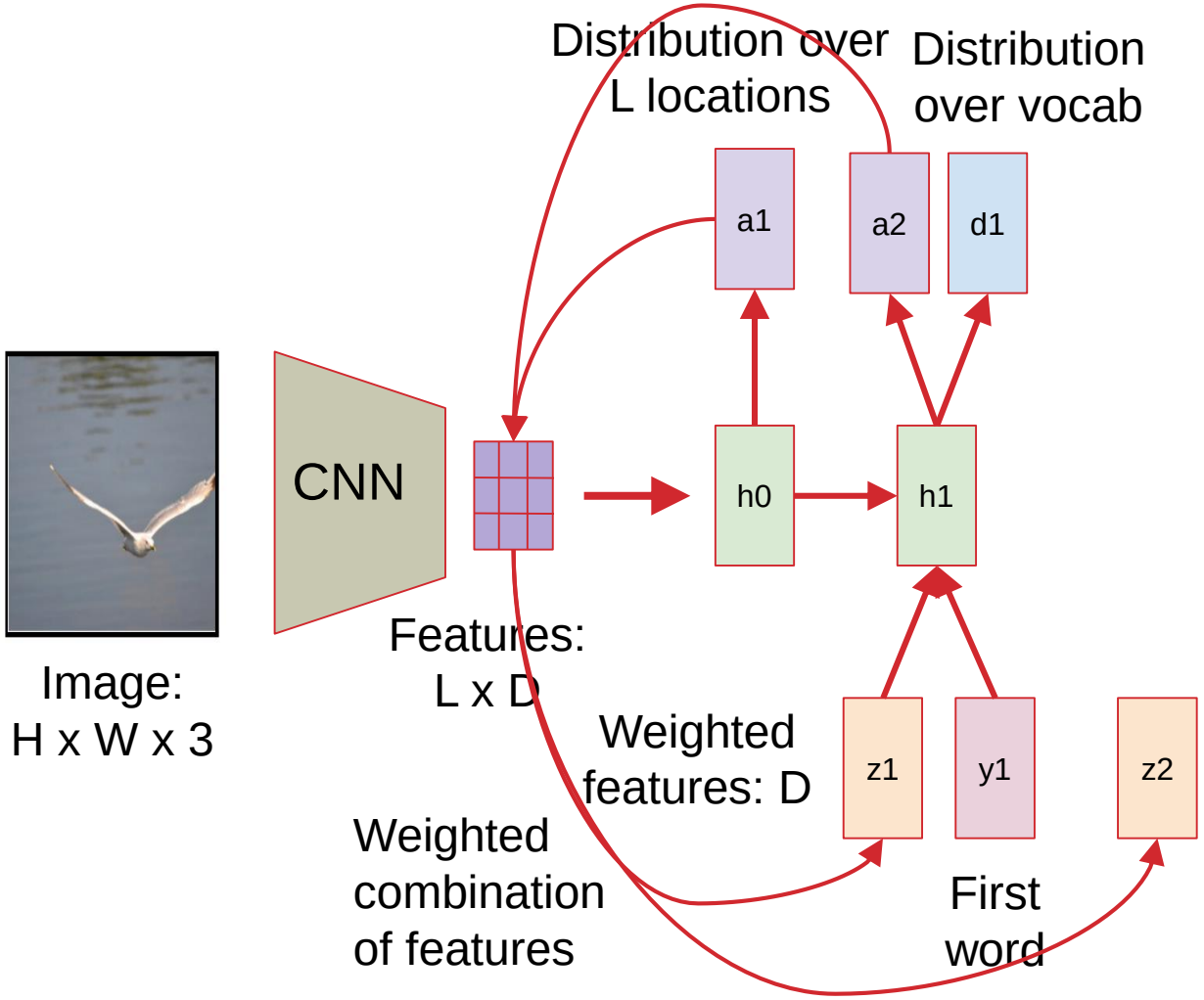


Image Captioning with Attention

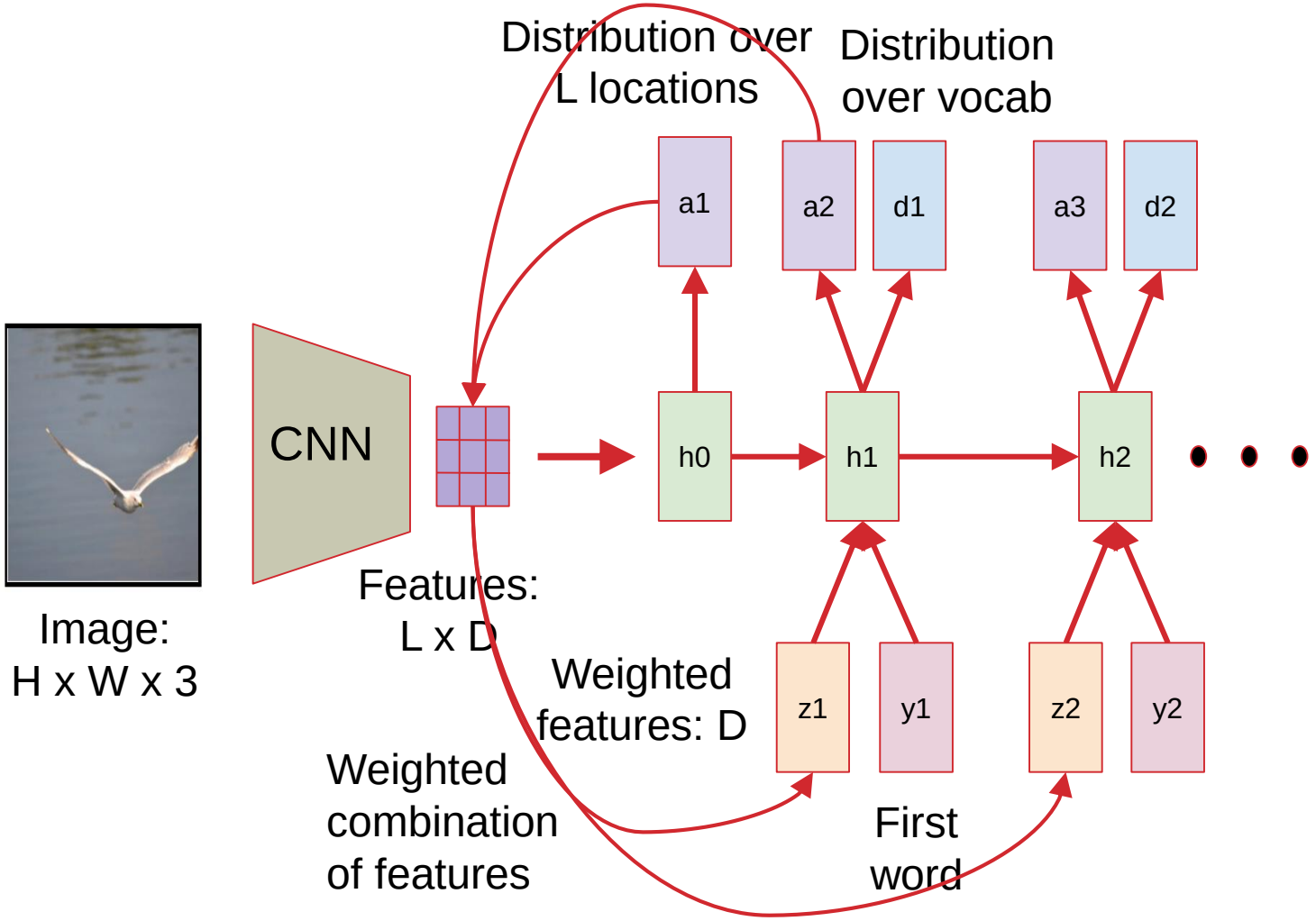


Image Captioning with Attention

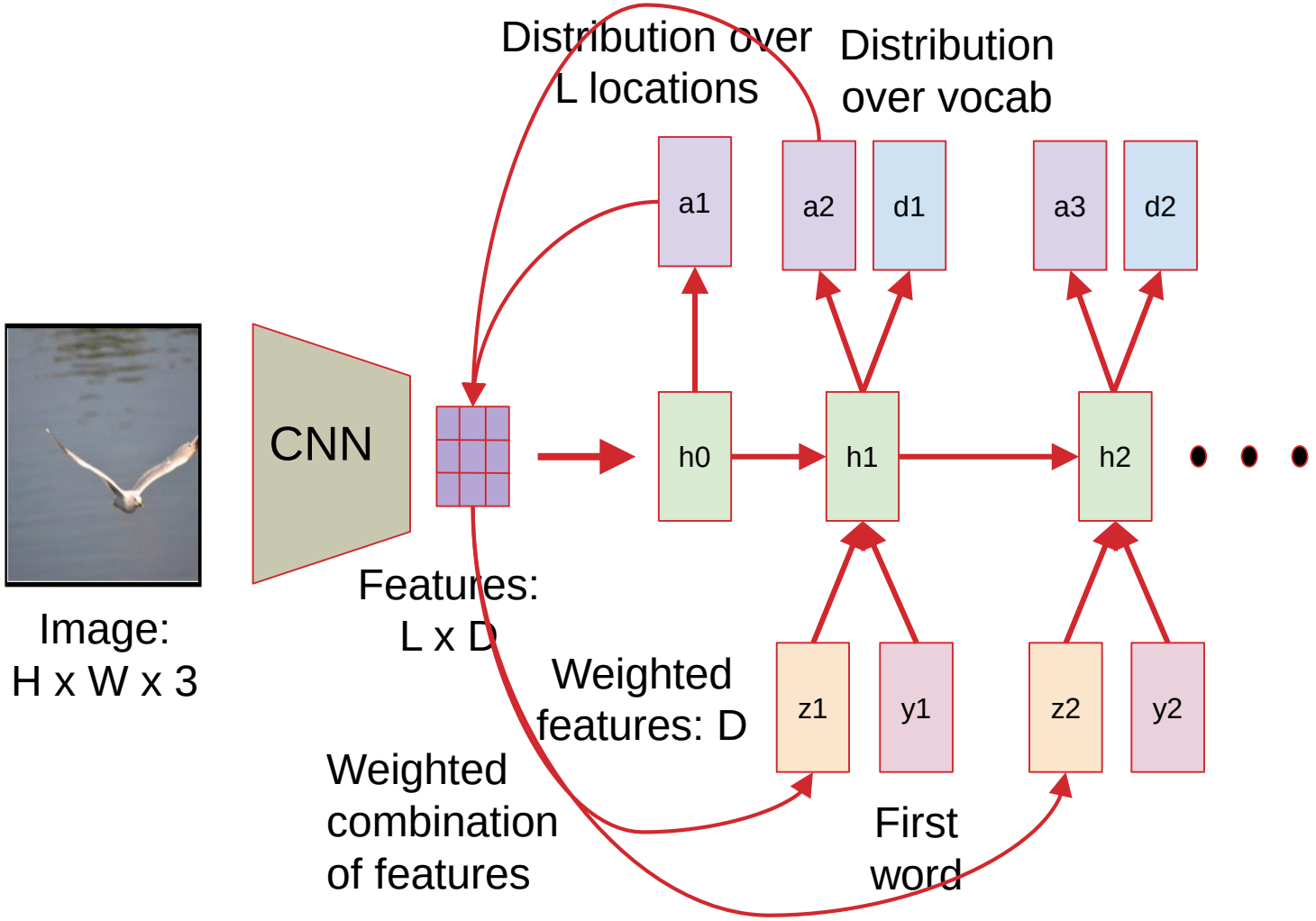


Image Captioning with Attention

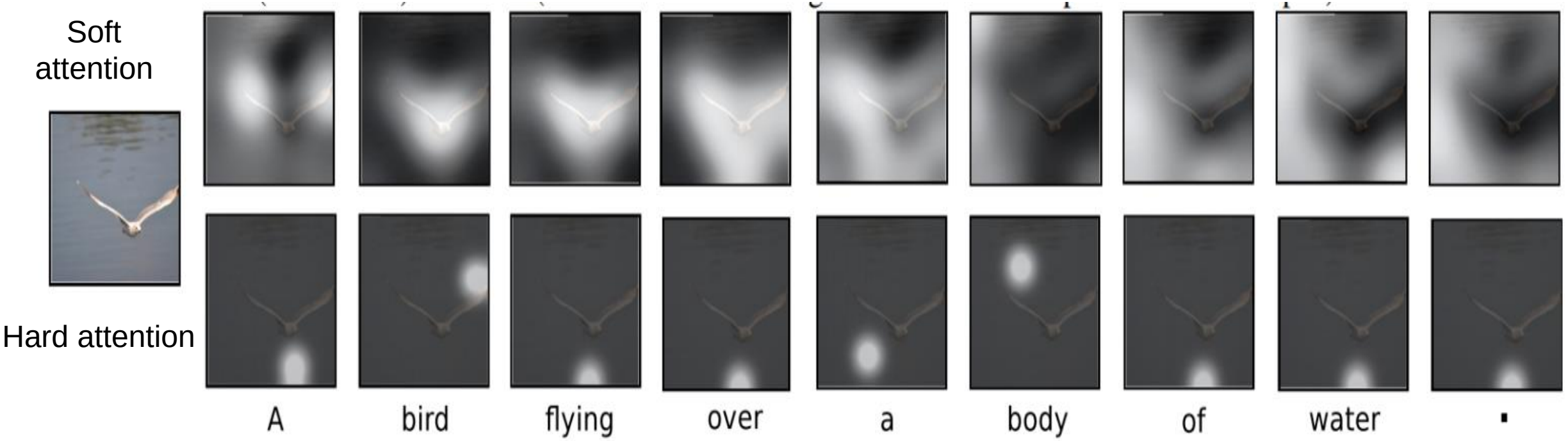


Image Captioning with Attention

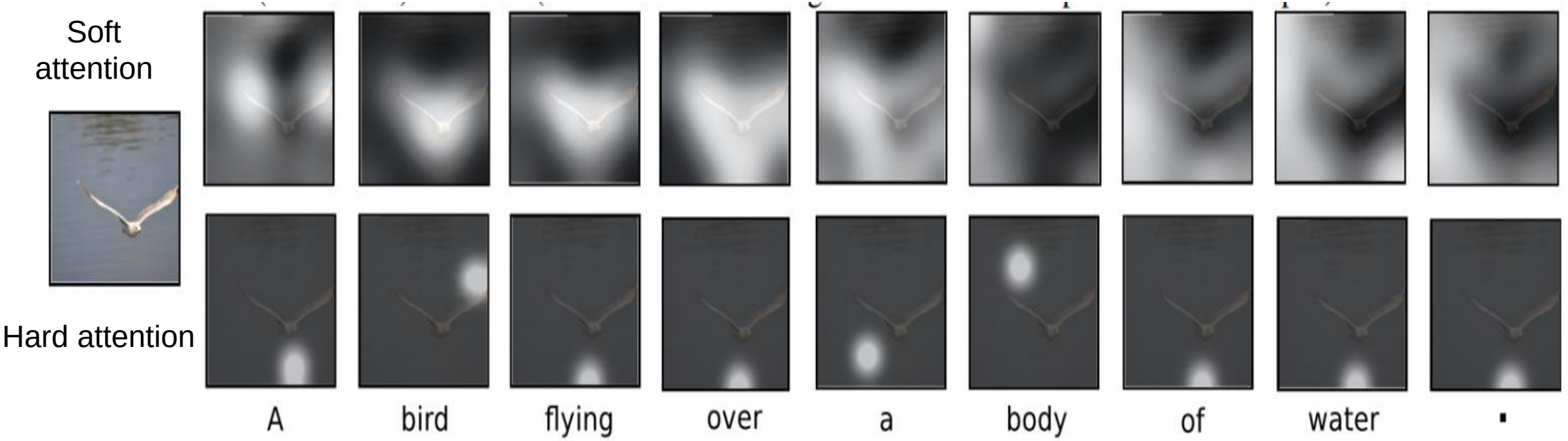


Image Captioning with Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

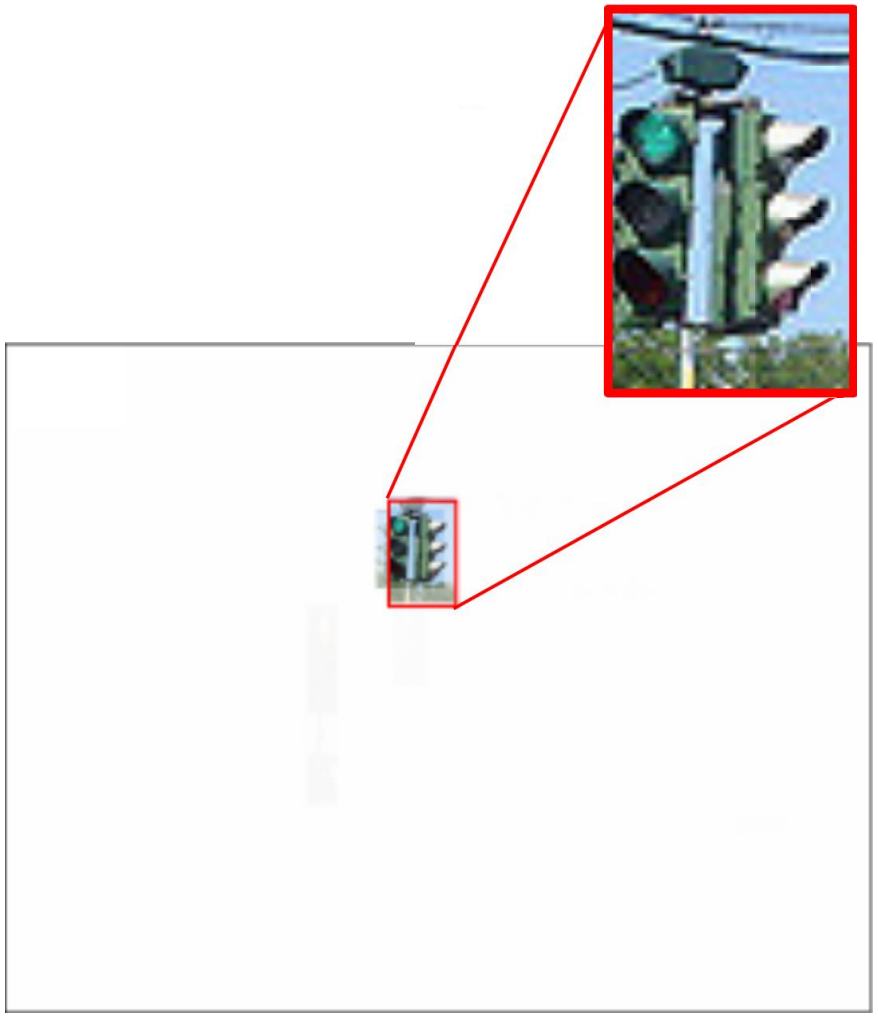
A giraffe standing in a forest with trees in the background.

RECAP, Visual Attention

- Vision and language tasks often require fine grained visual processing, e.g. VQA:

Q: What color is illuminated on the traffic light ?

A: green



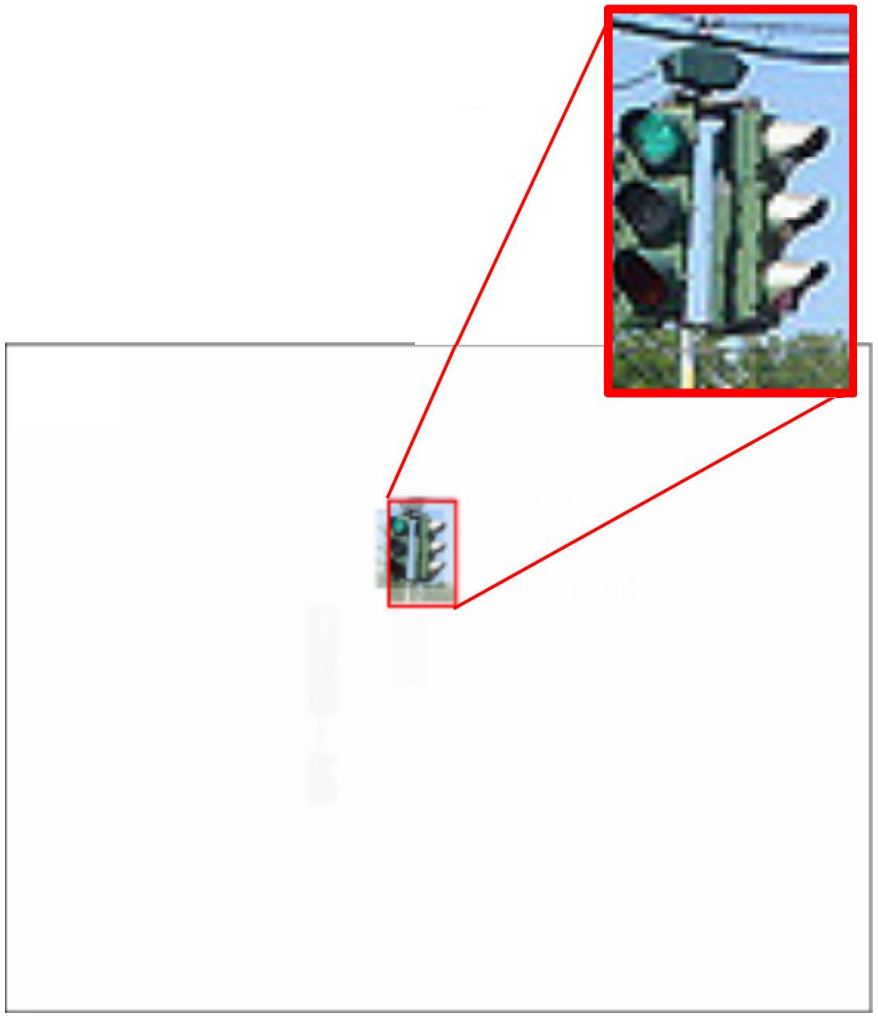
https://panderson.me/images/CVPR-Up-Down-talk.pdf

RECAP, Visual Attention

