

DSC291: Advanced Statistical Natural Language Processing

Language Modeling
Self-supervised Learning

Zhiting Hu

Lecture 4, April 7, 2022

UC San Diego

HALICIOĞLU DATA SCIENCE INSTITUTE

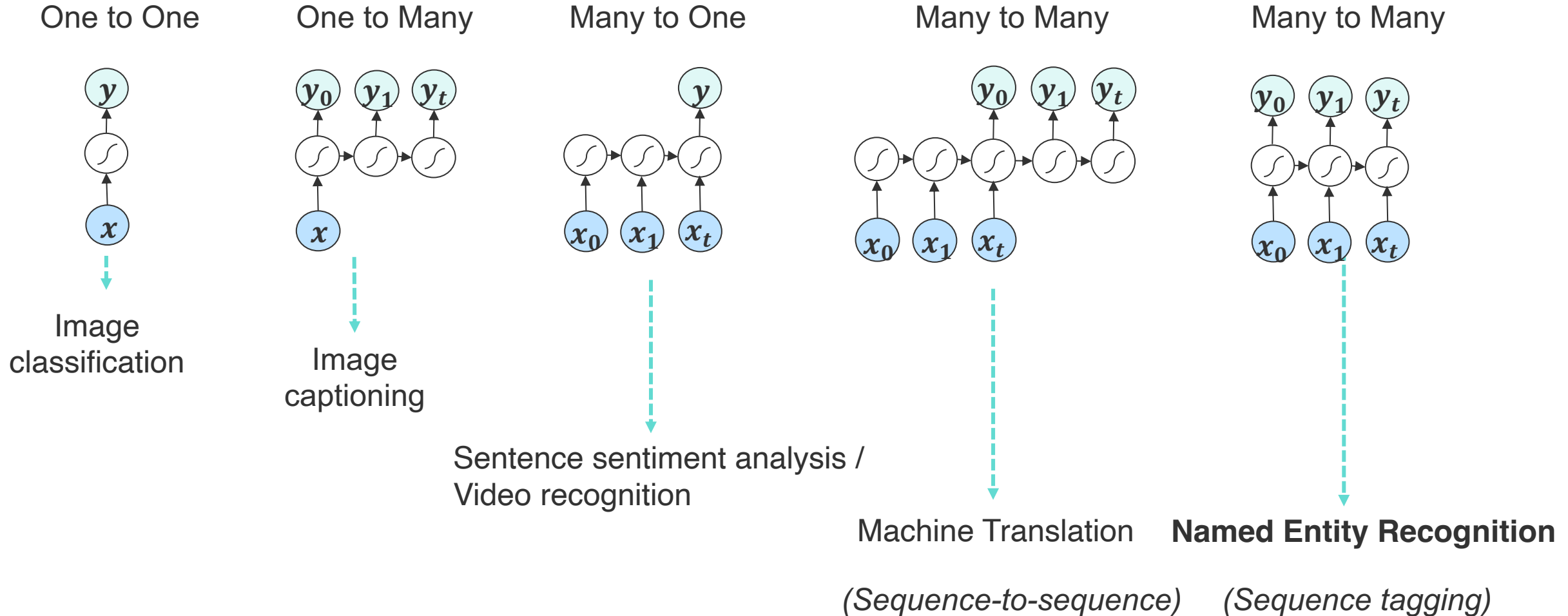
Outline

- Neural architectures

Outline

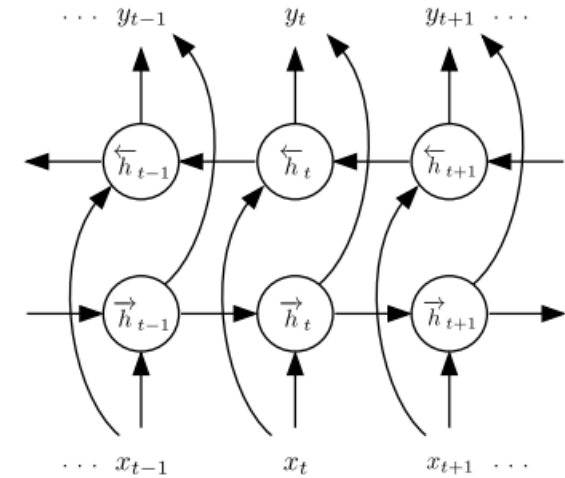
- Convolutional Networks (ConvNets)
- Recurrent Networks (RNNs)
 - Long-range dependency, vanishing gradients
 - LSTM
 - RNNs in different forms
- Attention Mechanisms
 - (Query, Key, Value)
 - Attention on Text and Images
- Transformers: Multi-head Attention
 - Transformer
 - BERT

RNNs in Various Forms



RNNs in Various Forms

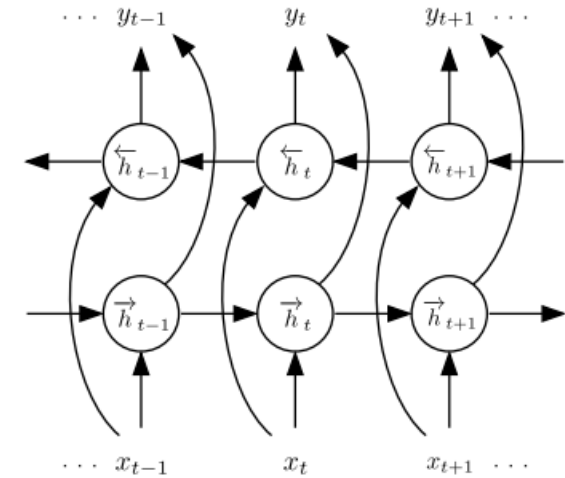
- Bi-directional RNN
 - Hidden state is the concatenation of both forward and backward hidden states.
 - Allows the hidden state to capture both **past** and **future** information.



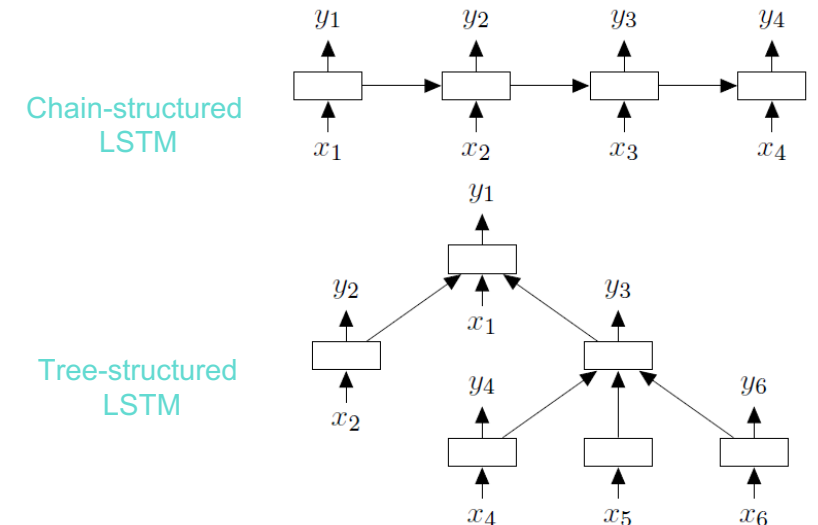
[Speech Recognition with Deep Recurrent Neural Networks, Alex Graves]

RNNs in Various Forms

- Bi-directional RNN
 - Hidden state is the concatenation of both forward and backward hidden states.
 - Allows the hidden state to capture both **past** and **future** information.
- Tree-structured RNN
 - Hidden states condition on both an input vector and the hidden states of **arbitrarily** many child units.
 - Standard LSTM = a special case of tree-LSTM where each internal node has exactly one child.



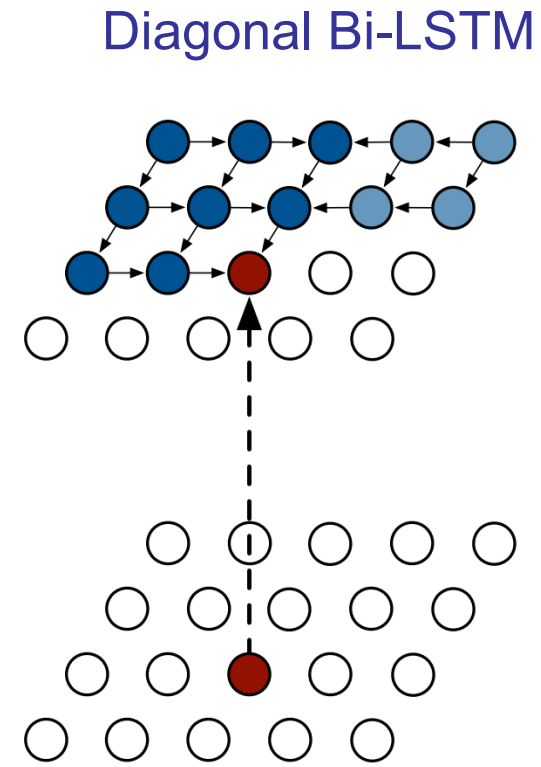
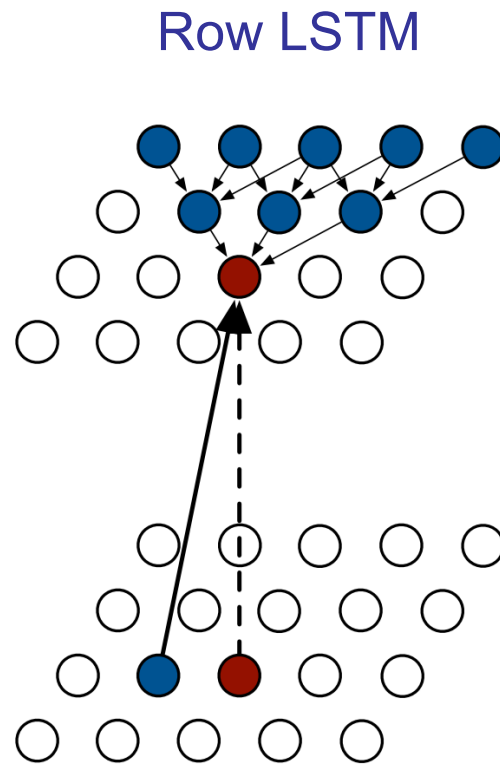
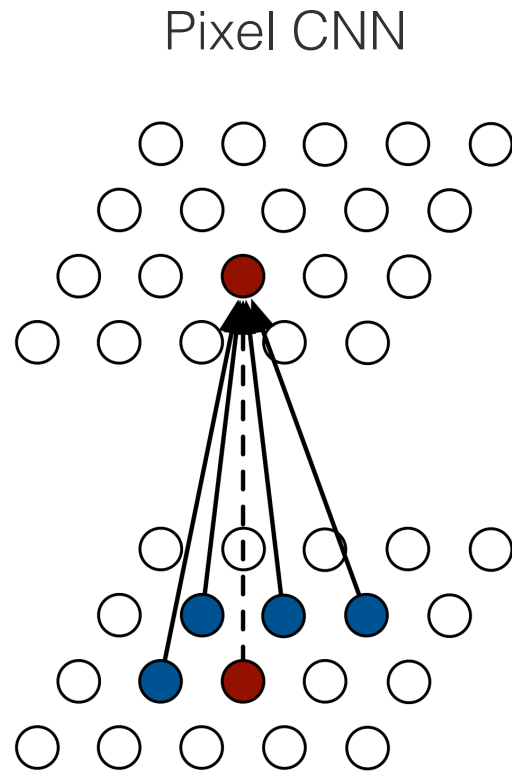
[Speech Recognition with Deep Recurrent Neural Networks, Alex Graves]



Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks, Tai. et al.

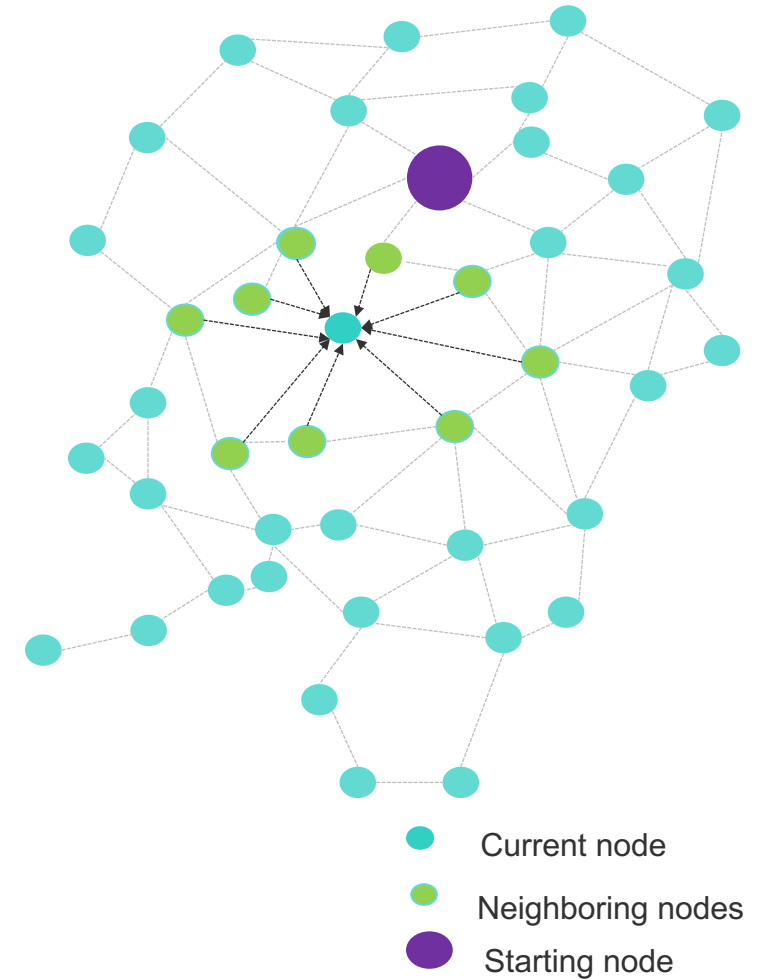
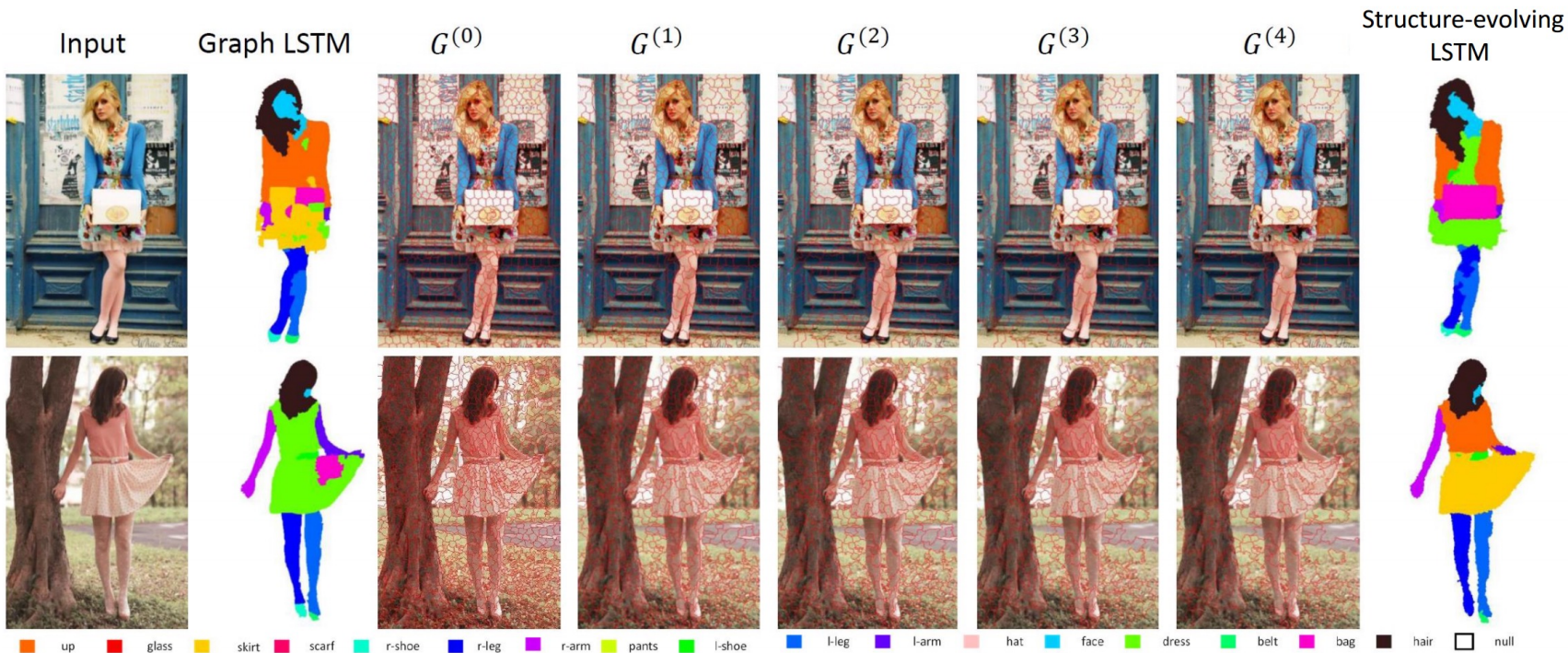
RNNs in Various Forms

- RNN for 2-D sequences



RNNs in Various Forms

- RNN for Graph Structures
 - Used in, e.g., image segmentation

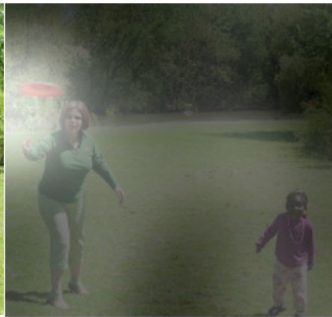


Outline

- Convolutional Networks (ConvNets)
- Recurrent Networks (RNNs)
 - Long-range dependency, vanishing
 - LSTM
 - RNNs in different forms
- Attention Mechanisms
 - (Query, Key, Value)
 - Attention on Text and Images
- Transformers: Multi-head Attention
 - Transformer
 - BERT

Attention: Examples

- Chooses which features to pay attention to



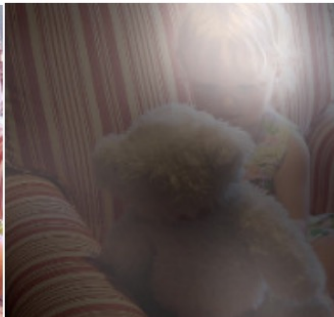
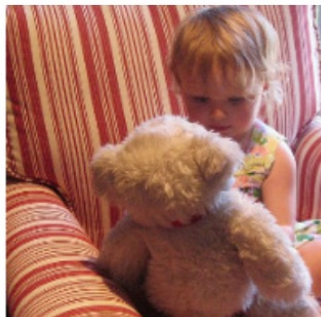
A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



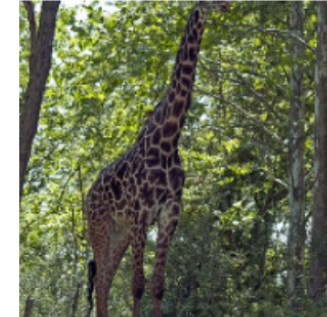
A stop sign is on a road with a mountain in the background.



