DSC291: Advanced Statistical Natural Language Processing

Language Modeling

Zhiting Hu

Lecture 3, April 5, 2022

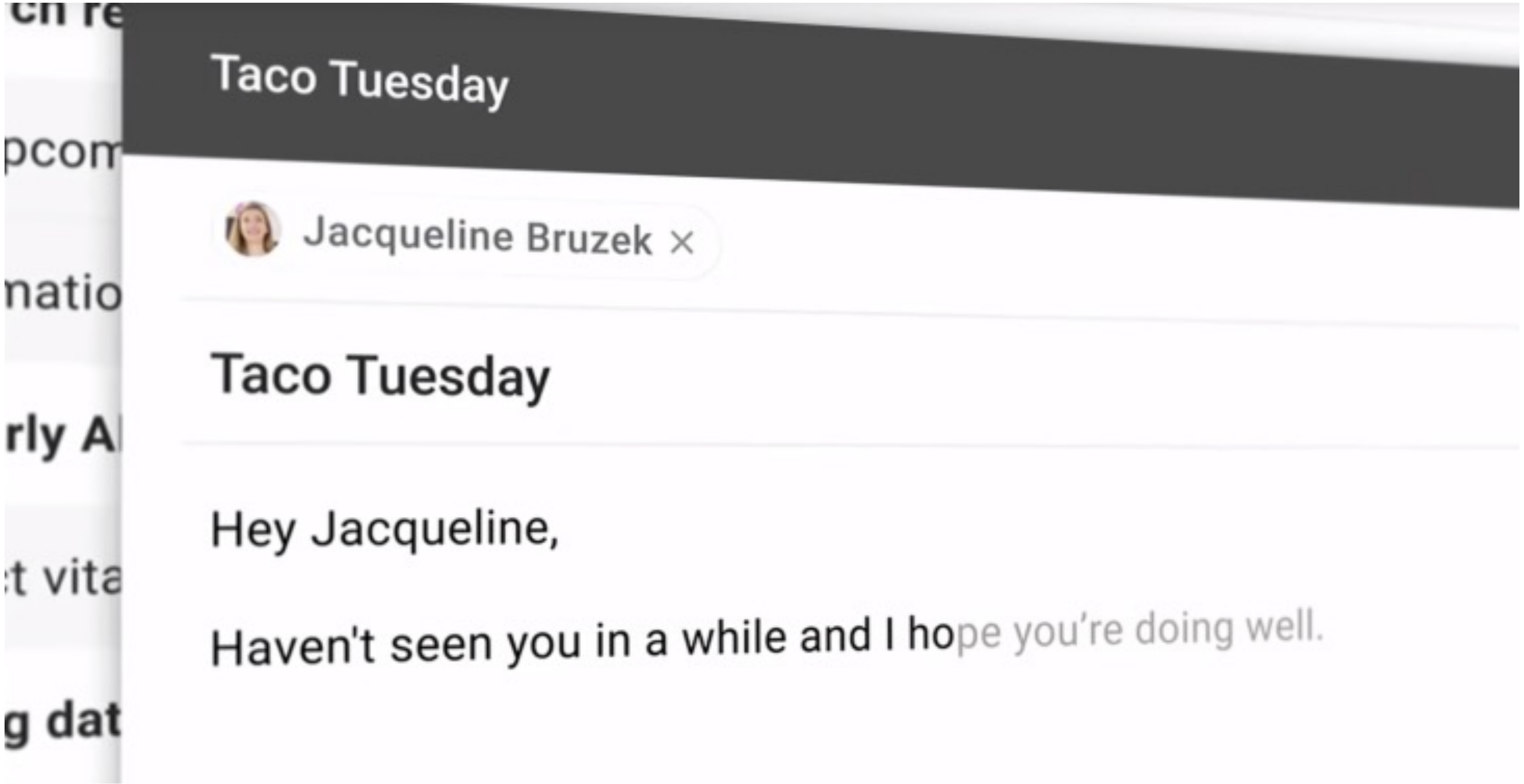
UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

Outline

- N-gram language models
- Neural language models
- Neural architectures (in general)

Motivations of Language Models

- Generation



Email auto-completion

Motivations of Language Models

- 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%).

Motivations of Language Models

- Evaluation of language fluency
  - Ex:

Successful speech recognition requires generating a word sequence that is:

- Faithful to the acoustic input
- Fluent

If we're mapping acoustics $\boldsymbol{a}$ to word sequences $\boldsymbol{w}$, then:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}} \operatorname{Faithfulness}(\boldsymbol{w}; \boldsymbol{a}) + \operatorname{Fluency}(\boldsymbol{w})$$

Language models can provide a "fluency" score.

[Courtesy: UW CSE 447 by Noah Smith]

Motivations of Language Models

- Evaluation of language fluency
  - Ex: Other tasks that have text (or speech) as output:
    - translation from one language to another
    - conversational systems
    - document summarization
    - image captioning
    - optical character recognition
    - spelling and grammar correction

    If we're mapping inputs $\boldsymbol{i}$ to word sequences $\boldsymbol{w}$, then:

$$\boldsymbol{w}^* = \operatorname*{argmax}_{\boldsymbol{w}} \text{Faithfulness}(\boldsymbol{w}; \boldsymbol{i}) + \text{Fluency}(\boldsymbol{w})$$

    Language models can provide a "fluency" score.

[Courtesy: UW CSE 447 by Noah Smith]

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

Motivations of Language Models

- Few-shot prediction (e.g., GPT3)

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.

[Nurecas.com]

Notations

- $\mathcal{V}$ is a finite set of (discrete) symbols (words or characters); $V = |\mathcal{V}|$
- $\mathcal{V}^*$ is the (infinite) set of sequences of symbols from $\mathcal{V}$

Notations

- $\mathcal{V}$ is a finite set of (discrete) symbols (words or characters); $V = |\mathcal{V}|$
- $\mathcal{V}^*$ is the (infinite) set of sequences of symbols from $\mathcal{V}$
- In language modeling, we imagine a sequence of random variables $X_1, X_2, \ldots$ that continues until some X_n takes the value "🛑" (a special end-of-sequence symbol).
- $\mathcal{V}^\dagger$ is the (infinite) set of sequences of $\mathcal{V}$ symbols, with a single 🛑, which is at the end.

The Language Modeling Problem

- Input: training data $\boldsymbol{x} = (x_1, x_2, \dots, x_N)$ in $\mathcal{V}^\dagger$
  - (assuming one instance $\boldsymbol{x}$ for simplicity of notations)
- Output: p: $\mathcal{V}^\dagger \to \mathbb{R}$
- Think of p as a measure of plausibility

Probabilistic Language Model

- We let p be a probability distribution, which means that

$$\forall \boldsymbol{x} \in \mathcal{V}^\dagger, p(\boldsymbol{x}) \geq 0$$

$$\sum_{\boldsymbol{x} \in \mathcal{V}^\dagger} p(\boldsymbol{x}) = 1$$

- Advantages:
  - Interpretability
  - We can apply the maximum likelihood principle to build a language model from data

[Courtesy: UW CSE 447 by Noah Smith]

Decomposing using the Chain Rule

$$p(\boldsymbol{X} = \boldsymbol{x}) = \left(\begin{array}{l} p(X_1 = x_1) \\ \cdot\, p(X_2 = x_2 \mid X_1 = x_1) \\ \cdot\, p(X_3 = x_3 \mid \boldsymbol{X}_{1:2} = \boldsymbol{x}_{1:2}) \\ \vdots \\ \cdot\, p(X_N = \bigcirc \mid \boldsymbol{X}_{1:N-1} = \boldsymbol{x}_{1:N-1}) \end{array} \right)$$

$$= \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1})$$

Example:

$\boldsymbol{x}$ = *(I, like, this, movie, …)*

$$p(\boldsymbol{x}) = \cdots p_\theta(like \mid I)\; p_\theta(this \mid I, like) \cdots$$

Predict each word based on the "history"

[Courtesy: UW CSE 447 by Noah Smith]

Unigram Model: Empty History

$$p(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1})$$

$$\overset{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i; \boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_{x_i}$$

Multinomial distribution

[Courtesy: UW CSE 447 by Noah Smith]

Unigram Model: Empty History

$$p(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1})$$

Multinomial distribution

$$\stackrel{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i; \boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_{x_i}$$

Maximum likelihood estimate: for every $v \in \mathcal{V}$,

$$\theta_v^* = \frac{\sum_{i=1}^{N} \mathbf{1}\{x_i = v\}}{N}$$

$$= \frac{\text{count}_{\boldsymbol{x}}(v)}{N}$$

[Courtesy: UW CSE 447 by Noah Smith]

Example

The probability of

$$\text{Presidents tell lies .}$$

is:

$$p(X_1 = \text{Presidents}) \cdot p(X_2 = \text{tell}) \cdot p(X_3 = \text{lies}) \cdot p(X_4 = .) \cdot p(X_5 = \;)$$

In unigram model notation:

$$\theta_{\text{Presidents}} \cdot \theta_{\text{tell}} \cdot \theta_{\text{lies}} \cdot \theta_{.} \cdot \theta_{\;}$$

Using the maximum likelihood estimate for $\boldsymbol{\theta}$, we could calculate:

$$\frac{\text{count}_{\boldsymbol{x}}(\text{Presidents})}{N} \cdot \frac{\text{count}_{\boldsymbol{x}}(\text{tell})}{N} \cdots \frac{\text{count}_{\boldsymbol{x}}(\;)}{N}$$

[Courtesy: UW CSE 447 by Noah Smith]

Unigram Models: Assessment

Pros:

- Easy to understand
- Cheap
- Good enough for information retrieval (maybe)

Cons:

- Fixed, known vocabulary assumption
- "Bag of words" assumption is linguistically inaccurate
  - $p(\text{the the the the}) \gg p(\text{I want ice cream})$

[Courtesy: UW CSE 447 by Noah Smith]

n-gram Models

$$p(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1})$$

$$\stackrel{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i \mid X_{i-\mathsf{n}+1:i-1} = \boldsymbol{x}_{i-\mathsf{n}+1:i-1}; \boldsymbol{\theta})$$

$$= \prod_{i=1}^{N} \theta_{x_i \mid \boldsymbol{x}_{i-\mathsf{n}+1:i-1}}$$

[Courtesy: UW CSE 447 by Noah Smith]

n-gram Models

$$p(\boldsymbol{X} = \boldsymbol{x}) = \prod_{i=1}^{N} p(X_i = x_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1})$$

$$\stackrel{\text{assumption}}{=} \prod_{i=1}^{N} p(X_i = x_i \mid X_{i-\mathsf{n}+1:i-1} = \boldsymbol{x}_{i-\mathsf{n}+1:i-1}; \boldsymbol{\theta})$$

$$= \prod_{i=1}^{N} \theta_{x_i \mid \boldsymbol{x}_{i-\mathsf{n}+1:i-1}}$$

$(\mathsf{n} - 1)$th-order Markov assumption $\equiv$ n-gram model

- Unigram model is the $\mathsf{n} = 1$ case
- For a long time, trigram models $(\mathsf{n} = 3)$ were widely used
- 5-gram models $(\mathsf{n} = 5)$ were common in MT for a time

[Courtesy: UW CSE 447 by Noah Smith]

n-gram Models

- Maximum likelihood estimate for the n-gram model's probability of v given a $(n - 1)$-length history $\boldsymbol{h}$

$$
\begin{aligned}
\theta_{v|\boldsymbol{h}} &= p(X_i = v \mid \boldsymbol{X}_{i-\mathsf{n}+1:i-1} = \boldsymbol{h}) \\
&= \frac{p(X_i = v, \boldsymbol{X}_{i-\mathsf{n}+1:i-1} = \boldsymbol{h})}{p(\boldsymbol{X}_{i-\mathsf{n}+1:i-1} = \boldsymbol{h})} \\
&= \frac{\text{count}_{\boldsymbol{x}}(\boldsymbol{h}v)}{N} \Bigg/ \frac{\text{count}_{\boldsymbol{x}}(\boldsymbol{h})}{N} \\
&= \frac{\text{count}_{\boldsymbol{x}}(\boldsymbol{h}v)}{\text{count}_{\boldsymbol{x}}(\boldsymbol{h})}
\end{aligned}
$$

[Courtesy: UW CSE 447 by Noah Smith]

Choosing n is a Balancing Act

If n is too small, your model can't learn very much about language.

As n gets larger:

- The number of parameters grows with $O(V^{\mathsf{n}})$.
- Most n-grams will never be observed, so you'll have lots of zero probability n-grams. This is an example of **data sparsity**.
- Your model depends increasingly on the training data; you need (lots) more data to learn to generalize well.

This is a beautiful illustration of the bias-variance tradeoff.

[Courtesy: UW CSE 447 by Noah Smith]

Other "tricks"

- Smoothing

  The game: prevent $\theta_{v|\boldsymbol{h}} = 0$ for any v and $\boldsymbol{h}$, while keeping $\sum_{\boldsymbol{x}} p(\boldsymbol{x}) = 1$ so that perplexity stays meaningful.

  - Simple method: add $\lambda > 0$ to every count (including counts of zero) before normalizing (the textbook calls this "Lidstone" smoothing)

- Dealing with Out-of-Vocabulary Terms
  - Define a special OOV or "unknown" symbol **unk**. Transform some (or all) rare words in the training data to **unk**.
  - Build a language model at the character level.
  - Some new methods use data-driven, deterministic tokenization schemes that segment some words into smaller parts to reduce the effective vocabulary size (Sennrich et al., 2016; Wu et al., 2016).

n-gram Models: Assessment

Pros:

- Easy to understand
- Cheap (with modern hardware; Lin and Dyer, 2010)
- Fine in some applications and when training data is scarce

Cons:

- Fixed, known vocabulary assumption
- Markov assumption is linguistically inaccurate
  - (But not as bad as unigram models!)
- Data sparseness problem

[Courtesy: UW CSE 447 by Noah Smith]

Neural Language Models

Neural Language Models

Instead of a lookup for a word and fixed-length history ($\theta_{v|\boldsymbol{h}}$), define a vector function:

$$p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1}) = \mathbf{NN}(\mathbf{enc}(\boldsymbol{x}_{1:i-1}); \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ do the work of *encoding* the history and *transforming* it into a distribution over the next word.

[Courtesy: UW CSE 447 by Noah Smith]

Neural Language Models

Instead of a lookup for a word and fixed-length history ($\theta_{v|\boldsymbol{h}}$), define a vector function:

$$p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1}) = \mathbf{NN}(\mathbf{enc}(\boldsymbol{x}_{1:i-1}); \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ do the work of *encoding* the history and *transforming* it into a distribution over the next word.
The transformation is described as a composed series of simple transformations or "layers."

[Courtesy: UW CSE 447 by Noah Smith]

Neural Network

Formally, it's a function $\mathbf{NN}$ from $\boldsymbol{\theta}$ (learned parameters) and inputs to outputs, all of which are real-valued vectors (or matrices, or tensors, or collections of them).

Almost always, $\mathbf{NN}$ is differentiable with respect to $\boldsymbol{\theta}$ and nonlinear with respect to the data input.

- "Nonlinear" means there does **not** exist a matrix $\mathbf{A}$ such that $\mathbf{NN}(\mathbf{v}; \boldsymbol{\theta}) = \mathbf{A}\mathbf{v}$, for all $\mathbf{v}$.

[Courtesy: UW CSE 447 by Noah Smith]

Neural Language Models

Instead of a lookup for a word and fixed-length history ($\theta_{v|\boldsymbol{h}}$), define a vector function:

$$p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{x}_{1:i-1}) = \mathbf{NN}(\mathbf{enc}(\boldsymbol{x}_{1:i-1}); \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ do the work of *encoding* the history and *transforming* it into a distribution over the next word.
The transformation is described as a composed series of simple transformations or "layers."

- We first map word histories $\boldsymbol{h}$ to vectors/matrices
- We interpret the output as $p(X_i \mid \boldsymbol{X}_{1:i-1} = \boldsymbol{h})$

[Courtesy: UW CSE 447 by Noah Smith]

Two Key Components

- "Embedding" words as vectors
- Layering to increase capacity (i.e., the set of distributions that can be represented).

"One Hot" Vectors

Let $\mathbf{e}_i \in \mathbb{R}^V$ be the ith column of the identity matrix $\mathbf{I}$.

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}; \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}; \quad \ldots; \quad \mathbf{e}_V = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$\mathbf{e}_i$ is the "one hot" vector for the ith word in $\mathcal{V}$.

A neural language model starts by "looking up" each word by multiplying its one hot vector by a matrix $\underset{V \times d}{\mathbf{M}}$; $\mathbf{e}_v^\top \mathbf{M} = \mathbf{m}_v$, the "embedding" of v.

$\mathbf{M}$ becomes part of the parameters ($\boldsymbol{\theta}$).