- Visual attention mechanisms learn to focus on image regions that are relevant to the task

Q: What color is illuminated on the traffic light ?

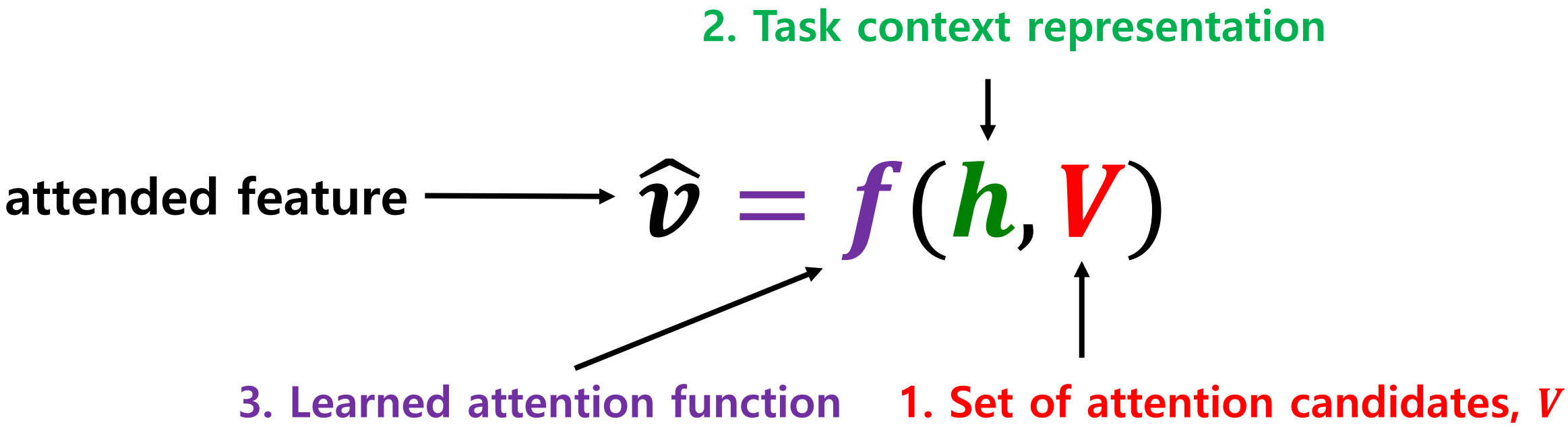
A: green



https://panderson.me/images/CVPR-Up-Down-talk.pdf

Components of visual attention

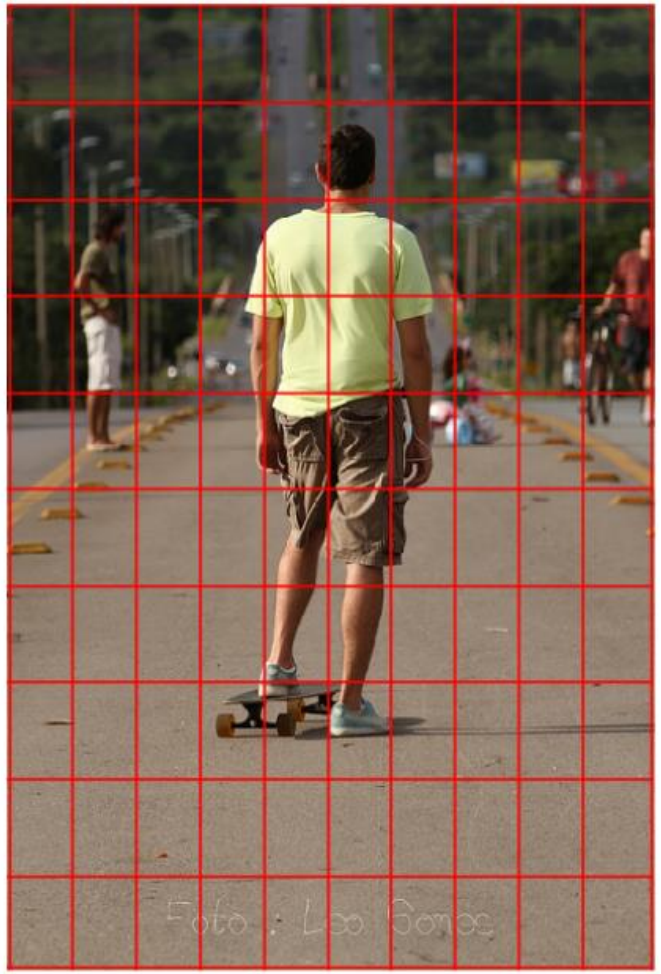
- Visual attention mechanisms **learn to focus** on **image regions** that are **relevant to the task**



https://panderson.me/images/CVPR-Up-Down-talk.pdf

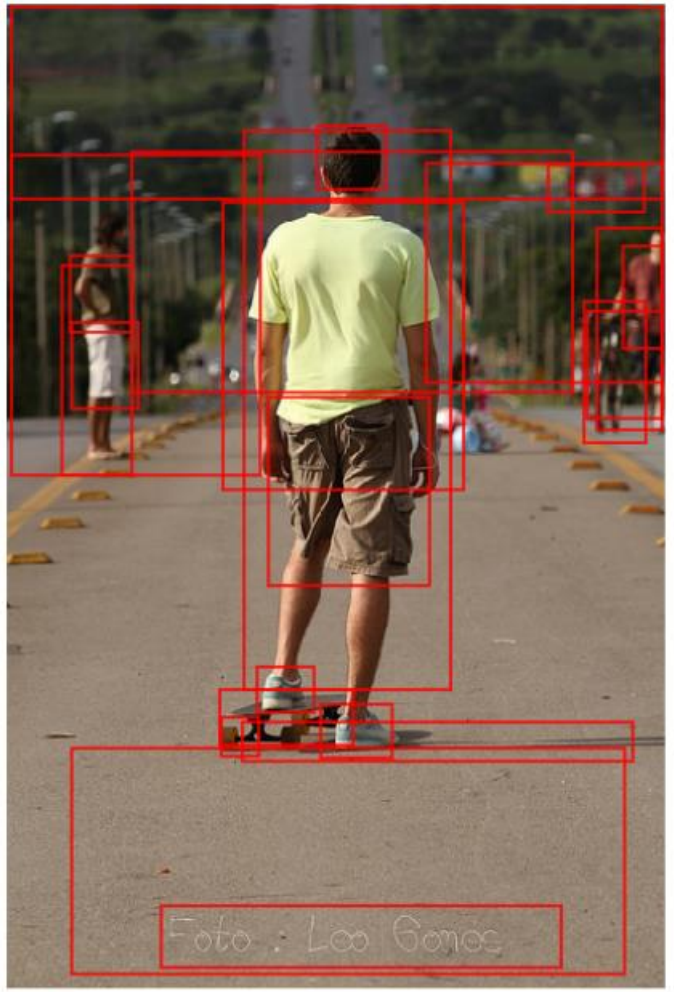
Attention Candidates , V

10



10

Standard approach:
Spatial output of a CNN



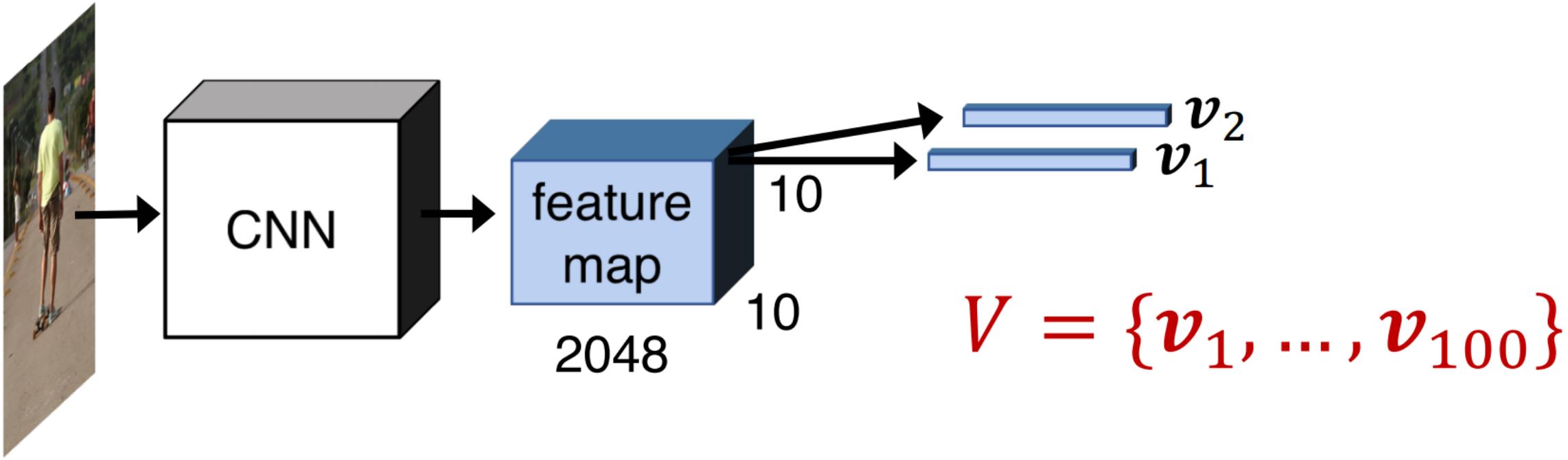
k **regions**

Bottom-up attention approach:
Object-based attention

https://panderson.me/images/CVPR-Up-Down-talk.pdf

Attention Candidates , V

Standard approach: use the spatial output of a CNN to extract vectors for each position in a grid