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



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

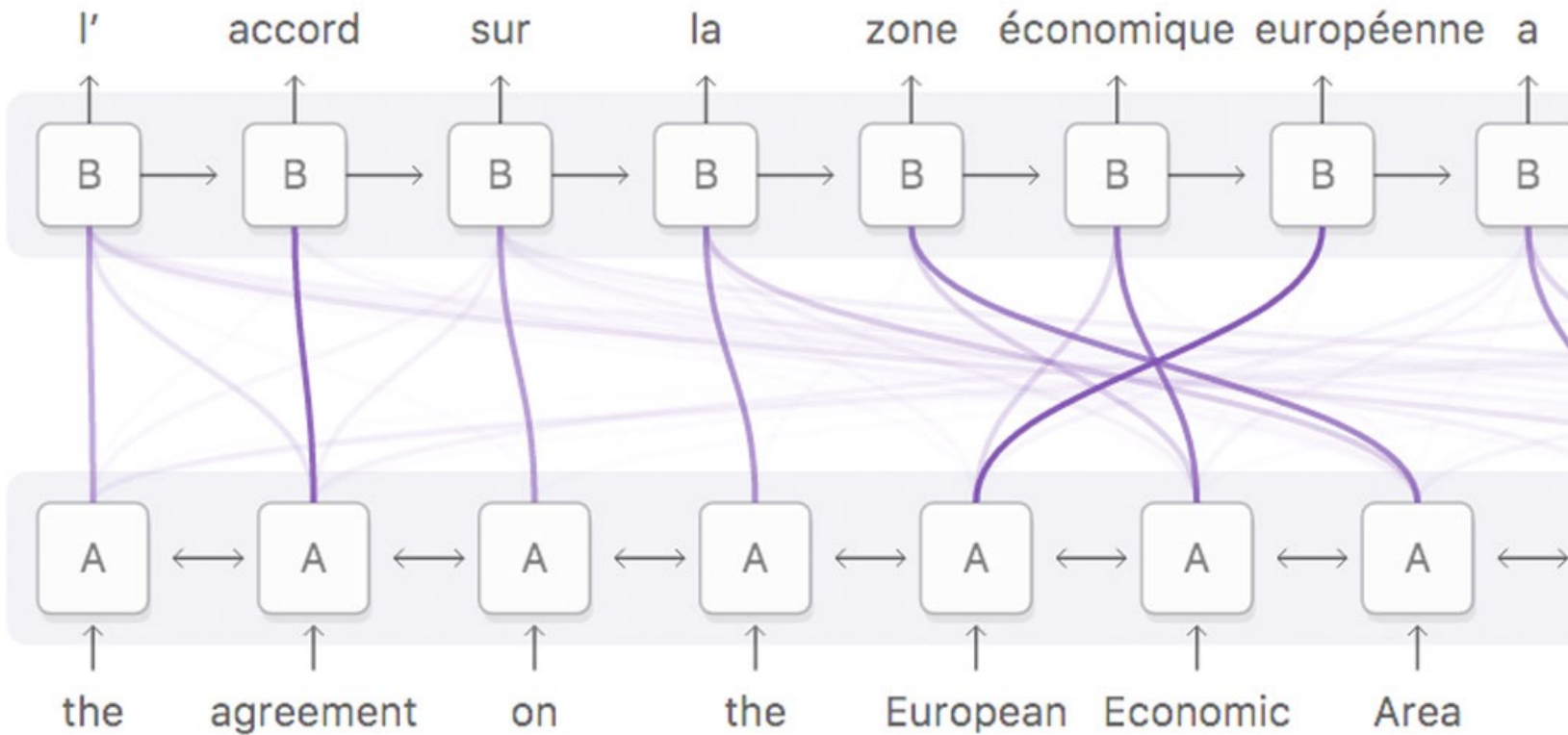


A giraffe standing in a forest with trees in the background.

Image captioning [Show, attend and tell. Xu et al. 15]

Attention: Examples

- Chooses which features to pay attention to

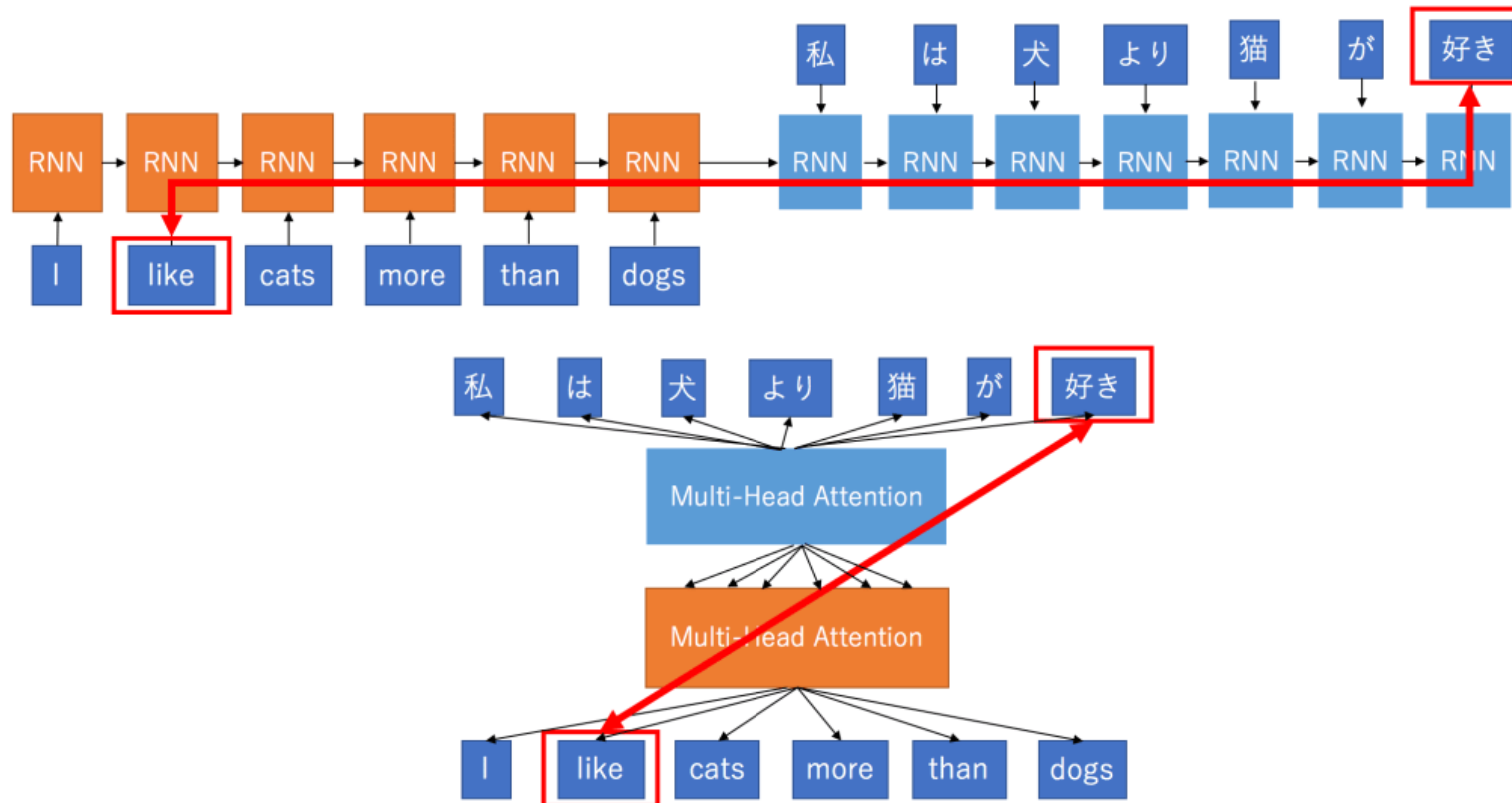


Machine Translation

Why Attention?

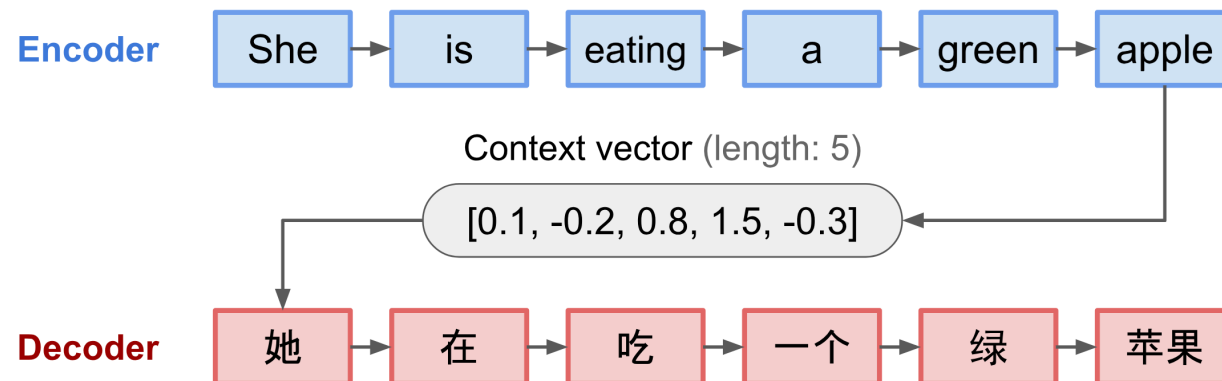
Why Attention?

- Long-range dependencies
 - Dealing with gradient vanishing problem



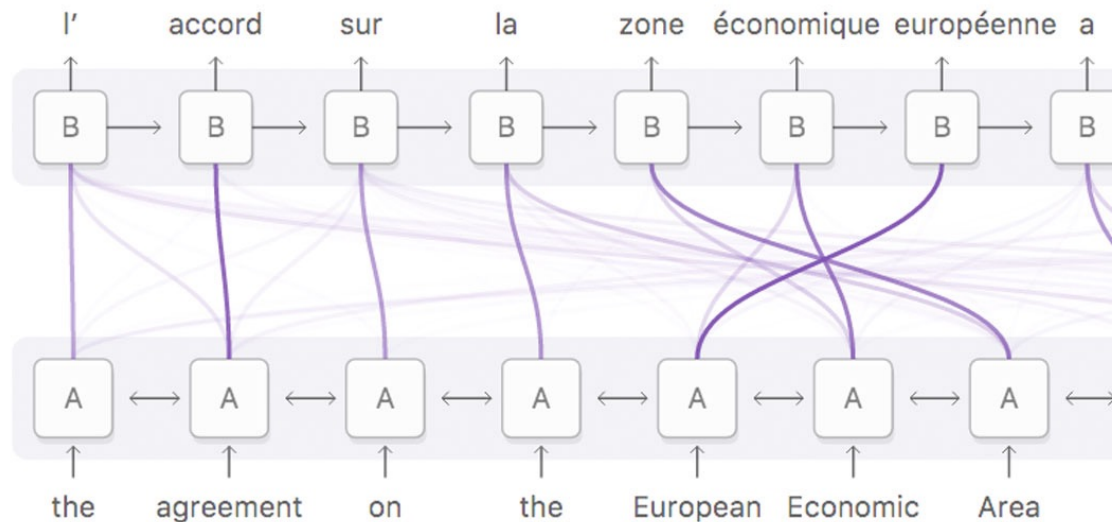
Why Attention?

- Long-range dependencies
 - Dealing with gradient vanishing problem
- Fine-grained representation instead of a single global representation
 - Attending to smaller parts of data: patches in images, words in sentences



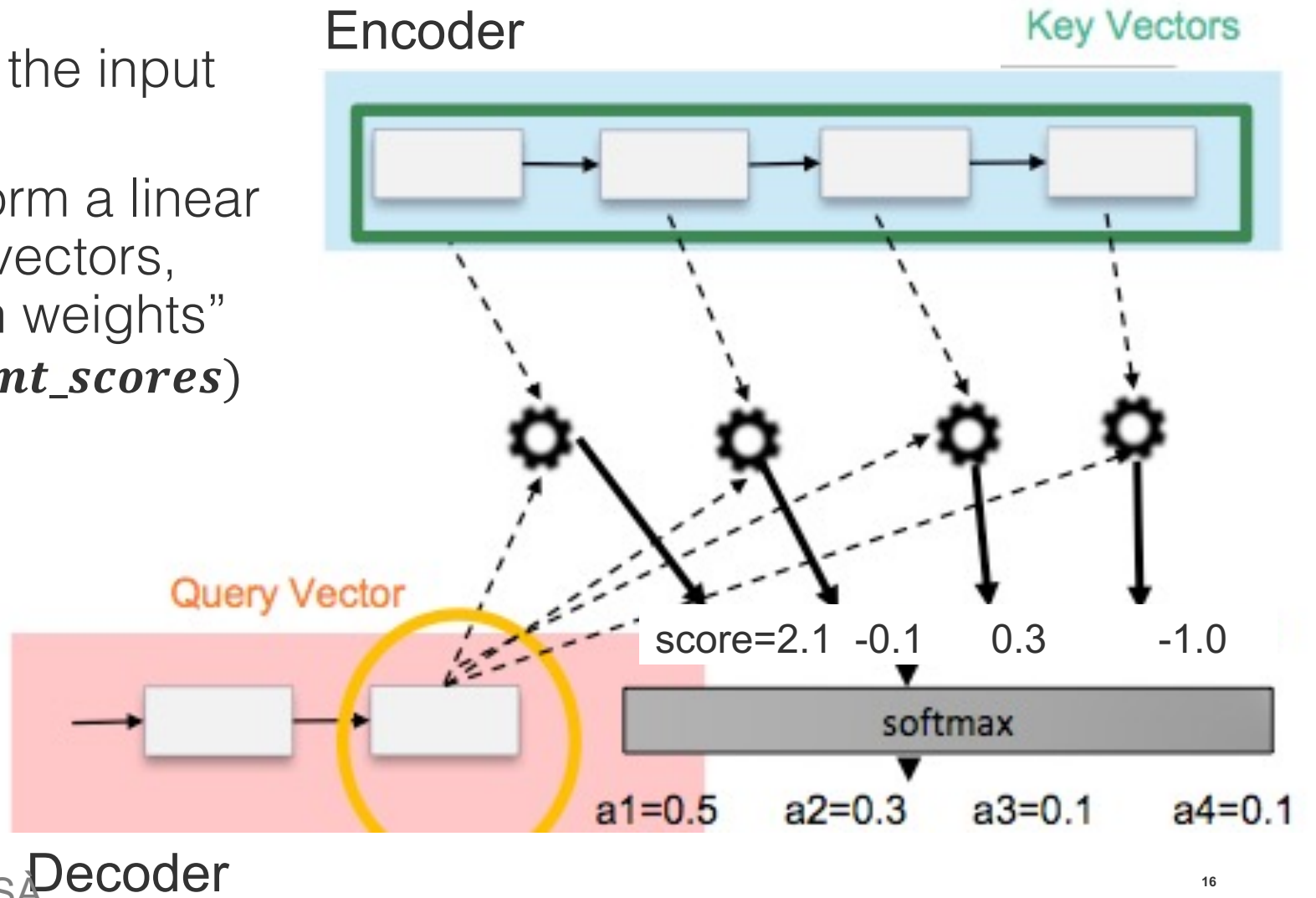
Why Attention?

- Long-range dependencies
 - Dealing with gradient vanishing problem
- Fine-grained representation instead of a single global representation
 - Attending to smaller parts of data: patches in images, words in sentences
- Improved Interpretability



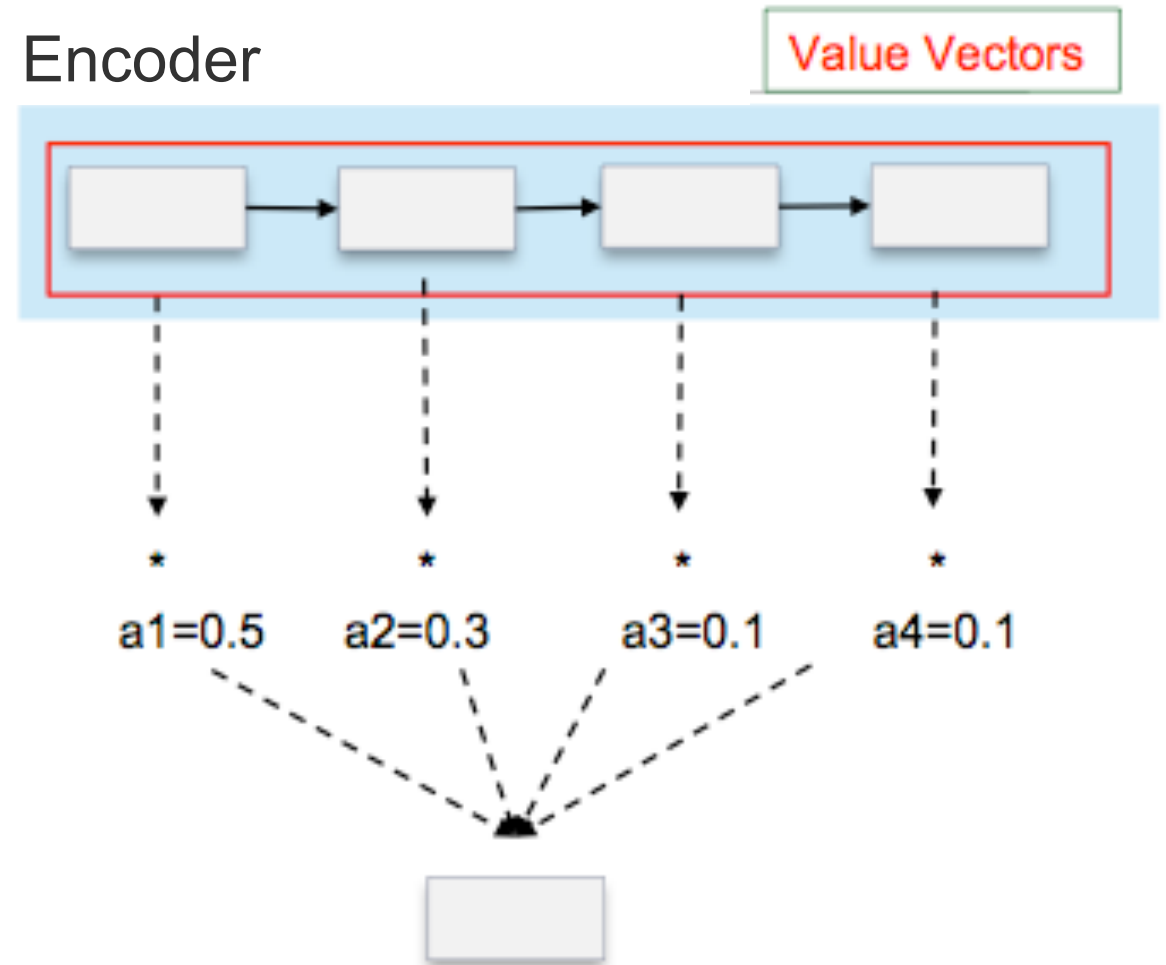
Attention Computation

- Encode each token in the input sentence into vectors
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”
 - $\mathbf{a} = \text{softmax}(\text{alignment_scores})$



