CS 4803 / 7643: Deep Learning

Topics:
- Attention
- Transformers
- Natural Language Applications

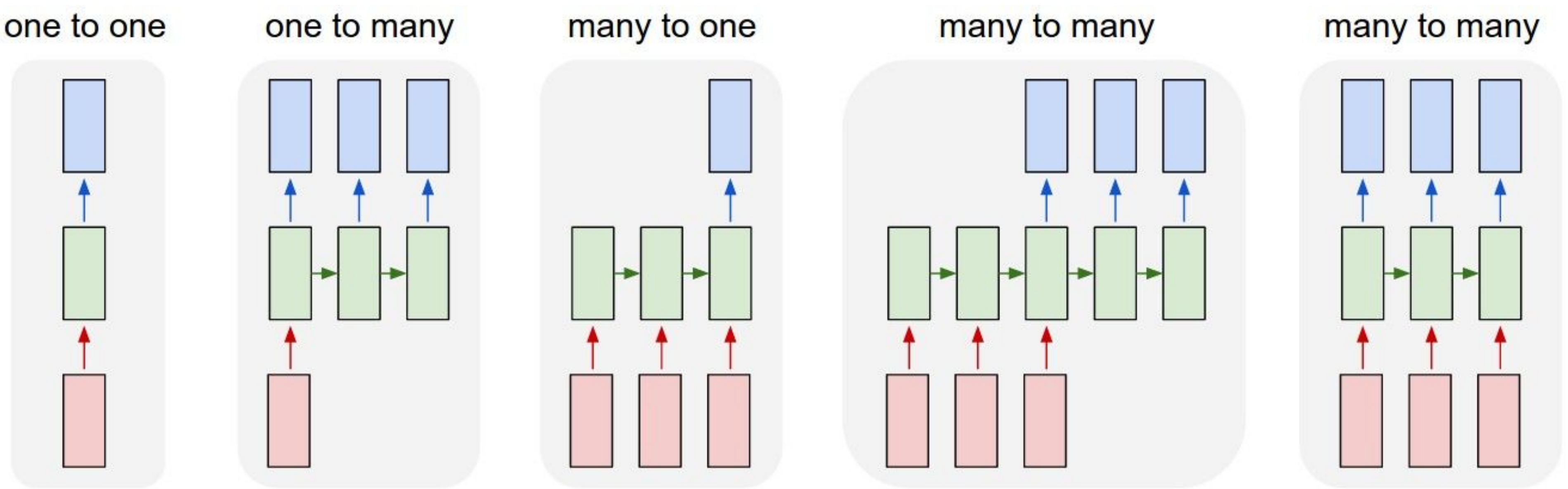
Sarah Wiegreffe
Georgia Tech

Plan for Today

- Recap
  - RNNs
  - Image Captioning
- RNNs with Attention
- Machine Translation
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- More natural language applications
  - Language Modeling (ELMo, GPT)
  - BERT ("Masked Language Modeling")