https://panderson.me/images/CVPR-Up-Down-talk.pdf

Objects are a natural basis for attention

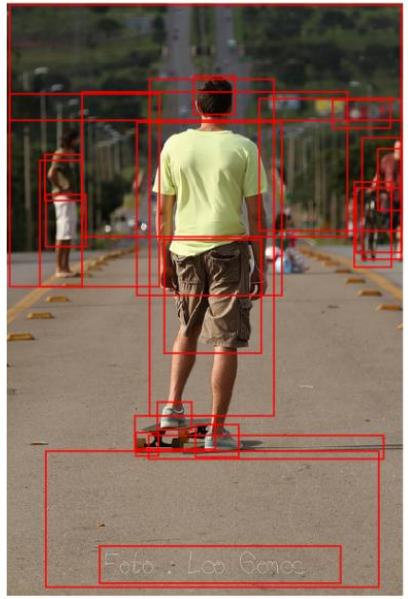
- **Human visual attention can select discrete objects, not just spatial regions**
- **Image captioning and VQA are concerned with objects**

A young man on a skateboard looking down street with people watching.

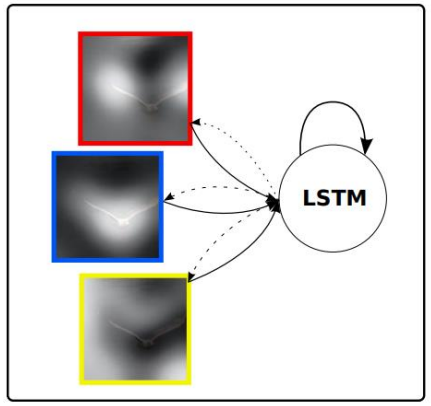
Q: Is the boy in the yellow shirt wearing head protective gear? **A:** No

Bottom-up and top-down attention

V



Bottom-up process: Extract all objects and other salient regions from the image (independent of the question / partially-completed caption)



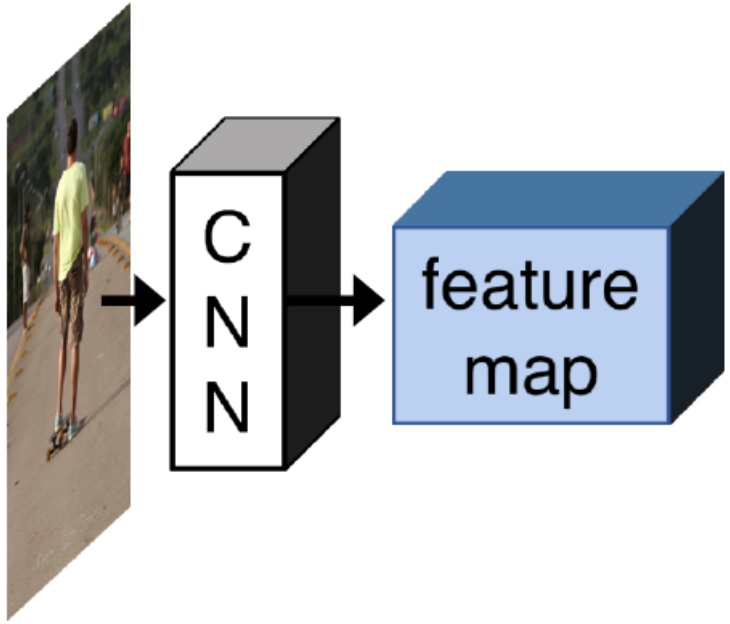
3. RNN with attention over the image

Top-down process: Given task context, weight the attention candidates (i.e., use existing VQA / captioning models)

https://panderson.me/images/CVPR-Up-Down-talk.pdf

Attention Candidates , V

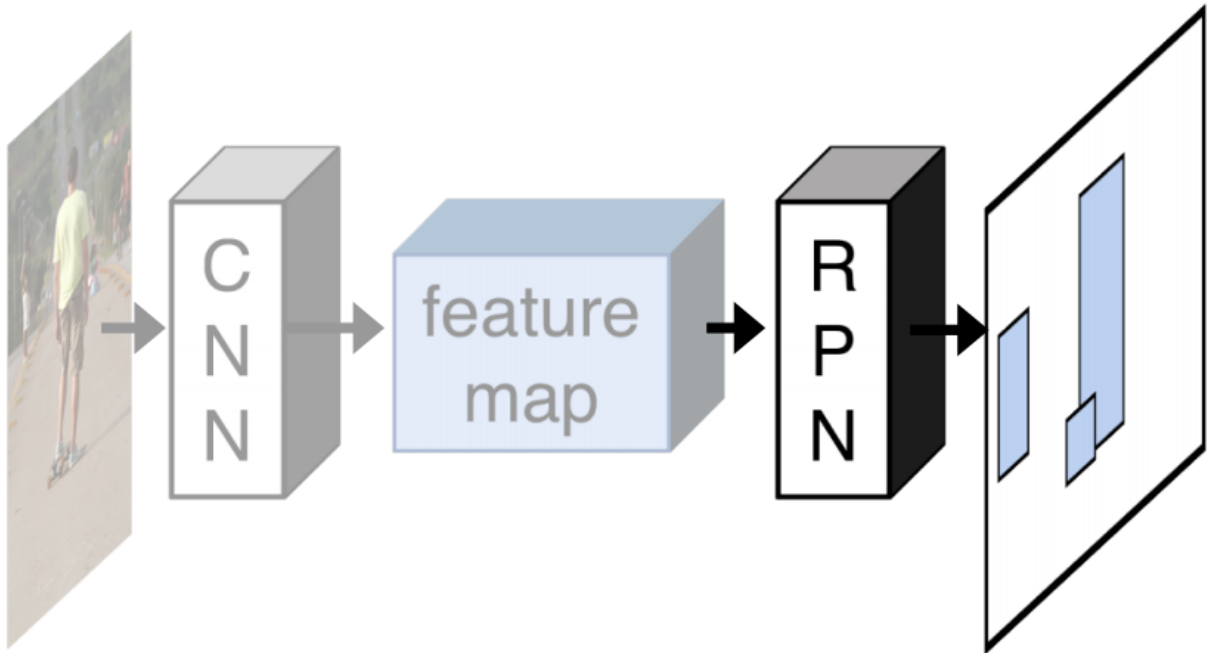
Bottom-up attention (using Faster R-CNN[2])



https://panderson.me/images/CVPR-Up-Down-talk.pdf

Attention Candidates , V

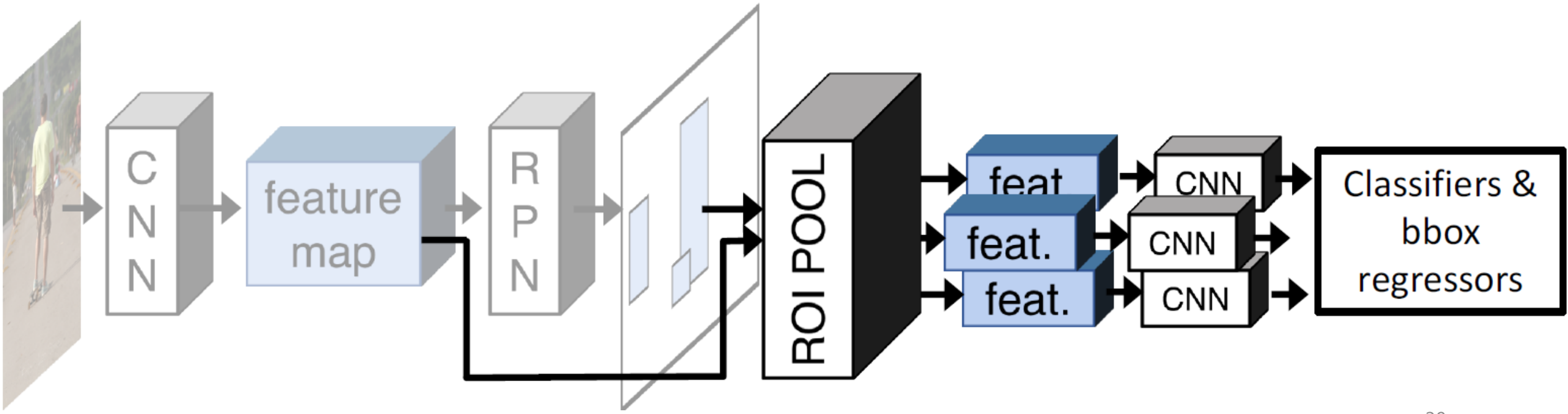
Bottom-up attention (using Faster R-CNN[2])



https://panderson.me/images/CVPR-Up-Down-talk.pdf

Attention Candidates , V

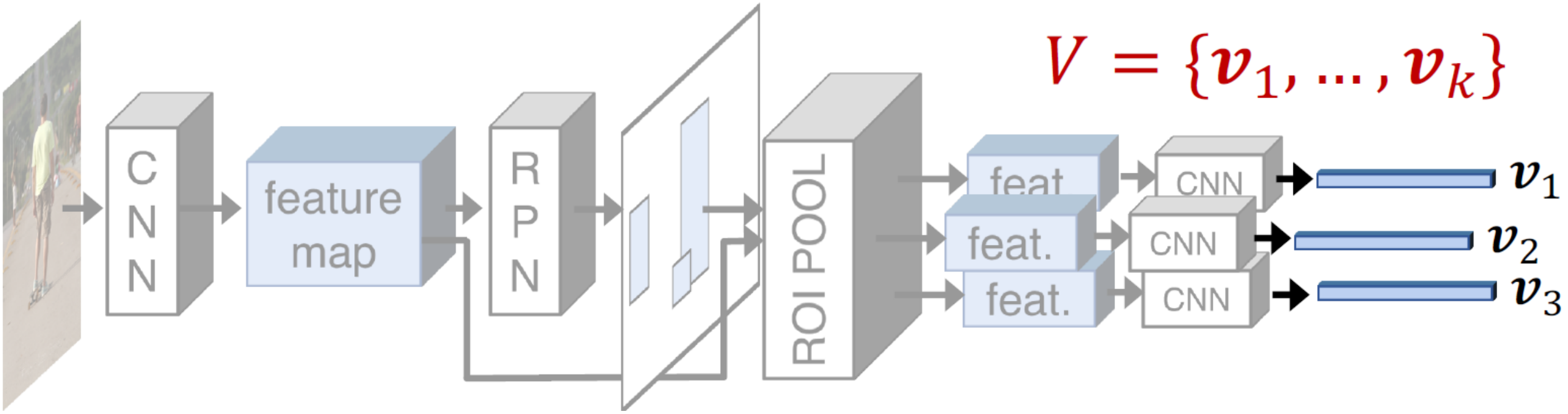
Bottom-up attention (using Faster R-CNN[2])



https://panderson.me/images/CVPR-Up-Down-talk.pdf

Attention Candidates , V

Bottom-up attention (using Faster R-CNN[2])