Sequences of Word Vectors

Given a word sequence $\langle v_1, v_2, \ldots, v_k \rangle$, we transform it into a sequence of word vectors,

$$\mathbf{m}_{v_1}, \mathbf{m}_{v_2}, \ldots, \mathbf{m}_{v_k}$$

Adding Layers

- Neural networks are built by composing functions, a mix of
  - Affine, $\boldsymbol{v}' = \boldsymbol{W}\boldsymbol{v} + \boldsymbol{b}$ (note that the dimensionality of $\boldsymbol{v}$ and $\boldsymbol{v}'$ might be different)
  - Nonlinearity, e.g.,
    - rectified linear ("relu") units $v_i' = \max(0, v_i)$
    - elementwise hyperbolic tangent $v_i' = \tanh(v_i) = \dfrac{e^{v_i} - e^{-v_i}}{e^{v_i} + e^{-v_i}}$
    - softmax $v_i' = \exp\{ v_i \} / \sum_j \exp\{ v_j \}$
  - More complex components (composed of the above operations):
    - Convolutional layers
    - Recurrent NNs
    - Attention

Summary so far

- language models utilities
  - Generation, evaluation of fluency, few-shot prediction (GPT3), …
- N-gram language models
  - Unigram LM
  - N-gram LM
- Neural language models:
  - Embedding: one-hot vectors -> embedding vectors
  - Neural networks

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

Convolutional Networks (ConvNets)

- Biologically-inspired variants of MLPs [LeCun et al. NIPS 1989]
  - Receptive field [Hubel & Wiesel 1962; Fukushima 1982]
    - Visual cortex contains a complex arrangement of cells
    - These cells are sensitive to small **sub-regions** of the visual field
  - The sub-regions are **tiled** to cover the entire visual field

Local Filters

Exploit the strong spatially local correlation present in natural images

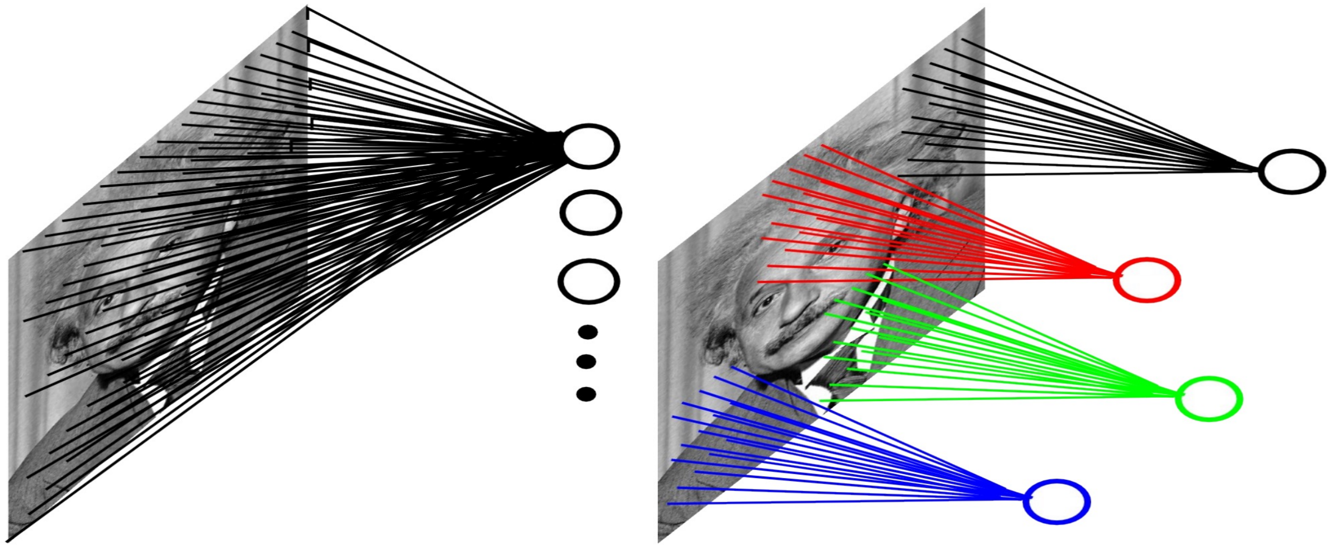


Figure courtesy: Yann LeCun

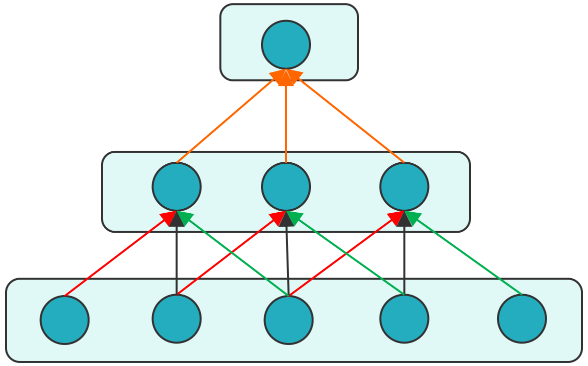
Convolutional Networks (ConvNets)

- Sparse connectivity
- Shared weights
- Increasingly “global” receptive fields
  - simple cells detect local features
  - complex cells “pool” the outputs of simple cells within a retinotopic neighborhood.

Feature maps $m+1$

Feature maps m

Feature maps $m-1$



Convolutional Networks (ConvNets)

- Hierarchical Representation Learning [Zeiler & Fergus 2013]

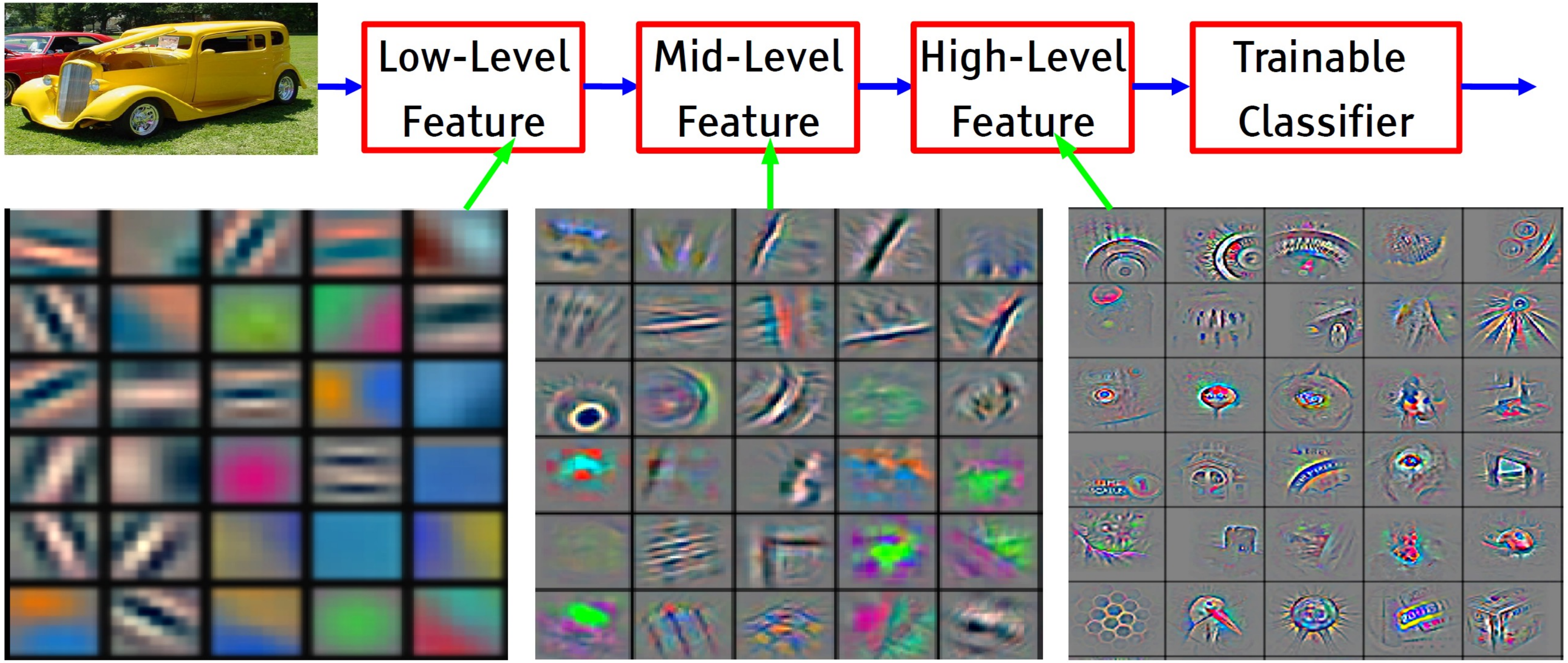
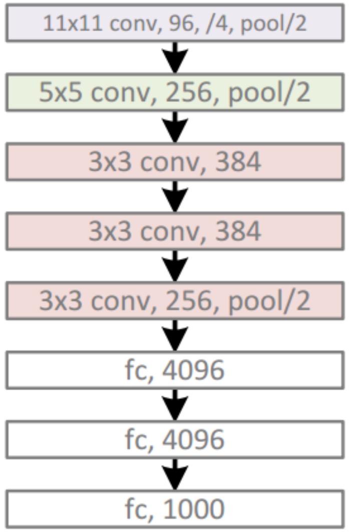