Slide Credit: Sarah Wiegreffe

Recap: RNNs



Input: No sequence

Output: No sequence

Example: “standard” classification / regression problems

Input: No sequence

Output: Sequence

Example: Im2Caption

Input: Sequence

Output: No sequence

Example: sentence classification, multiple-choice question answering

Input: Sequence

Output: Sequence

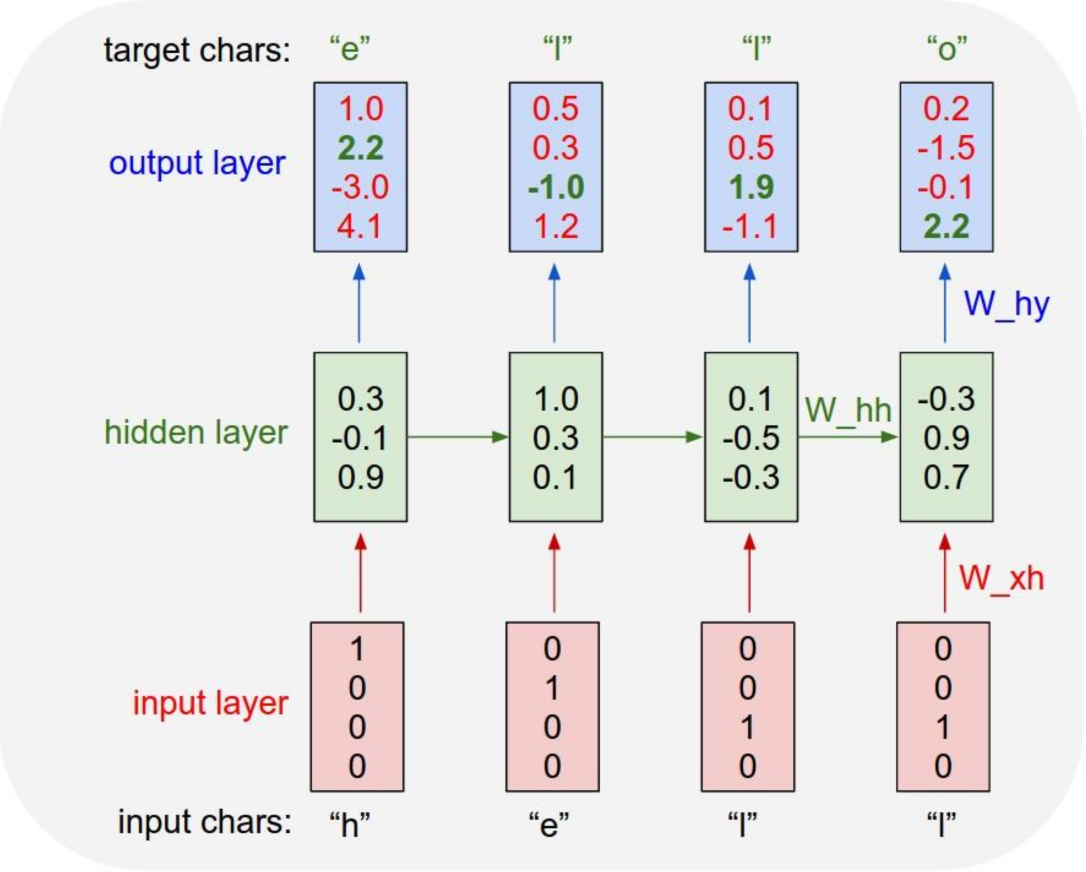
Example: machine translation, video classification, video captioning, open-ended question answering