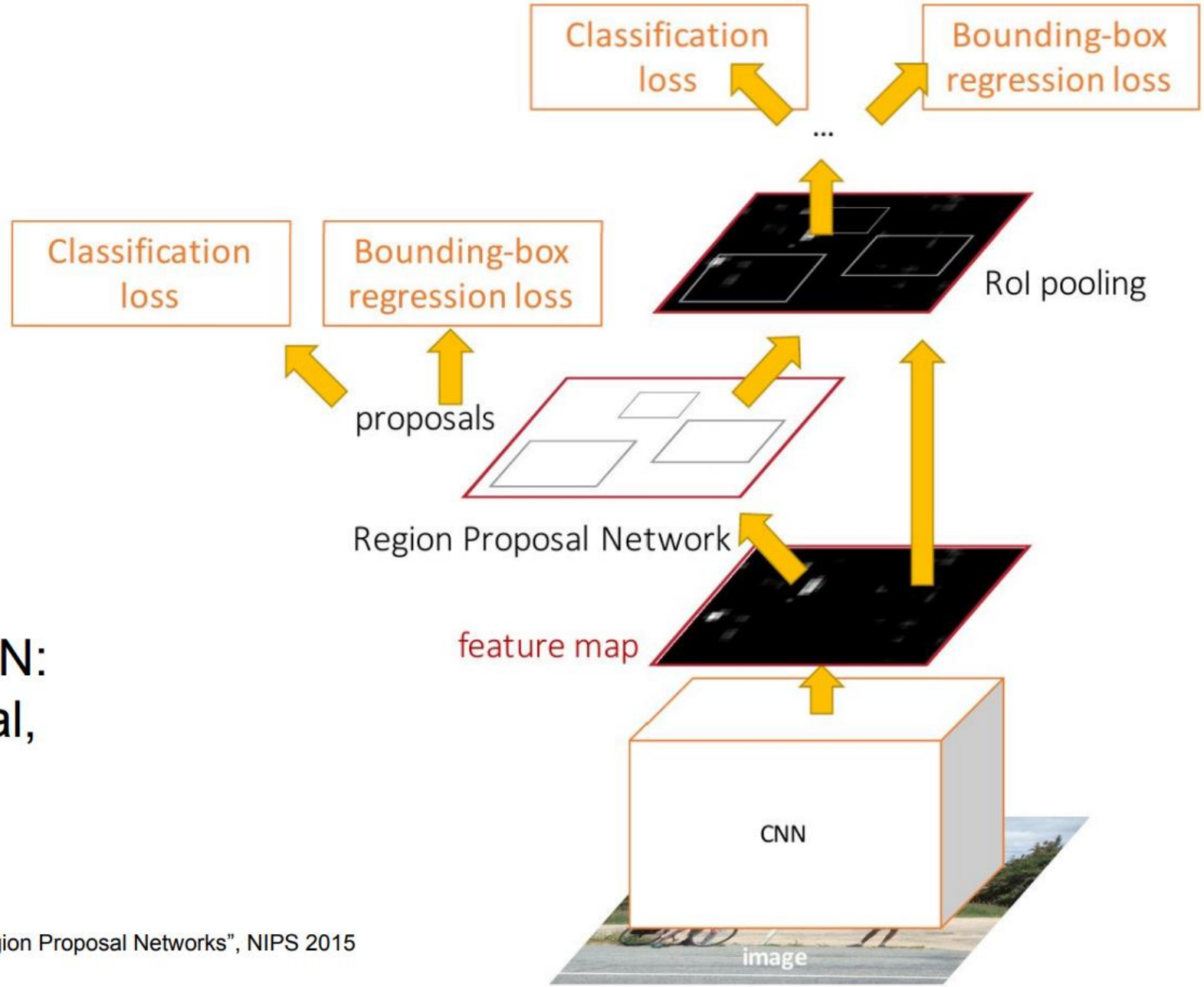
https://panderson.me/images/CVPR-Up-Down-talk.pdf

Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one

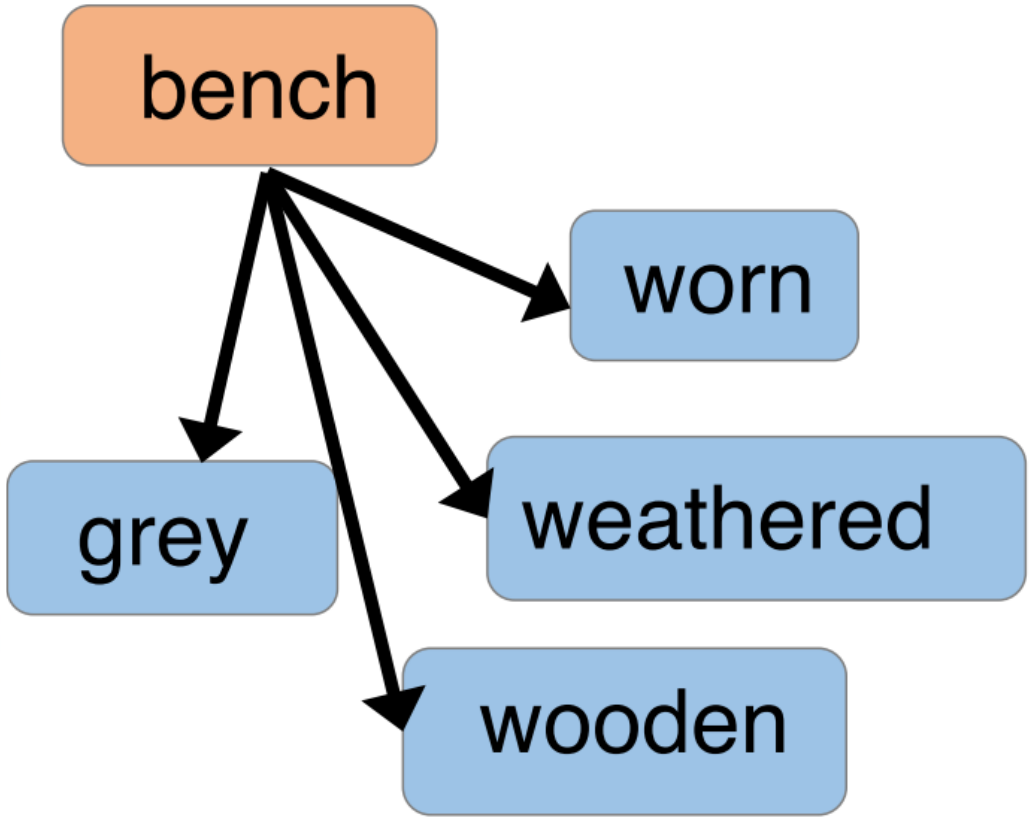


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Faster RCNN pre-training

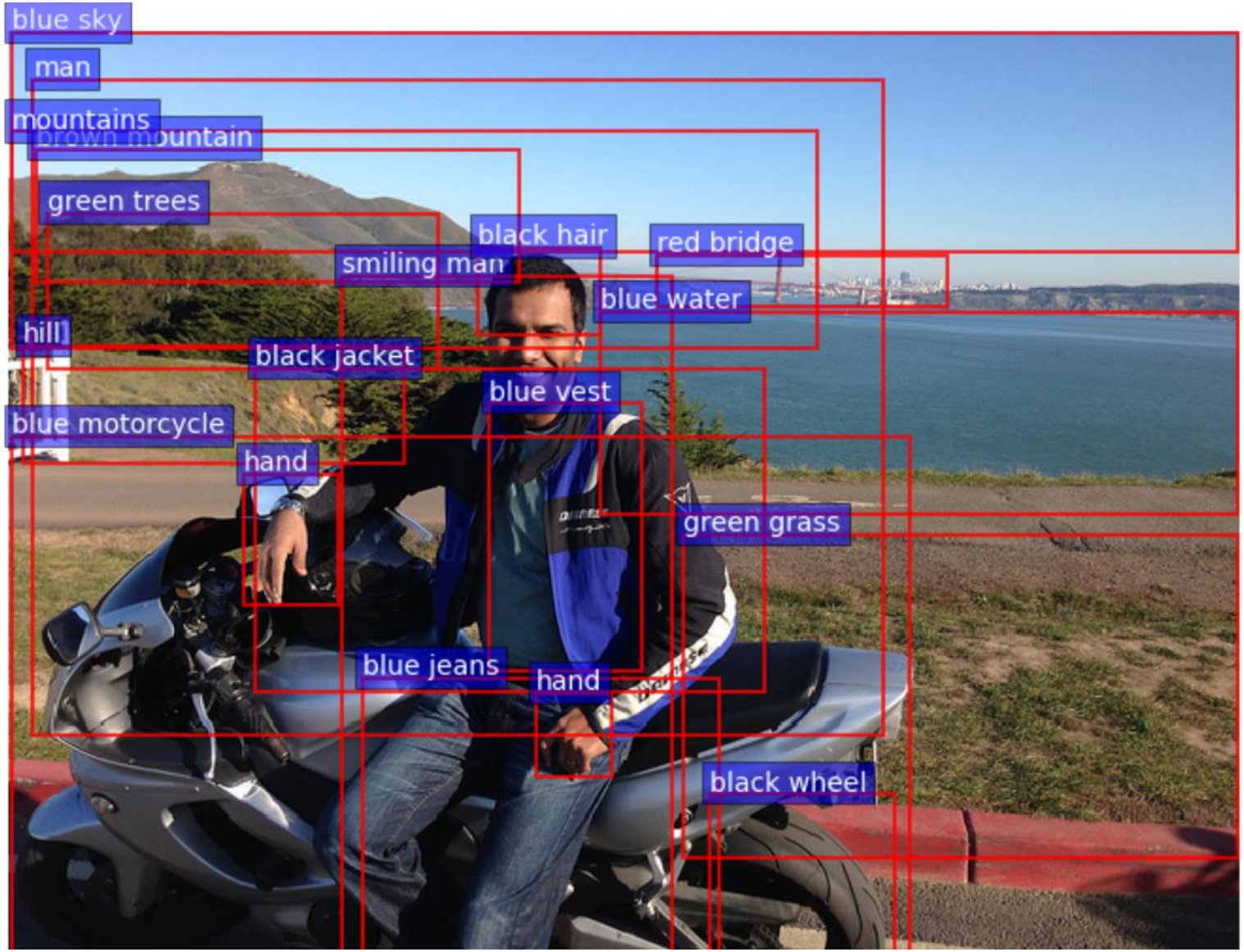
Using Visual Genome with:

- 1600 filtered object classes
- 400 filtered attribute classes



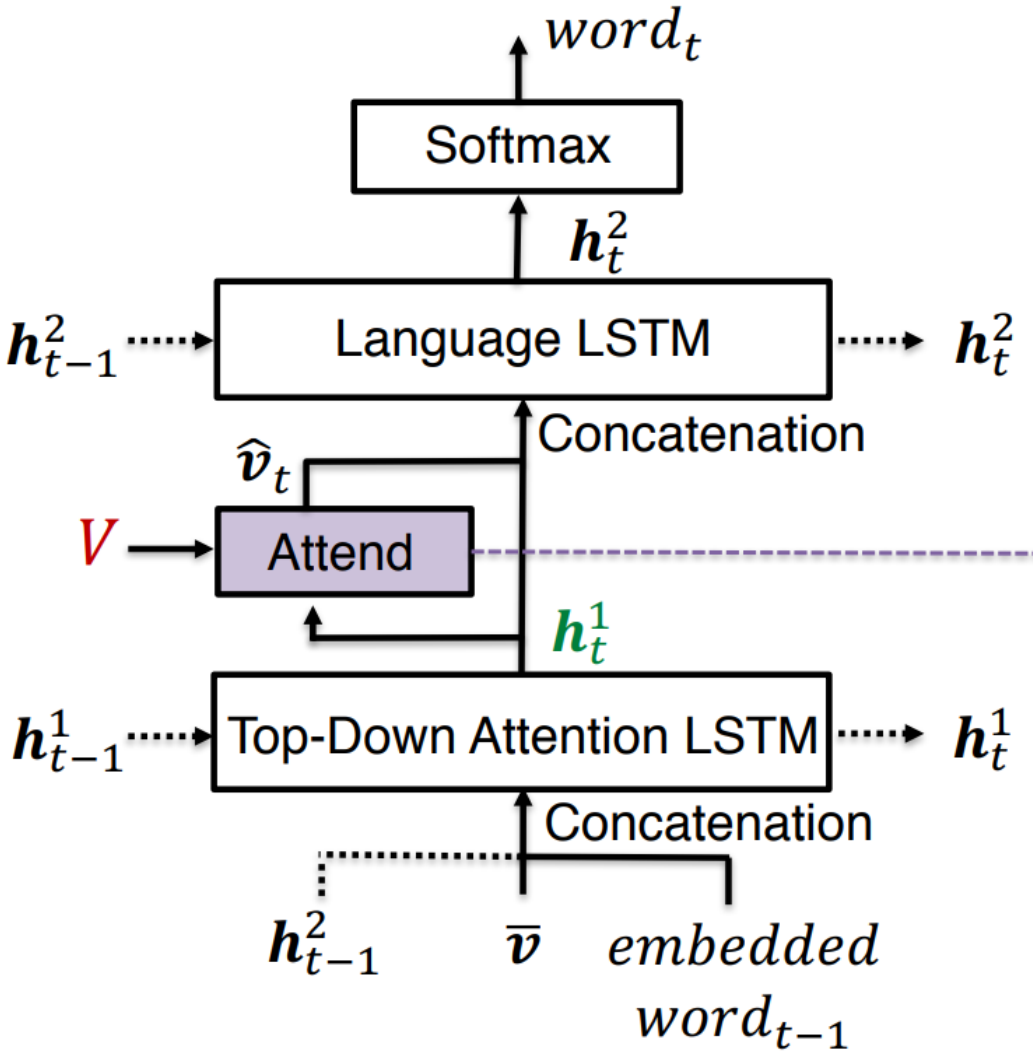
https://panderson.me/images/CVPR-Up-Down-talk.pdf

Result of Faster RCNN



Result of Faster RCNN

Image captioning model:



Attend block

$$a_i = \boldsymbol{w}^T \tanh(W_v \boldsymbol{v}_i + W_h \boldsymbol{h})$$

$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$

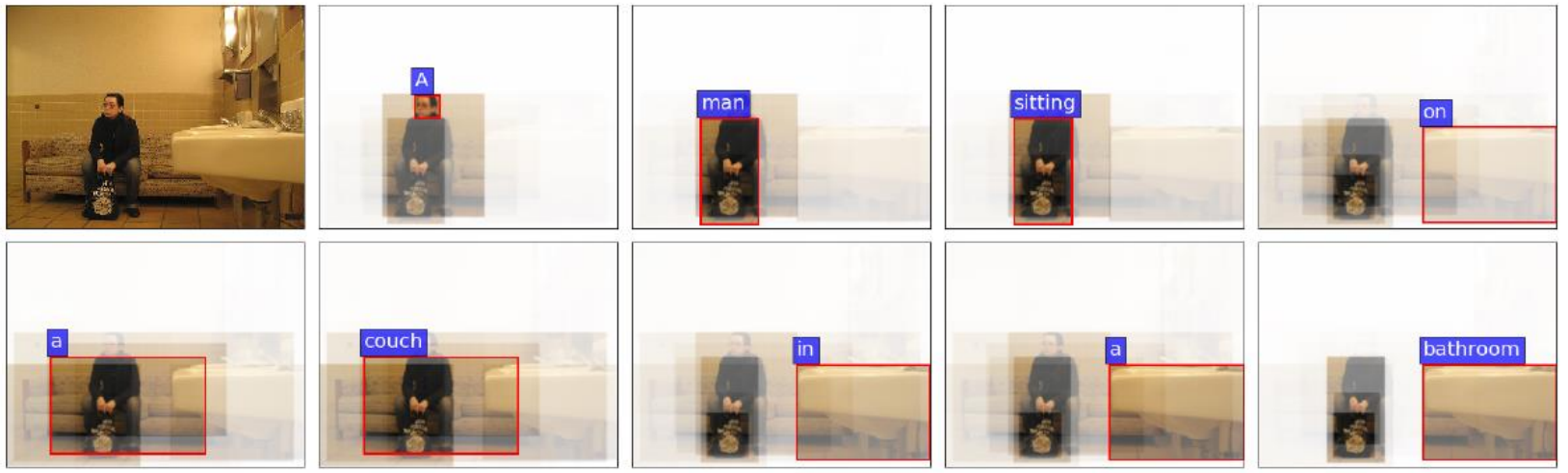
$$\hat{\boldsymbol{v}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{v}_i$$

https://panderson.me/images/CVPR-Up-Down-talk.pdf

Results of Up-Down attention model

ResNet (10×10): A man sitting on a ~~toilet~~ in a bathroom.