Figure courtesy: Yann LeCun

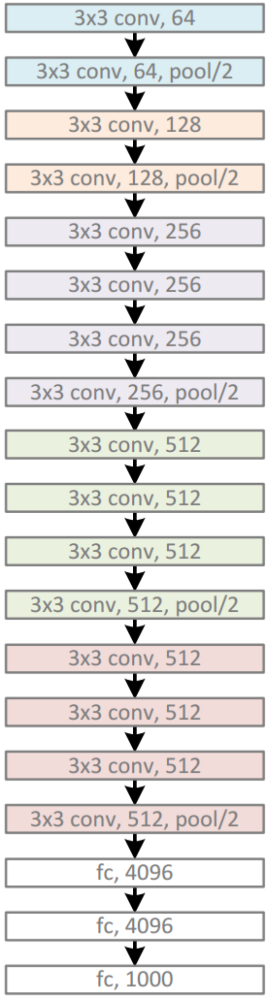
Evolution of ConvNets

AlexNet, 8 layers



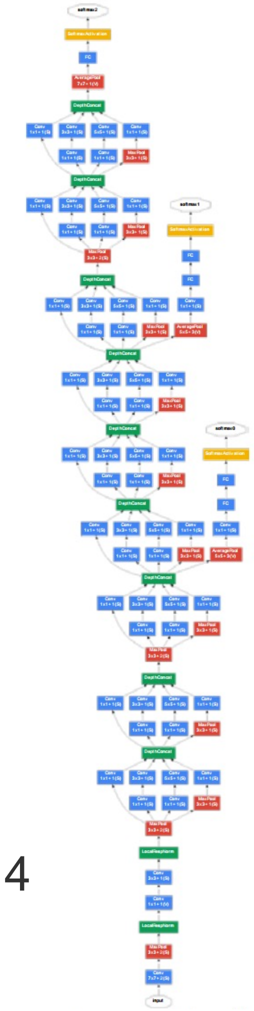
2012

VGG, 19 layers



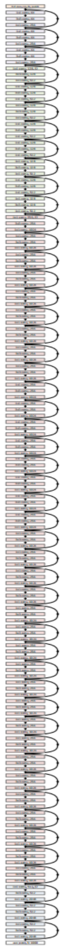
2015

GoogleNet, 22 layers



2014

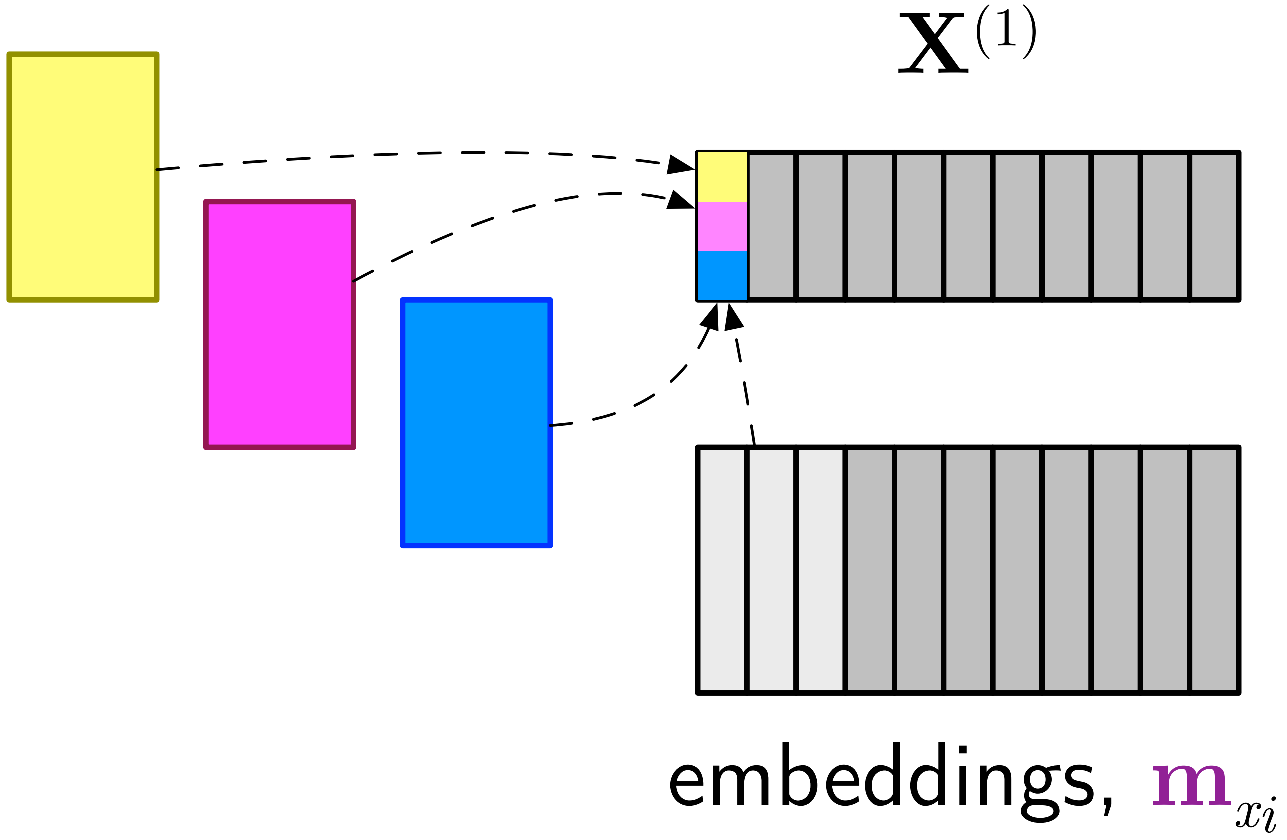
ResNet, 152 layers



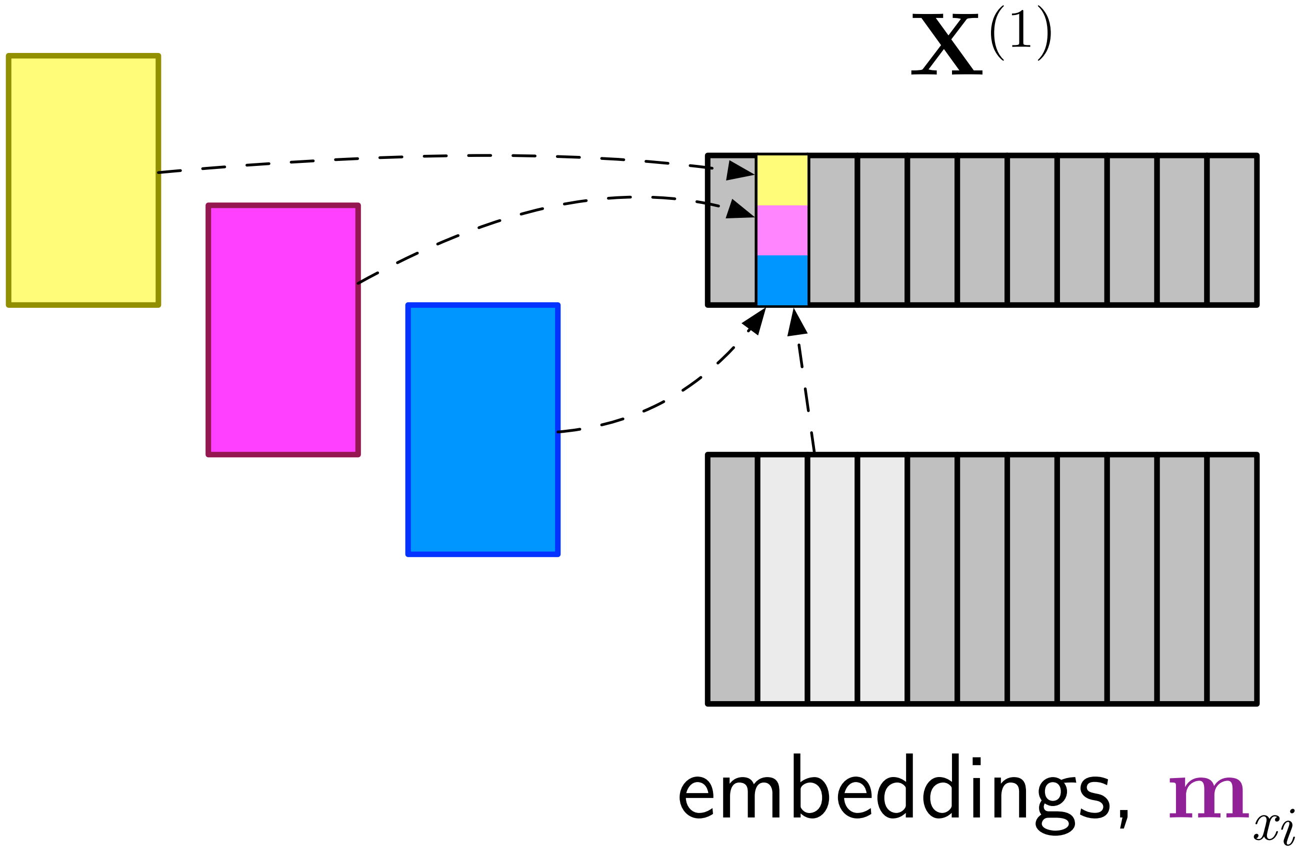
2015

Figure courtesy: Kaiming He

Conv layers for Text



Conv layers for Text

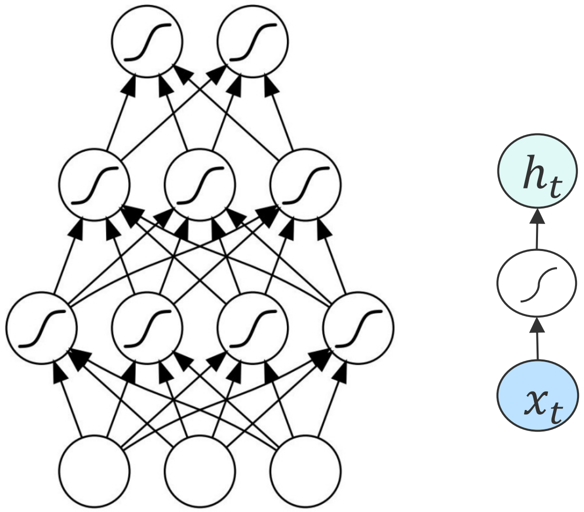


Outline

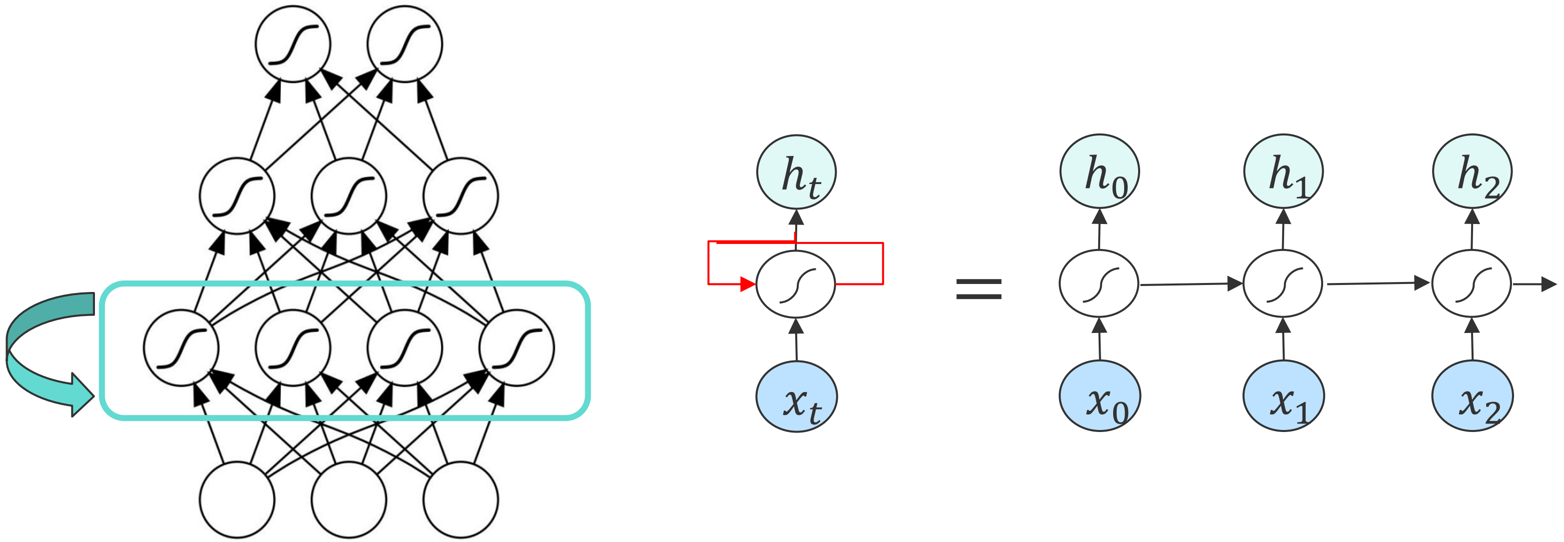
- Convolutional Networks (ConvNets)
- Recurrent Networks (RNNs)
  - Long-range dependency, vanishing
  - LSTM
  - RNNs in different forms
- Attention Mechanisms
  - (Query, Key, Value)
  - Attention on Text and Images
- Transformers: Multi-head Attention
  - Transformer
  - BERT

ConvNets v.s. Recurrent Networks (RNNs)

- Spatial Modeling *vs.* Sequential Modeling
- Fixed *vs.* variable number of computation steps.



The output depends ONLY on the current input



The hidden layers and the output additionally depend on previous states of the hidden layers

RNNs in Various Forms

One to One

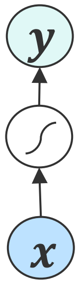


Image
classification

One to Many

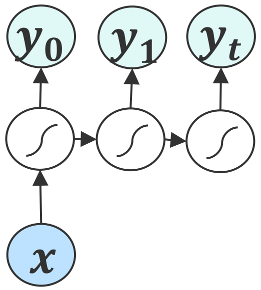
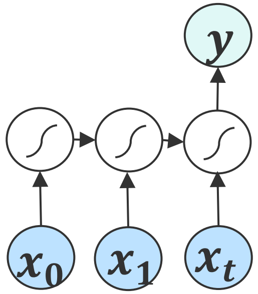


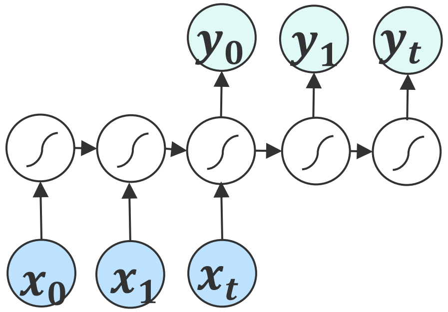
Image
captioning

Many to One



Sentence sentiment analysis /
Video recognition

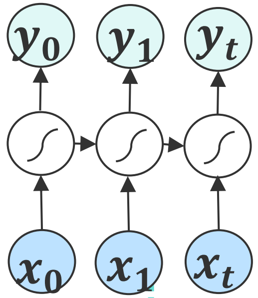
Many to Many



Machine Translation

(Sequence-to-sequence)

Many to Many

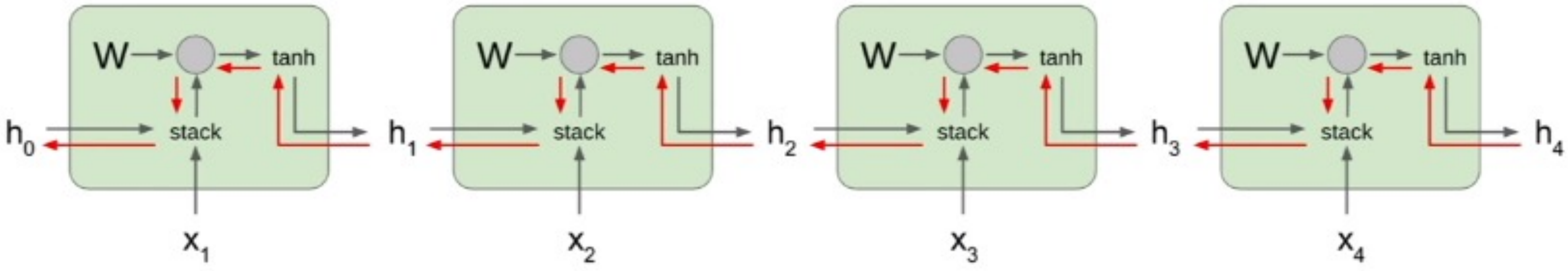


Named Entity Recognition

(Sequence tagging)

Vanishing / Exploding Gradients in RNNs

$$\boldsymbol{h}_t = tanh(W^{hh}\boldsymbol{h}_{t-1} + W^{hx}\boldsymbol{x}_t)$$

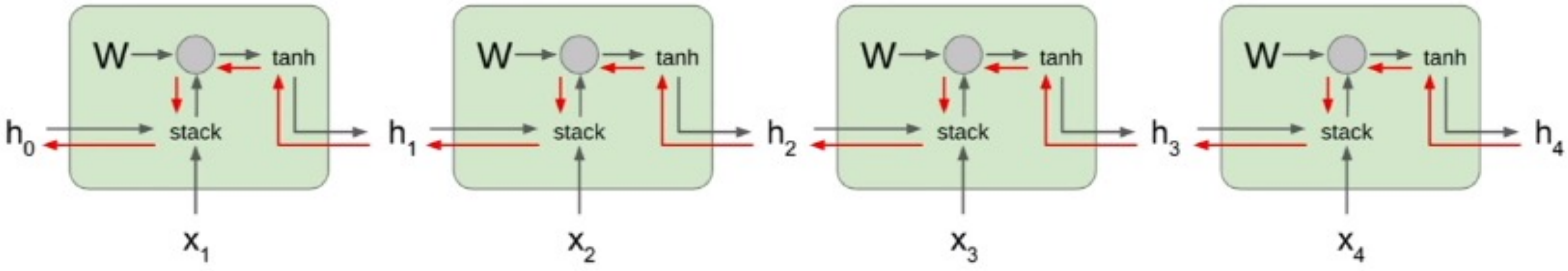


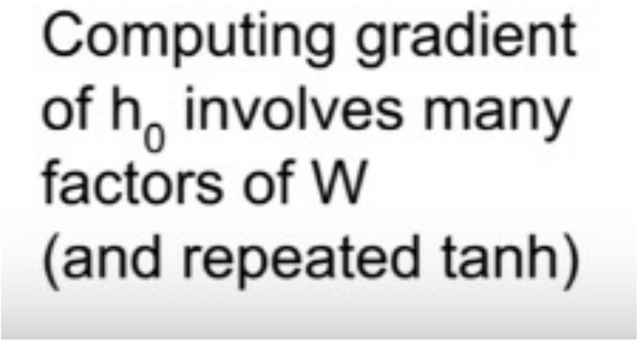
Source: CS231N Stanford

Bengio et al., 1994 “Learning long-term dependencies with gradient descent is difficult”
Pascanu et al., 2013 “On the difficulty of training recurrent neural networks”

Vanishing / Exploding Gradients in RNNs

$$\boldsymbol{h}_t = tanh(W^{hh}\boldsymbol{h}_{t-1} + W^{hx}\boldsymbol{x}_t)$$



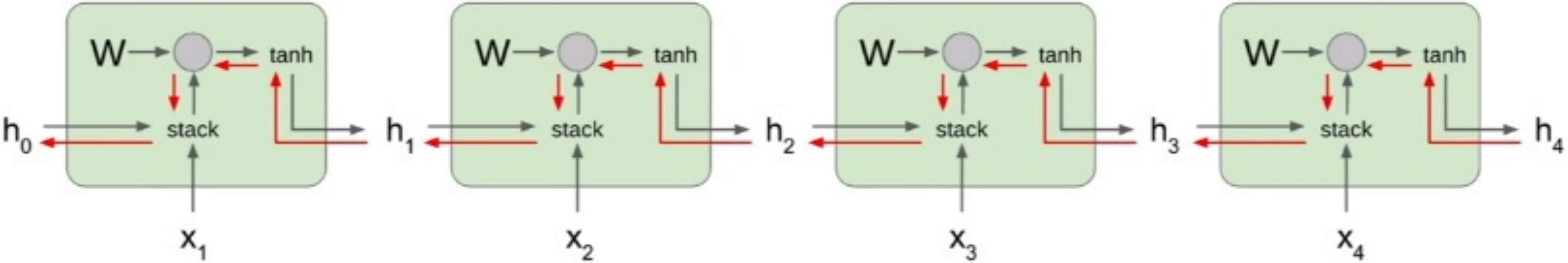


Bengio et al., 1994 “Learning long-term dependencies with gradient descent is difficult”
Pascanu et al., 2013 “On the difficulty of training recurrent neural networks”