Recap: RNNs

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **"hello"**



Neural Image Captioning

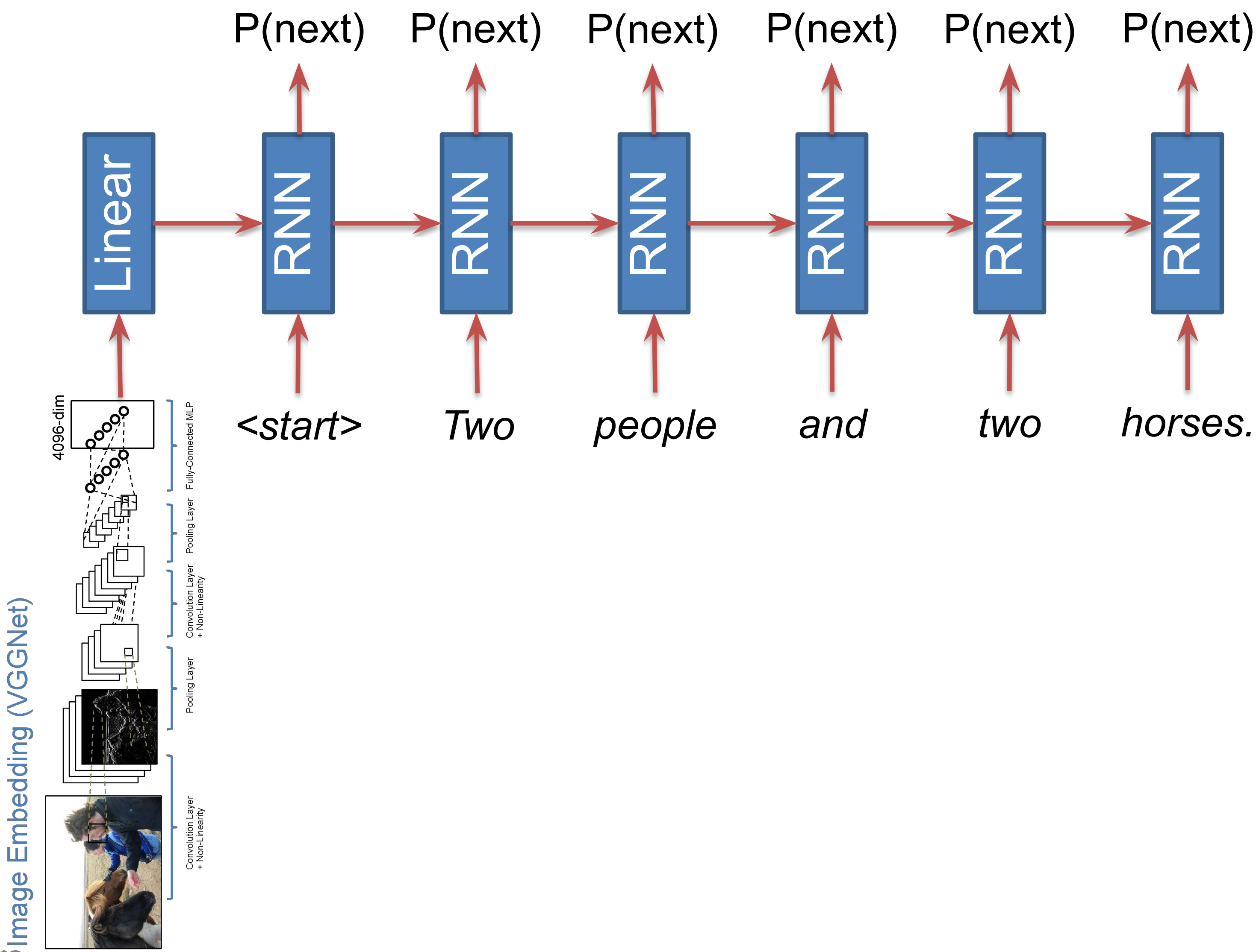


Image Captioning: Example Results

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

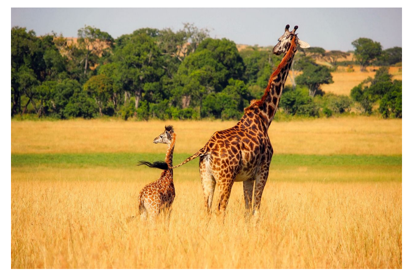
A cat is sitting on a tree branch

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



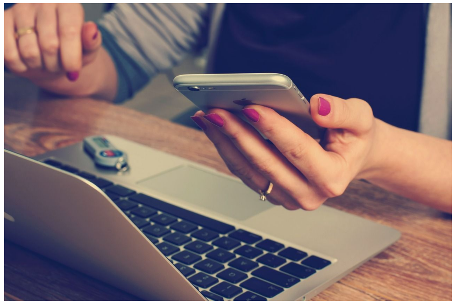
Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