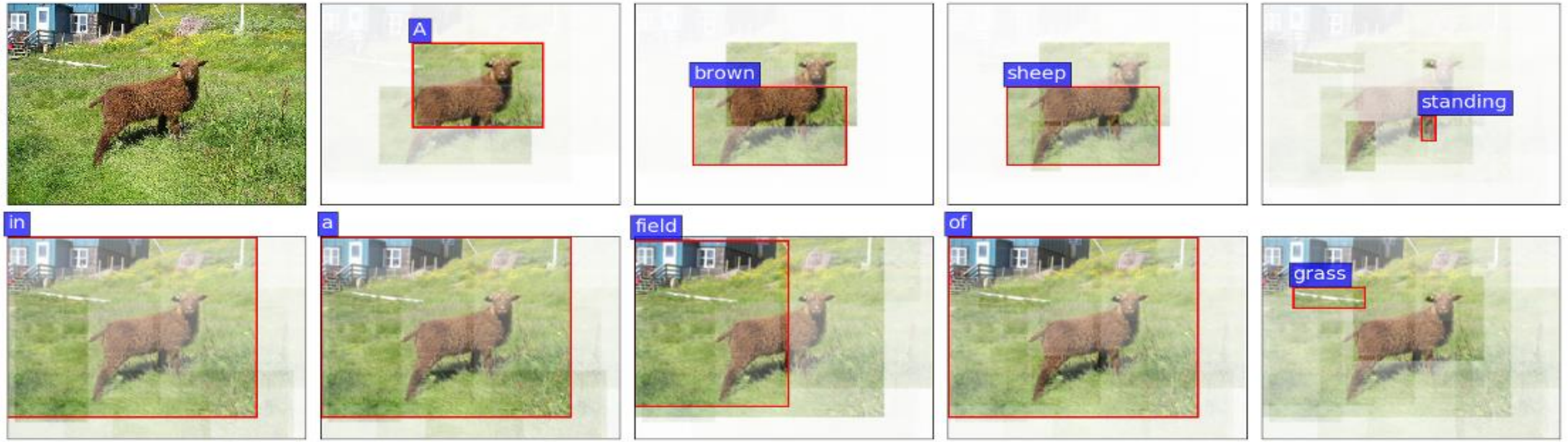
Up-Down : A man sitting on a **couch** in a bathroom.



https://panderson.me/images/CVPR-Up-Down-talk.pdf

Results of Up-Down attention model

Up-Down : `A brown sheep standing in a field of grass.`



https://panderson.me/images/CVPR-Up-Down-talk.pdf

Results of Up-Down attention model

1st COCO Captions leaderboard (July 2017)

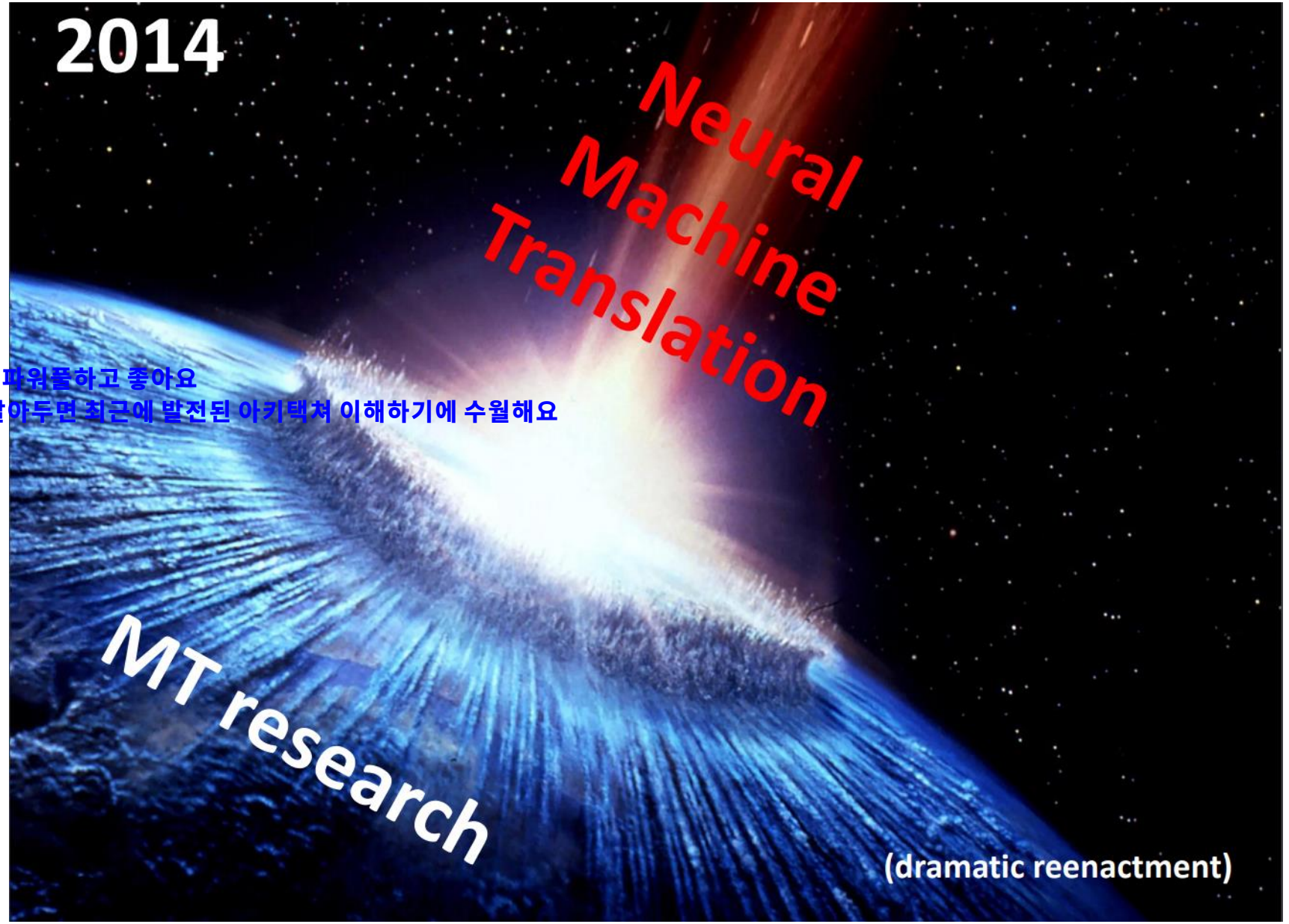
COCO Captions "Karpathy" test set (single-model):

	BLEU-4	METEOR	CIDEr	SPICE
ResNet (10×10)	34.0	26.5	111.1	20.2
Up-Down (Ours)	**36.3**	**27.7**	120.1	21.4
			+8%	+6%

https://panderson.me/images/CVPR-Up-Down-talk.pdf

Neural Machine Translation

Neural Machine Translation



- 어탠션은 파워풀하고 좋아요
- 이거 잘 알아두면 최근에 발전된 아키텍쳐 이해하기에 수월해요

Neural Machine Translation

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network*

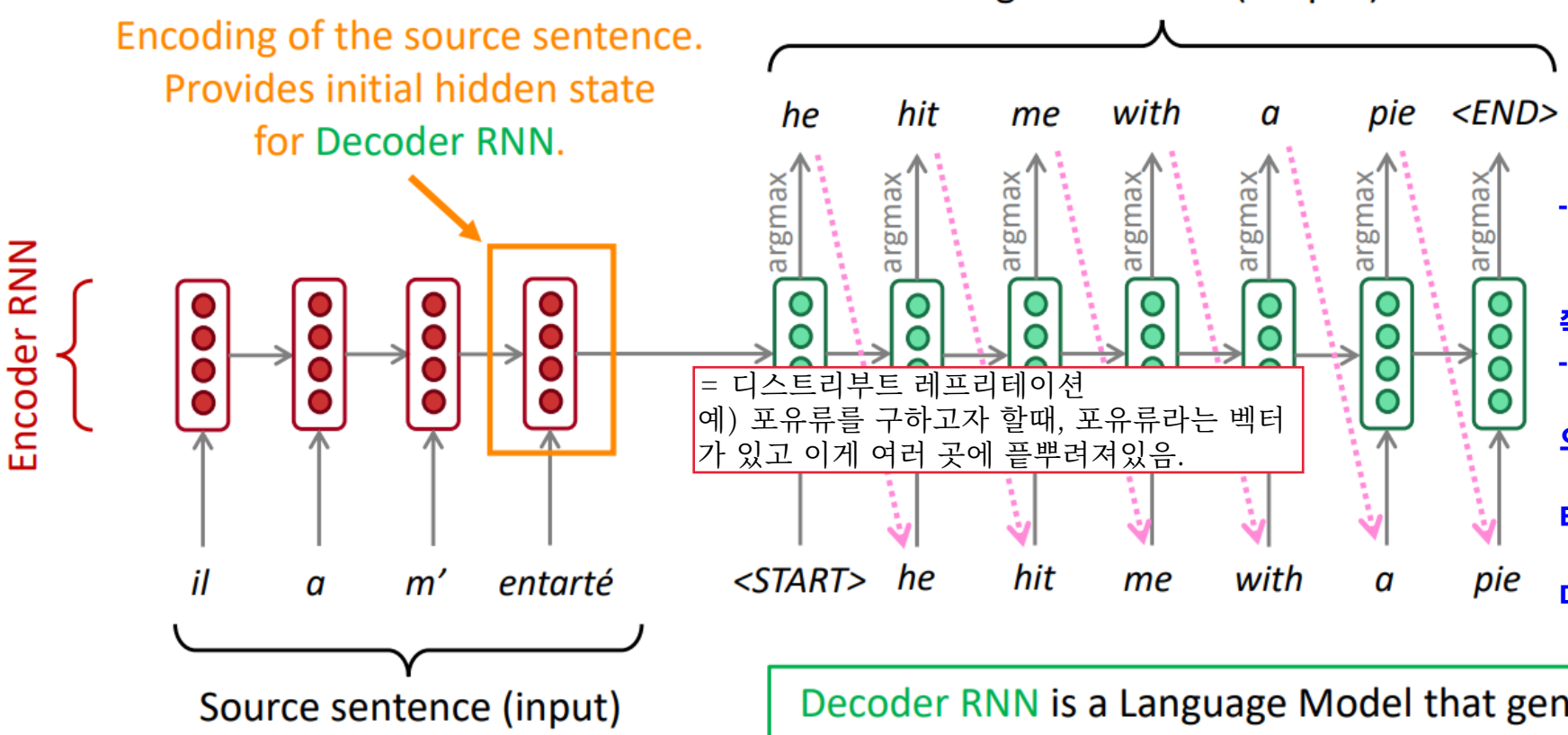
- The neural network architecture is called sequence-to-sequence (aka seq2seq) and it involves *two* RNNs.

머신트랜스레이션
- 매니투매니
- 시퀀스에 대해 타겟이 존재해요
- 그러다보니까 우리가 인코더와 디코더가 존재하는데 인코더는 우리가 주어진 워드의 시퀀스 고려해어 적당한 단어 매핑
- i go school이면 내가 학교를 간다는거 벡터 만들고
- 각 정보를 하이로 인코딩해주는 벡터인코딩하는게 인코더.
#인코더 알엔엔과 디코더 알엔엔이 존재합니다
- 인코더 알엔엔은 주어진 문제 잘 이해해서 벡터로 표시
- 디코더: 주어진 문장에 주어진 벡터 기반으로 생성또는 번역수행
우리가 어떤 정보 인코딩하고 이해할 수 있께 바꾸는게 디코더
- 한국어를 영어로 번역하면 의미론적으로 스페이스로 매핑하고
- 디코더는 매핑된 애를 다시 한국어 정보 및 영어로 보내는 역할
- 자연어 세계에서 영어를 임베딩 스페이스 벡터로 보내는게 인코더
- 디코더는 이걸 사람이 이해할 수 있는 한글로 바꿔주는 것.

Neural Machine Translation

시퀀스투 시퀀스
- 인코더와 디코더의 구조 각각 하나씩 총 두개의 구조인 시퀀스투 시퀀스
- 주어진 인풋을 아웃풋으로 바꿔서 그리 부름
- 이 모델 특징은 인코더와 디코더로 구성
- 그래서 가각ㄱ 알엔엔 및 네트워크고 그 파라미터는 공유 안됨

The sequence-to-sequence model



= 디스트리부트 레프리테이션
예) 포유류를 구하고자 할때, 포유류라는 벡터가 있고 이게 여러 곳에 픝뿌려져있음.

- 피에이디: 패딩
  > 아무것도 없는 정보.
  > 컴퓨터 비전에서 씨앤앤 돌릴때 한쪽 0으로 돌린 역할과 같음
- 디코더는 스타트와 앤드 토큰 중요
  - 예측을 할때 초창기엔 생성한게 없으니
  디코더에게 생성하라고 알려주는 스타트 토큰
  - 다 마치고 나면 앤드토큰으로 번역 마침을 알려 문장의 끝 고지.

Encoder RNN produces an encoding of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

Note: This diagram shows **test time** behavior: decoder output is fed in ⋯⋯➤ as next step's input

특이한 토큰
- 앤엘피엔 스페셜 토큰 존재: 스타트 토큰, 앤드 토큰.
- sos, eos, pad
- 스타트 토큰: 스타트 오브 센텐스: 문자의 시작
-이오에스: 엔드 오브 센텐스: 문장의 끝
- 피에이디: 패딩

Neural Machine Translation

- Sequence-to-sequence is useful for *more than just MT*

- Many NLP tasks can be phrased as sequence-to-sequence:
  - Summarization (long text → short text)
  - Dialogue (previous utterances → next utterance)
  - Parsing (input text → output parse as sequence)
  - Code generation (natural language → Python code)

- 기존 랭귀지는 초창기에 아무것도 없을때 엑스원구하고, 엑스 투랑 엑스 원알때 엑스쓰리 구하는식
  - 비슷한데 기븐으로서 항상 엑스가 주어진 조건부 랭귀지 모델.
- 기존 딥러닝이 모든 경우에 대해서 좋냐는건 아니에요.
  - 큰 챌린지: 적은 데이터에 대해서도 피팅을 잘해야함
  - 많은 데이터가 있으면 통계 기반의 모델들을 가뿐히 뛰어넘어요
  - 데이터가 적으면 딥러닝 결과가 안좋곤함
  - 일단 엔엠티등에는 빅 포커스 필요.