Source: CS231N Stanford

Vanishing / Exploding Gradients in RNNs

$$\boldsymbol{h}_t = tanh(W^{hh}\boldsymbol{h}_{t-1} + W^{hx}\boldsymbol{x}_t)$$



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

→ **Gradient clipping**: Scale gradient if its norm is too big

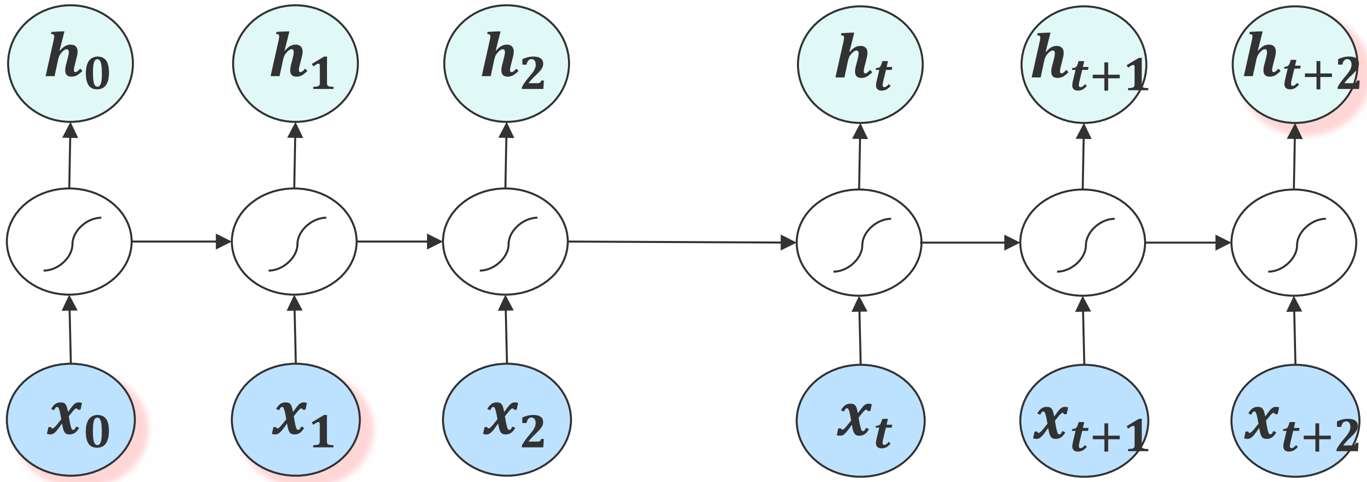
```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

Largest singular value < 1:
Vanishing gradients

Source: CS231N Stanford

Bengio et al., 1994 “Learning long-term dependencies with gradient descent is difficult”
Pascanu et al., 2013 “On the difficulty of training recurrent neural networks”

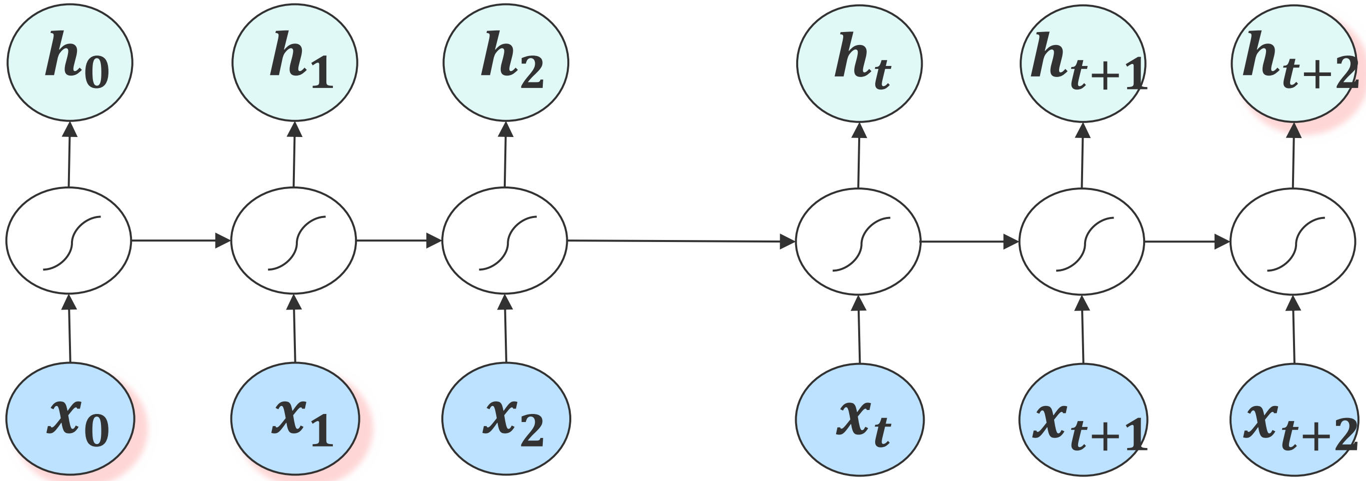
Long-term Dependency Problem



I live in France and I know ____________

Example courtesy: Manik Soni

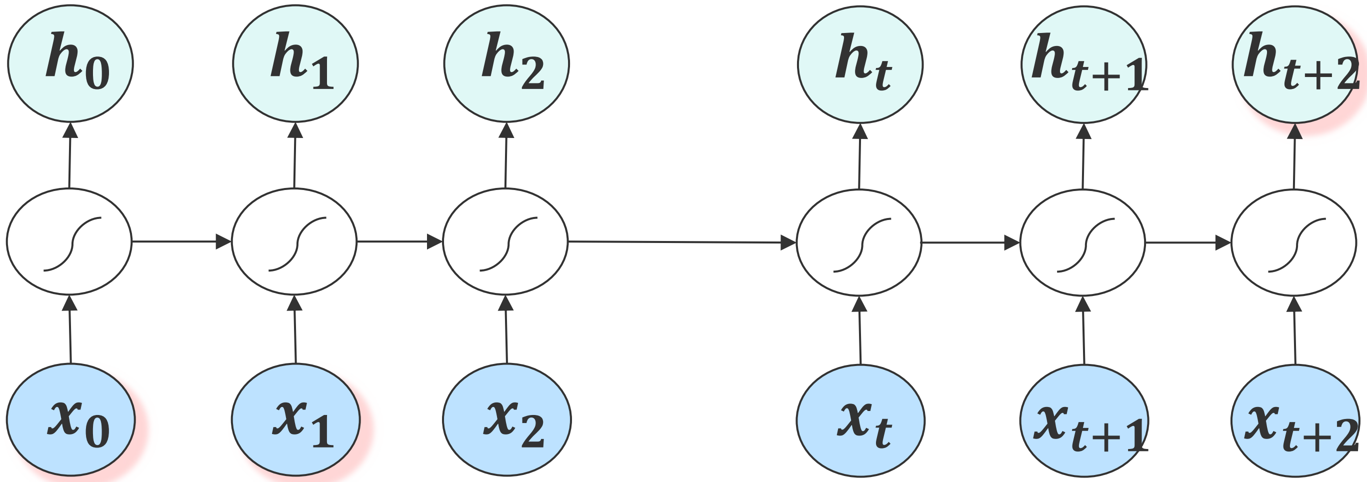
Long-term Dependency Problem



I live in France and I know ___*French*___

Example courtesy: Manik Soni

Long-term Dependency Problem



I live in France and I know __*French*__

I live in France, a beautiful country, and I know __*French*__

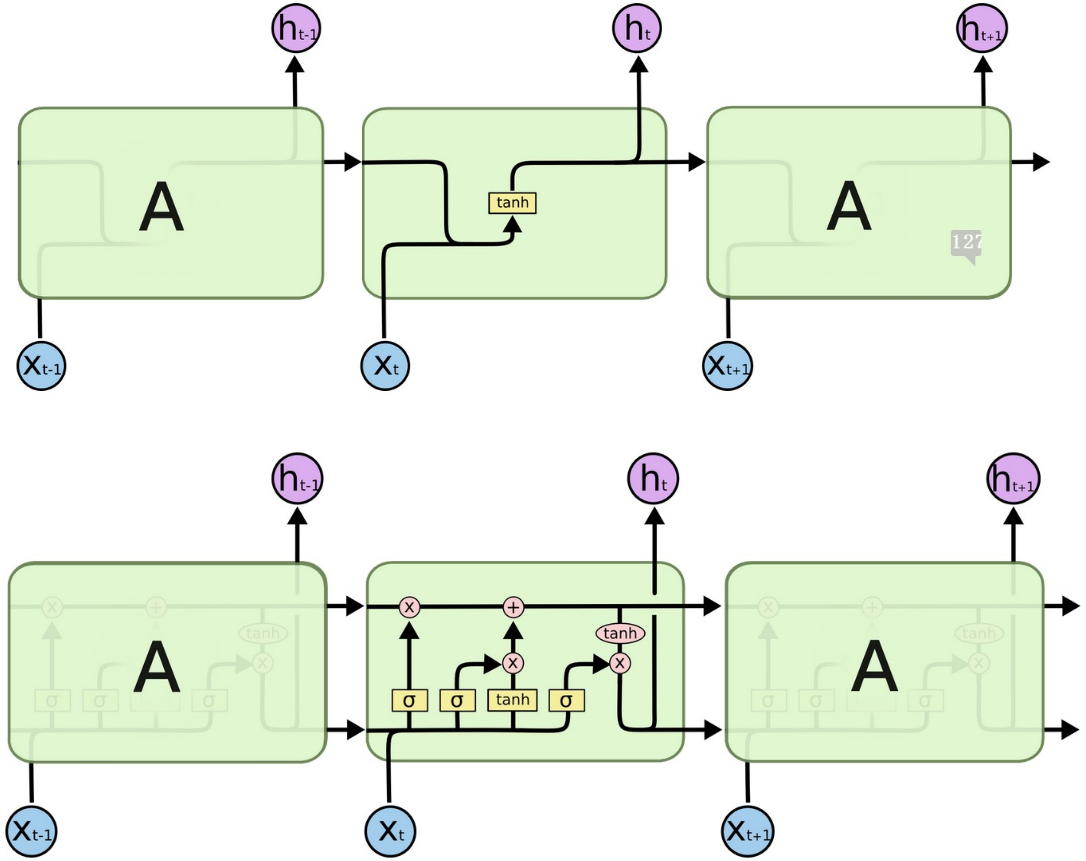
Example courtesy: Manik Soni

Long Short Term Memory (LSTM)

- LSTMs are designed to explicitly alleviate the long-term dependency problem [Horchreiter & Schmidhuber (1997)]

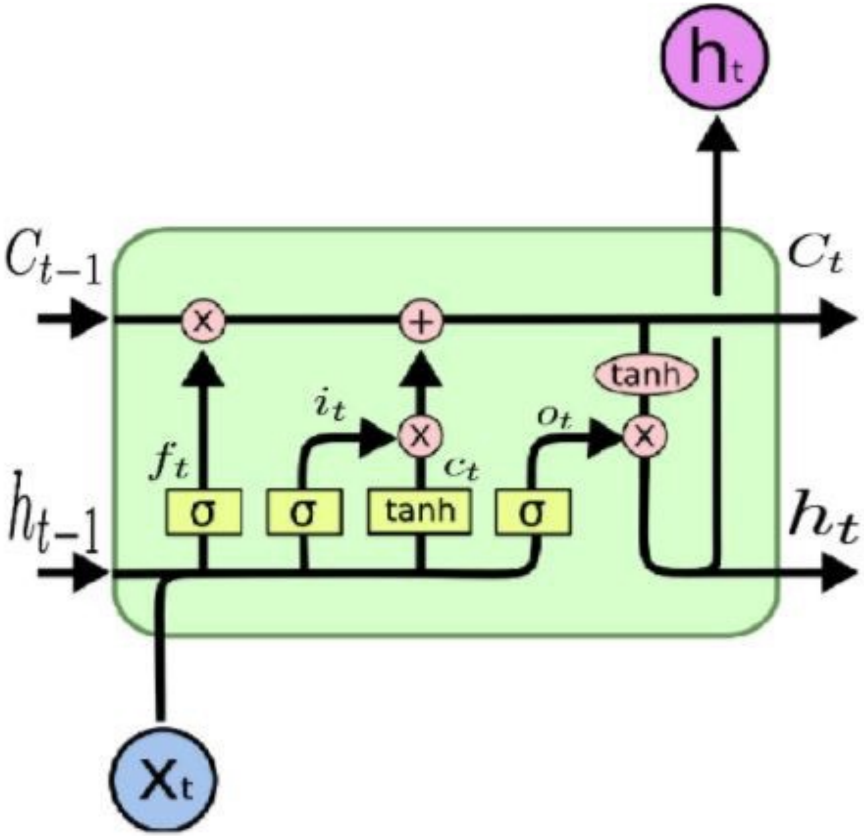
Standard RNN

LSTM



Long Short Term Memory (LSTM)

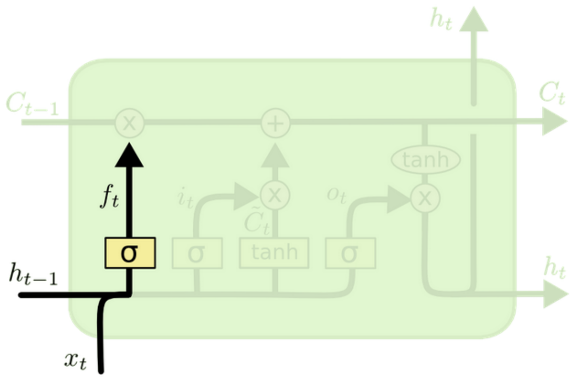
- Gate functions make decisions of reading, writing, and resetting information



- Forget gate: whether to erase cell (reset)
- Input gate: whether to write to cell (write)
- Output gate: how much to reveal cell (read)

Long Short Term Memory (LSTM)

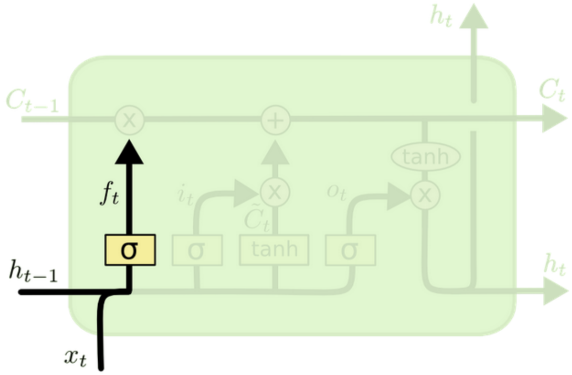
- **Forget gate:** decides what must be removed from $\boldsymbol{h}_{t-1}$



$$\boldsymbol{f_t} = \boldsymbol{\sigma}(\boldsymbol{W_f} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_f})$$