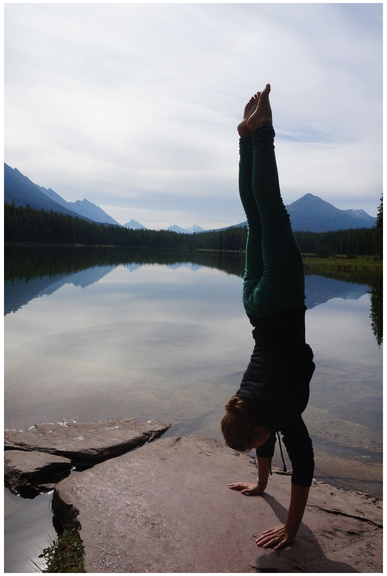
Image Captioning: Failure Cases

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

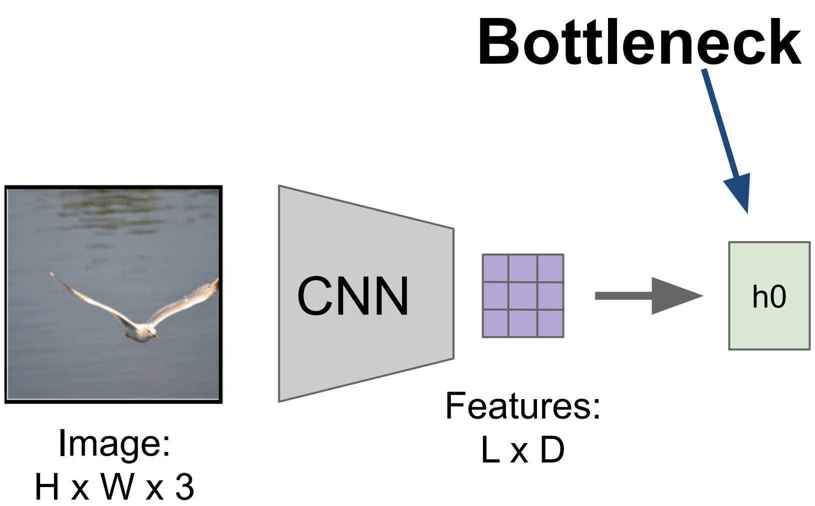
A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Plan for Today

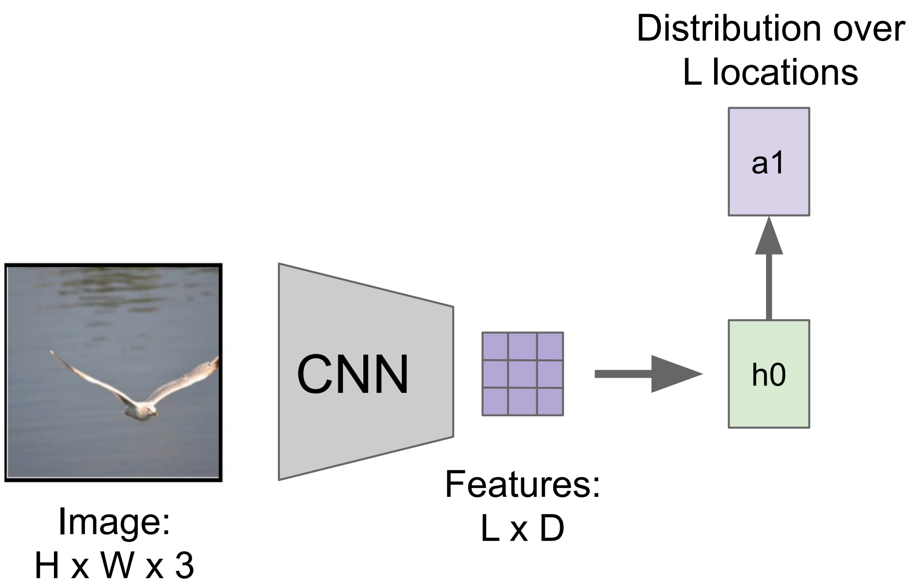
- Recap
  - RNNs, Image Captioning
- Image Captioning with Attention
- Machine Translation
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- More natural language applications (if time)
  - Language Modeling (ELMo, GPT)
  - BERT ("Masked Language Modeling")