#시퀀스투 시퀀스 단점

Neural Machine Translation

- The sequence-to-sequence model is an example of a **Conditional Language Model**.
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are also conditioned on the source sentence *x*

- NMT directly calculates $P(y|x)$:

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots P(y_T|y_1,\ldots,y_{T-1},x)$$

Probability of next target word, given target words so far and source sentence x

- **Question**: How to train a NMT system?
- **Answer**: Get a big parallel corpus…

특이한 토큰
- 엔엘피엔 스페셜 토큰 존재: 스타트 토큰, 앤드 토큰.
- sos, eos, pad
- 스타트 토큰: 스타트 오브 센텐스: 문자으이 시작
-이오에스: 엔드 오브 센텐스: 문장의 끝
- 패딩: 의미없는
  > 아무것도 없는 정보.
  > 컴퓨터 비전에서 씨앤앤 돌릴때 한쪽 0으로 돌린 역할과 같음
- 디코더는 스타트와 앤드 토큰 중요
  - 예측을 할때 초창기엔 생성한게 없으니
  디코더에게 생성하라고 알려주는 스타트 토큰
  - 다 마치고 나면 앤드토큰으로 번역 마침을 알려 문장의 끝 고지.

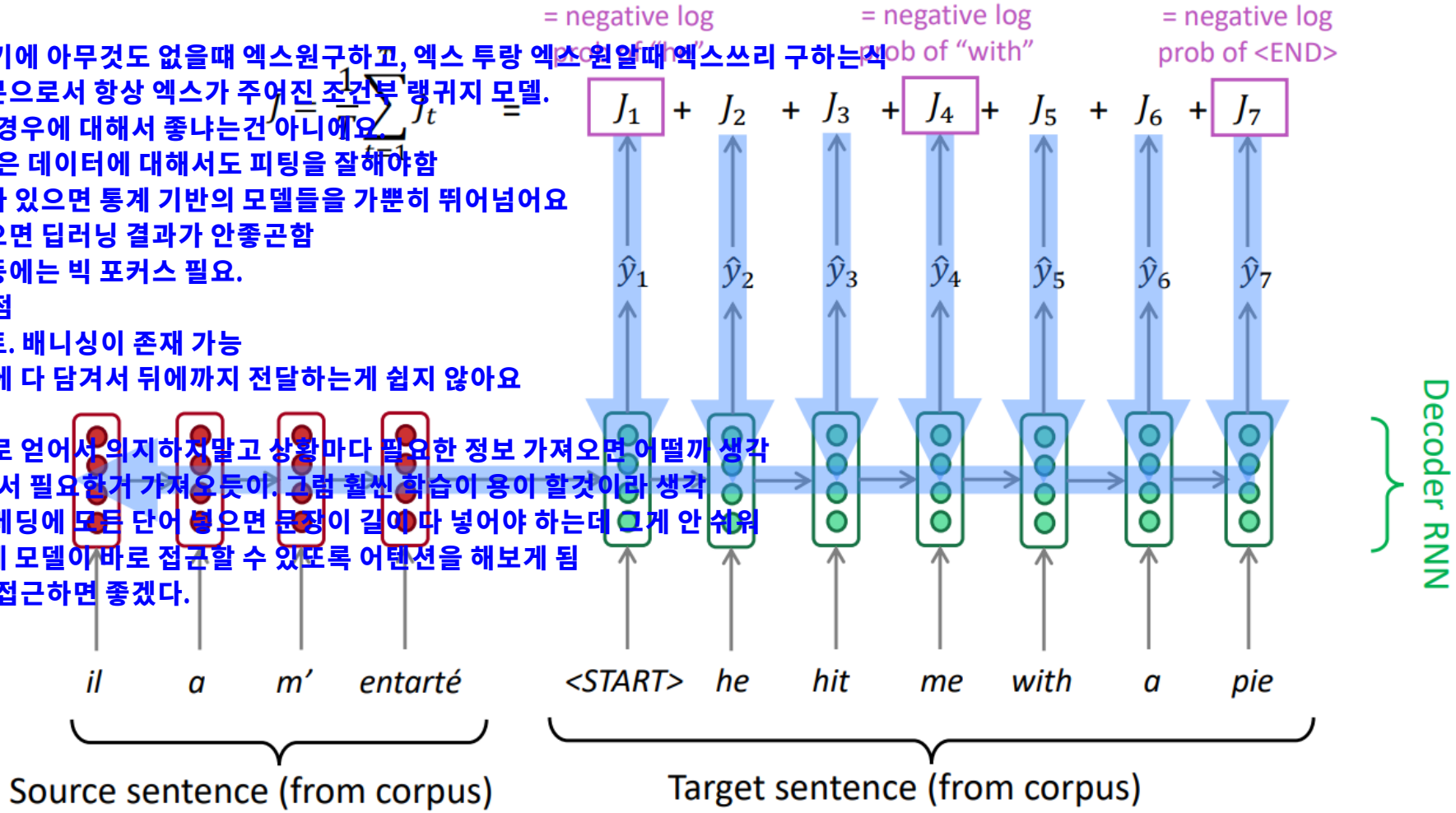
#MT
- NMT는 컨디셔널 랭귀지라 할 수 있음
- 이건 주어진 인풋 서퀀스로부터 타겟의 랭귀지에 적절한 워드 미 문장 단어를 만들어 생성하는 것.
- 우리가 주어진 엑스로 부터 와이를 예측하면
  - 와이는 와이 엔부터 영까지 주어진 센텐스
  - 소스인 엑스는 한국어이고, 와이원은 첫번째 단어. 그 리고 와이원아는 채로 와이투 예측
  마지막은 긴 길이까지 예측하는게 엔엘티 목적

Neural Machine Translation

- 기존 랭귀지는 초창기에 아무것도 없을때 엑스원구하고, 엑스 투랑 엑스원알때 엑스쓰리 구하는식
  - 비슷한데 기븐으로서 항상 엑스가 주어진 조건부 랭귀지 모델.
- 기존 딥러닝이 모든 경우에 대해서 좋냐는건 아니에요.
  - 큰 챌린지: 적은 데이터에 대해서도 피팅을 잘해야함
  - 많은 데이터가 있으면 통계 기반의 모델들을 가뿐히 뛰어넘어요
  - 데이터가 적으면 딥러닝 결과가 안좋곤함
  - 일단 엔엠티등에는 빅 포커스 필요.

#시퀀스투 시퀀스 단점
- 길어져서 그레디언트. 배니싱이 존재 가능
- 그러면 임베딩정보에 다 담겨서 뒤에까지 전달하는게 쉽지 않아요

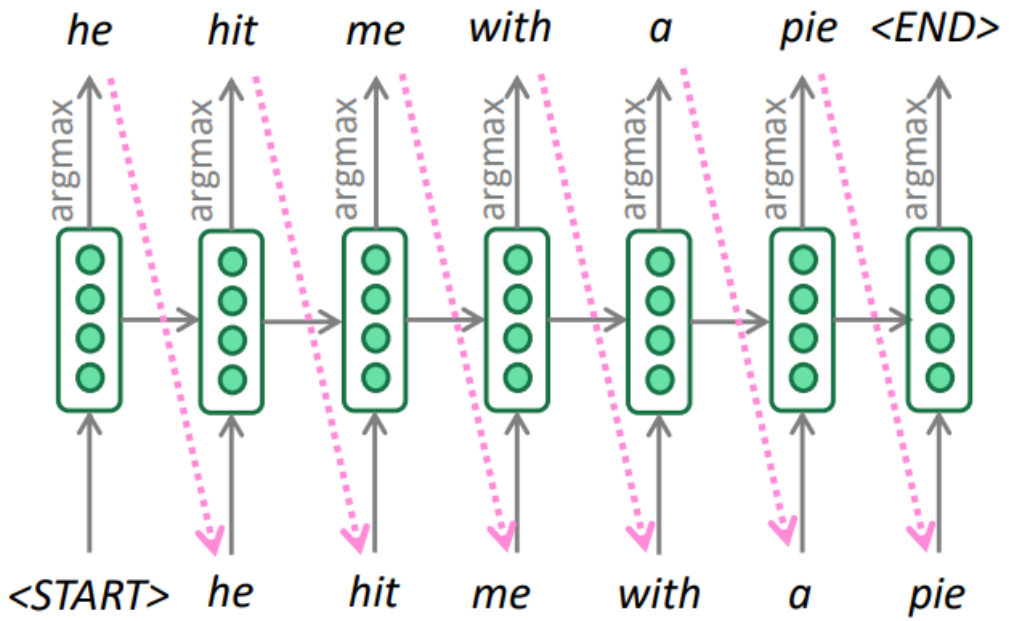
고민
- 우리가 정보를 하나로 얻어서 의지하지말고 상황마다 필요한 정보 가져오면 어떨까 생각
- 예) 데이터 베이스에서 필요한거 가져오듯이. 그럼 훨씬 학습이 용이 할것이라 생각
- 아니면 조그만한 임베딩에 모든 단어 넣으면 문장이 길어 다 넣어야 하는데 그게 안 쉬워
- 따라서 다이렉트하게 모델이 바로 접근할 수 있또록 어텐션을 해보게 됨
- 필요한 정보를 바로 접근하면 좋겠다.



Seq2seq is optimized as a **single system.**
Backpropagation operates "*end-to-end*".

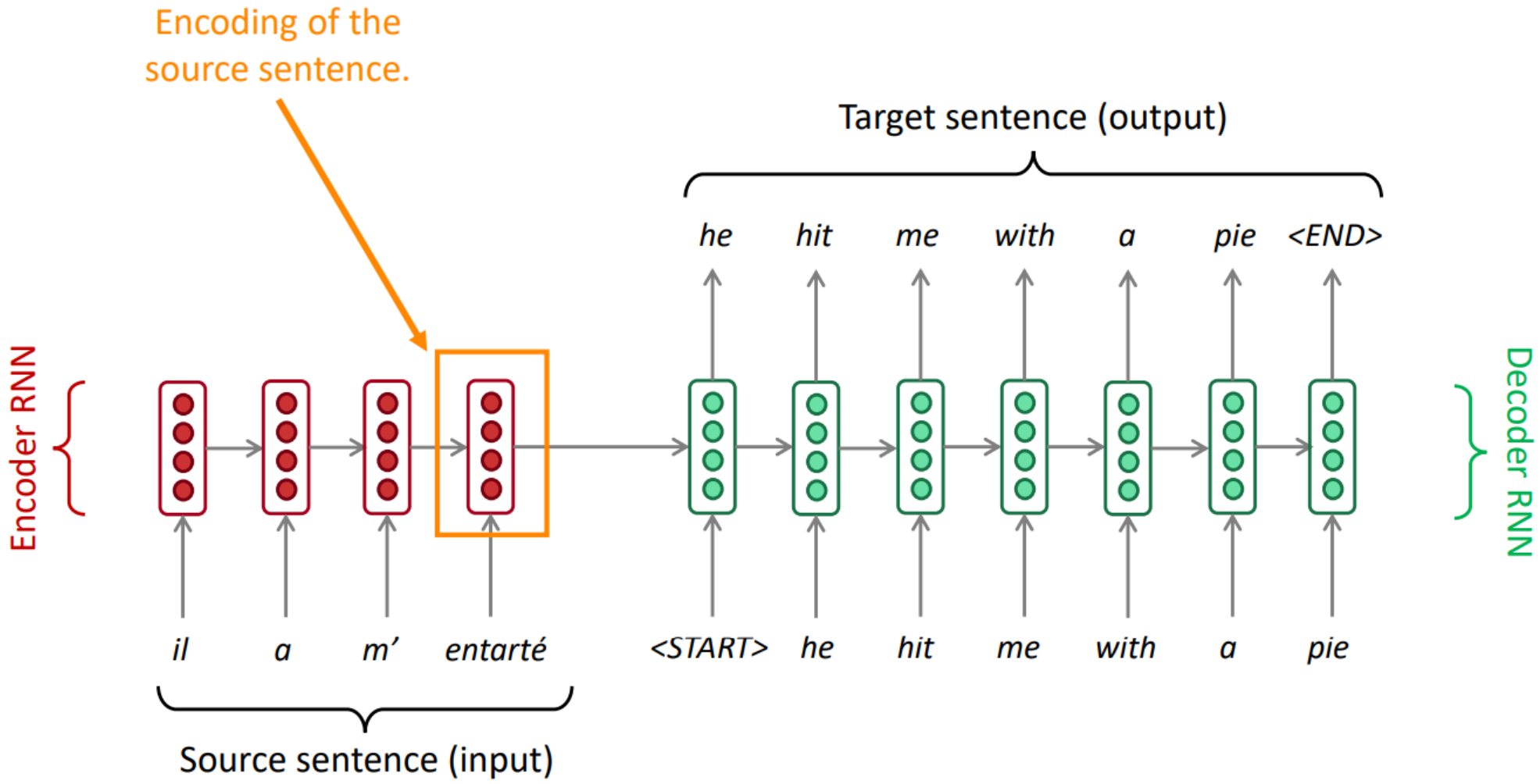
Greedy Decoding

- We saw how to generate (or "decode") the target sentence by taking argmax on each step of the decoder



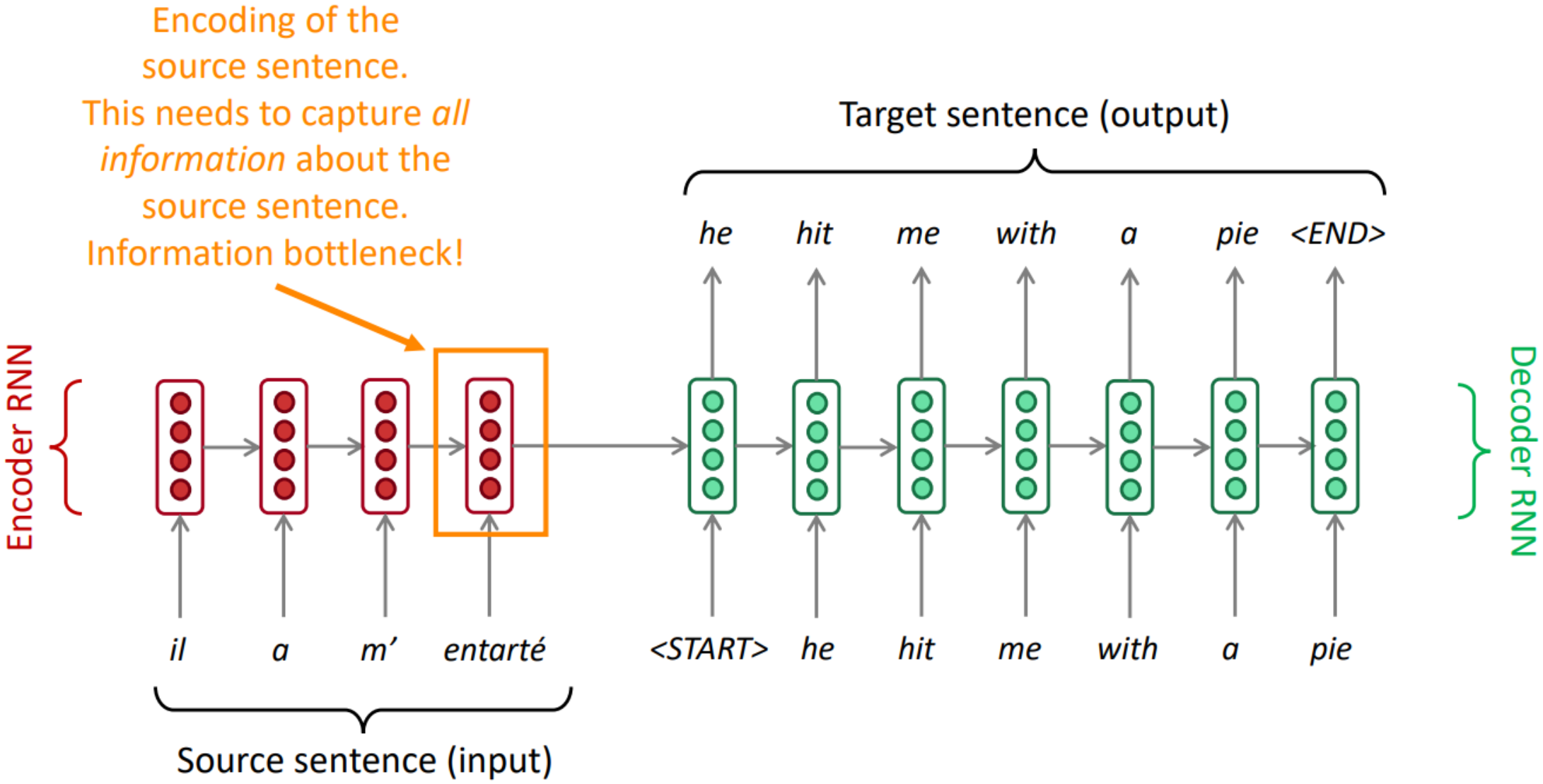
- This is greedy decoding (take most probable word on each step)
- **Problems with this method?**

Neural Machine Translation



Problems with this architecture?