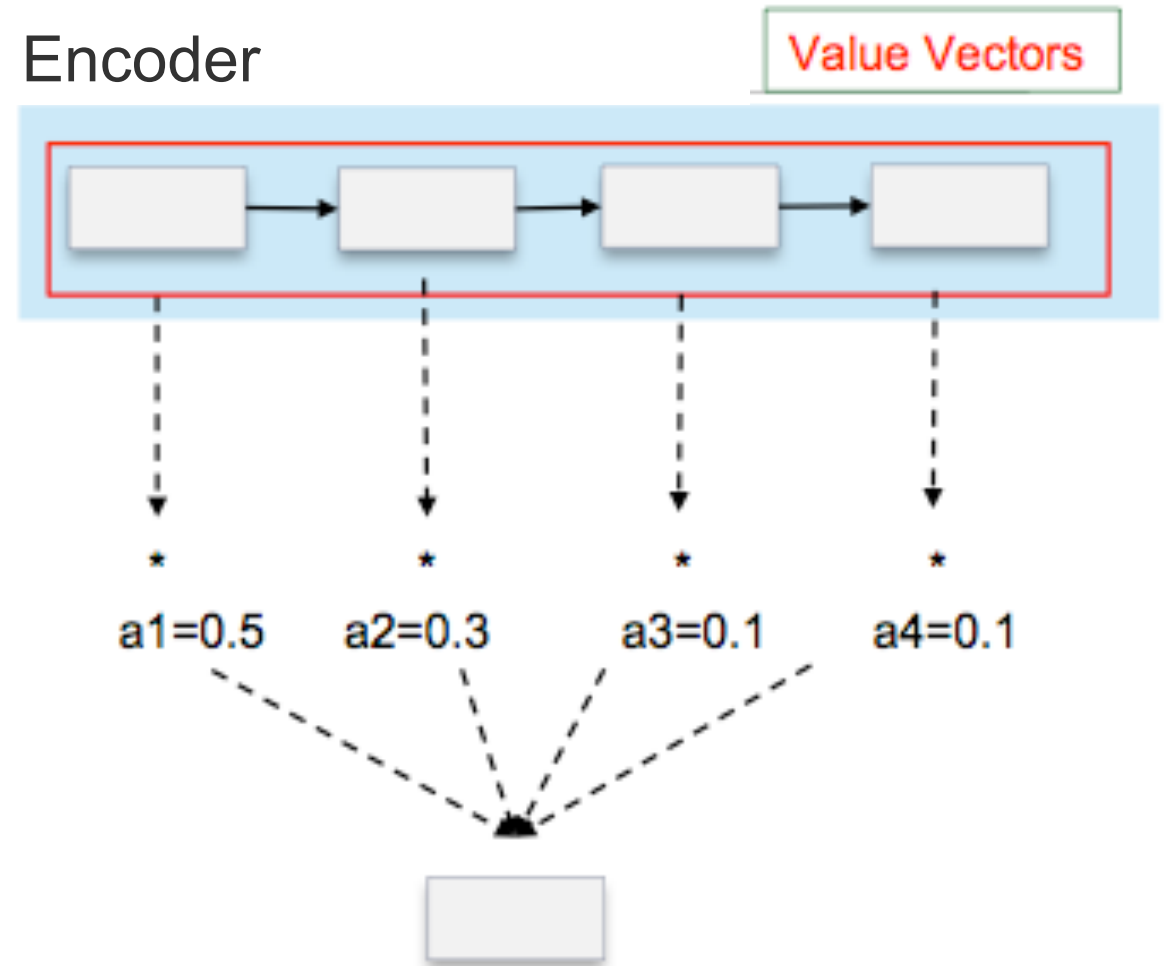
Attention Computation (cont'd)

- Combine together value by taking the weighted sum



Attention Computation (cont'd)

- Combine together value by taking the weighted sum
- Query: decoder state
- Key: all encoder states
- Value: all encoder states



Attention Variants

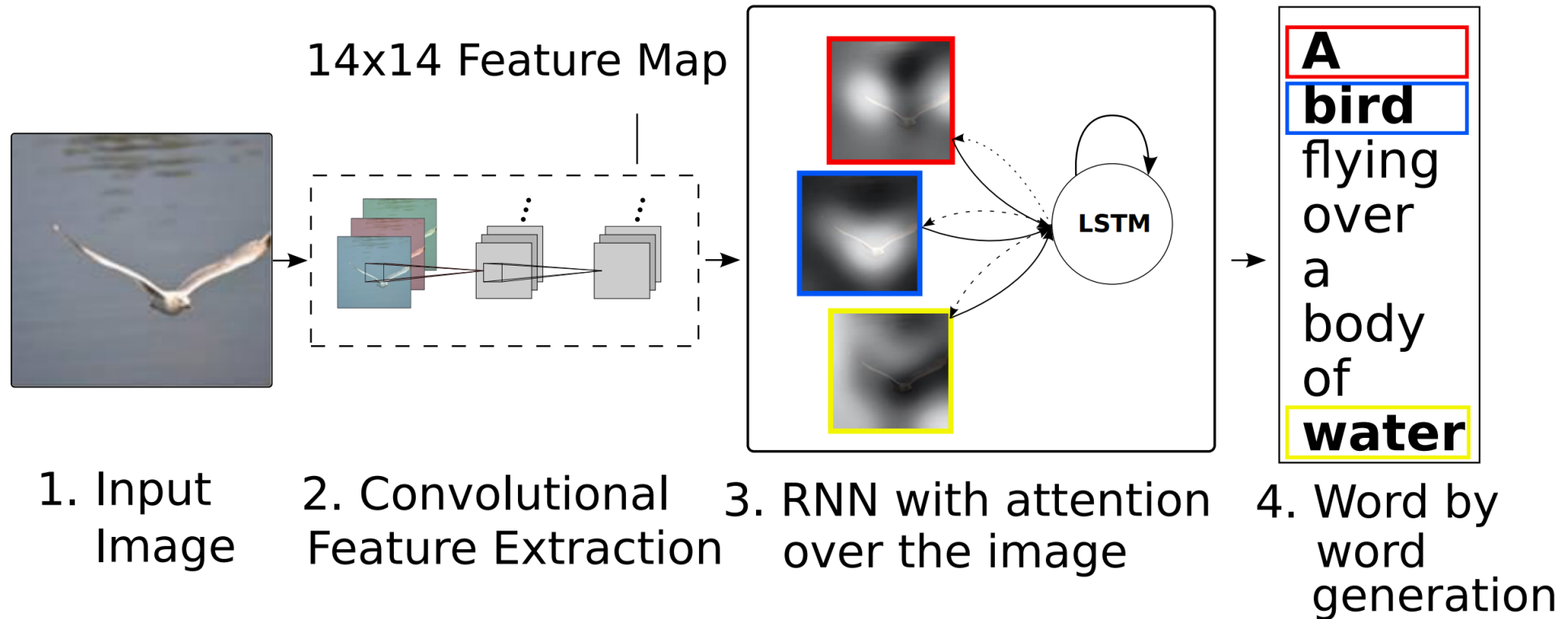
- Popular attention mechanisms with different alignment score functions

Alignment score = f(Query, Keys)

- Query: decoder state s_t
- Key: all encoder states h_i
- Value: all encoder states h_i

Name	Alignment score function	Citation
Content-base attention	$\text{score}(s_t, h_i) = \text{cosine}[s_t, h_i]$	Graves2014
Additive(*)	$\text{score}(s_t, h_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a [s_t; h_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a s_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(s_t, h_i) = s_t^\top \mathbf{W}_a h_i$ where \mathbf{W}_a is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(s_t, h_i) = s_t^\top h_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(s_t, h_i) = \frac{s_t^\top h_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Attention on Images: Image Captioning

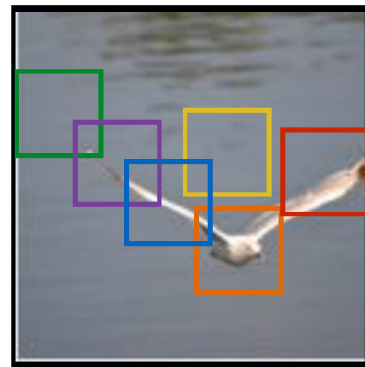


- Query: decoder state
- Key: visual feature maps
- Value: visual feature maps

Attention on Images: Image Captioning

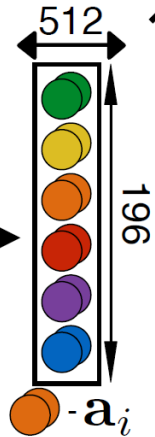
Hard attention vs Soft attention

A bird flying over a body of water.



conv-512
conv-512
maxpool

14x14x512 =
196 x 512 (L x D)
annotations



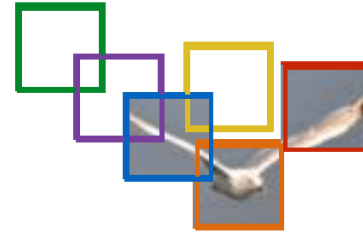
Hard

Soft

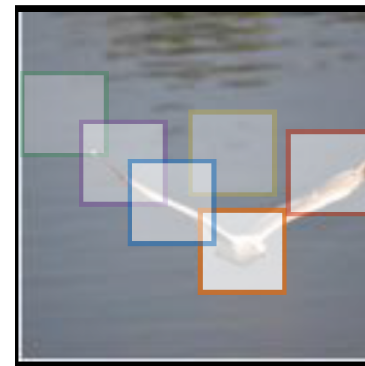
$\hat{\mathbf{z}}_t = \phi(\{\mathbf{a}_i\}, \{\alpha_i\})$

Sample regions of attention

$\hat{\mathbf{z}}_t = \text{orange}, \text{orange}, \text{red}, \text{blue}$



$$L_z = \sum_{z \in \{\text{orange}, \text{orange}, \text{red}, \text{blue}\}} \log p(\mathbf{y} | z)$$



$$L_s = \sum_s p(s | \mathbf{a}) \log p(\mathbf{y} | s, \mathbf{a})$$

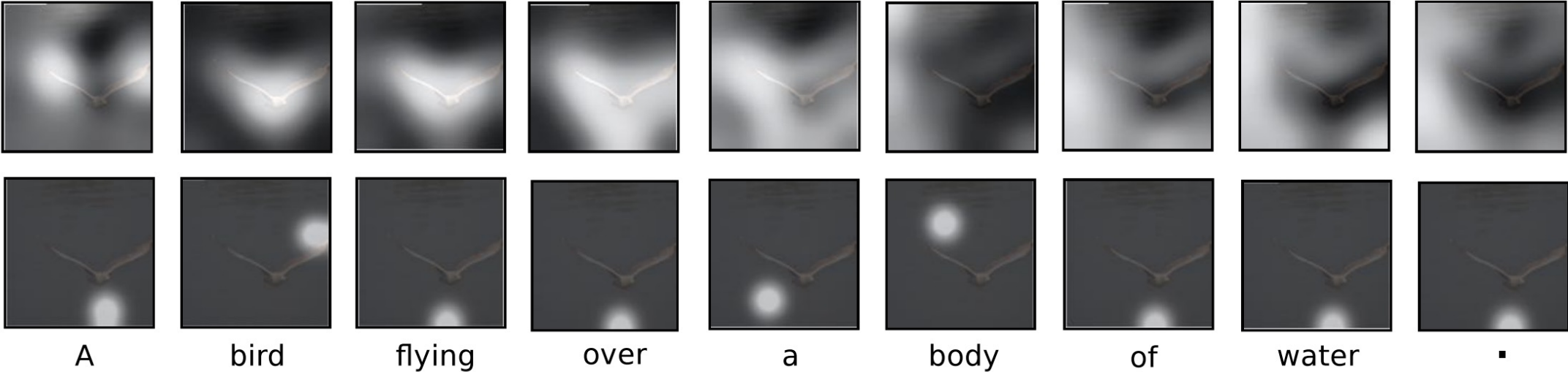
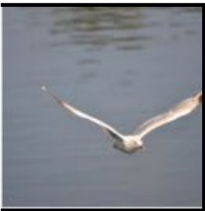
A variational lower bound of maximum likelihood

$\hat{\mathbf{z}}_t = \langle [p_1 \ p_2 \ p_3 \ p_4 \ p_5 \ p_6], [\text{green}, \text{yellow}, \text{orange}, \text{red}, \text{purple}, \text{blue}] \rangle$

Computes the expected attention

Attention on Images: Image Captioning

Hard attention vs Soft attention



Attention on Images: Image Paragraph Generation

- Generate a long paragraph to describe an image
 - Long-term visual and language reasoning
 - Contentful descriptions -- ground sentences on visual features



This picture is taken for three baseball players on a field. The man on the left is wearing a blue baseball cap. The man has a red shirt and white pants. The man in the middle is in a wheelchair and holding a baseball bat. Two men are bending down behind a fence. There are words band on the fence.

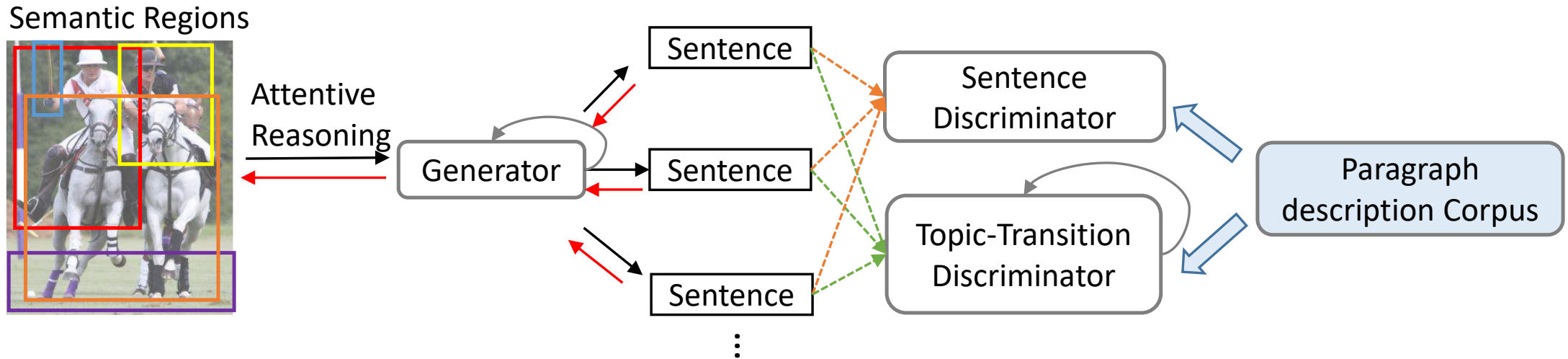


A tennis player is attempting to hit the tennis ball with his left foot hand. He is holding a tennis racket. He is wearing a white shirt and white shorts. He has his right arm extended up. There is a crowd of people watching the game. A man is sitting on the chair.

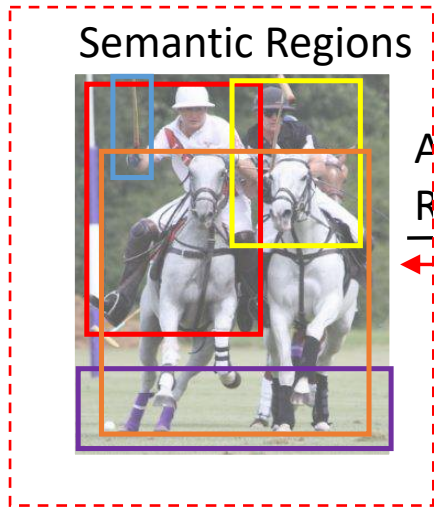


A couple of zebra are standing next to each other on dirt ground near rocks. There are trees behind the zebras. There is a large log on the ground in front of the zebra. There is a large rock formation to the left of the zebra. There is a small hill near a small pond and a wooden log. There are green leaves on the tree.

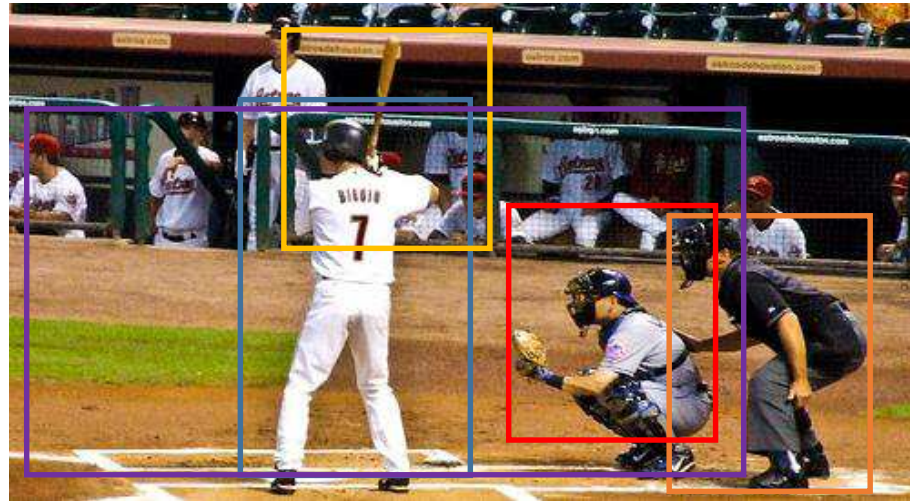
Attention on Images: Image Paragraph Generation