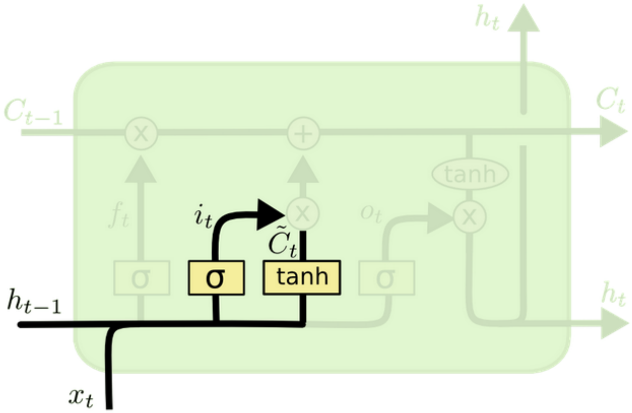
Long Short Term Memory (LSTM)

- **Forget gate:** decides what must be removed from $\boldsymbol{h}_{t-1}$



$$\boldsymbol{f_t} = \boldsymbol{\sigma}(\boldsymbol{W_f} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_f})$$

- **Input gate:** decides what new information to store in the cell

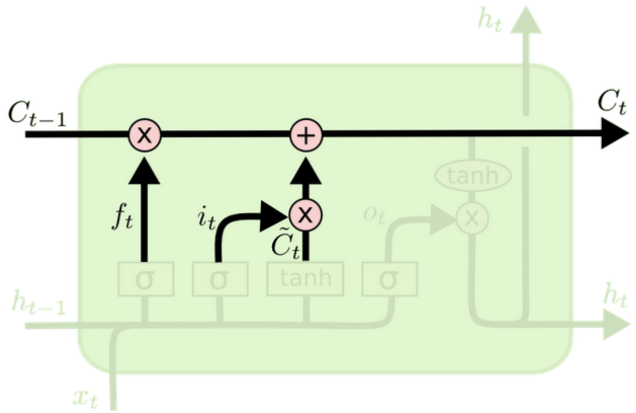


$$\boldsymbol{i_t} = \boldsymbol{\sigma}(\boldsymbol{W_i} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_i})$$

$$\widetilde{\boldsymbol{C}}_{\boldsymbol{t}} = \tanh(\boldsymbol{W_C} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_C})$$

Long Short Term Memory (LSTM)

- Update cell state:



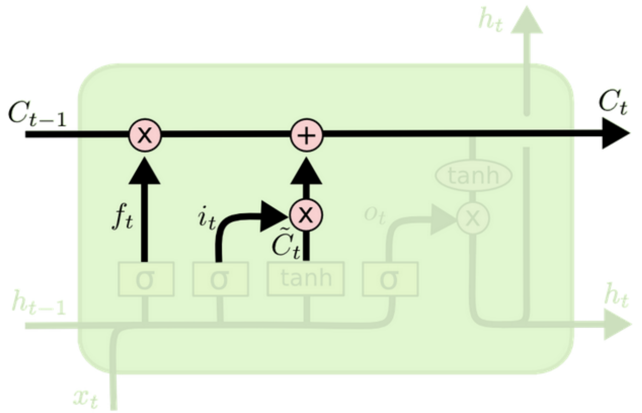
$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

forgetting unneeded things

scaling the new candidate values by how much we decided to update each state value.

Long Short Term Memory (LSTM)

- Update cell state:

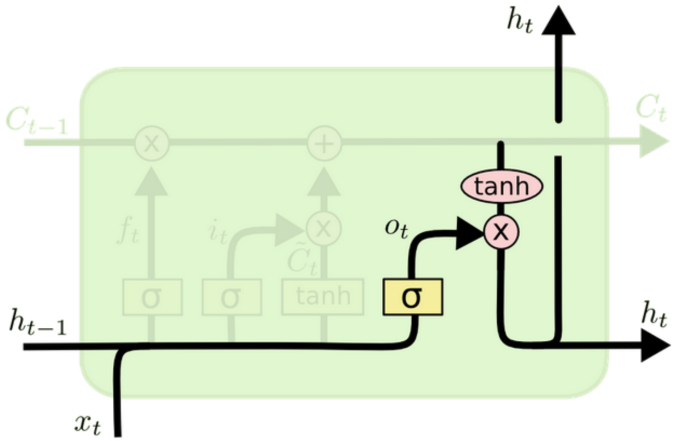


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

forgetting unneeded things

scaling the new candidate values by how much we decided to update each state value.

- Output gate: decides what to output from our cell state

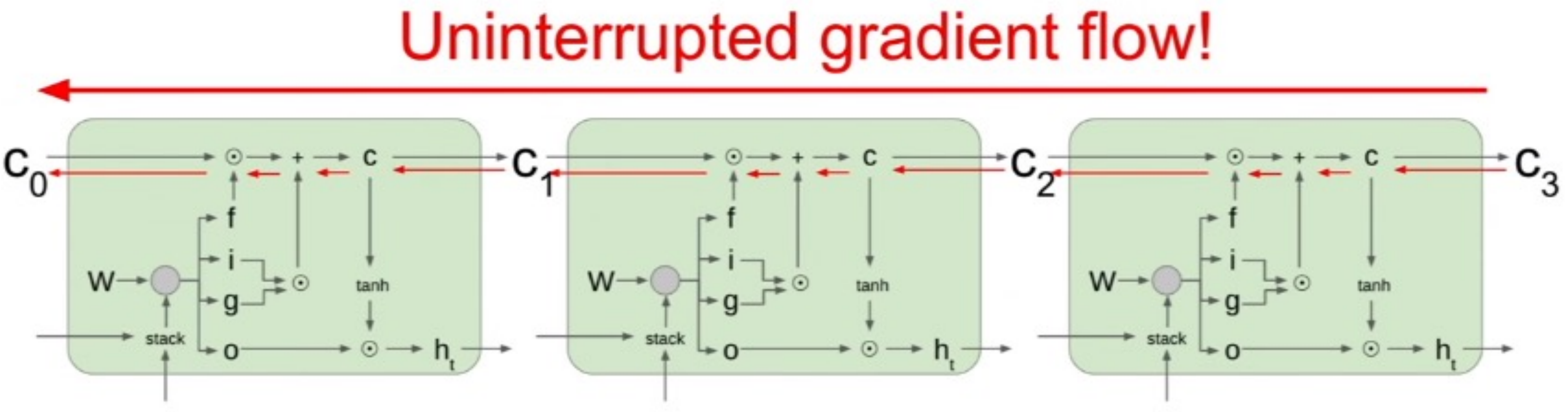


$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

sigmoid decides what parts of the cell state we're going to output

Backpropagation in LSTM



- No multiplication with matrix W during backprop
- Multiplied by different values of forget gate -> less prone to vanishing/exploding gradient

Source: CS231N Stanford

RNNs in Various Forms

One to One | One to Many | Many to One | Many to Many | Many to Many

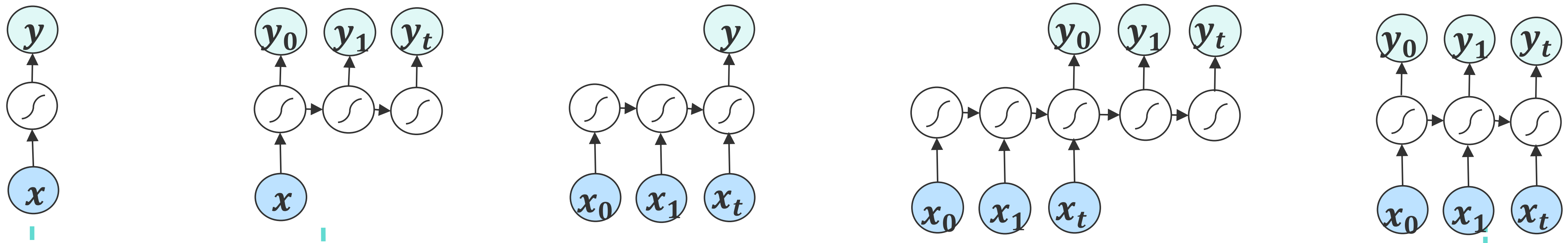


Image classification

Image captioning

Sentence sentiment analysis / Video recognition

Machine Translation