Neural Machine Translation



Attention

- **Attention** provides a solution to the bottleneck problem.

- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

#어텐션
- 일종의 태깅
- 인코더의 어디를 포커스 할지 선택
- 디코딩시 인코딩의 어디 부분을 다이렉트하게 포커싱 할까

- First we will show via diagram (no equations), then we will show with equations

Neural Machine Translation (NMT)

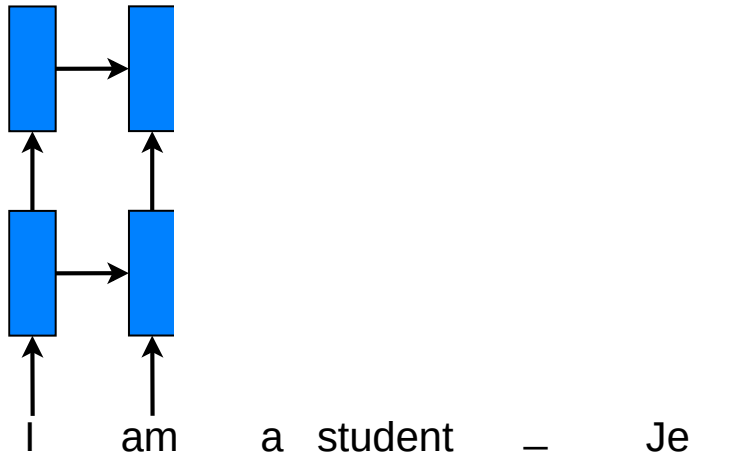
I am a student _ Je suis étudiant

- Big RNNs trained end-to-end.

Neural Machine Translation (NMT)

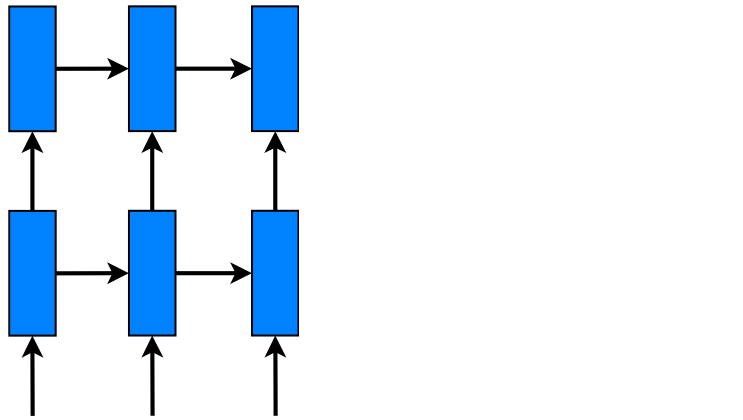
- Big RNNs trained end-to-end.

Neural Machine Translation (NMT)



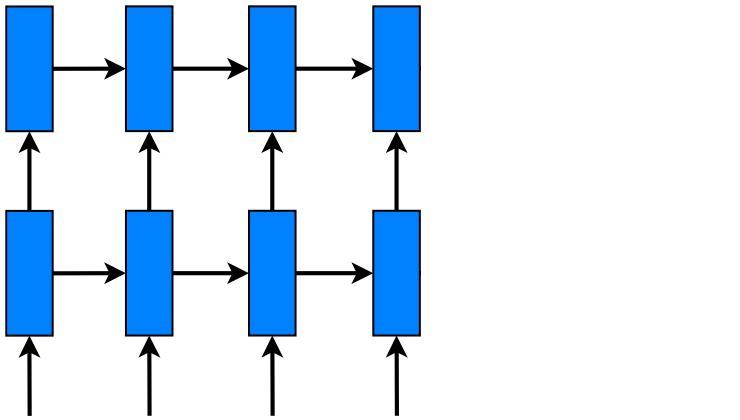
- Big RNNs trained end-to-end.

Neural Machine Translation (NMT)



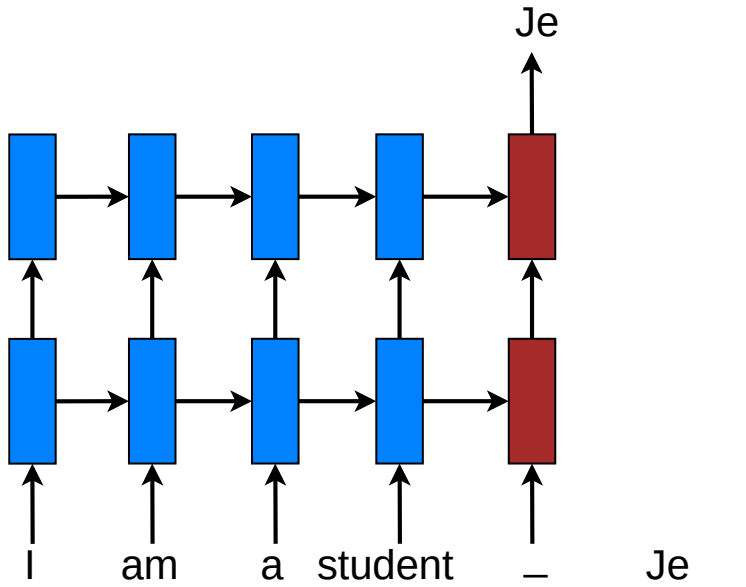
- Big RNNs trained end-to-end.

Neural Machine Translation (NMT)



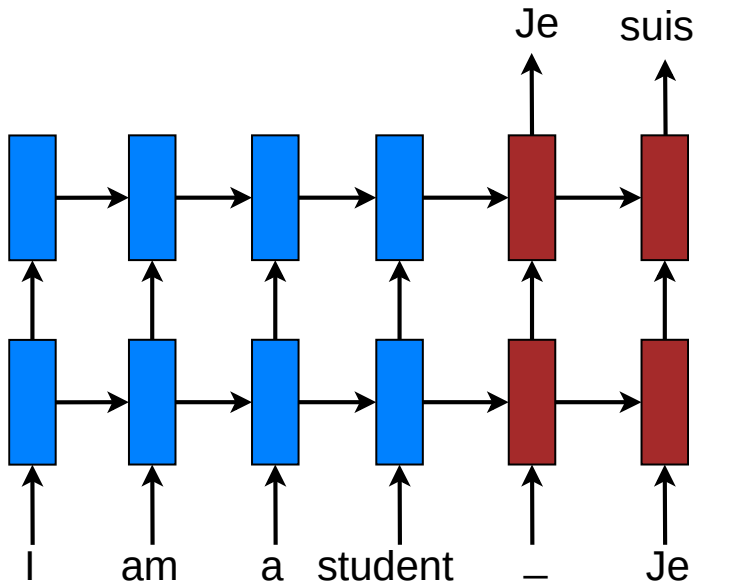
- Big RNNs trained end-to-end.

Neural Machine Translation (NMT)



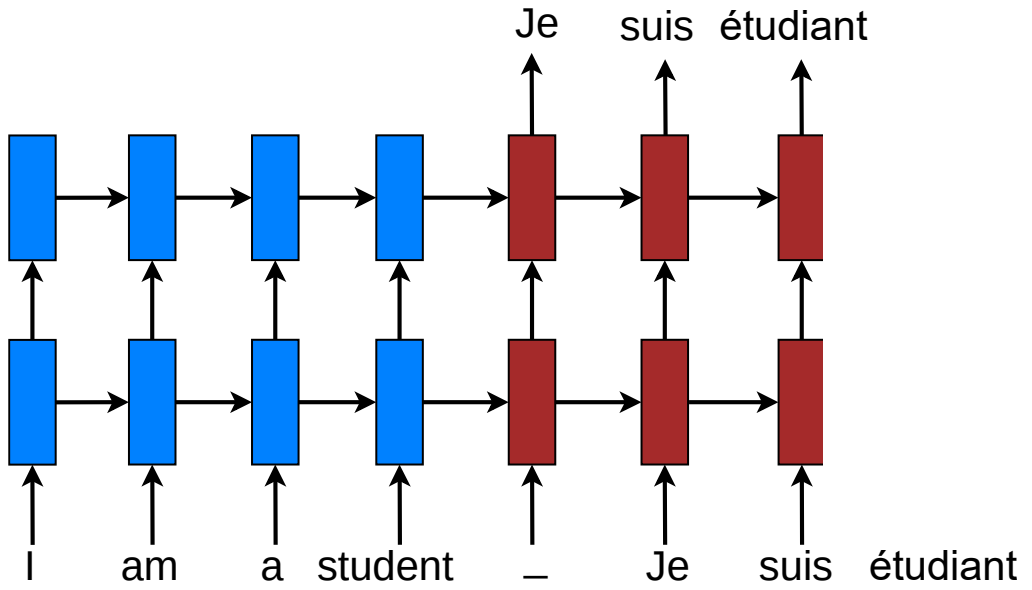
- Big RNNs trained end-to-end: encoder-decoder.

Neural Machine Translation (NMT)



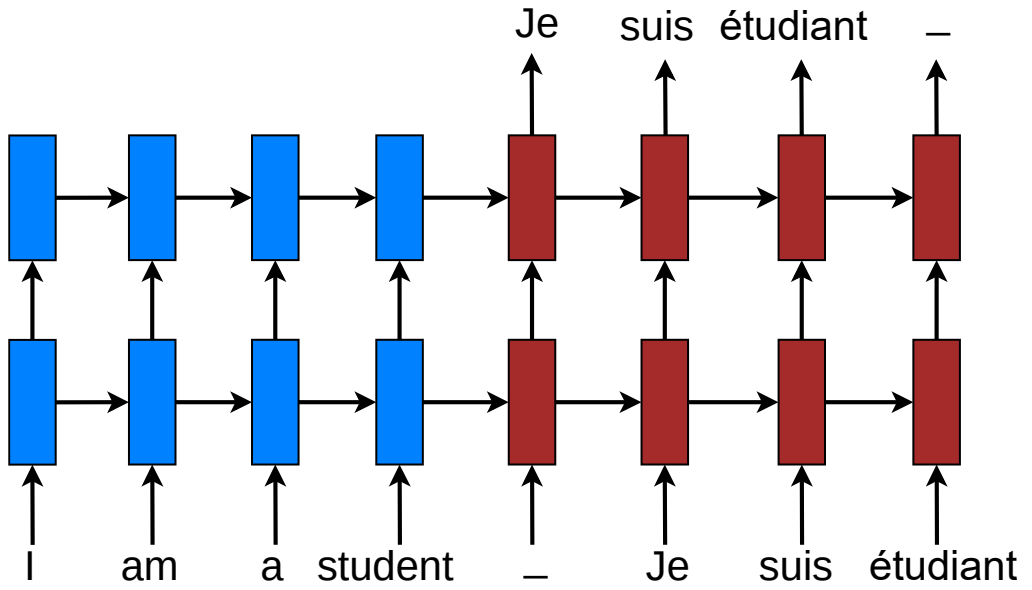
- Big RNNs trained end-to-end: encoder-decoder.

Neural Machine Translation (NMT)



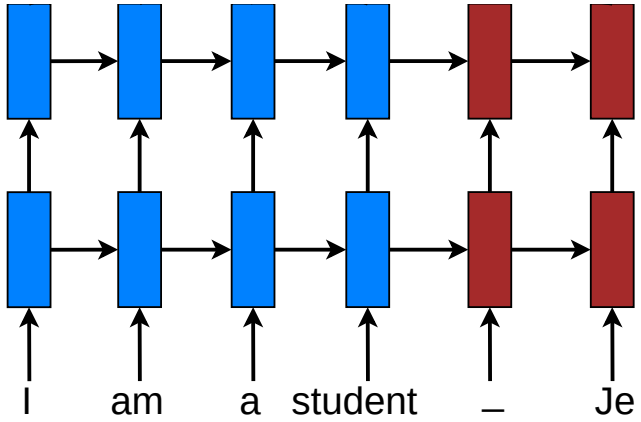
- Big RNNs trained end-to-end: encoder-decoder.

Neural Machine Translation (NMT)



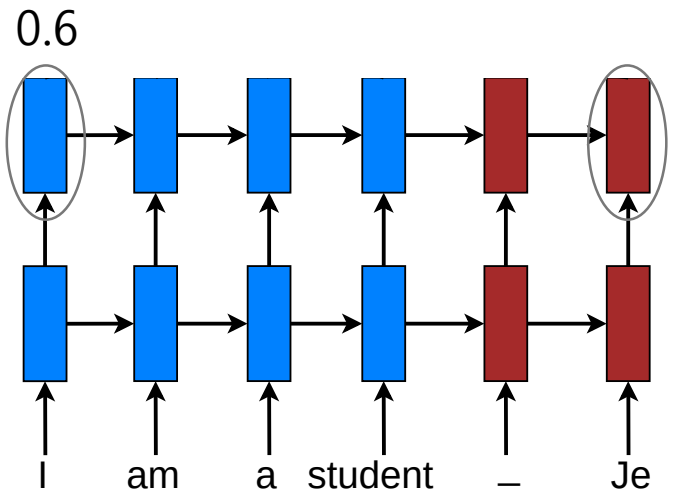
- Big RNNs trained end-to-end: encoder-decoder.
  - Generalize well to long sequences.
  - Small memory footprint.
  - Simple decoder.