Attention on Images: Image Paragraph Generation



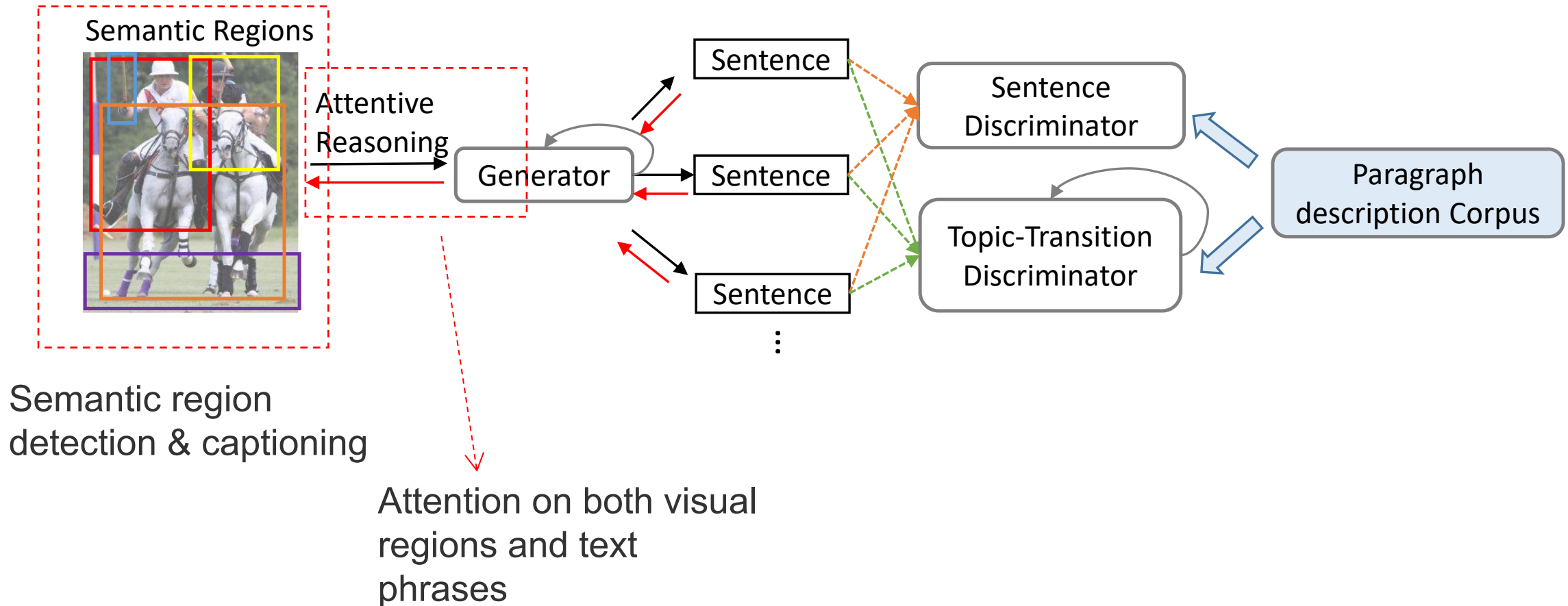
Semantic region
detection & captioning



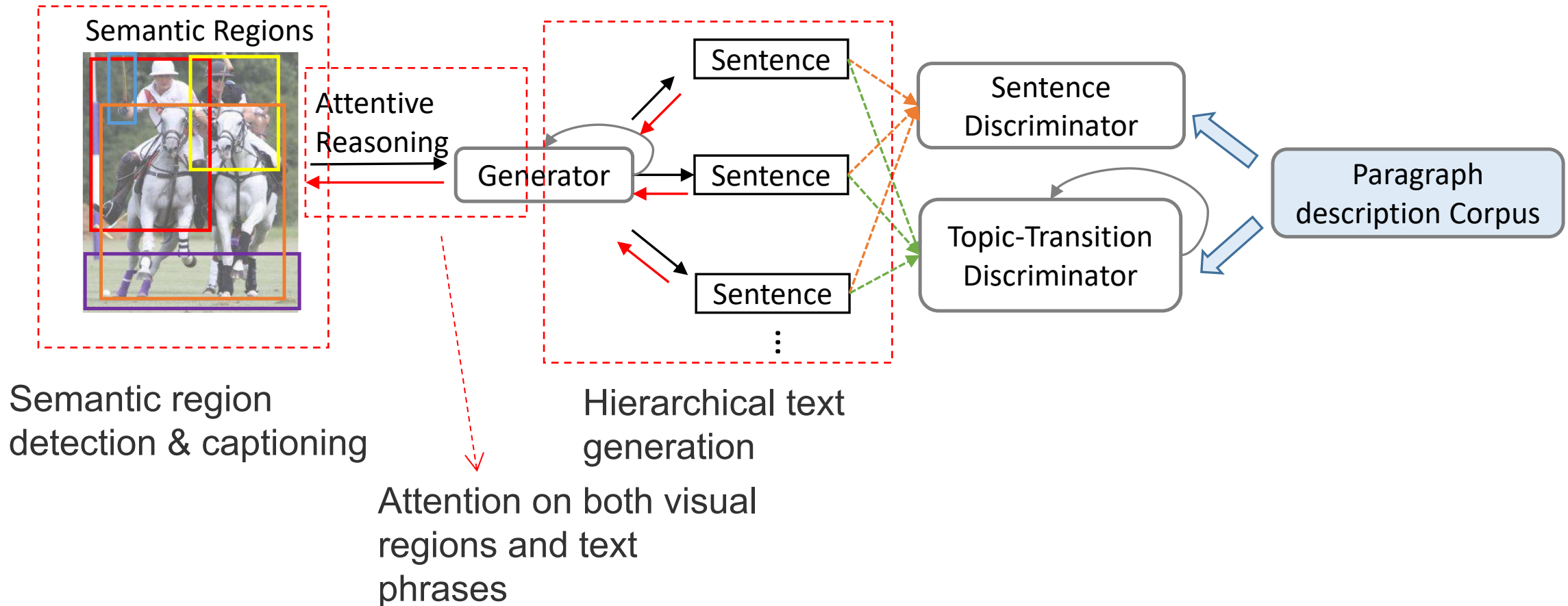
**Local
Phrases**

- people playing baseball
- a man wearing white shirt and pants
- man holding a baseball bat
- person wearing a helmet in the field
- a man bending over

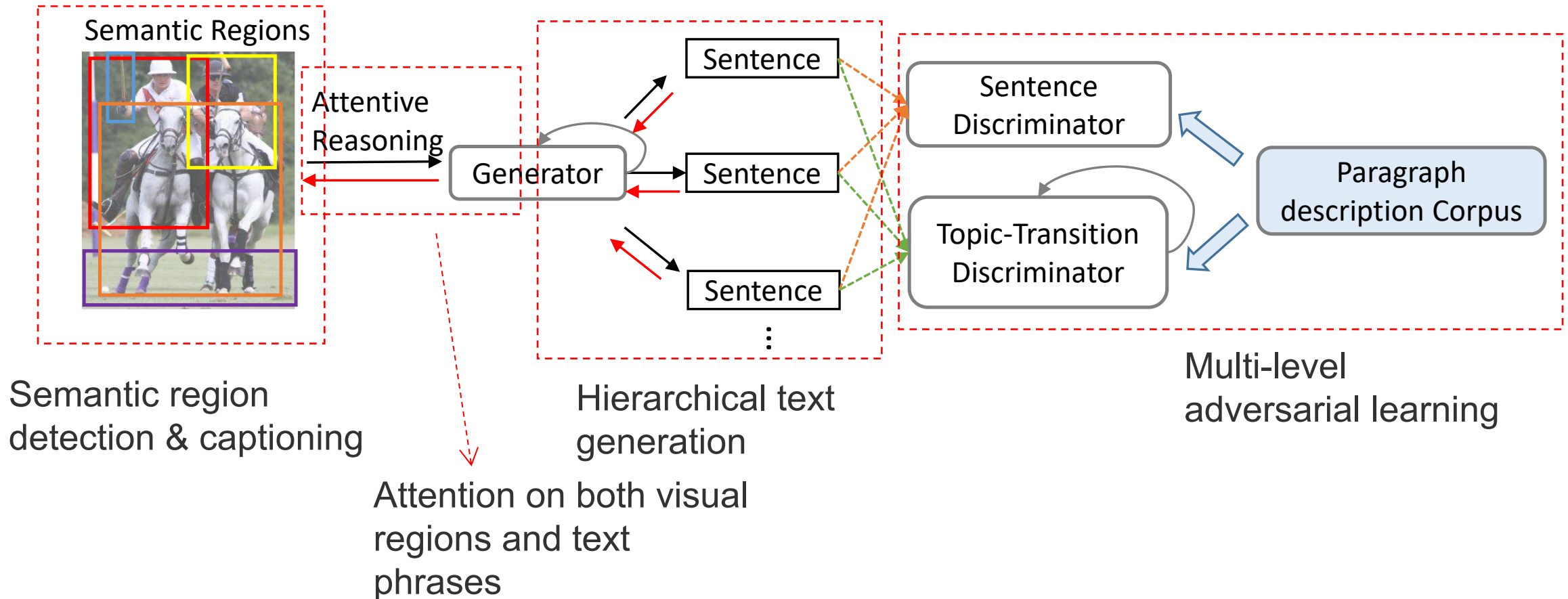
Attention on Images: Image Paragraph Generation



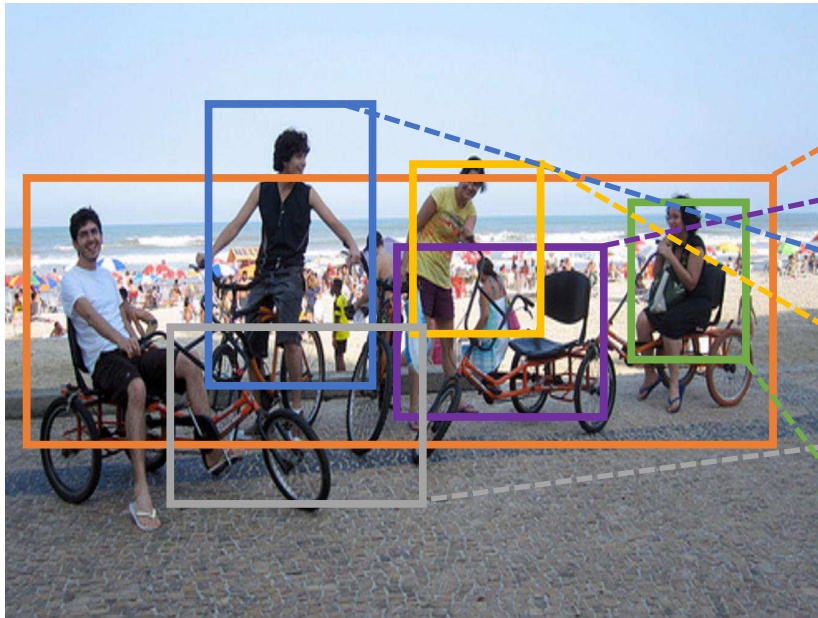
Attention on Images: Image Paragraph Generation



Attention on Images: Image Paragraph Generation



Attention on Images: Image Paragraph Generation



- 1) people riding a bike
- 2) a bicycle parked on the sidewalk
- 3) man wearing a black shirt
- 4) a woman wearing a yellow shirt
- 5) a red and black bike
- 6) a woman wearing a shirt

Paragraph: *A group of people are riding bikes. There are two people riding bikes parked on the sidewalk. He is wearing a black shirt and jeans. A woman is wearing a short sleeve yellow shirt and shorts. There are many other people on the red and black bikes. A woman wearing a shirt is riding a bicycle.*

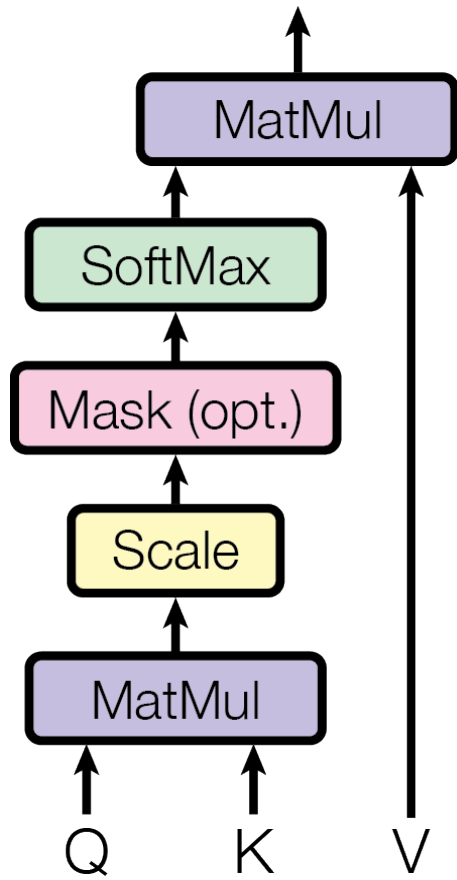
Outline

- Convolutional Networks (ConvNets)
- Recurrent Networks (RNNs)
 - Long-range dependency, vanishing gradients
 - LSTM
 - RNNs in different forms
- Attention Mechanisms
 - (Query, Key, Value)
 - Attention on Text and Images
- Transformers: Multi-head Attention

Transformers – Multi-head (Self-)Attention

- State-of-the-art Results by Transformers
 - [Vaswani et al., 2017] Attention Is All You Need
 - Machine Translation
 - [Devlin et al., 2018] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
 - Pre-trained Text Representation
 - [Radford et al., 2019] Language Models are Unsupervised Multitask Learners
 - Language Models

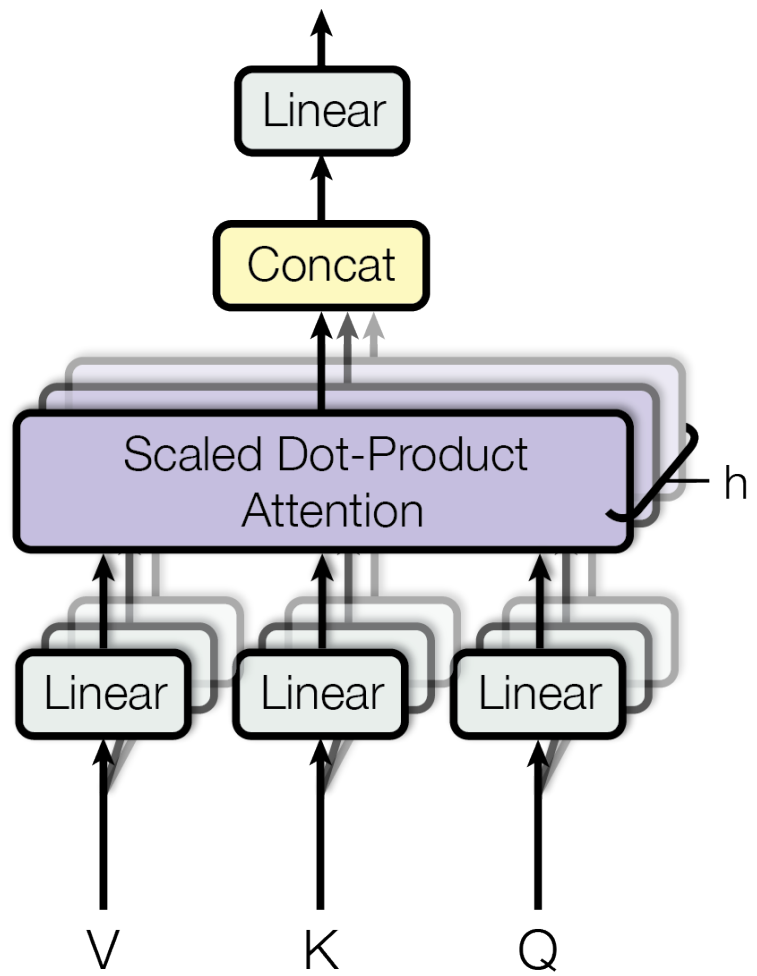
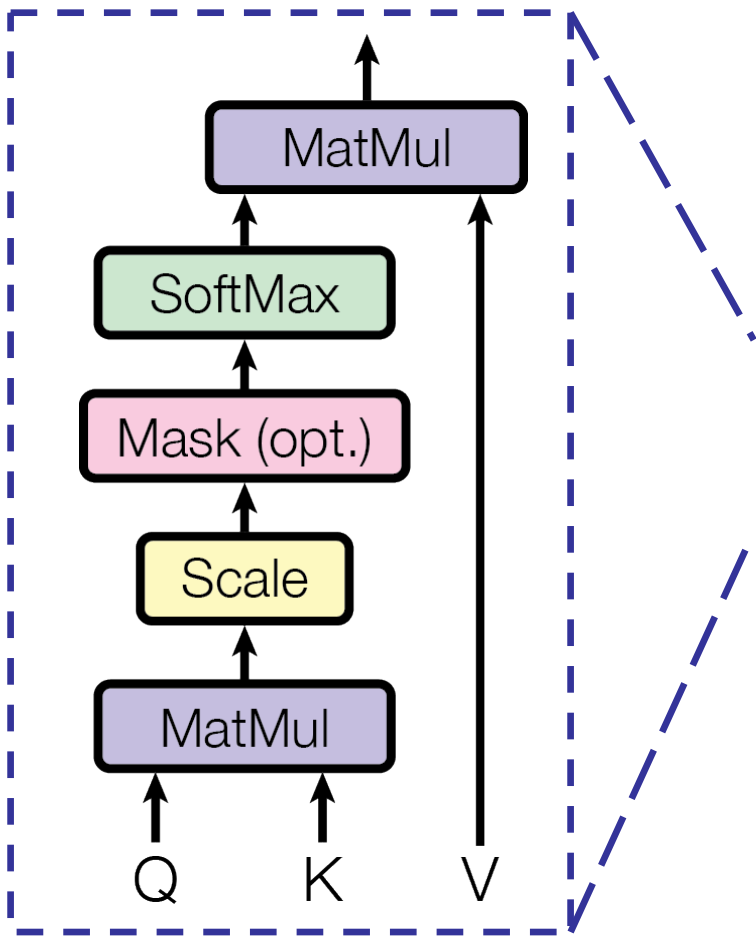
Multi-head Attention



Scaled Dot-Product Attention

Image source: [Vaswani, et al., 2017](#)