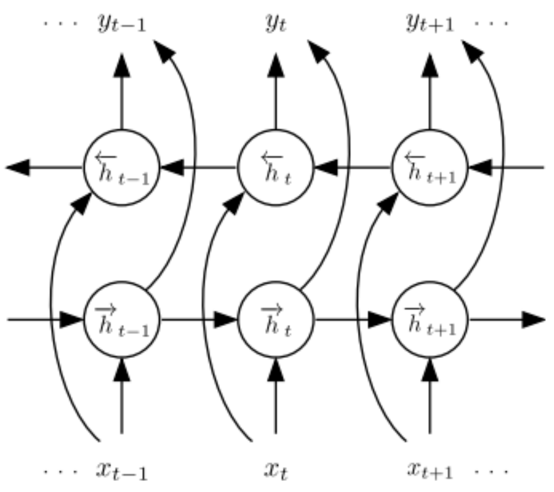
(Sequence-to-sequence)

Named Entity Recognition

(Sequence tagging)

RNNs in Various Forms

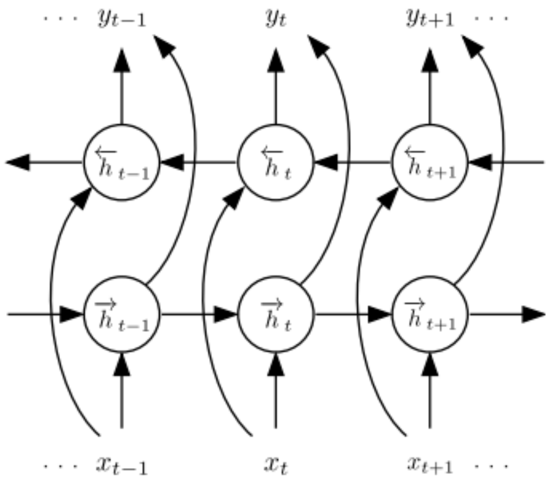
- Bi-directional RNN
  - Hidden state is the concatenation of both forward and backward hidden states.
  - Allows the hidden state to capture both past and future information.



[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.

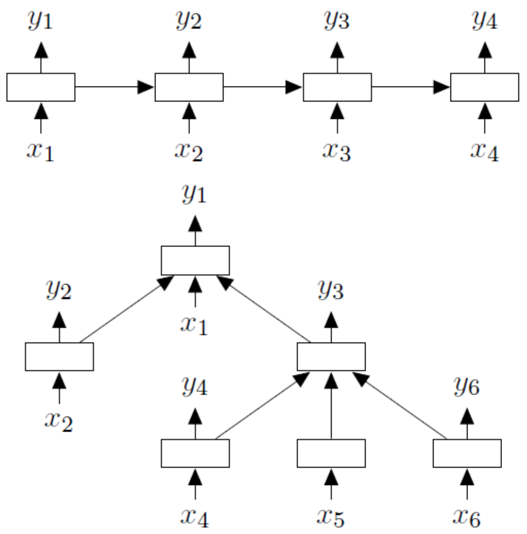


[Speech Recognition with Deep Recurrent Neural Networks, Alex Graves]

- 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.

Chain-structured LSTM

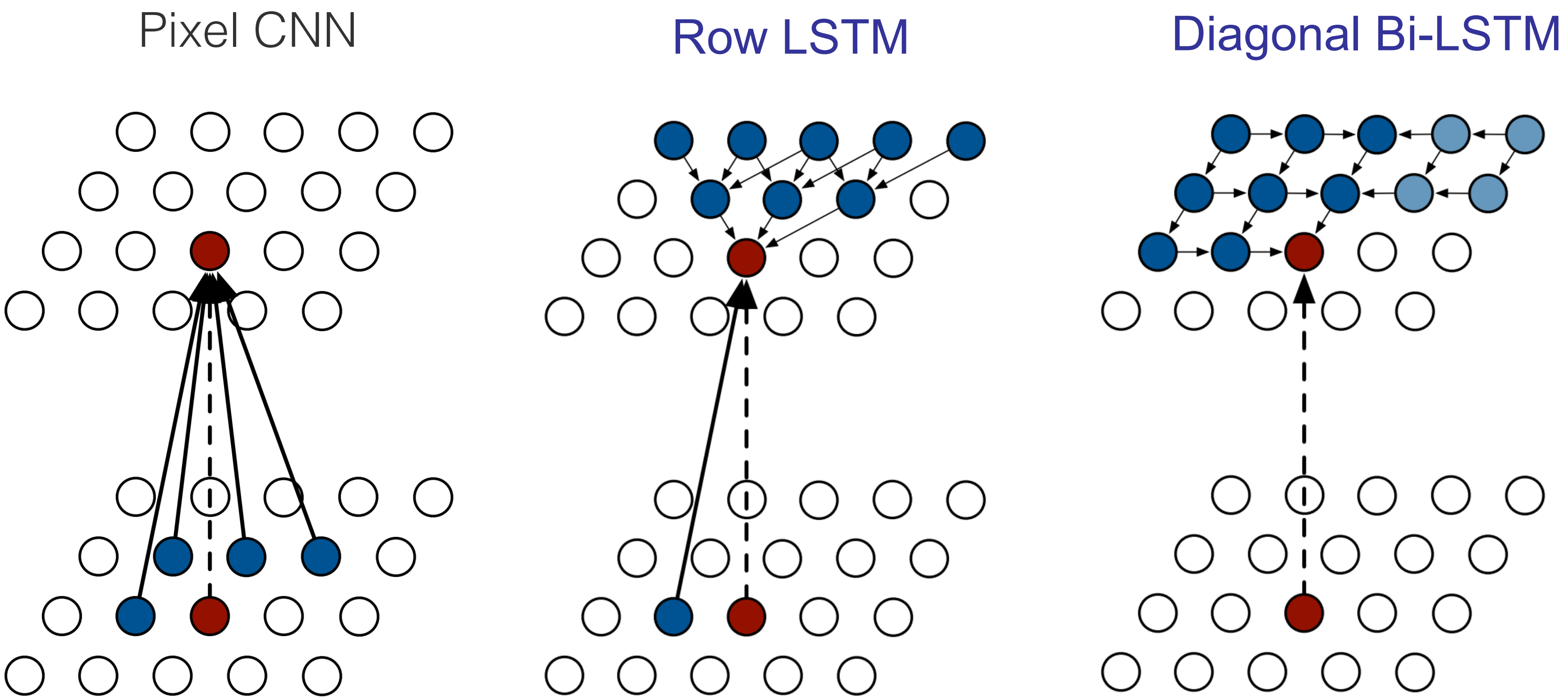
Tree-structured LSTM



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

RNNs in Various Forms

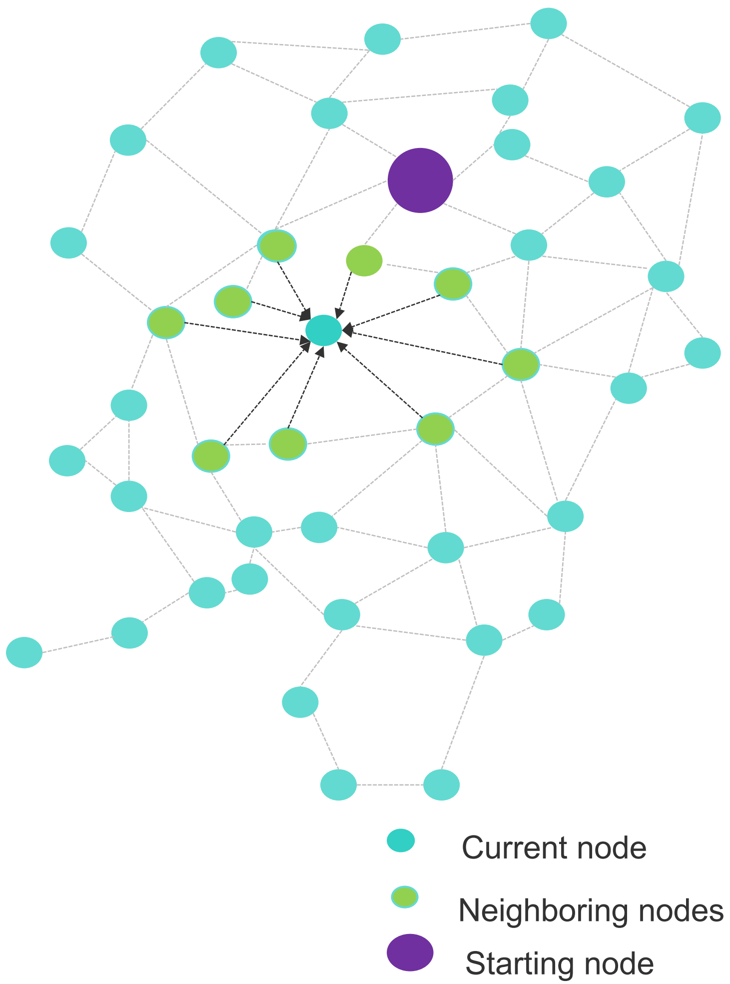
- RNN for 2-D sequences



[Pixel Recurrent Neural Networks, van den Oord. et al. 2016]

RNNs in Various Forms

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



[Semantic Object Parsing with Graph LSTM. Liang et al. 2016]

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.

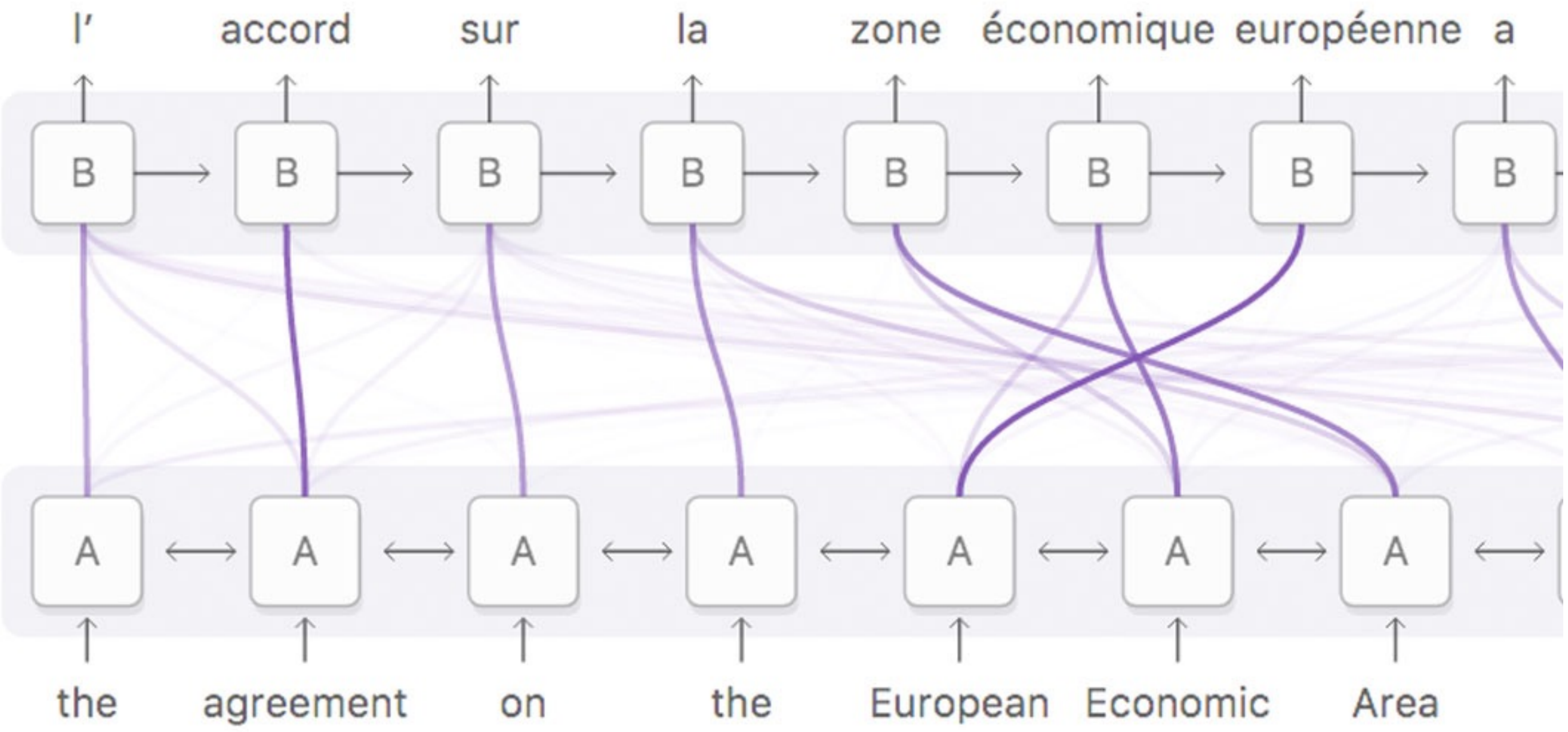
A group of people sitting on a boat in the water.

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

Figure courtesy: Olah & Carter, 2016

Why Attention?

Figure courtesy: keitakurita

Why Attention?

- Long-range dependencies
  - Dealing with gradient vanishing problem

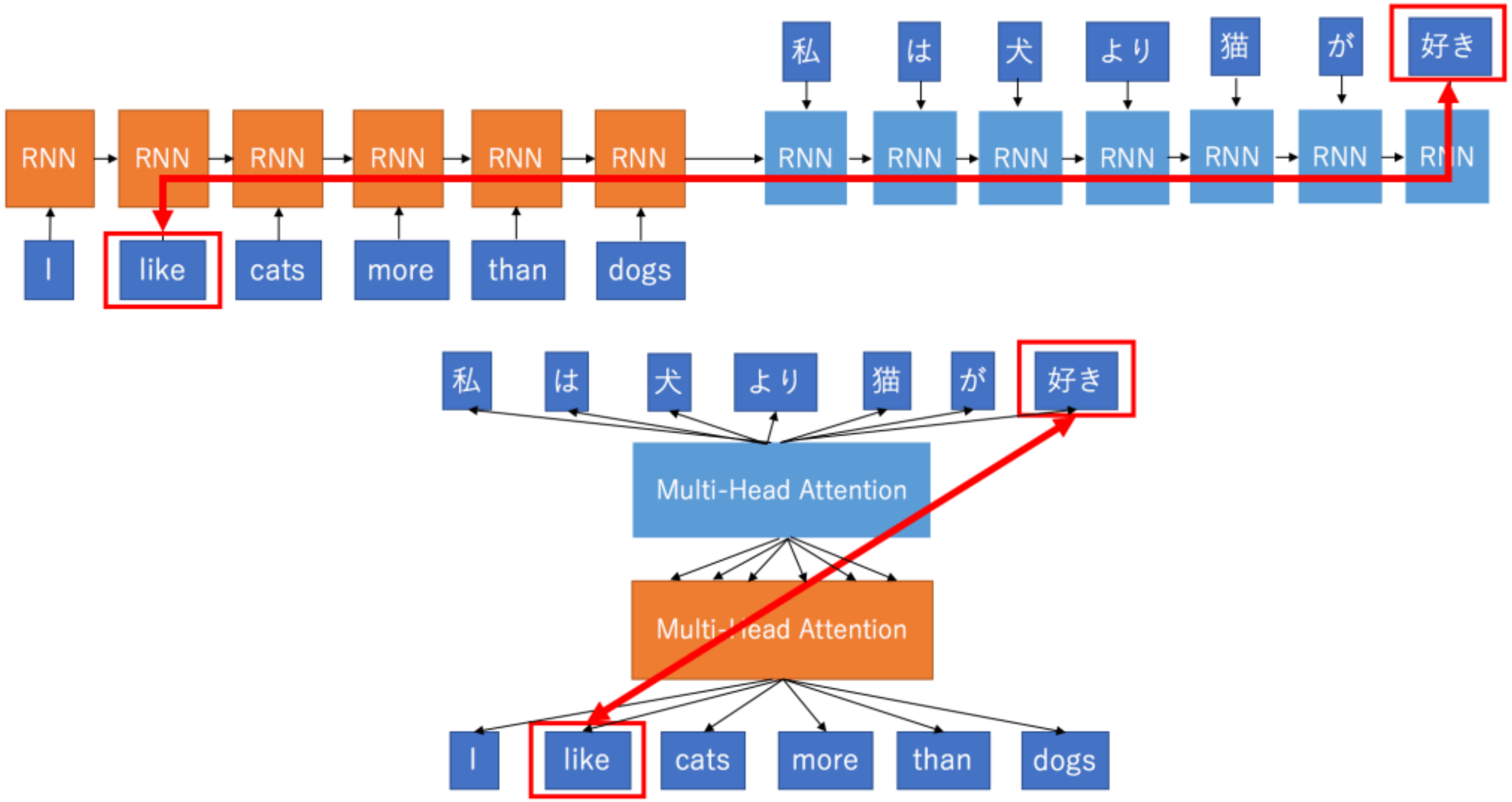


Figure courtesy: keitakurita

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

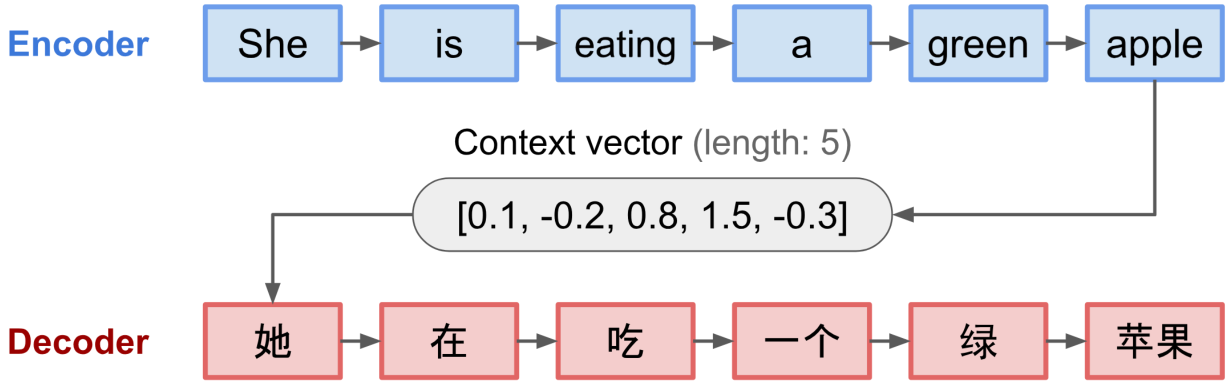


Figure courtesy: Lilian Weng

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

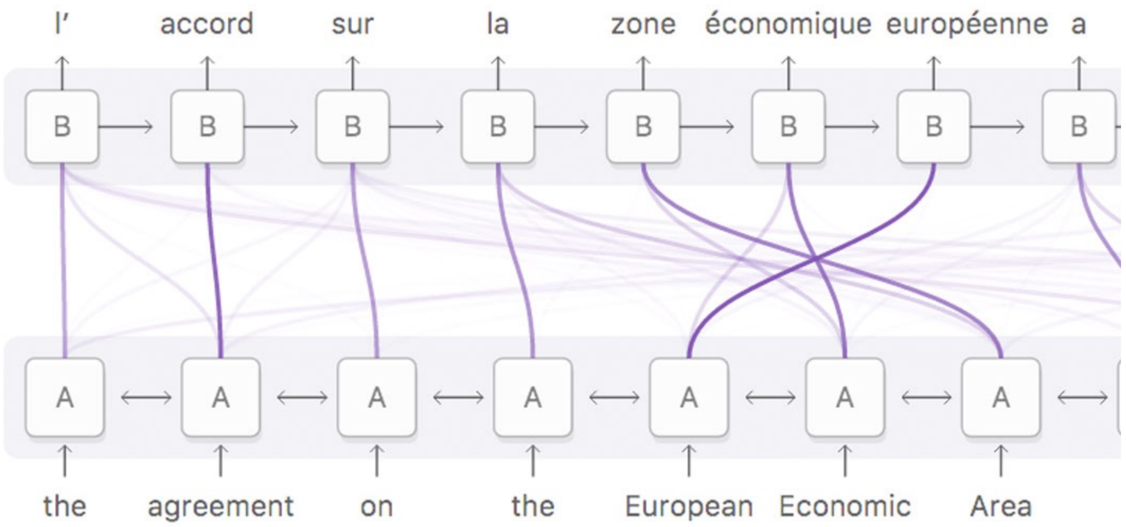


Figure courtesy: Olah & Carter, 2016

Attention Computation

- Encode each token in the input sentence into vectors