Attention Mechanism



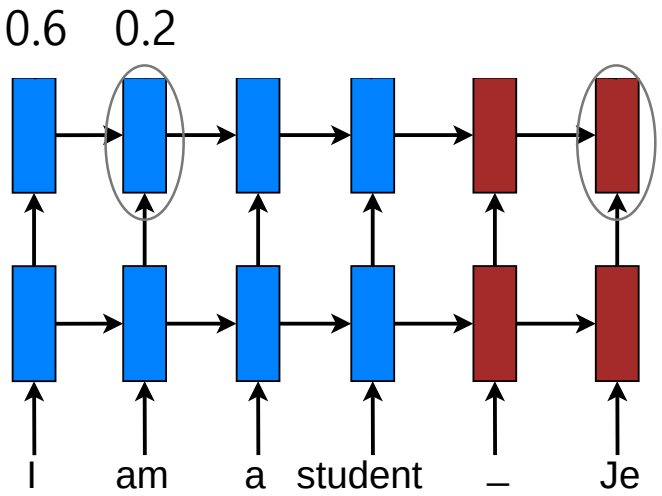
- Maintain a memory of source hidden states

Attention Mechanism



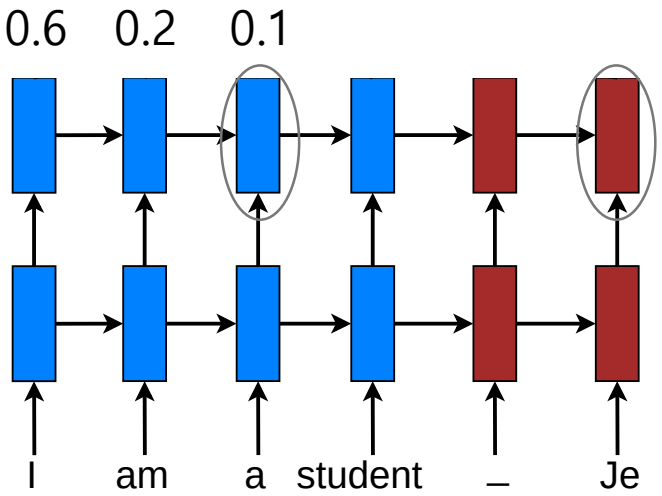
- Maintain a memory of source hidden states
  - Compare target and source hidden states

Attention Mechanism



- Maintain a memory of source hidden states
  - Compare target and source hidden states

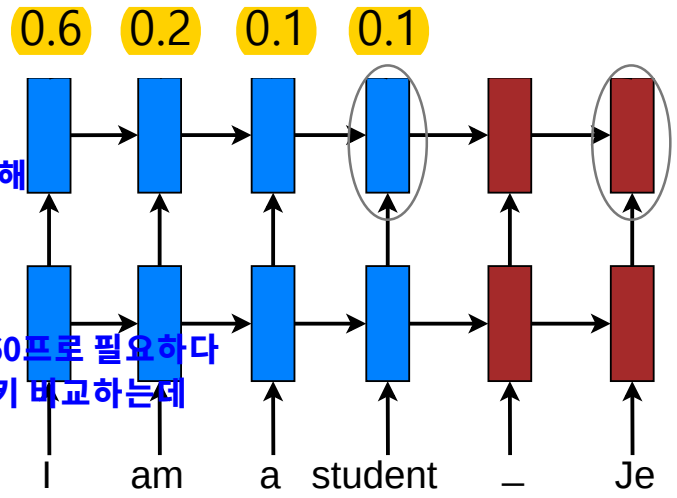
Attention Mechanism



- Maintain a memory of source hidden states
  - Compare target and source hidden states

Attention Mechanism

#어텐션
- 일종의 태깅
- 인코더의 어디를 포커스 할지 선택
- 디코딩시 인코딩의 어디 부분을 다이렉트하게 포커싱 할까
- 엔드투엔드로 학습한다 생각해봐요
-문제) 롱 시퀀스에 대해서는 빅 알엔엔으로 가능하나 문장이 길면 프로블럼 가질 수 밖에
내가 이 시퀀스 시퀀스
- 지가 나오고 그다음 나오는게 아이라고 하면
- 나는 동사 부분이 필요해
- 가지고 올때 중요도를 곱해줘서(0.6),(0.2)가져온대
- 이걸 하이레벨적으로 생각하면 데이터 베이스 생각하면 편해
-인코더 쪽에서 데이터 베이스 테이블인데 키/밸류가 있겠지
-쿼리문이 있어. 이 쿼리 기반으로 키랑 비교해요.
- 그 다음에 키 페어중에서 맞는 거 힛해서 가져오죠
- 이건 정도를 따져요 이 쿼리가 이거면 키의 이게 필요한게 60프로 필요하다
- 두번째 부분은 20프로, 세번째는 10프로 필요한거 쿼리와 키 비교하는데
- 쿼리 백터가 3개 있고 키 백터가 마찬가지로 3개 있따 치면
- 들어가고 이게 유사도인거에요
- 전체 섬이 1이 되게 소프트 맥스하고 **
-



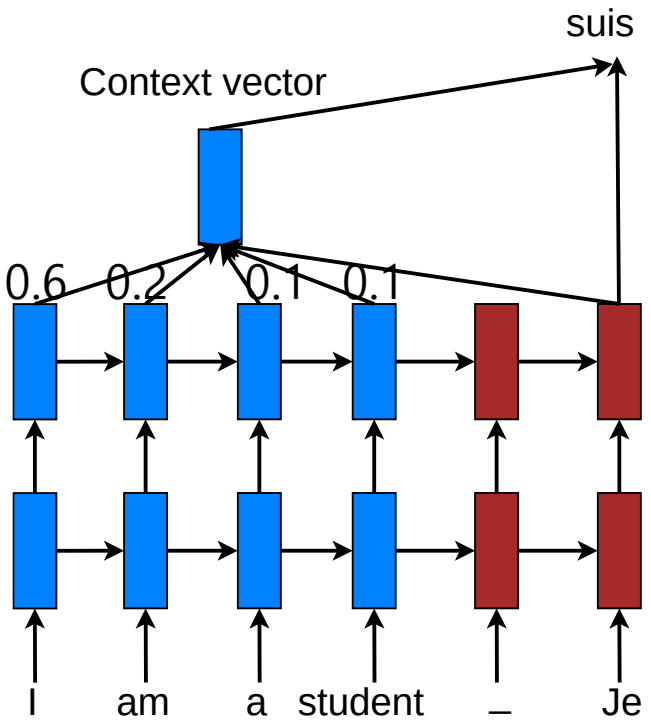
각 데이터 베이스에 대해서 0.6/0.2/0.1/0.1로 가져옵니다.
- 그게 바로 내가 번역시 필요한 정보
- 이걸 바탕으로 다음 단어를 예측합니다.
- 지금 매커니즘이 어탠션 어렵게 생각 안하고 데이터베이스 기준으로
인코더가 데이터 베이스에 올라가는 정보 매핑
- 그럼 벡터로 표시되고 키와 벨류는 사실 동일해요 (디비랑 다르지만)
- 그 중에서 번역을 할때마다 쿼리를 만들어요.
- 그래서 필요한거 가져오는데 쿼리랑 키를 비교해서 가져오고
- 쿼리의 유사도 정보가 있을거에요. 이거 필요하다는 유사도
- 이 정도가 필요한 정도가 될거니까
- 이런 정도를 유사도로가져오고, 소프트맥스하면 확률값이 나와요

- 소프트하게 그 값을 가져와서 그게 데이터베이스로부터 가져온 값
- 이렇게 하면 롱시퀀스.메모리가 커져도 혹은 아무리 긴 단어를 번역해도이전까지는 하나의 데이터 베이스 하나의 블록의존이,

- Maintain a memory of source hidden states
  - Compare target and source hidden states

이제는 필요할때마다 다이렉트하게
정보를 빼올 수 있어 편지죠
- 기존의 시퀀스는 디비 없고 하나의 테이블만 있어서 그거 전달해 번역했는데 그거보다는 더 빠르게 접근할 수 있어서 속도도 빠르고 좋을 수 있단그죠
- 정보를 다이나믹하게 빼오는 알고리즘이다!

Attention Mechanism



- Maintain a memory of source hidden states
  - Able to translate long sentences.

No other attention architectures b eside (Bahdanau et al., 2015)

Attention

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

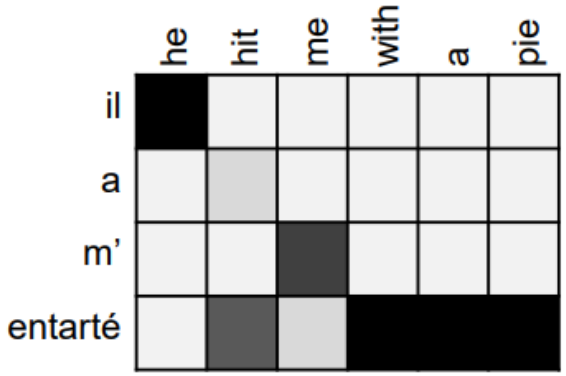
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is great

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Attention Models

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

- **More general definition of attention**:
  - Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

- We sometimes say that the query *attends to* the values.
- For example, in the seq2seq + attention model, each decoder hidden state (query) *attends to* all the encoder hidden states (values).

Attention is a general Deep Learning technique

More general definition of attention:

Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

Intuition:

- The weighted sum is a *selective summary* of the information contained in the values, where the query determines which values to focus on.
- Attention is a way to obtain a *fixed-size representation of an arbitrary set of representations* (the values), dependent on some other representation (the query).

There are several attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Attention always involves:
  1. Computing the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$
  2. Taking softmax to get *attention distribution* α:

There are multiple ways to do this

$$\alpha = \text{softmax}(\boldsymbol{e}) \in \mathbb{R}^N$$

  3. Using attention distribution to take weighted sum of values:

$$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1}$$

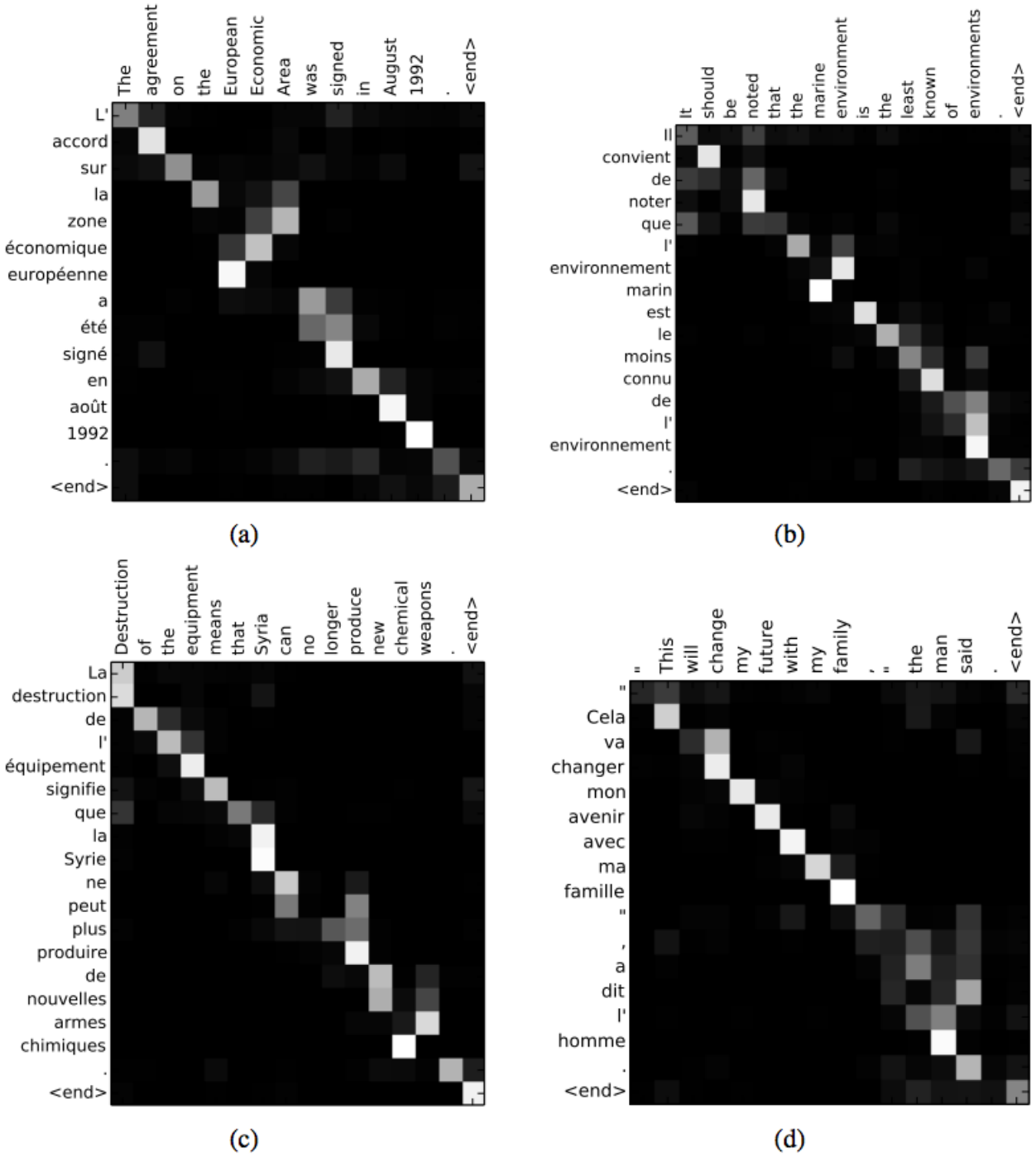
thus obtaining the *attention output* $\boldsymbol{a}$ (sometimes called the *context vector*)

Attention variants

There are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$ from $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and $\boldsymbol{s} \in \mathbb{R}^{d_2}$:

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier
- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter

Attention Example in Machine Translation



References

Stanford University CS231n: Convolutional Neural Networks for Visual Recognition

Deep Learning Summer School, Montreal 2016 - VideoLectures.NET

Understanding LSTM Networks -- colah's blog

https://panderson.me/images/CVPR-Up-Down-talk.pdf