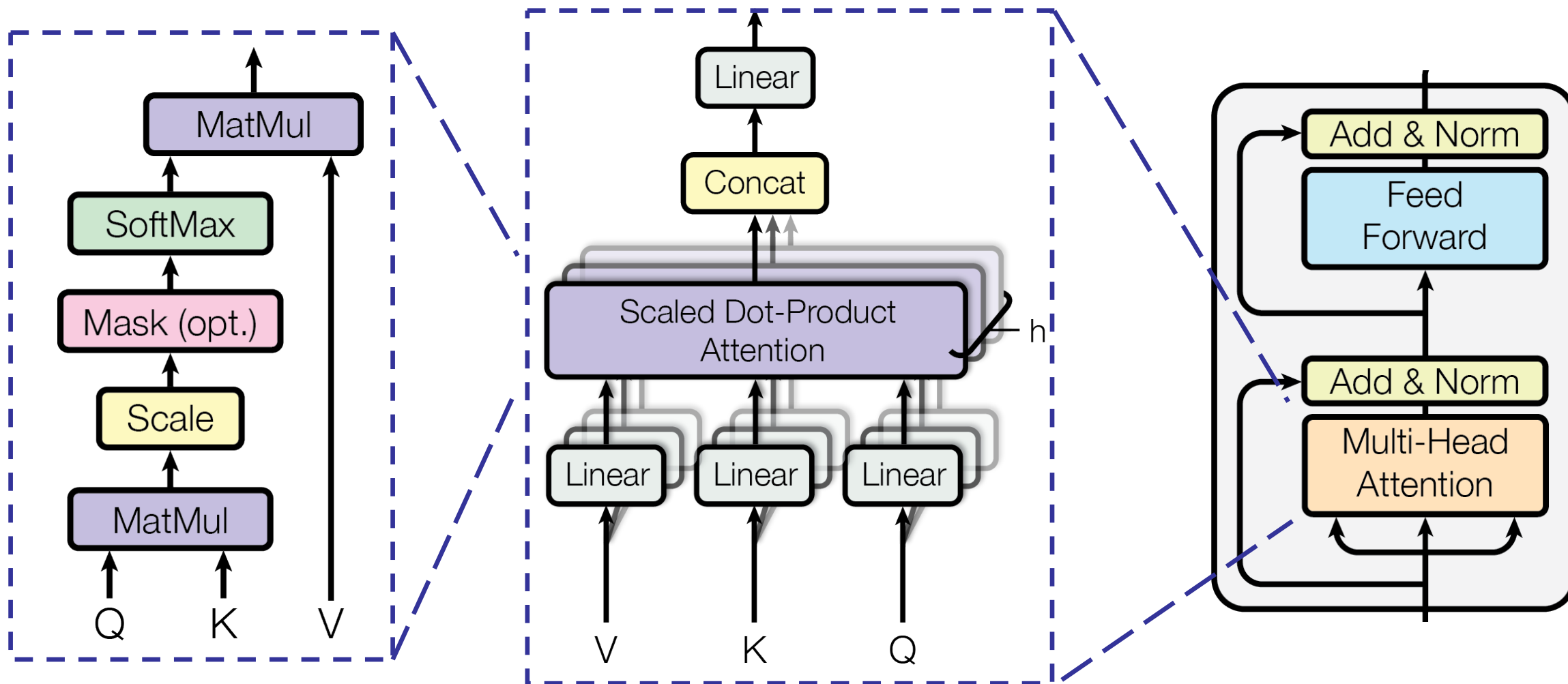
Multi-head Attention



Scaled Dot-Product Attention

Multi-head Attention

Multi-head Attention

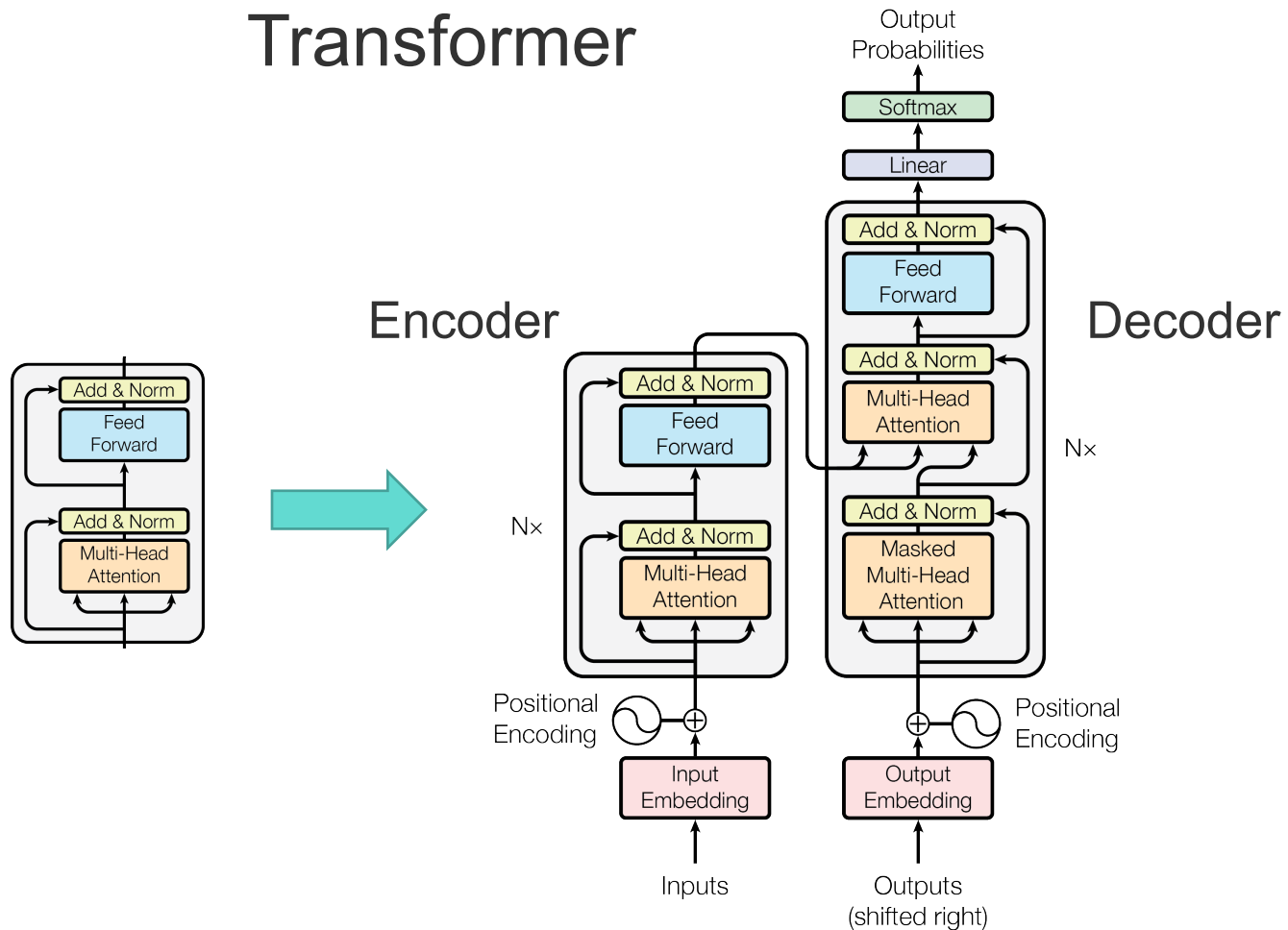


Scaled Dot-Product Attention

Multi-head Attention

Multi-head Attention in Encoders and Decoders

Transformer



Multi-head Attention in Encoders and Decoders

Transformer

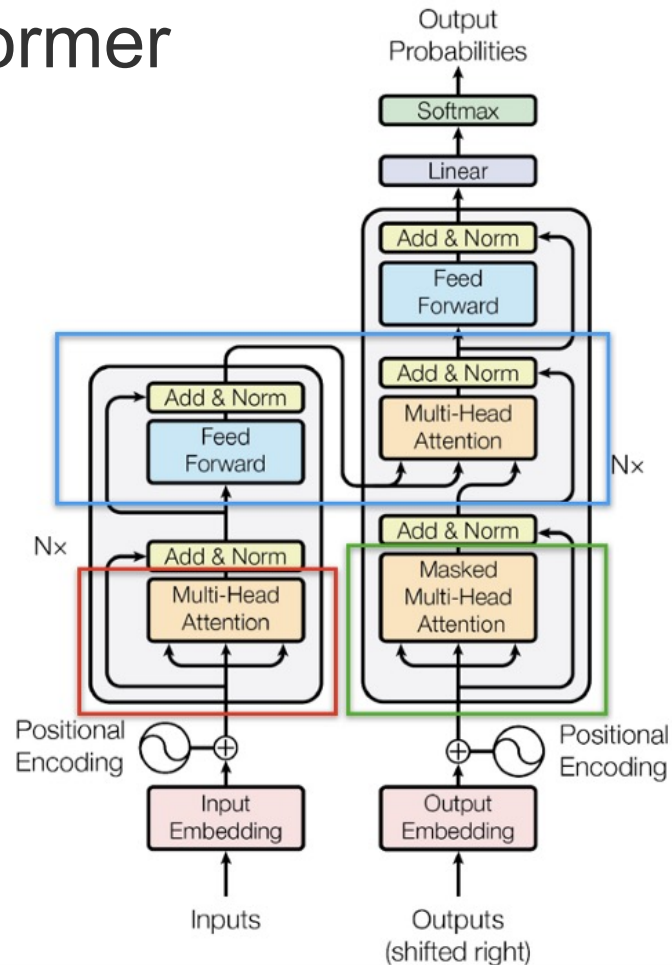


Figure 1: The Transformer - model architecture.

encoder self attention

1. Multi-head Attention
2. **Q**uery=**K**ey=**V**alue

decoder self attention

1. **M**asked Multi-head Attention
2. **Q**uery=**K**ey=**V**alue

encoder-decoder attention

1. Multi-head Attention
2. Encoder Self attention=**K**ey=**V**alue
3. Decoder Self attention=**Q**uery

Key Takeaways: Neural Architectures

- Convolutional Networks (ConvNets)
- Recurrent Networks (RNNs)
 - LSTM designed for long-range dependency, vanishing gradients
 - RNNs not only for sequence data, but also 2D sequences, trees, graphs
- Attention Mechanisms
 - Three core elements: (Query, Key, Value)
 - Many variants based on alignment score functions
 - Attention on text and images
- Transformers: Multi-head Attention

Key Takeaways: Neural Language Models

- $p(X_i | \mathbf{X}_{1:i-1} = \mathbf{x}_{1:i-1}) = \mathbf{NN}(\mathbf{enc}(\mathbf{x}_{1:i-1}); \boldsymbol{\theta})$
- Two key components:
 - “Embedding” words as vectors
 - Layering to increase capacity (i.e., the set of distributions that can be represented)
- Training: maximum likelihood estimation

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{MLE}} = -\log p_{\boldsymbol{\theta}}(\mathbf{x}^*) = -\prod_{i=1}^N p_{\boldsymbol{\theta}}(x_i^* | \mathbf{x}_{1:i-1}^*)$$

- Next word prediction

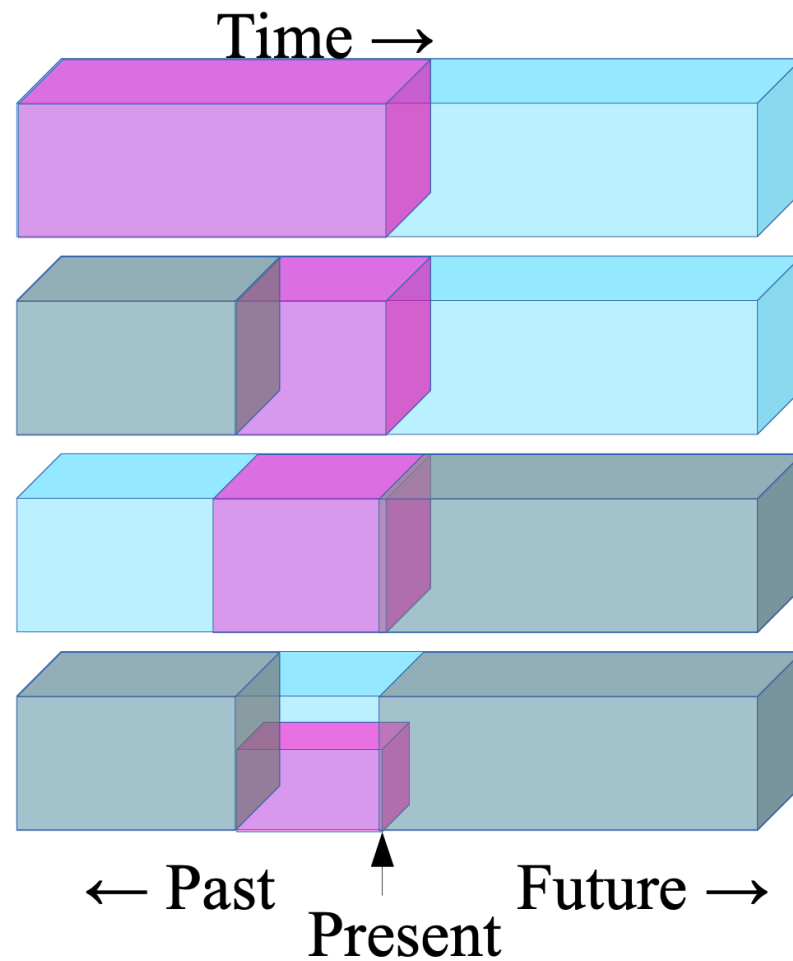
Representation Learning with Self-supervised Learning

Self-Supervised Learning

- Given an observed data instance \mathbf{t}
- One could derive various supervision signals based on the structure of the data
- By applying a “split” function that artificially partition \mathbf{t} into two parts
 - $(\mathbf{x}, \mathbf{y}) = \text{split}(\mathbf{t})$
 - sometimes split in a stochastic way
- Treat \mathbf{x} as the input and \mathbf{y} as the output
- Train a model $p_{\theta}(\mathbf{y}|\mathbf{x})$

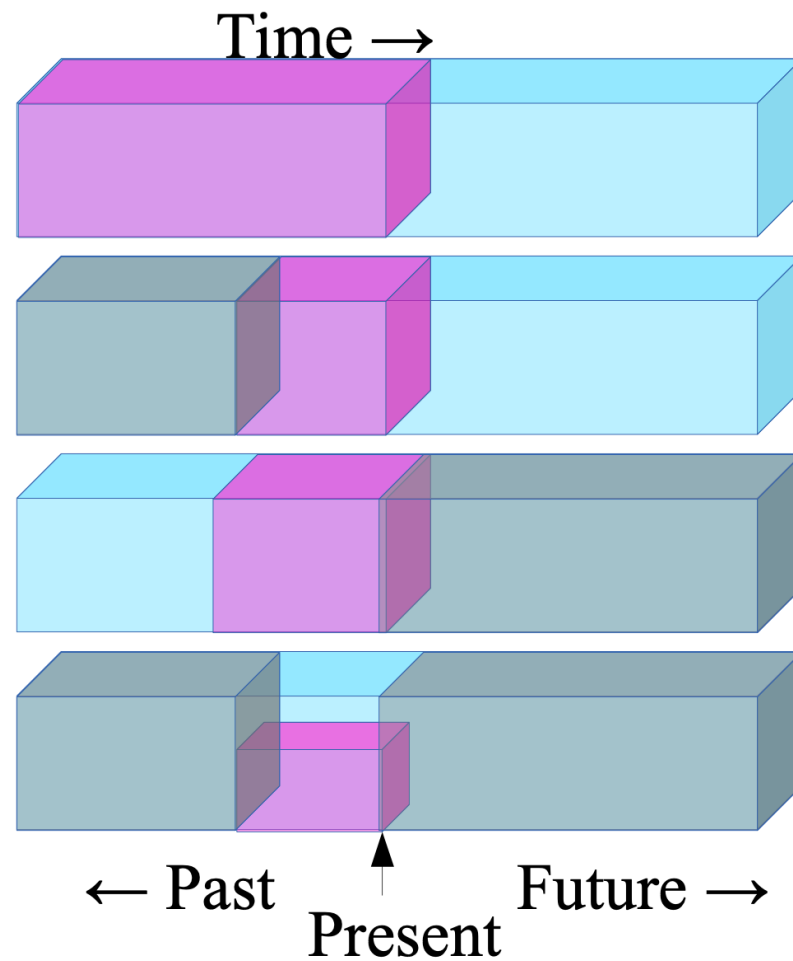
Self-Supervised Learning: Examples

- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.



Self-Supervised Learning: Examples

- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the occluded from the visible
- ▶ **Pretend there is a part of the input you don't know and predict that.**



Self-Supervised Learning: Motivation (I)

- ▶ **Our brains do this all the time**
- ▶ **Filling in the visual field at the retinal blind spot**
- ▶ **Filling in occluded images, missing segments in speech**
- ▶ **Predicting the state of the world from partial (textual) descriptions**
- ▶ **Predicting the consequences of our actions**
- ▶ **Predicting the sequence of actions leading to a result**
- ▶ **Predicting any part of the past, present or future percepts from whatever information is available.**



Self-Supervised Learning: Motivation (I)

- Successfully learning to predict everything from everything else would result in **the accumulation of lots of background knowledge about how the world works**
- The model is forced to learn what we really care about, e.g. a semantic representation, in order to solve the prediction problem

[Courtesy: Lecun “Self-supervised Learning”]

[Courtesy: Zisserman “Self-supervised Learning”]

Self-Supervised Learning: Motivation (II)

- The machine predicts any part of its input from any observed part
 - A lot of supervision signals in each data instance
- Untapped/availability of vast numbers of unlabeled text/images/videos..
 - Facebook: one billion images uploaded per day
 - 300 hours of video are uploaded to YouTube every minute

Self-Supervised Learning from Text

Examples:

- Language models
- Learning text representations

Language Models: Training

- Given data example \mathbf{y}^*
- Minimizes negative log-likelihood of the data

$$\min_{\theta} \mathcal{L}_{\text{MLE}} = -\log p_{\theta}(\mathbf{y}^*) = -\prod_{t=1}^T p_{\theta}(y_t^* | \mathbf{y}_{1:t-1}^*)$$

- Next word prediction
- Inference: teacher-forcing decoding
 - For every step t , feed in the previous ground-truth tokens $\mathbf{y}_{1:t-1}^*$ to decode next step

Language Models: GPT3

- A Transformer-based LM with 125M to 175B parameters
- Trained on massive text data

Dataset	# Tokens (Billions)
Total	499
Common Crawl (filtered by quality)	410
WebText2	19
Books1	12
Books2	55
Wikipedia	3

Brown et al., 2020 "Language Models Are Few-Shot Learners"

[Table from <https://lambdalabs.com/blog/demystifying-gpt-3/>]

Language Models: GPT3

- Generation

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.

The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Figure 3.14: The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

Language Models: GPT3

- Few-shot prediction

Prompts in Red. GPT3 responses in Blue

Q. How do you draw a bicycle?

A. You draw a bicycle with two circles connected by a line.

Q. How do you draw a ball?

A. You draw a ball with one circle.

Q. How do you draw a house?

A. You draw a house with a trapezium on top of a square.

Q. How do you draw a rabbit?

A. You draw a rabbit with four circles: one large circle for the head, a small one for the nose, a large circle for the body, and two tiny circles for the feet.

Q. How do you draw a snail?

A. You draw a snail with six circles: one large circle for the shell, two small circles for the antennae, one large circle for the head, one small circle for the tail, and a small circle for the mouth.

Q. How do you draw a monkey?

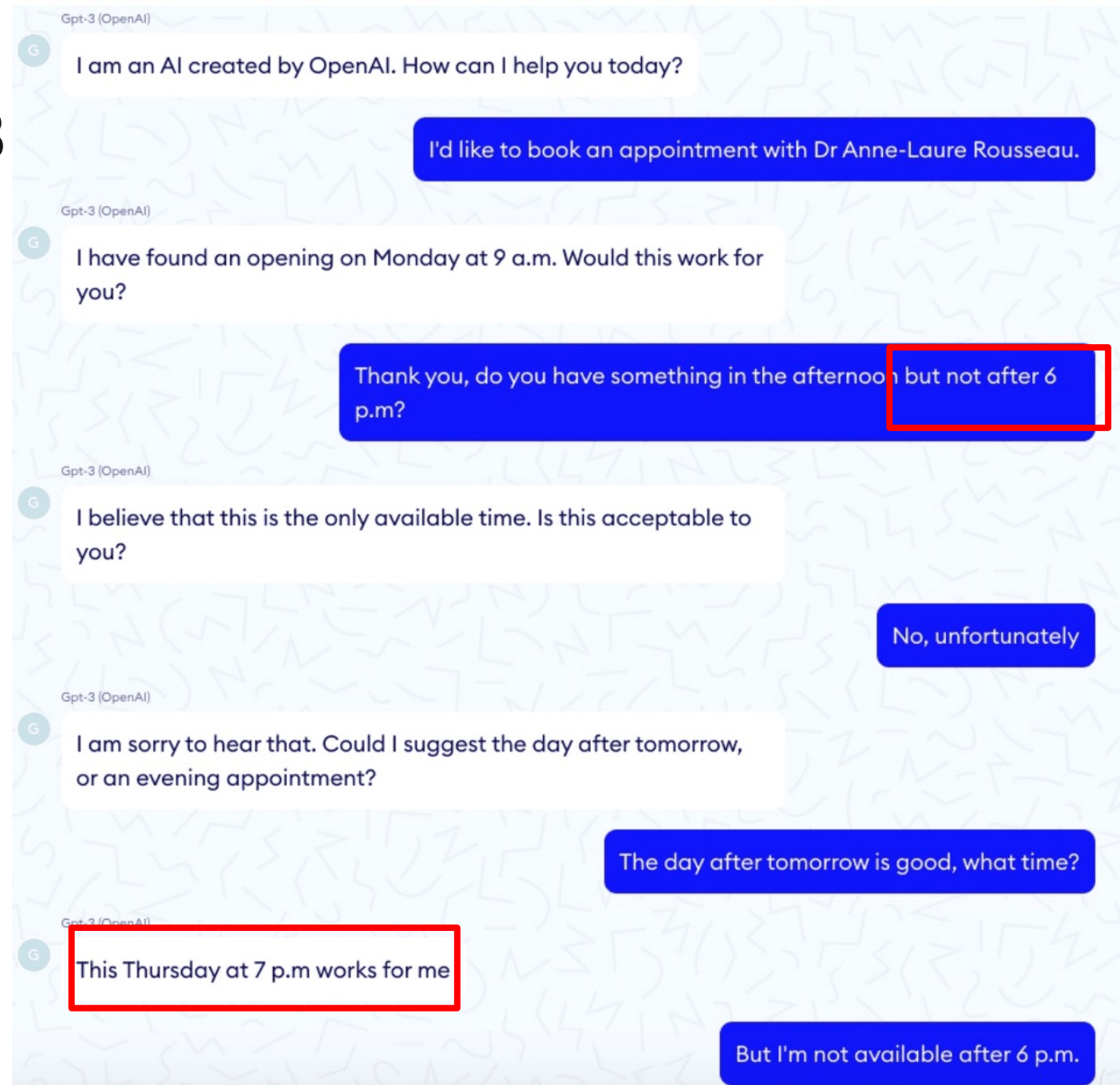
A. You draw a monkey with three circles: one large circle for the head, a small circle for the nose, and a medium-sized circle for the body.