- When decoding, perform a linear combination of these vectors, weighted by "attention weights"
  - $\boldsymbol{a} = \mathrm{softmax}(\boldsymbol{alignment\_scores})$

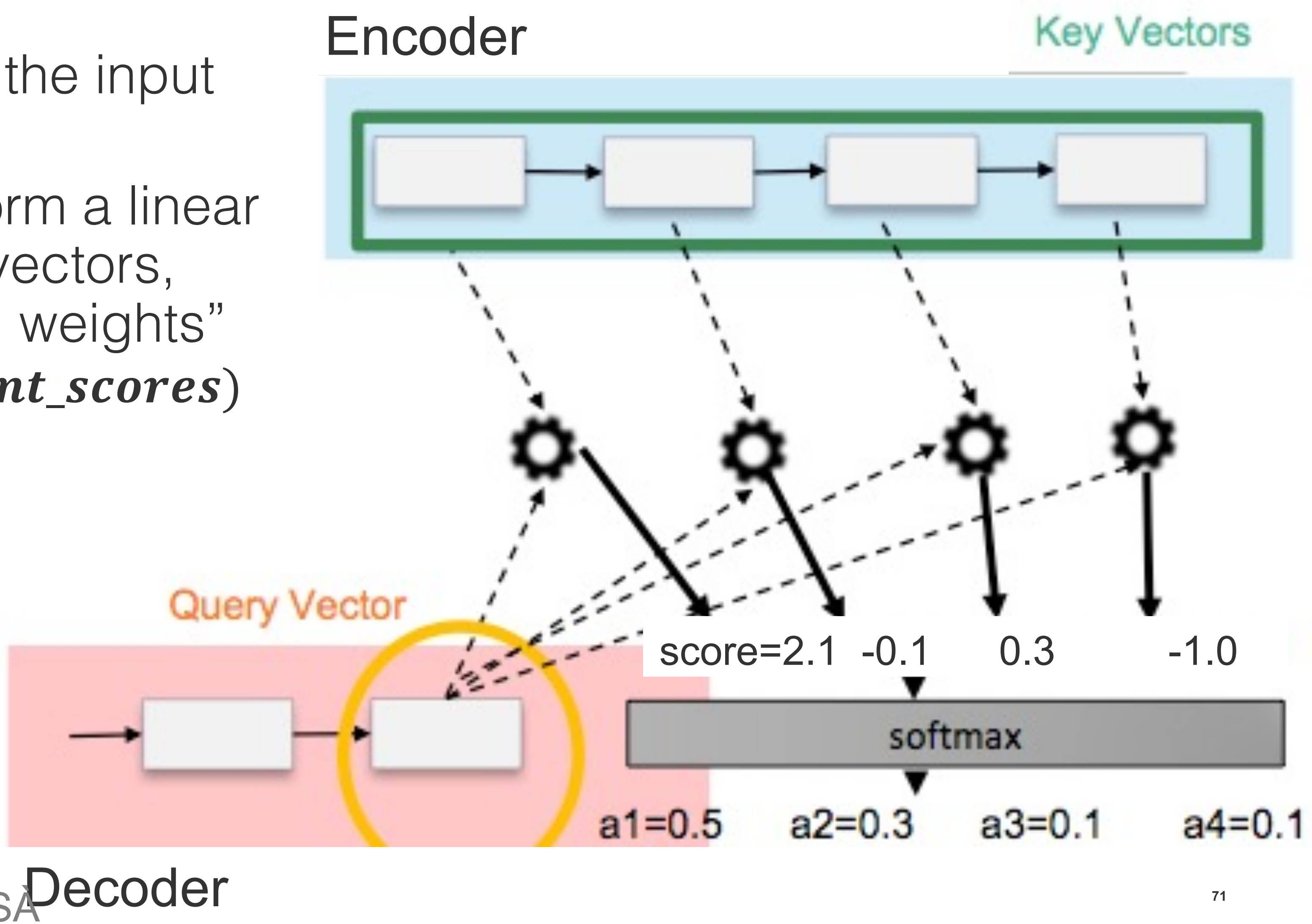
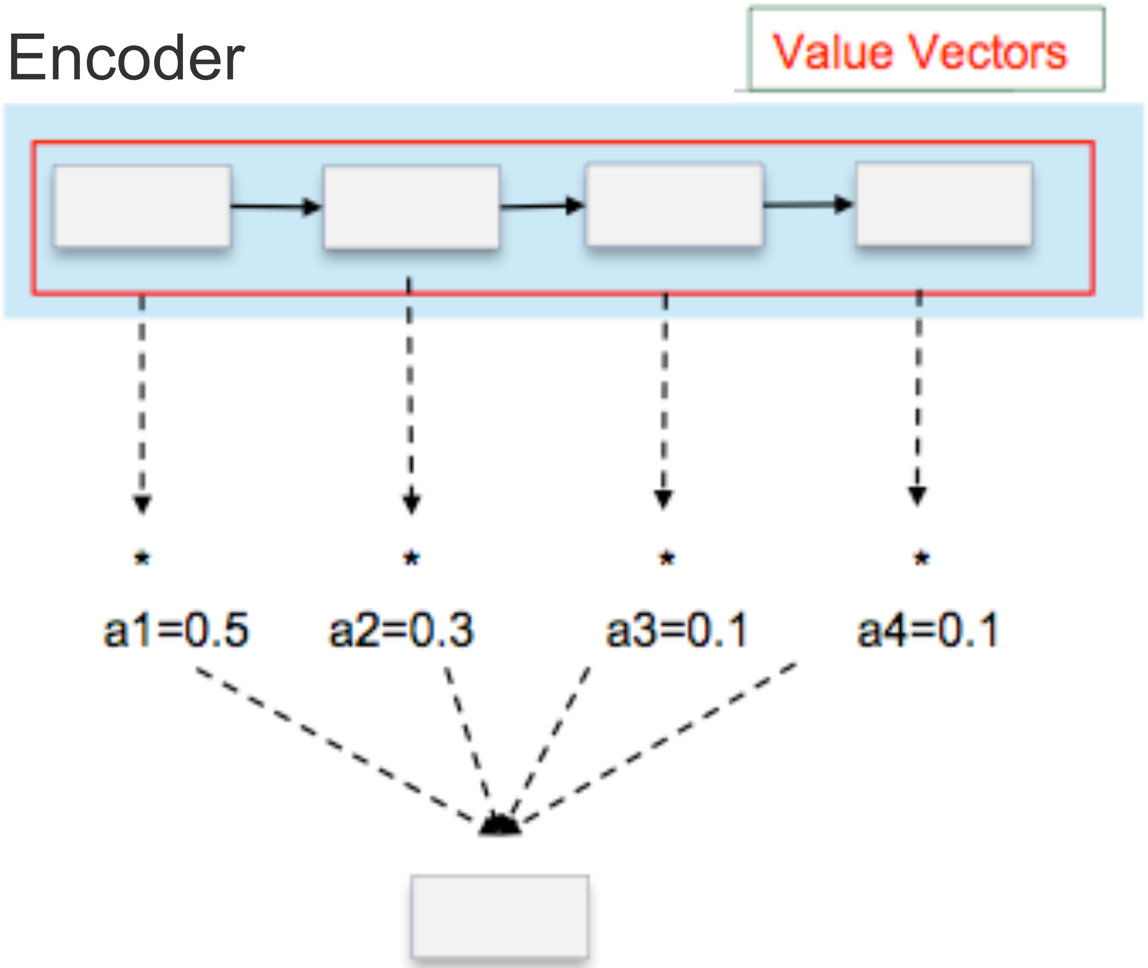


Figure courtesy: MARTA R. COSTA-JUSSÀ

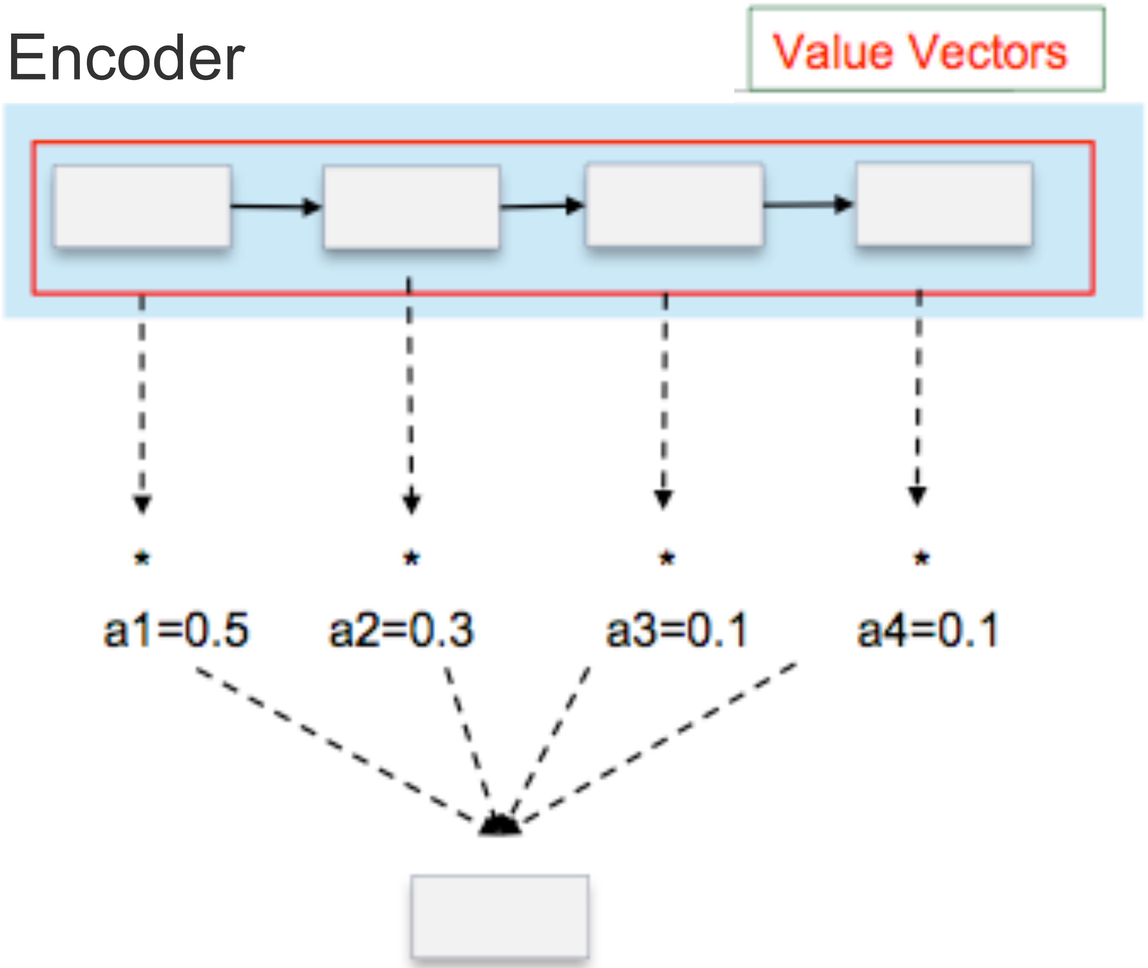
Attention Computation (cont'd)

- 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

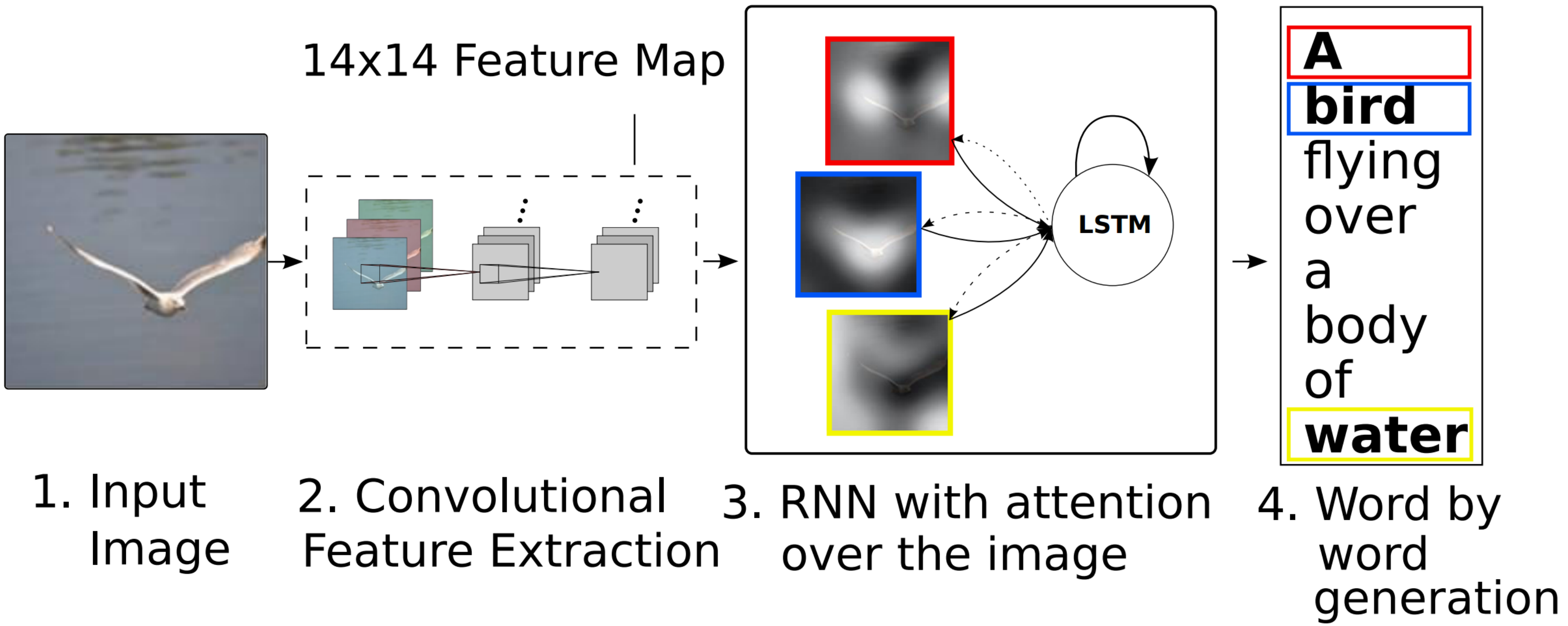
Alignment score = f(Query, Keys)

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

Courtesy: Lilian Weng

Attention on Images – Image Captioning

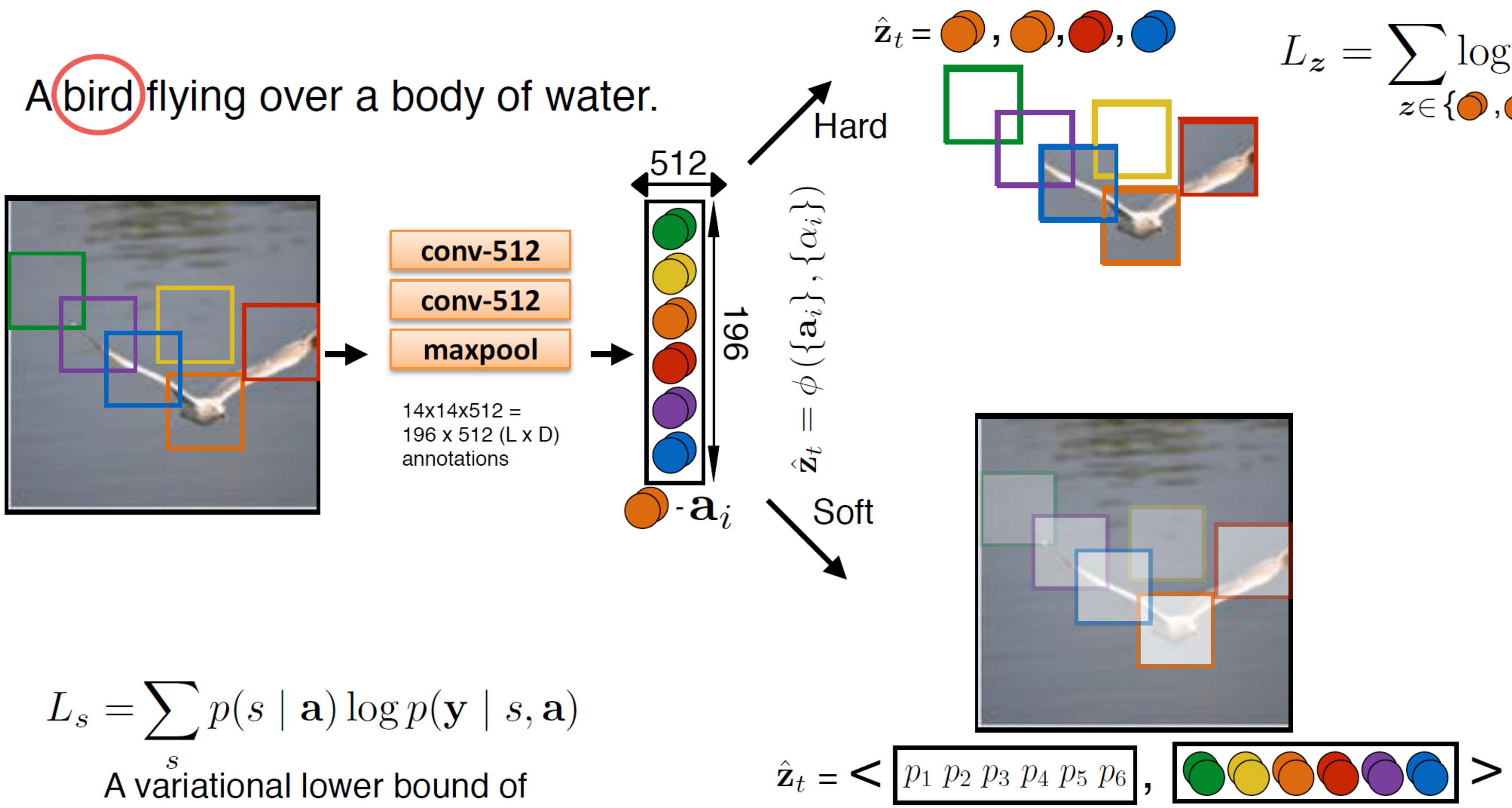


- Query: decoder state
- Key: visual feature maps
- Value: visual feature maps

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

Attention on Images – Image Captioning

Hard attention *vs* Soft attention



$$L_z = \sum_{z \in \{\cdots\}} \log p(\boldsymbol{y} \mid \boldsymbol{z})$$

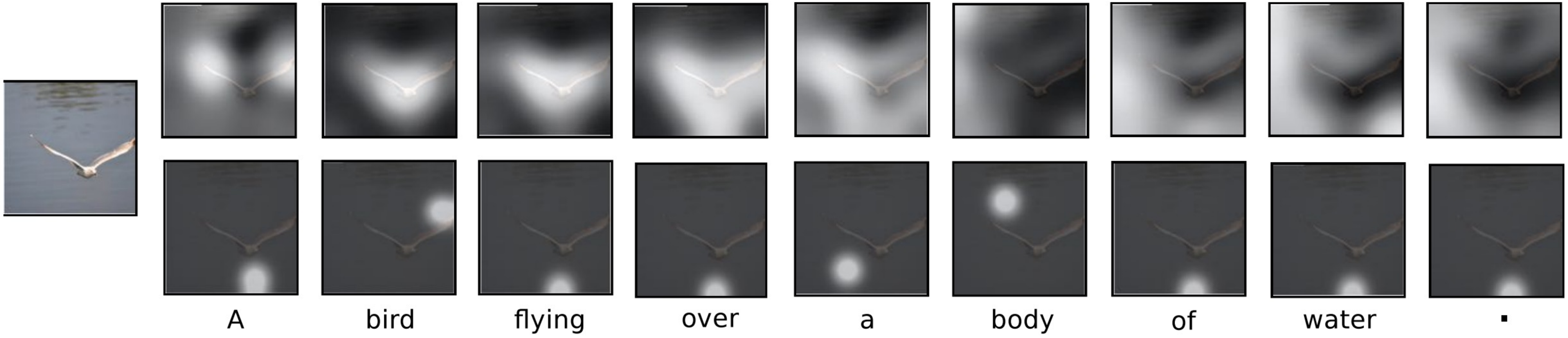
$$L_s = \sum_s p(s \mid \mathbf{a}) \log p(\mathbf{y} \mid s, \mathbf{a})$$

A variational lower bound of maximum likelihood

Computes the expected attention

Attention on Images – Image Captioning

Hard attention *vs* Soft attention



Attention on Images – Image Paragraph Generation

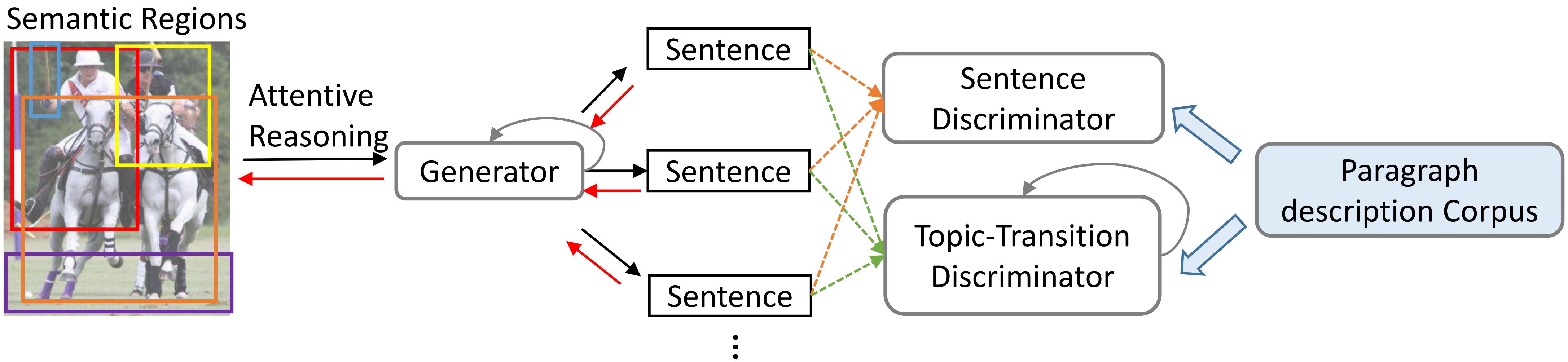
- Generate a long paragraph to describe an image
    - Long-term visual and language reasoning
    - Contentful descriptions -- ground sentences on visual features

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

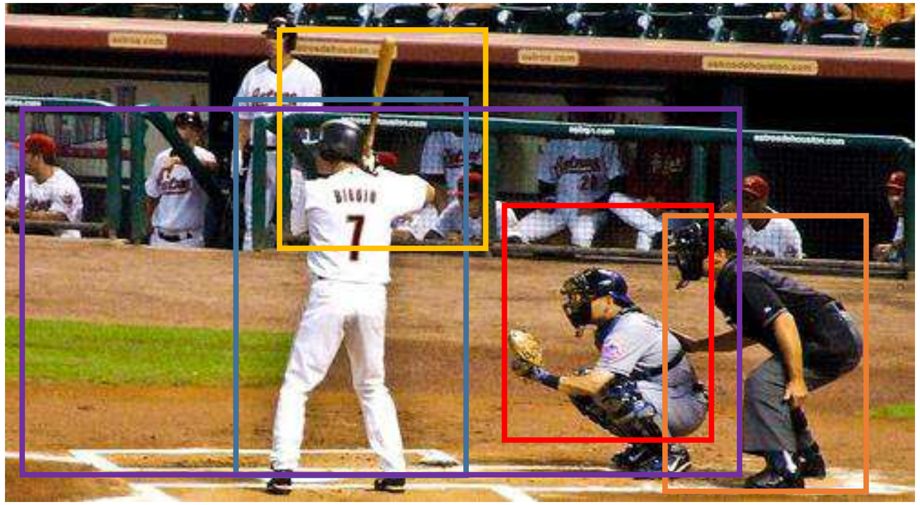


[Recurrent Topic-Transition GAN for Visual Paragraph Generation. Liang et al. 2017]

Attention on Images – Image Paragraph Generation

Semantic Regions

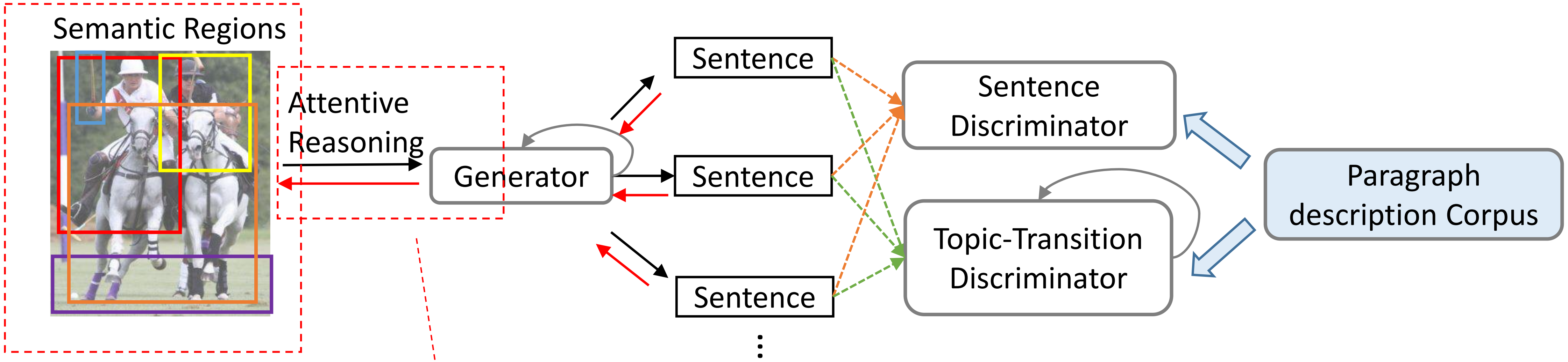
Semantic region detection & captioning



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

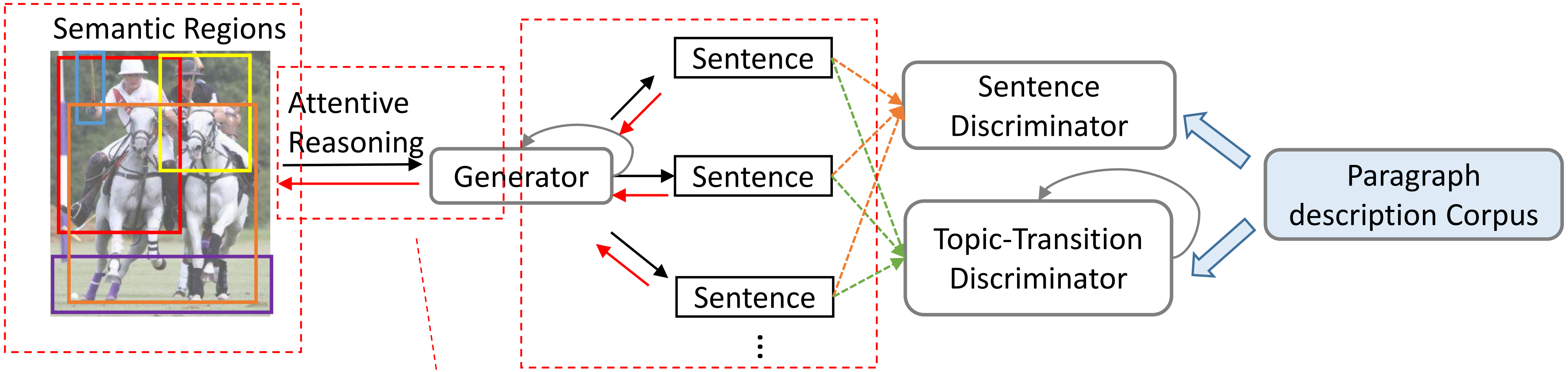