Q. How do you draw a person?

A. You draw a person with three circles: one large circle for the head, a medium-sized circle for the body, and a small circle for the feet.

Language Models: GPT3

- Weakness



Self-Supervised Learning from Text

Examples:

- Language models
- Learning text representations

Word Embedding

- A pre-trained matrix, each row is an embedding vector of a word

	0	1	2	3	4	5	6	7	8	9	..
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	..
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	..
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	..
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	..
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	..
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	..
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	..
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	..
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	..
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	..
toast	0.130740	-0.193730	0.253270	0.090102	-0.272580	-0.030571	0.096945	-0.115060	0.484000	0.848380	..
today	-0.156570	0.594890	-0.031445	-0.077586	0.278630	-0.509210	-0.066350	-0.081890	-0.047986	2.803600	..
blue	0.129450	0.036518	0.032298	-0.060034	0.399840	-0.103020	-0.507880	0.076630	-0.422920	0.815730	..
green	-0.072368	0.233200	0.137260	-0.156630	0.248440	0.349870	-0.241700	-0.091426	-0.530150	1.341300	..
kings	0.259230	-0.854690	0.360010	-0.642000	0.568530	-0.321420	0.173250	0.133030	-0.089720	1.528600	..
dog	-0.057120	0.052685	0.003026	-0.048517	0.007043	0.041856	-0.024704	-0.039783	0.009614	0.308416	..
sausages	-0.174290	-0.064869	-0.046976	0.287420	-0.128150	0.647630	0.056315	-0.240440	-0.025094	0.502220	..
lazy	-0.353320	-0.299710	-0.176230	-0.321940	-0.385640	0.586110	0.411160	-0.418680	0.073093	1.486500	..
love	0.139490	0.534530	-0.252470	-0.125650	0.048748	0.152440	0.199060	-0.065970	0.128830	2.055900	..
quick	-0.445630	0.191510	-0.249210	0.465900	0.161950	0.212780	-0.046480	0.021170	0.417660	1.686900	..

Word Embedding

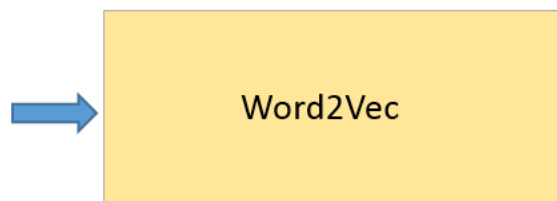
- A pre-trained matrix, each row is an embedding vector of a word

	0	1	2	3	4	5	6	7	8	9	..
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	..
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	..
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	..
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	..
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	..
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	..
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	..
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	..
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	..
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	..

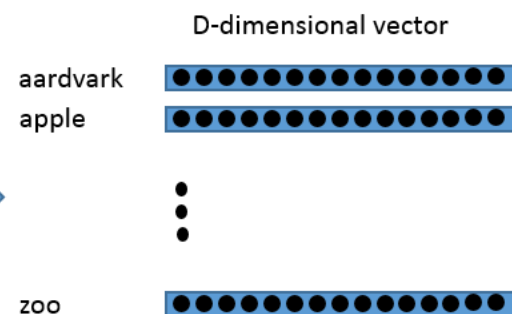
English Wikipedia Corpus

The Annual Reminder continued through July 4, 1969. This final Annual Reminder took place less than a week after the June 28 Stonewall riots, in which the patrons of the Stonewall Inn, a gay bar in Greenwich Village, fought against police who raided the bar. Rodwell received several telephone calls threatening him and the other New York participants, but he was able to arrange for police protection for the chartered bus all the way to Philadelphia. About 45 people participated, including the deputy mayor of Philadelphia and his wife. The dress code was still in effect at the Reminder, but two women from the New York contingent broke from the single-file picket line and held hands. When Kameny tried to break them apart, Rodwell furiously denounced him to onlooking members of the press.

Following the 1969 Annual Reminder, there was a sense, particularly among the younger and more radical participants, that the time for silent picketing had passed. Dissent and dissatisfaction had begun to take new and more emphatic forms in society. "The conference passed a resolution drafted by Rodwell, his partner Fred Sargeant, Broidy and Linda Rhodes to move the demonstration from July 4 in Philadelphia to the last weekend in June in New York City, as well as proposing to "other organizations throughout the country... suggesting that they hold parallel demonstrations on that day" to commemorate the Stonewall riot.



Embedding Matrix



350	-0.081890	-0.047986	2.803600	..
7880	0.076630	-0.422920	0.815730	..
1700	-0.091426	-0.530150	1.341300	..
3250	0.133030	-0.089720	1.528600	..
1704	-0.039783	0.009614	0.308416	..
3315	-0.240440	-0.025094	0.502220	..
1160	-0.418680	0.073093	1.486500	..
1060	-0.065970	0.128830	2.055900	..
3480	0.021170	0.417660	1.686900	..

Word Embedding

- Distributional semantics:
 - Words that are used and occur in the same **contexts** tend to purport similar meanings
- Learning word embedding using local context:
 - Word embedding algorithms are designed to “guess” a word at position i given words at positions in $\{i - w, \dots, i - 1\} \cup \{i + 1, \dots, i + w\}$

Word2vec: Skip-Gram Model

- (Mikolov et al., 2013a,b)

$$p(C = c \mid X = v) = \frac{1}{Z_v} \exp \mathbf{c}_c^\top \mathbf{v}_v$$

- ▶ Two different vectors for each element of \mathcal{V} : one when it is “ v ” (\mathbf{v}) and one when it is “ c ” (\mathbf{c}).
- ▶ This should remind you of a neural network; SGD on the likelihood function is the conventional approach to estimating the vectors.
- ▶ Normalization term Z_v is expensive, so approximations are required for efficiency.
- ▶ Can expand this to be over the whole sentence or document, or otherwise choose which words “count” as context.

Word Embedding Evaluation

Several popular methods for *intrinsic* evaluations:

- ▶ Do (cosine) similarities of pairs of words' vectors correlate with judgments of similarity by humans?
- ▶ TOEFL-like synonym tests, e.g., *rug* $\xrightarrow{?}$ {*sofa*, *ottoman*, *carpet*, *hallway*}
- ▶ Syntactic analogies, e.g., “*walking* is to *walked* as *eating* is to what?” Solved via:

$$\max_{v \in \mathcal{V}} \cos(\mathbf{v}_v, -\mathbf{v}_{\text{walking}} + \mathbf{v}_{\text{walked}} + \mathbf{v}_{\text{eating}})$$

Word Embedding Evaluation

- Extrinsic evaluation:
 1. Use large unannotated corpus to get your word vectors (sometimes called **pretraining**).
 2. Use them in a text classifier (or some other NLP system).

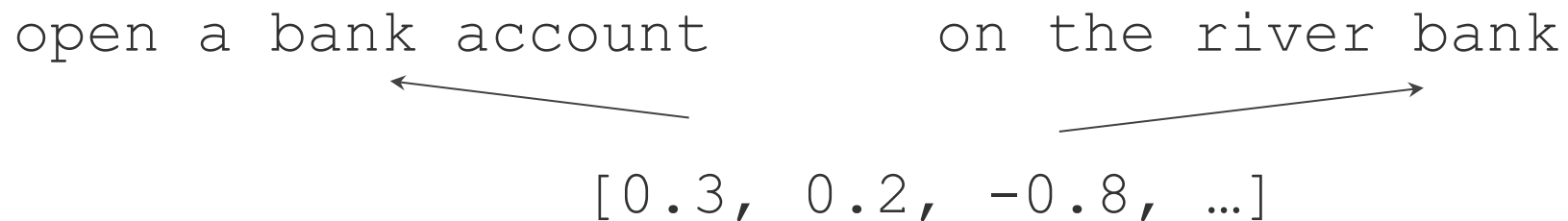
Two options:

 - ▶ Plug in word vectors as “frozen” features, and estimate the other parameters of your model.
 - ▶ Treat them as parameters of the text classifier; pretraining gives initial values, but they get updated, or “finetuned” during supervised learning.
 3. Does that system’s performance improve?

Word Embedding

- Problem: word embeddings are applied in a context free manner

open a bank account on the river bank



[0.3, 0.2, -0.8, ...]

Word Embedding

- Problem: word embeddings are applied in a context free manner

open a bank account on the river bank
 ← →
 [0.3, 0.2, -0.8, ...]

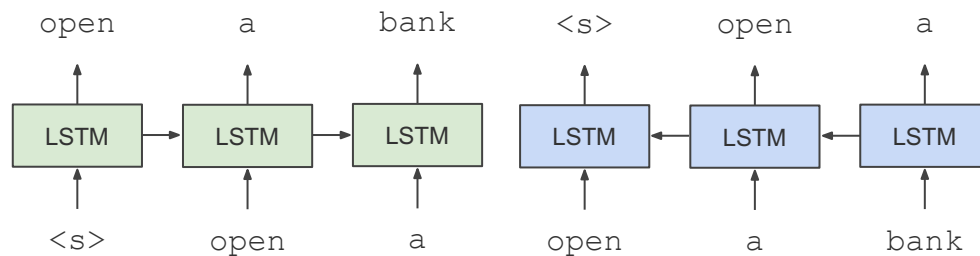
- Solution: Train contextual representations on text corpus

[0.9, -0.2, 1.6, ...] [-1.9, -0.4, 0.1, ...]
 ↑ ↑
open a bank account on the river bank

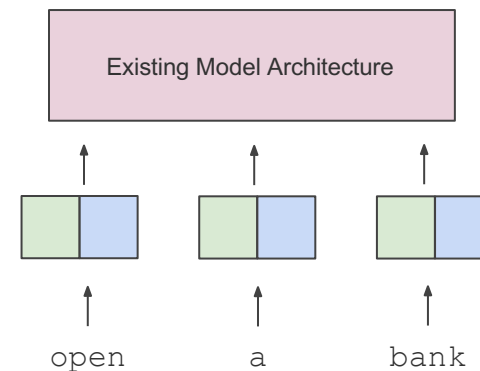
Contextual Representations

- *ELMo: Deep Contextual Word Embeddings*, AI2 & University of Washington, 2017

Train Separate Left-to-Right and Right-to-Left LMs

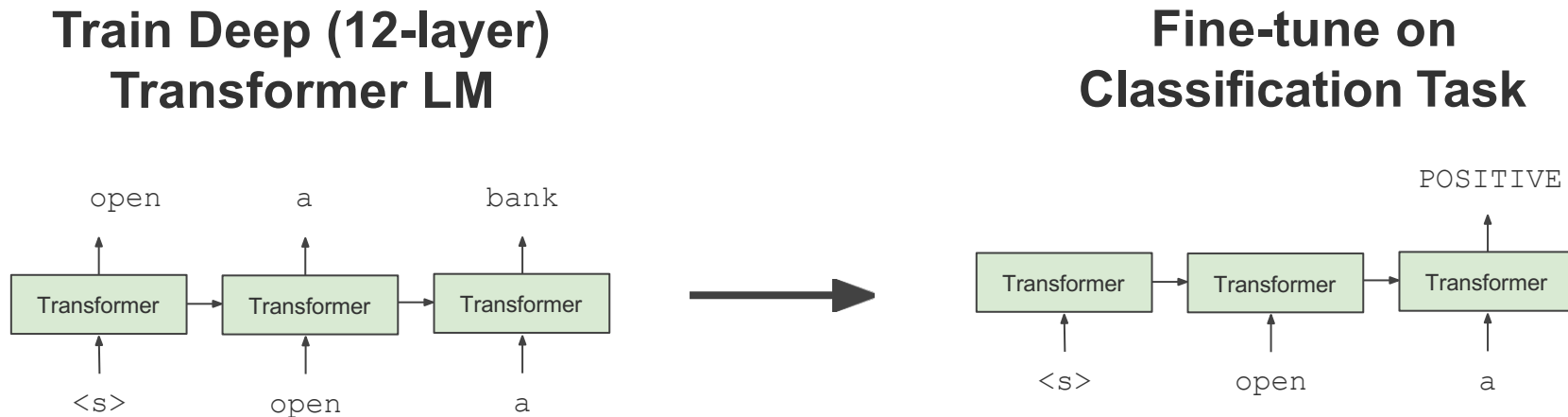


Apply as “Pre-trained Embeddings”



Contextual Representations

- *Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018

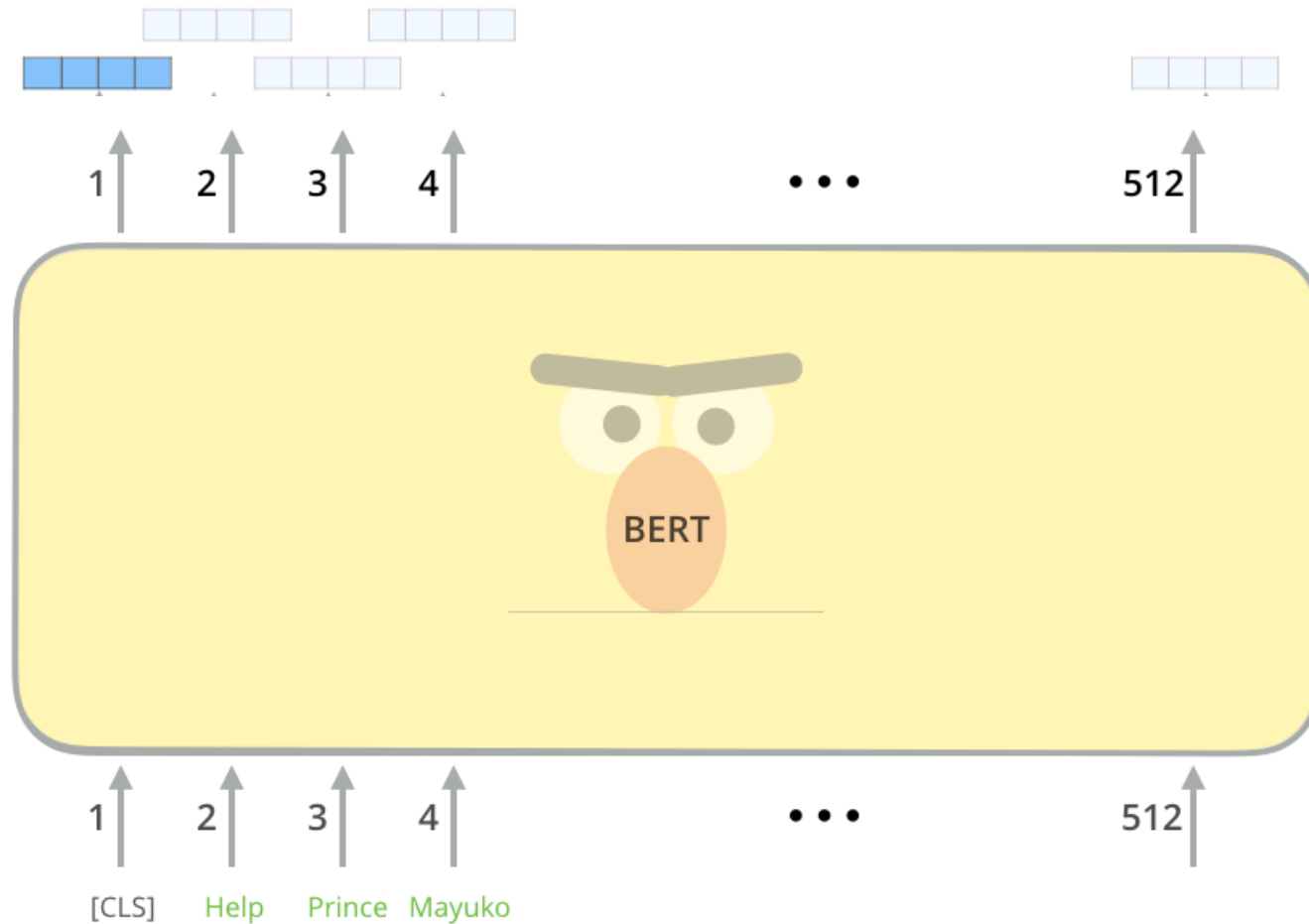


Problem with Previous Methods

- **Problem:** Language models only use left context *or* right context, but language understanding is bidirectional.