Attention on Images – Image Paragraph Generation



Semantic region
detection & captioning

Attention on both visual
regions and text
phrases

Attention on Images – Image Paragraph Generation

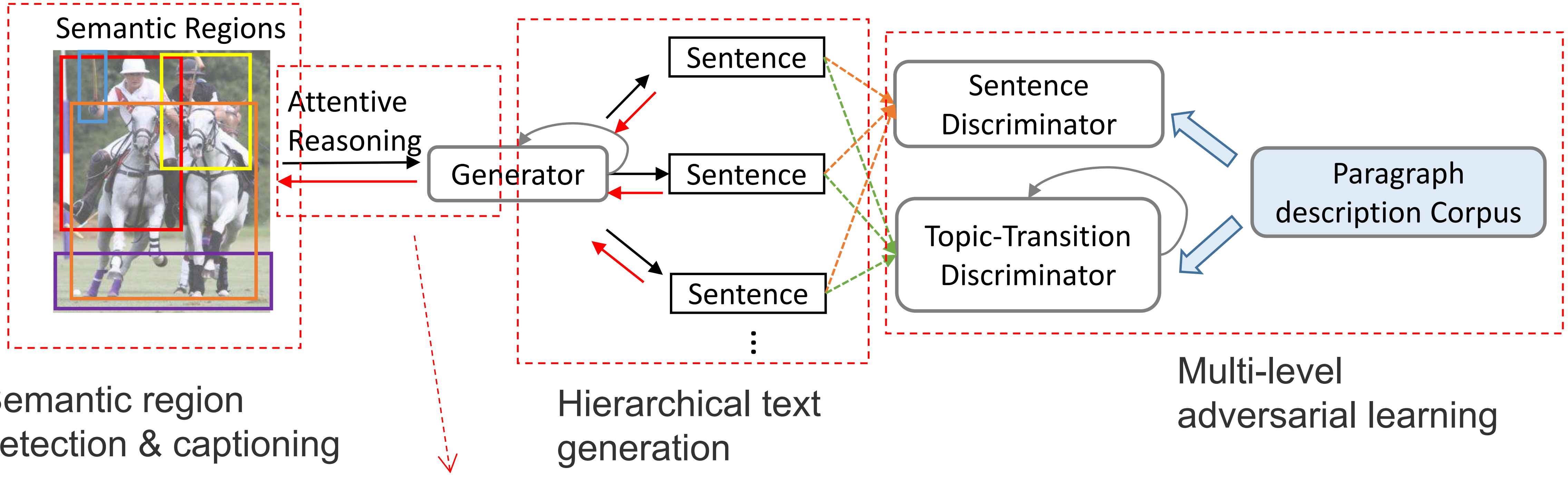


Semantic region detection & captioning

Hierarchical text generation

Attention on both visual regions and text phrases

Attention on Images – Image Paragraph Generation



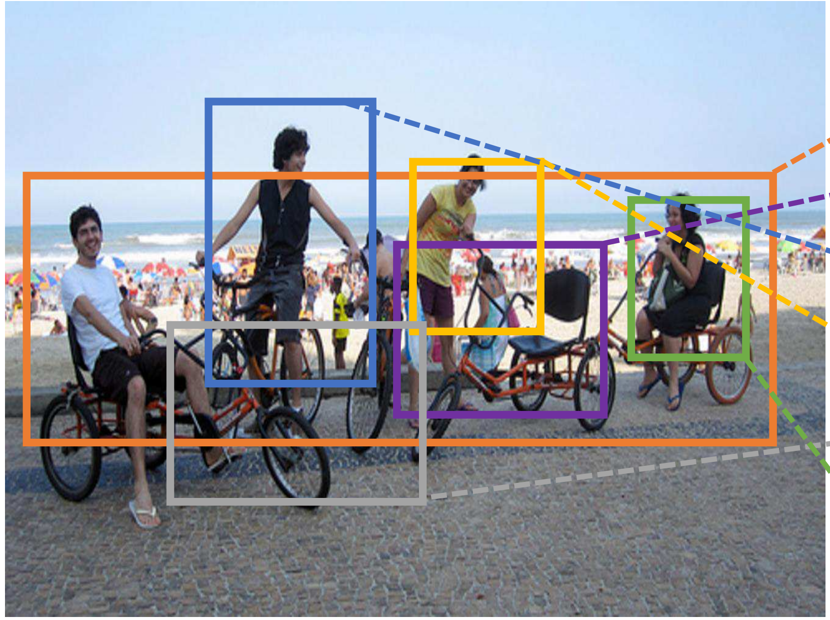
Semantic region
detection & captioning

Hierarchical text
generation

Multi-level
adversarial learning

Attention on both visual
regions and text
phrases

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.*

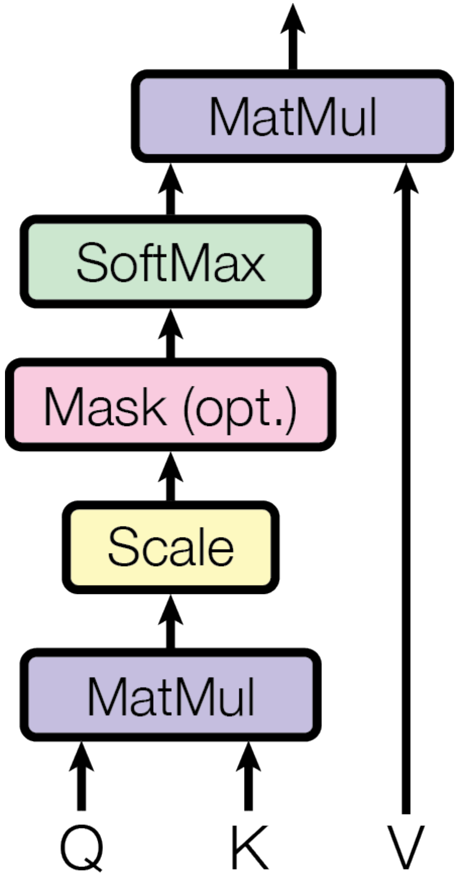
Outline

- Convolutional Networks (ConvNets)
- Recurrent Networks (RNNs)
  - Long-range dependency, vanishing
  - 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

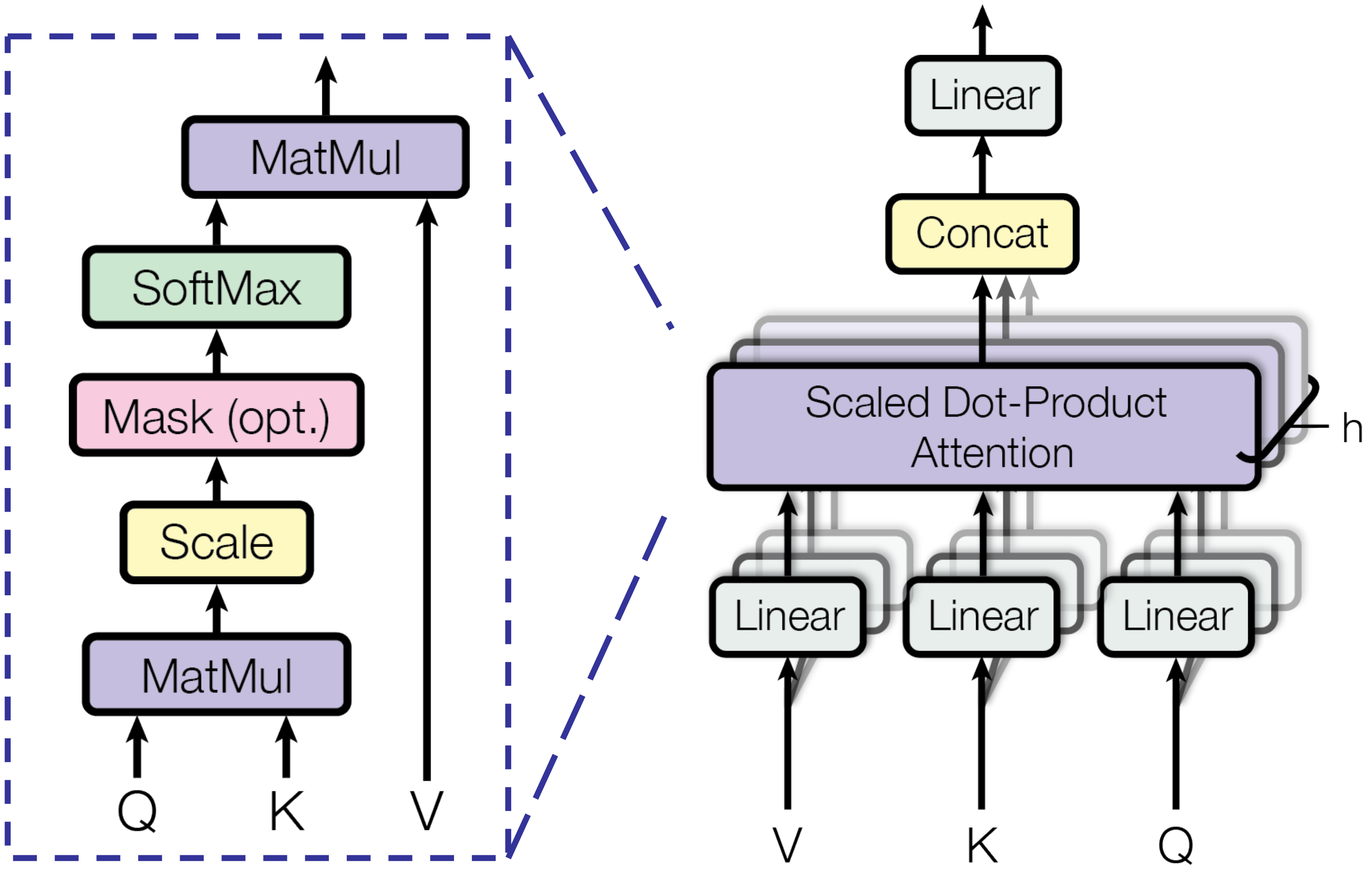
Multi-head Attention



Scaled Dot-Product Attention

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

Multi-head Attention

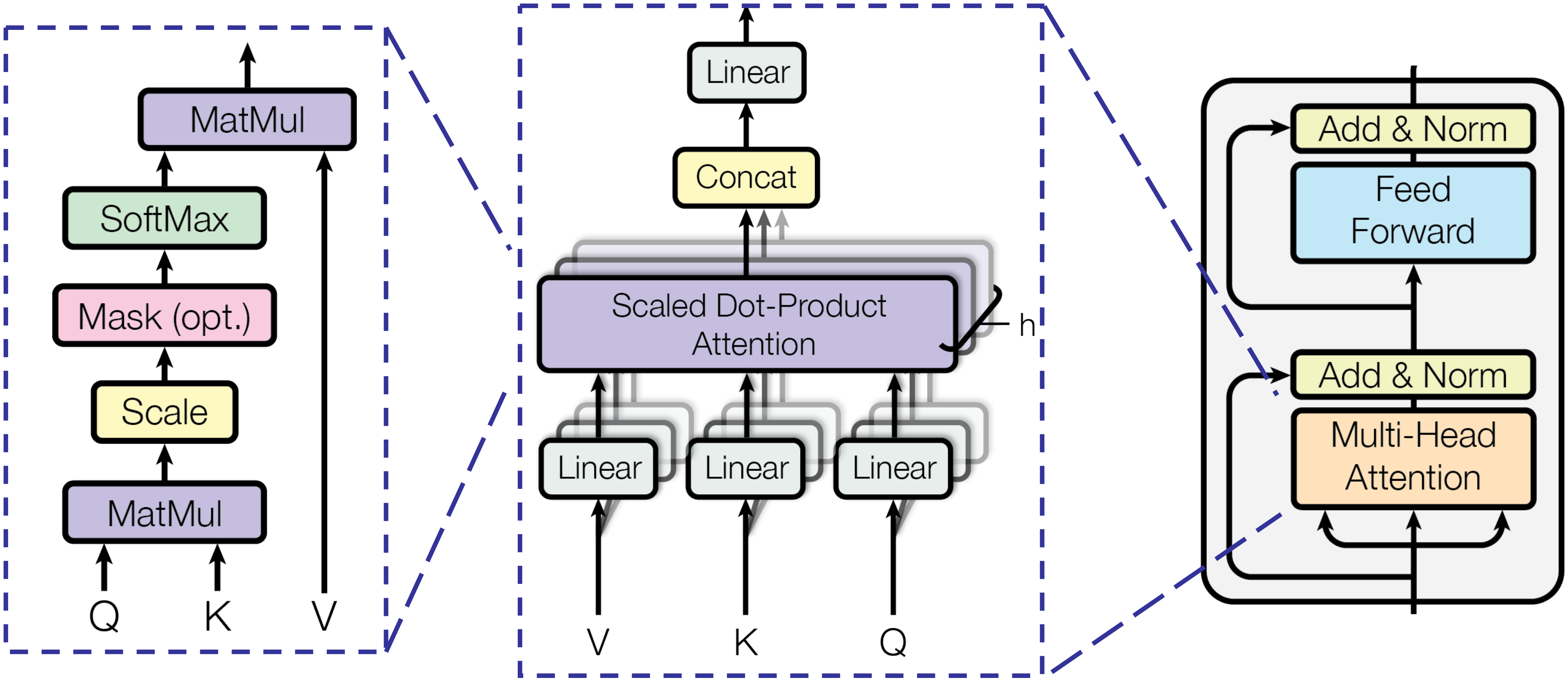


Scaled Dot-Product Attention

Multi-head Attention

Image source: Vaswani, et al., 2017

Multi-head Attention



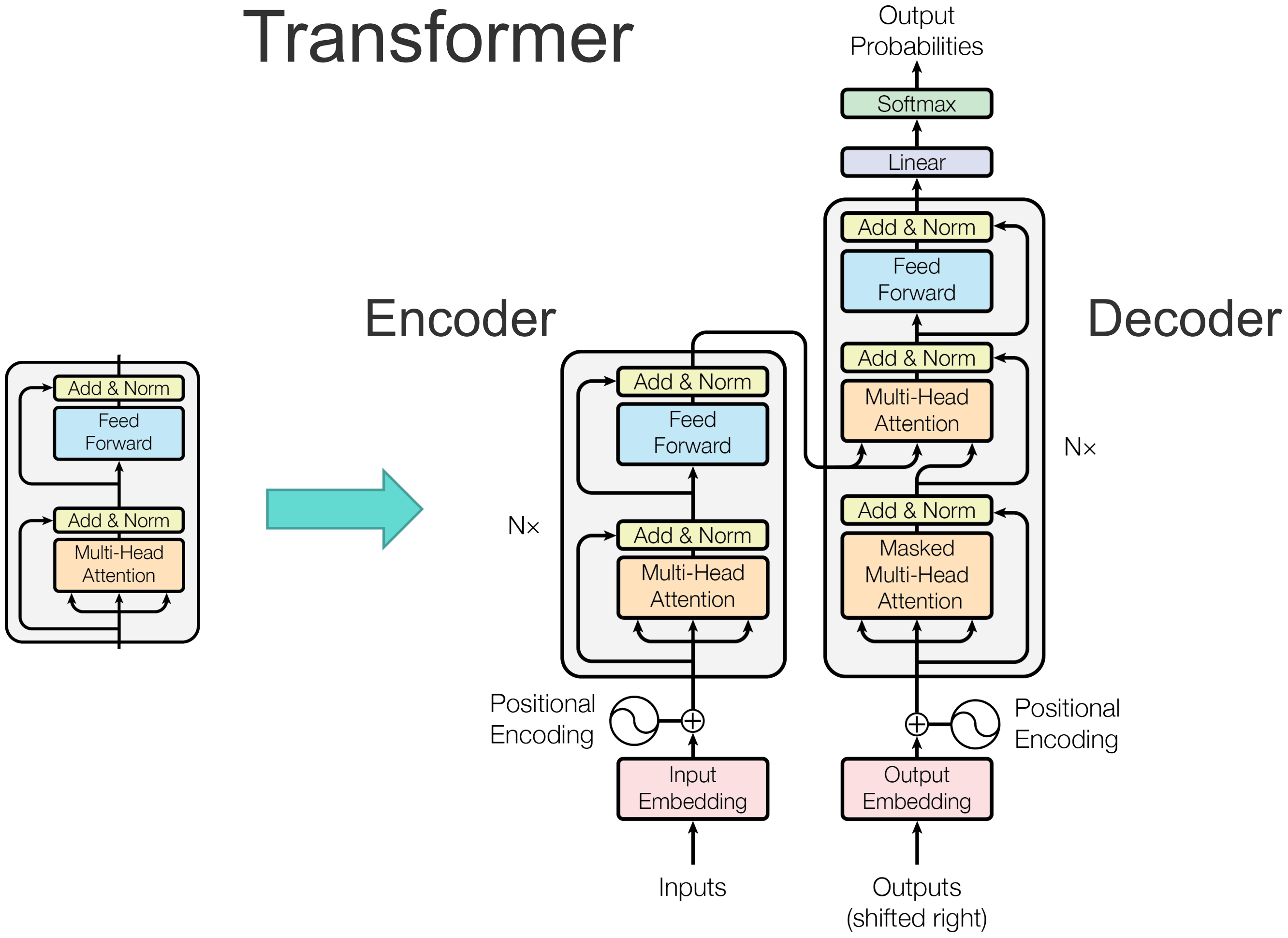
Scaled Dot-Product Attention

Multi-head Attention

Image source: Vaswani, et al., 2017

Multi-head Attention in Encoders and Decoders

Transformer



Multi-head Attention in Encoders and Decoders

Transformer

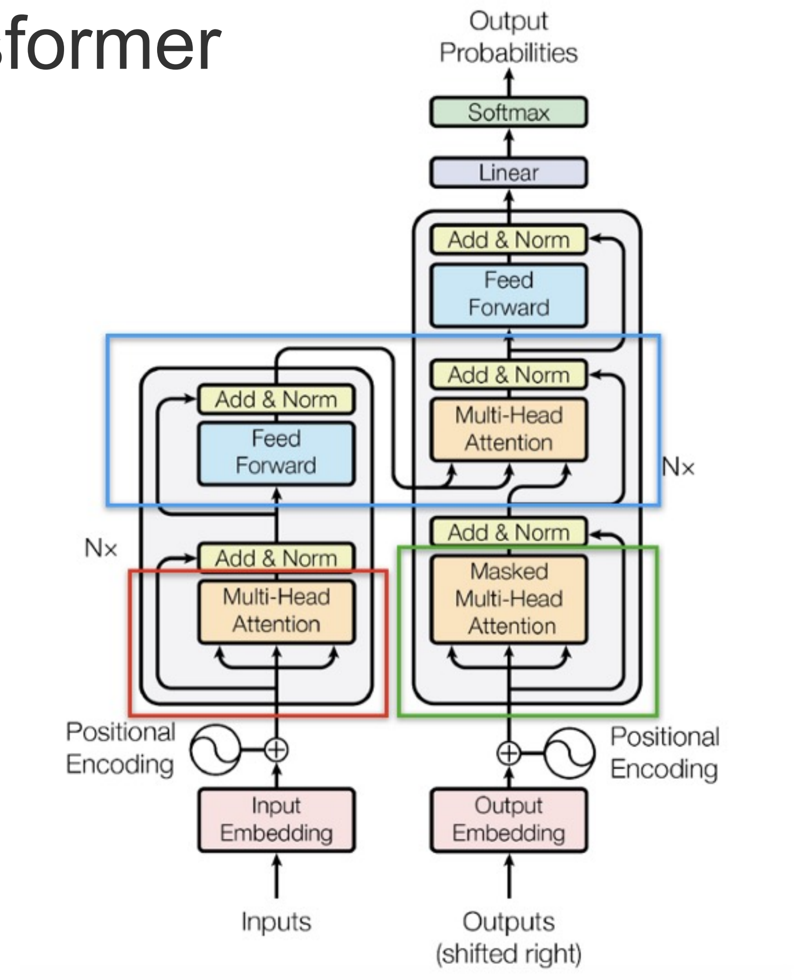


Figure 1: The Transformer - model architecture.

encoder self attention

1. Multi-head Attention
2. Query=Key=Value

decoder self attention

1. Masked Multi-head Attention
2. Query=Key=Value

encoder-decoder attention

1. Multi-head Attention
2. Encoder Self attention=Key=Value
3. Decoder Self attention=Query

Image source: Bgg

Questions?