BERT

- BERT: A bidirectional model to extract contextual word embedding



BERT: Pre-training Procedure

- Dataset:
 - Wikipedia (2.5B words) + a collection of free ebooks (800M words)

BERT: Pre-training Procedure

- Dataset:
 - Wikipedia (2.5B words) + a collection of free ebooks (800M words)
- Training procedure
 - **masked language model** (masked LM)
 - Masks some percent of words from the input and has to reconstruct those words from context

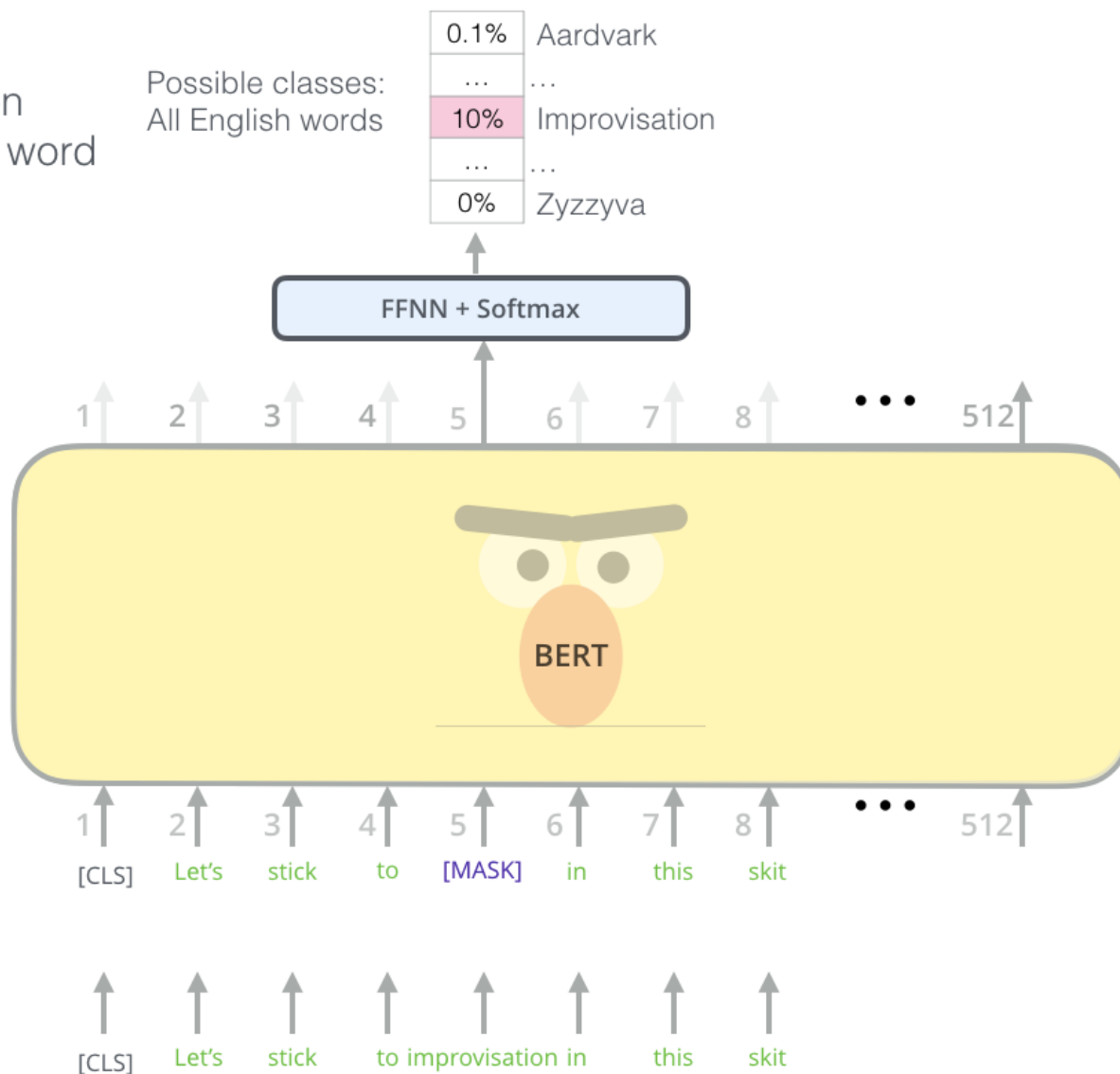
BERT: Pre-training Procedure

- Masked LM

Use the output of the masked word's position to predict the masked word

Randomly mask 15% of tokens

Input



BERT: Pre-training Procedure

- Masked LM
- 15% masking:
 - Too little masking: Too expensive to train (few supervision signals per example)
 - Too much masking: Not enough context
- Problem: Mask token never seen at fine-tuning
- Solution: don't replace with [MASK] 100% of the time. Instead:
- 80% of the time, replace with [MASK]
 - went to the store → went to the [MASK]
- 10% of the time, replace random word
 - went to the store → went to the running
- 10% of the time, keep same
 - went to the store → went to the store

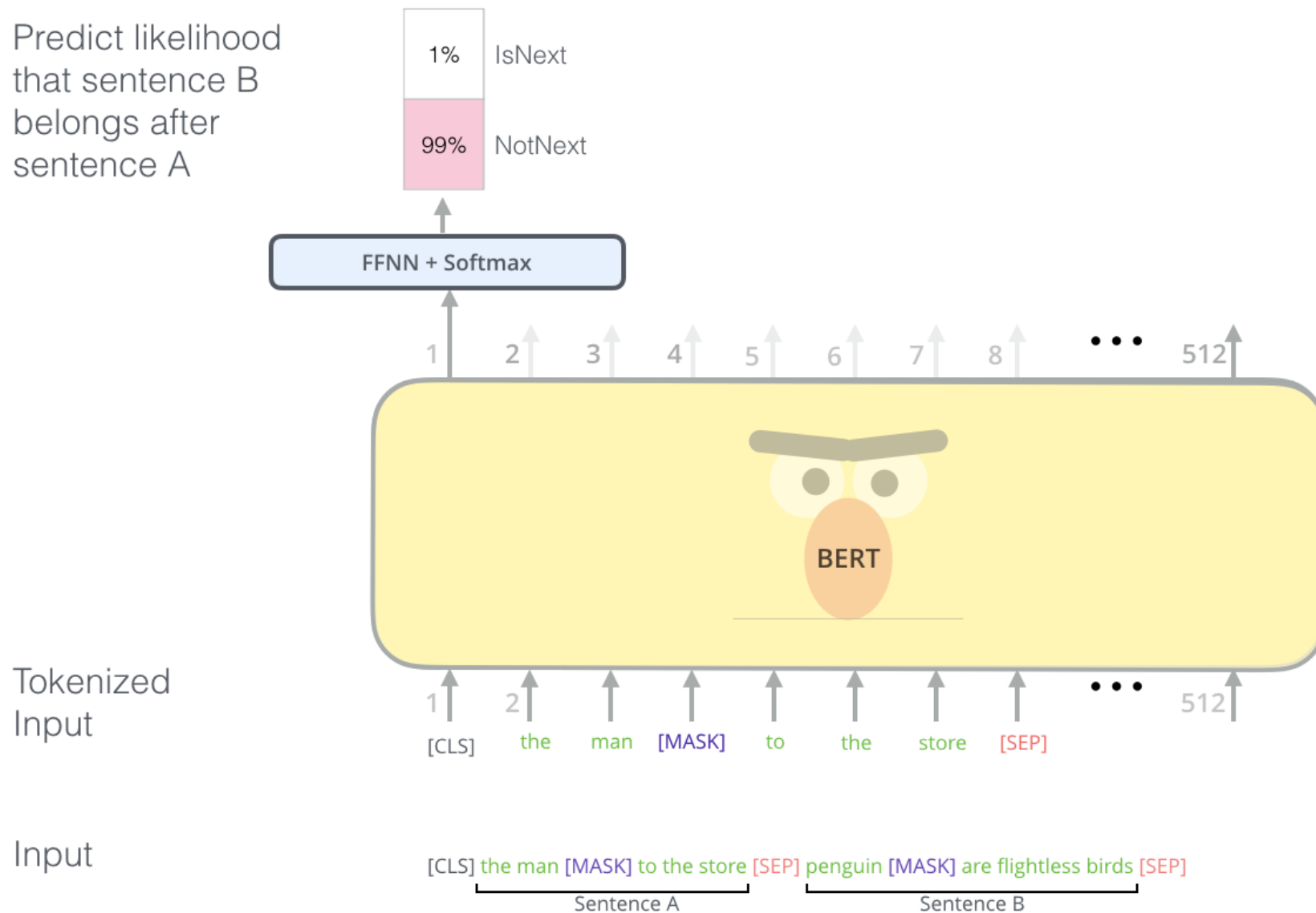
BERT: Pre-training Procedure

- Dataset:
 - Wikipedia (2.5B words) + a collection of free ebooks (800M words)
- Training procedure
 - **masked language model** (masked LM)
 - Masks some percent of words from the input and has to reconstruct those words from context
 - **Two-sentence task**
 - To understand relationships between sentences
 - Concatenate two sentences A and B and predict whether B actually comes after A in the original text

BERT: Pre-training Procedure

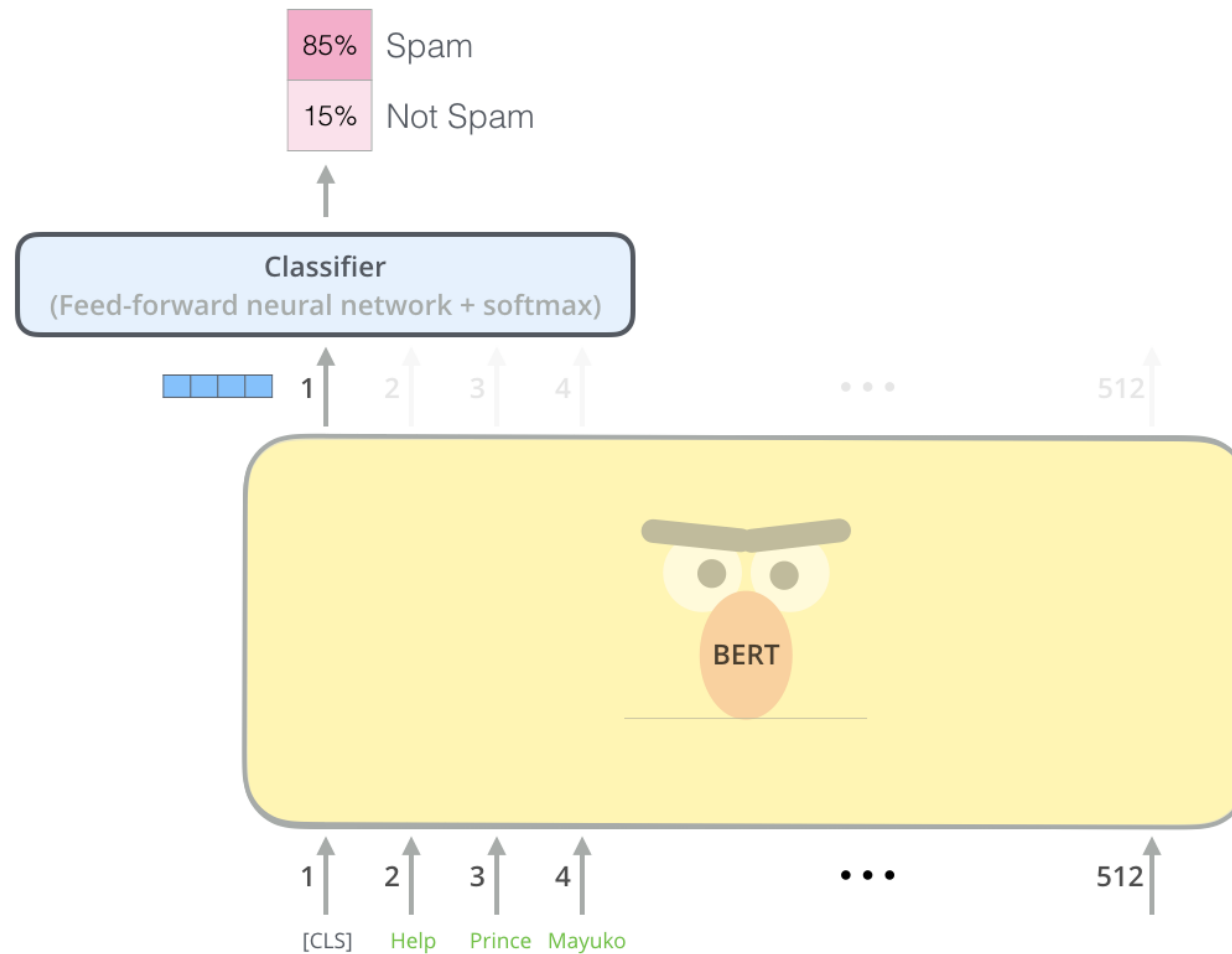
- Two sentence task

Predict likelihood that sentence B belongs after sentence A

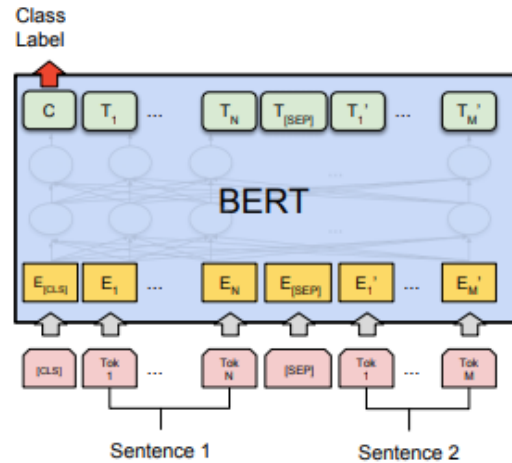


BERT: Downstream Fine-tuning

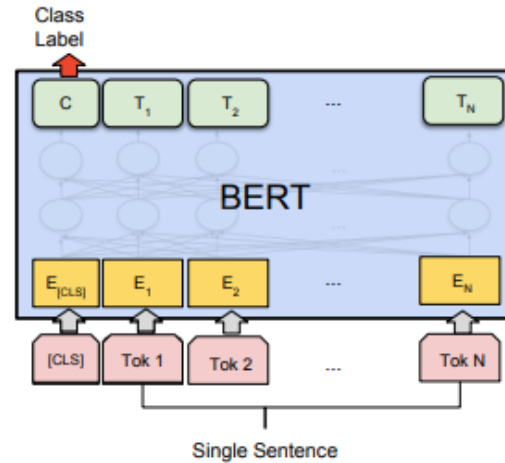
- Use BERT for sentence classification



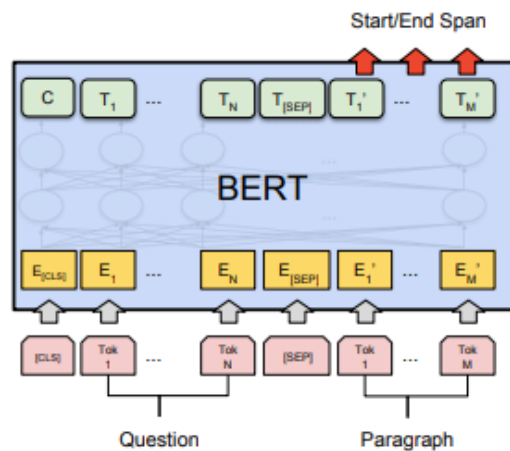
BERT: Downstream Fine-tuning



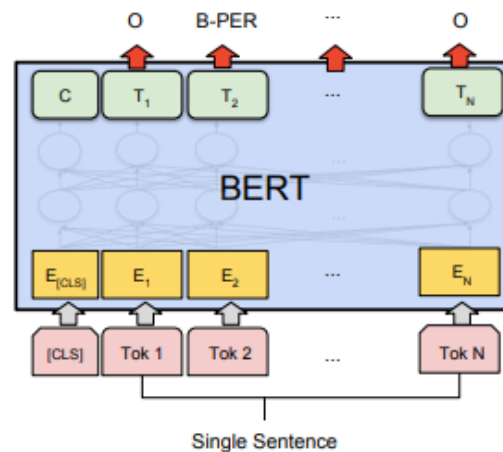
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

BERT Results

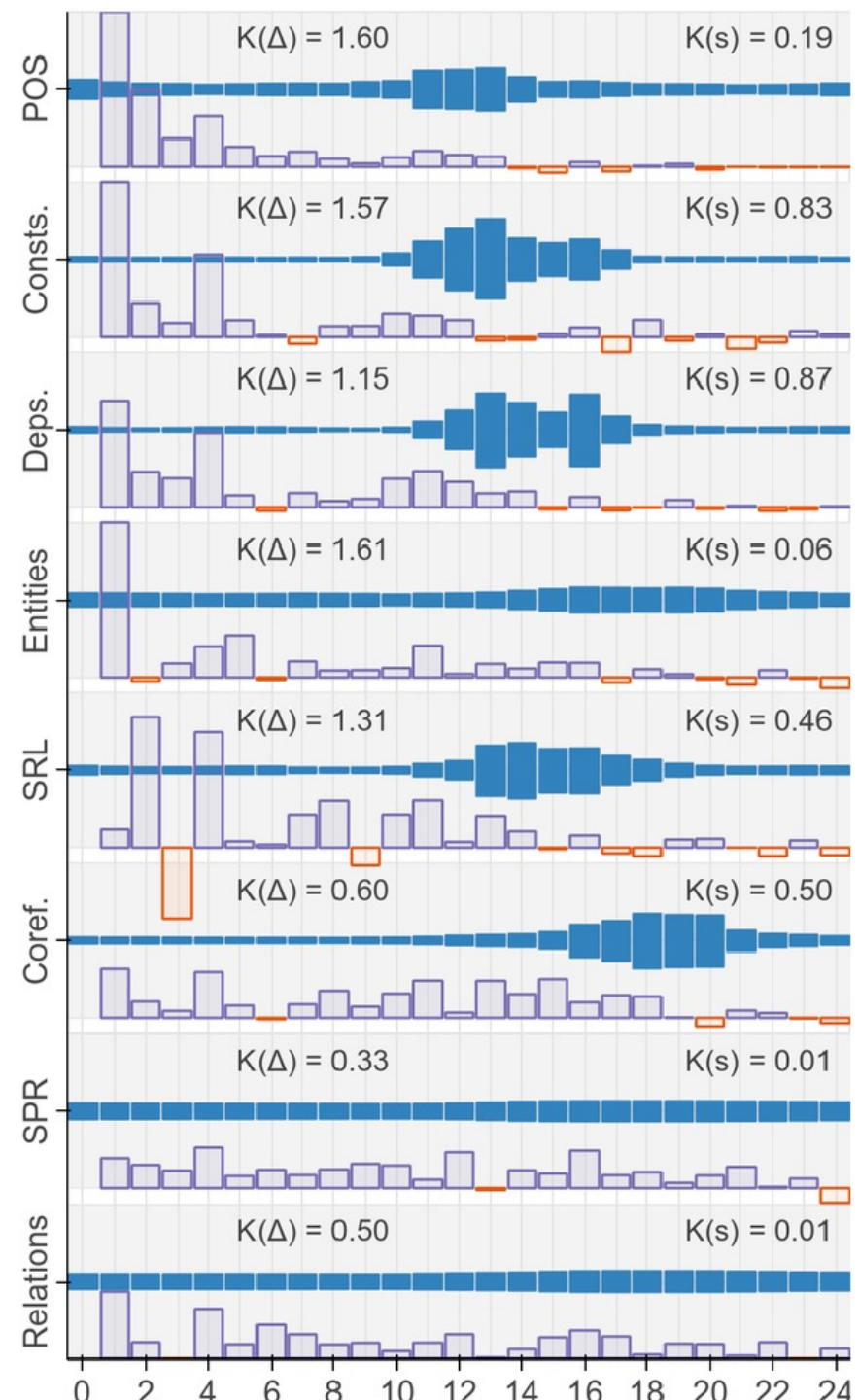
- Huge improvements over SOTA on 12 NLP task

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{BASE}	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); BERT_{BASE} = (L=12, H=768, A=12); BERT_{LARGE} = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from <https://gluebenchmark.com/leaderboard> and <https://blog.openai.com/language-unsupervised/>.

Analysis

- *BERT Rediscovered the Classical NLP Pipeline.* Tenney et al., 2019

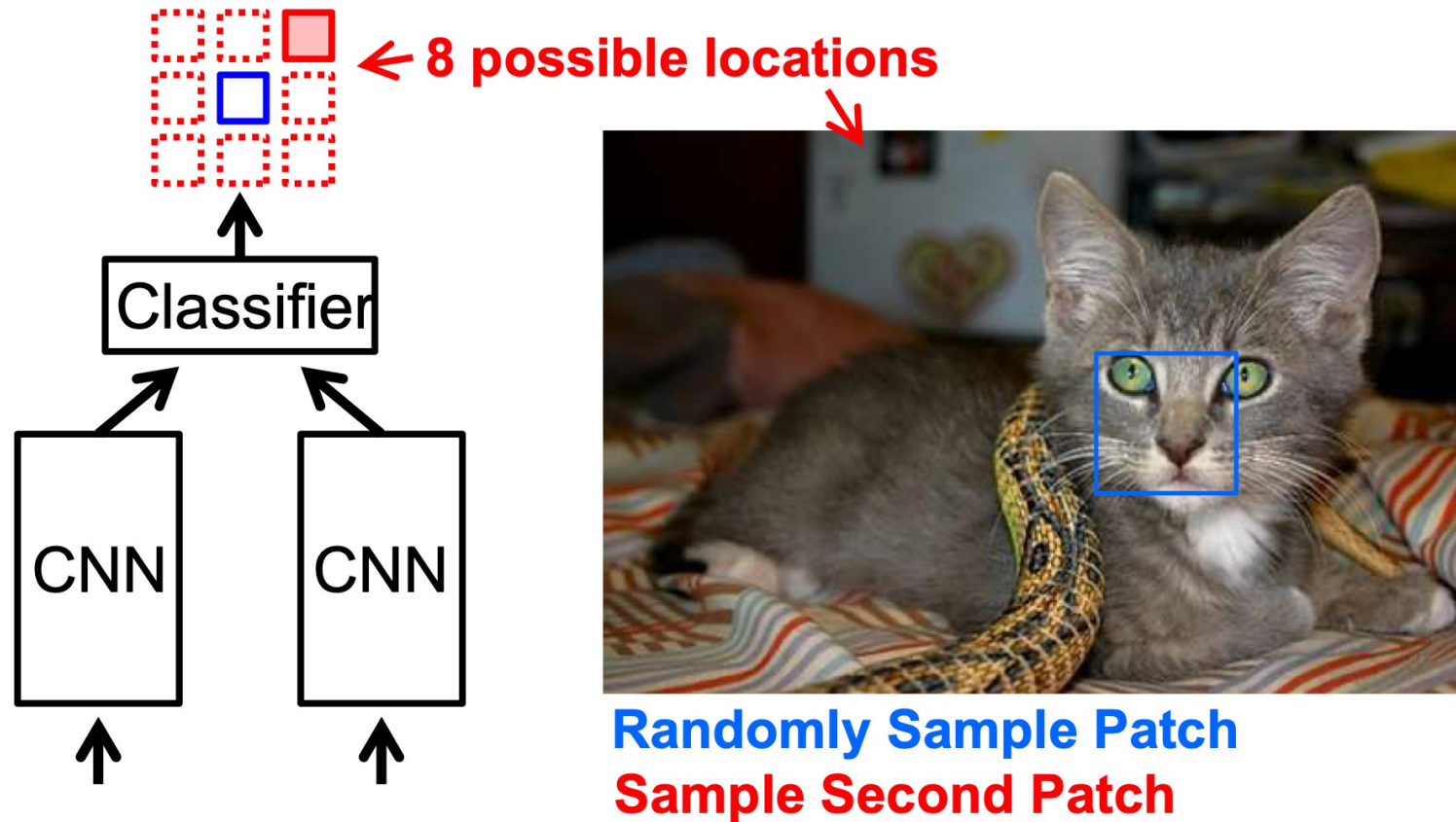


Self-supervised learning for other modalities: quick overview

- SSL on images
- SSL on videos

SSL from Images, EX (I): relative positioning

Train network to predict relative position of two regions in the same image



Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

SSL from Images, EX (I): relative positioning

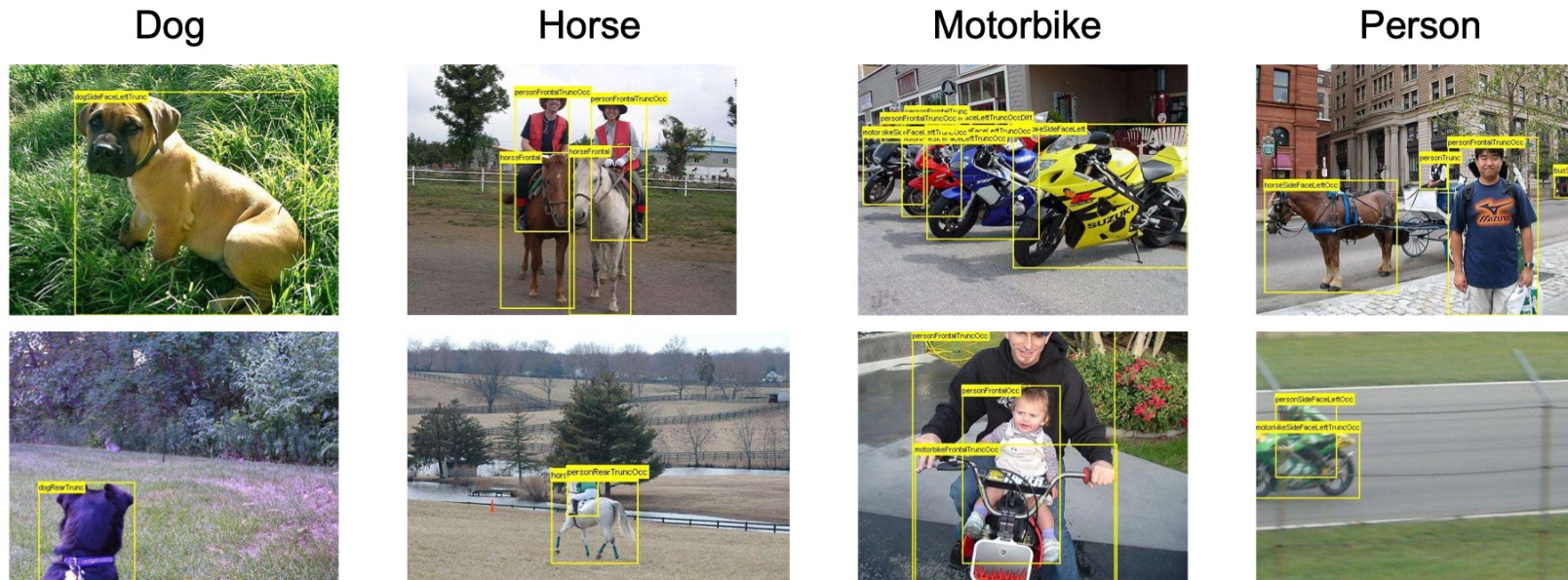


Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

SSL from Images, EX (I): relative positioning

Evaluation: PASCAL VOC Detection

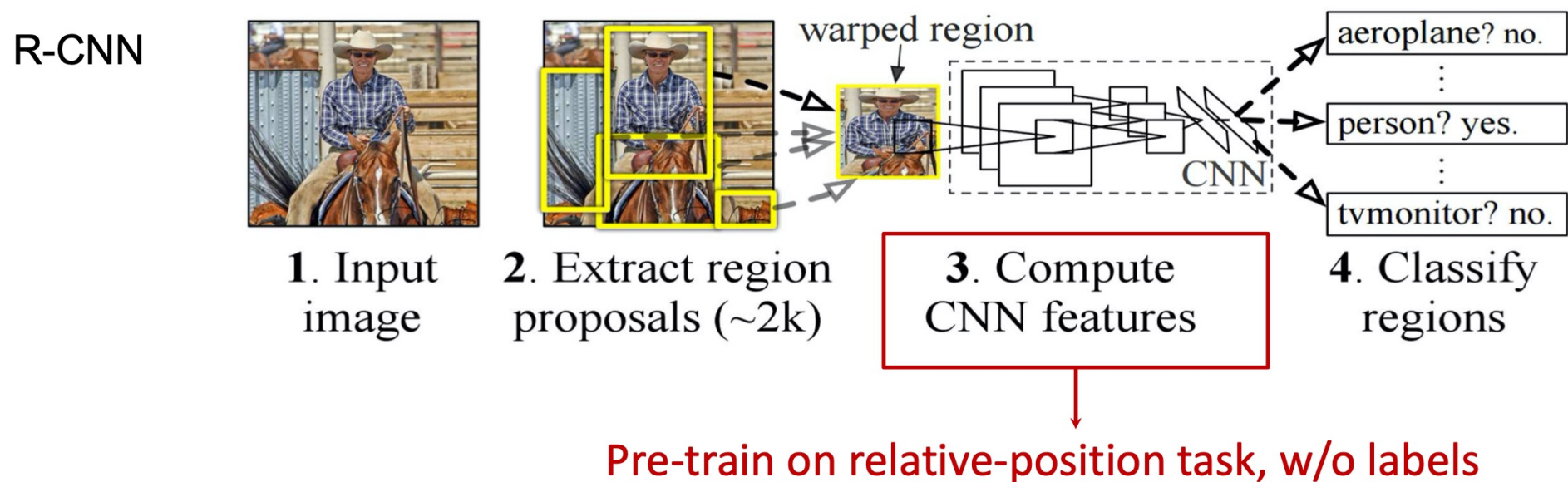
- 20 object classes (car, bicycle, person, horse ...)
- Predict the bounding boxes of all objects of a given class in an image (if any)



SSL from Images, EX (I): relative positioning

Evaluation: PASCAL VOC Detection

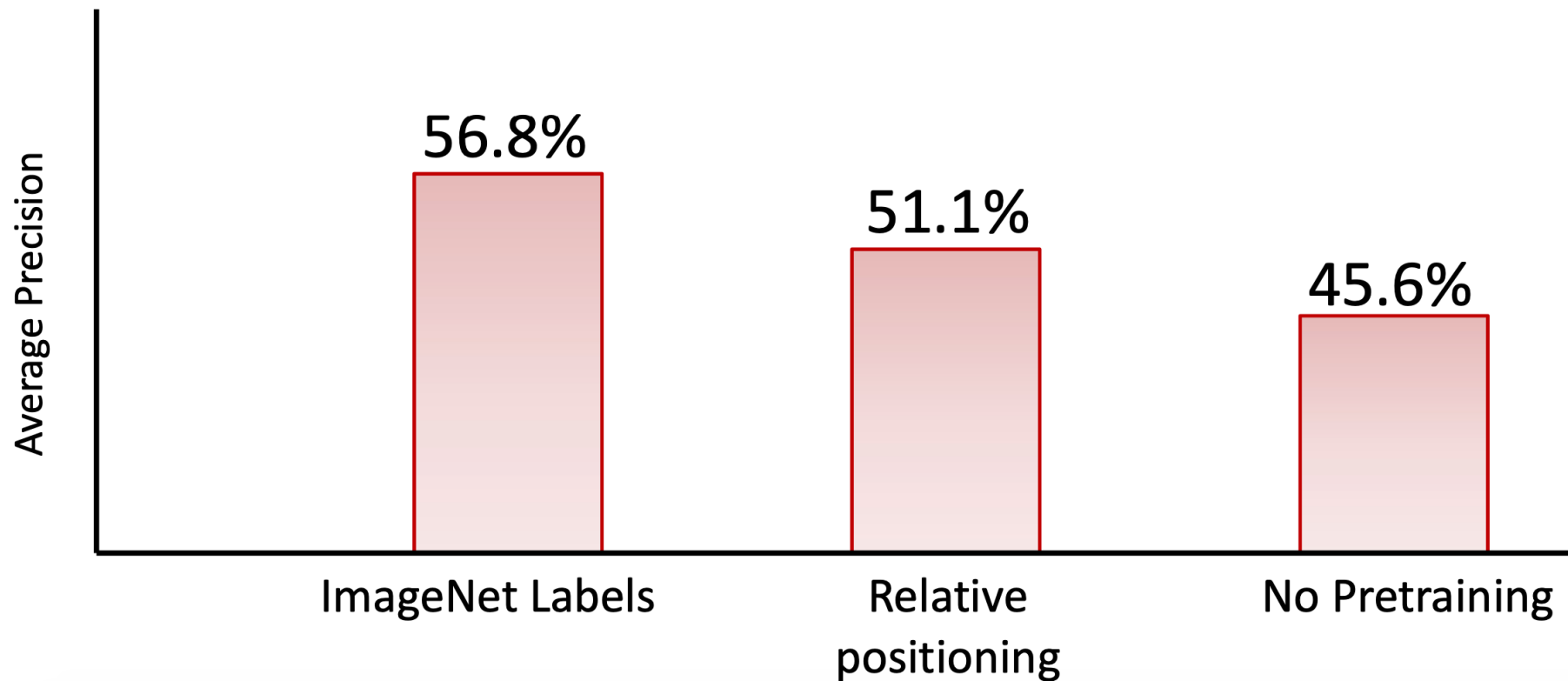
- Pre-train CNN using self-supervision (no labels)
- Train CNN for detection in R-CNN object category detection pipeline



[Girshick et al. 2014]

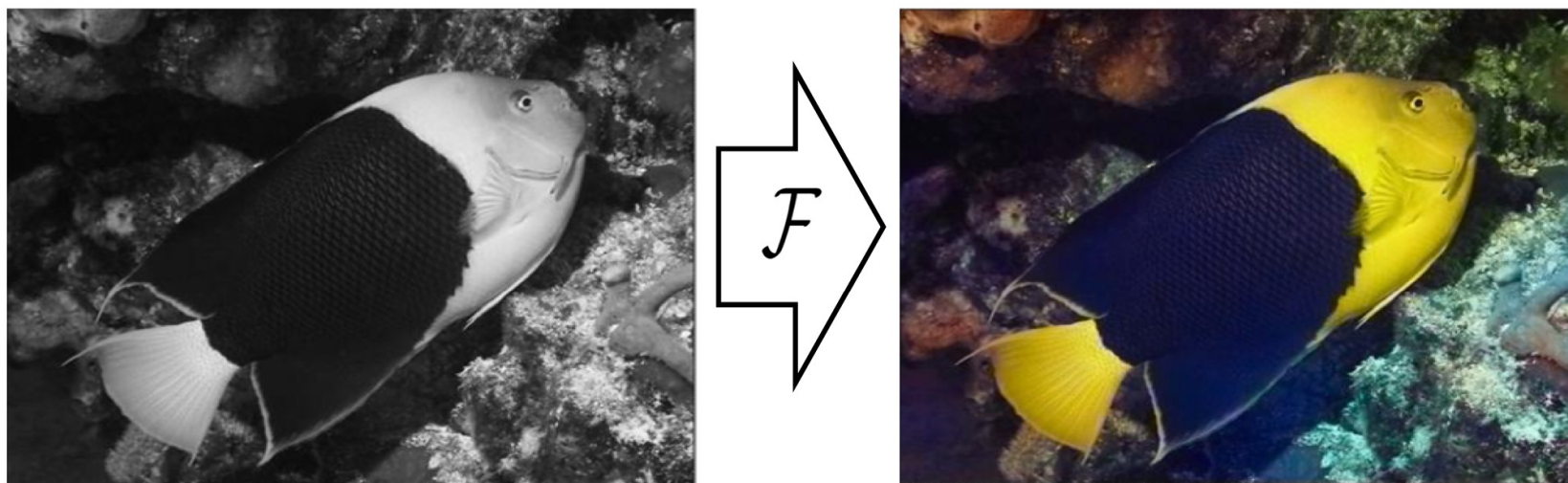
SSL from Images, EX (I): relative positioning

Evaluation: PASCAL VOC Detection



SSL from Images, EX (II): colorization

Train network to predict pixel colour from a monochrome input

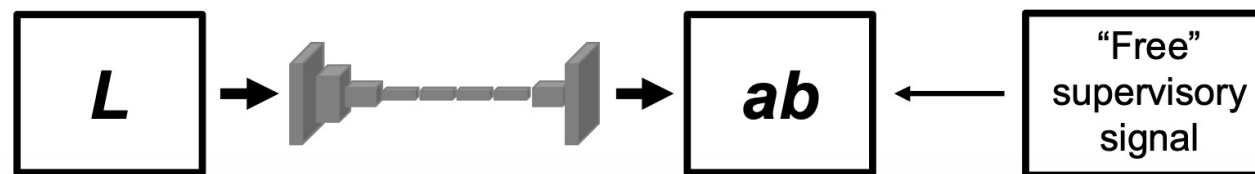


Grayscale image: L channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$

Concatenate (L, ab)

$$(\mathbf{X}, \hat{\mathbf{Y}})$$



SSL from Images, EX (II): colorization

Train network to predict pixel colour from a monochrome input



SSL from Images, EX (III): exemplar networks

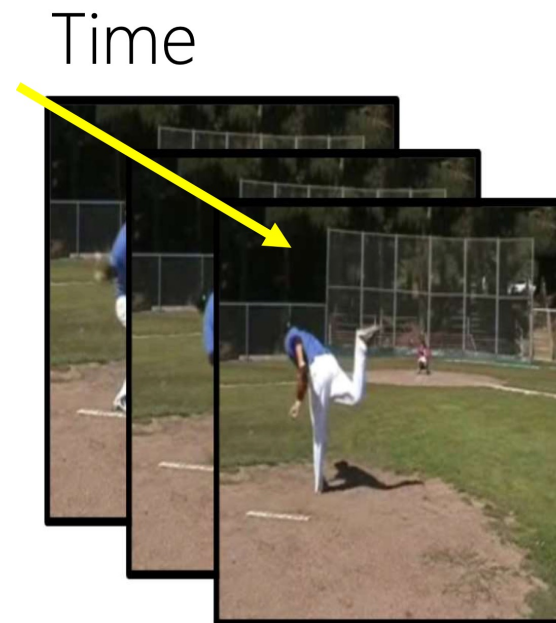
- Exemplar Networks (Dosovitskiy et al., 2014)
- Perturb/distort image patches, e.g. by cropping and affine transformations
- Train to classify these exemplars as same class



SSL from Videos

Three example tasks:

- Video sequence order
 - Sequential Verification: Is this a valid sequence?



“Sequence” of data

SSL from Videos

Three example tasks:

- Video sequence order
 - Sequential Verification: Is this a valid sequence?
- Video direction
 - Predict if video playing forwards or backwards

SSL from Videos

Three example tasks:

- Video sequence order
 - Sequential Verification: Is this a valid sequence?
- Video direction
 - Predict if video playing forwards or backwards
- Video tracking
 - Given a color video, colorize all frames of a gray scale version using a reference frame



Key Takeaways

- Self supervision learning
 - Predicting any part of the observations given any available information
 - The prediction task forces models to learn semantic representations
 - Massive/unlimited data supervisions
- SSL for text:
 - Language models: next word prediction
 - Word embedding: skip-gram
 - BERT text representations: masked language model (MLM)
- SSL for images/videos:
 - Various ways of defining the prediction task

Questions?