Image Captioning with Attention



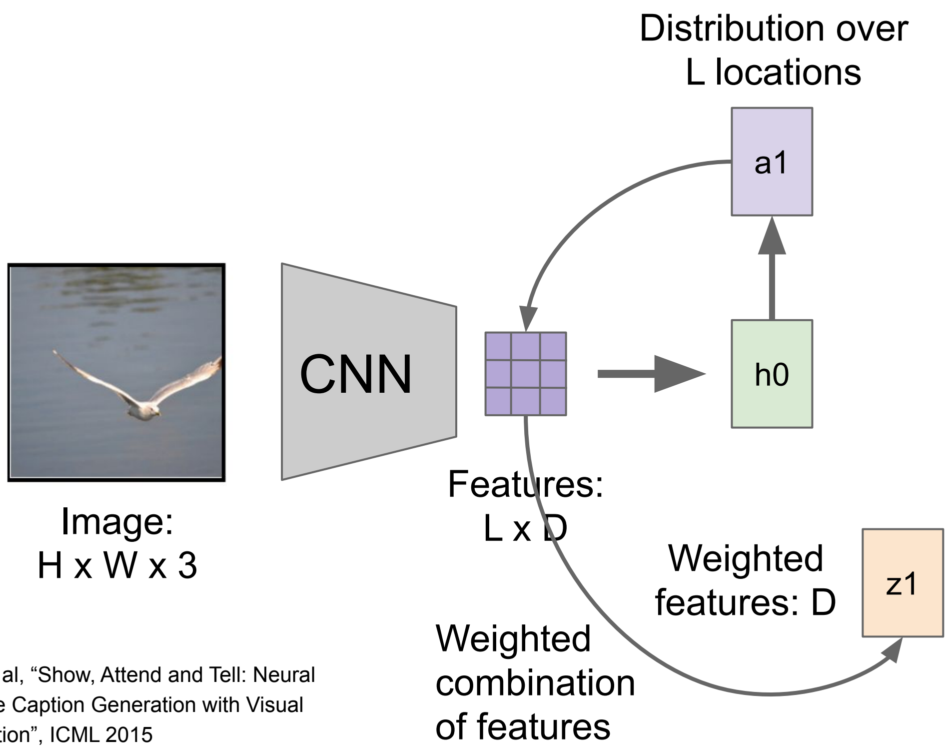
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



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

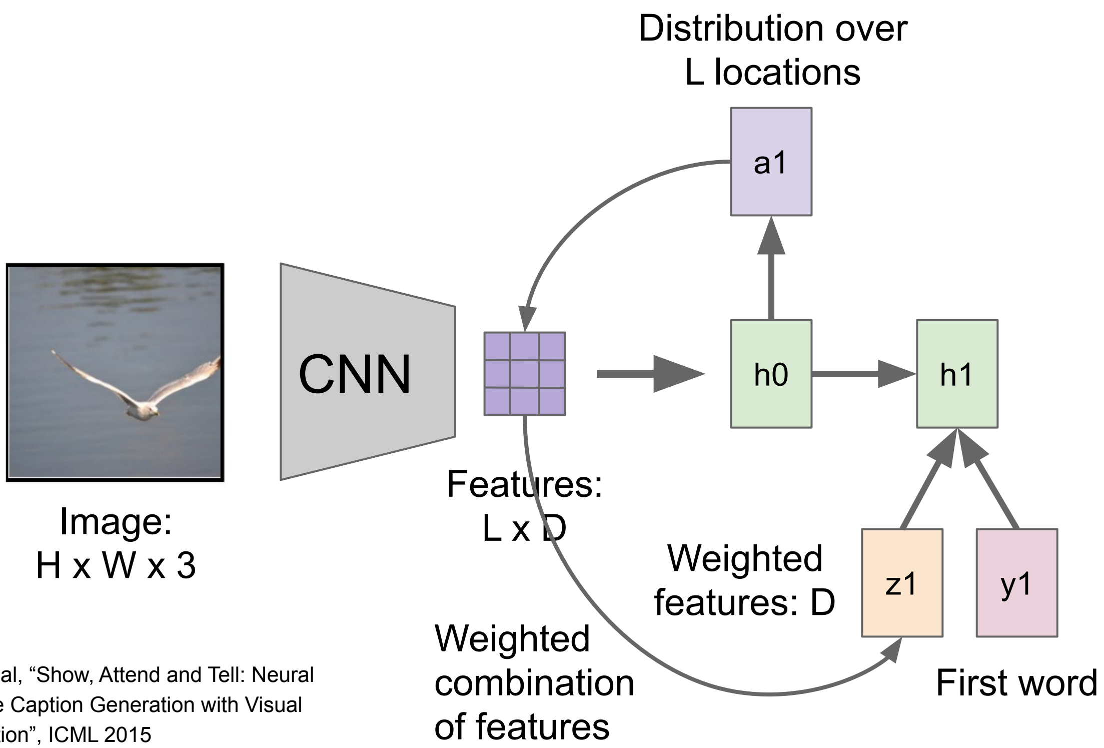
Image Captioning with Attention



$$z = \sum_{i=1}^{L} p_i v_i$$

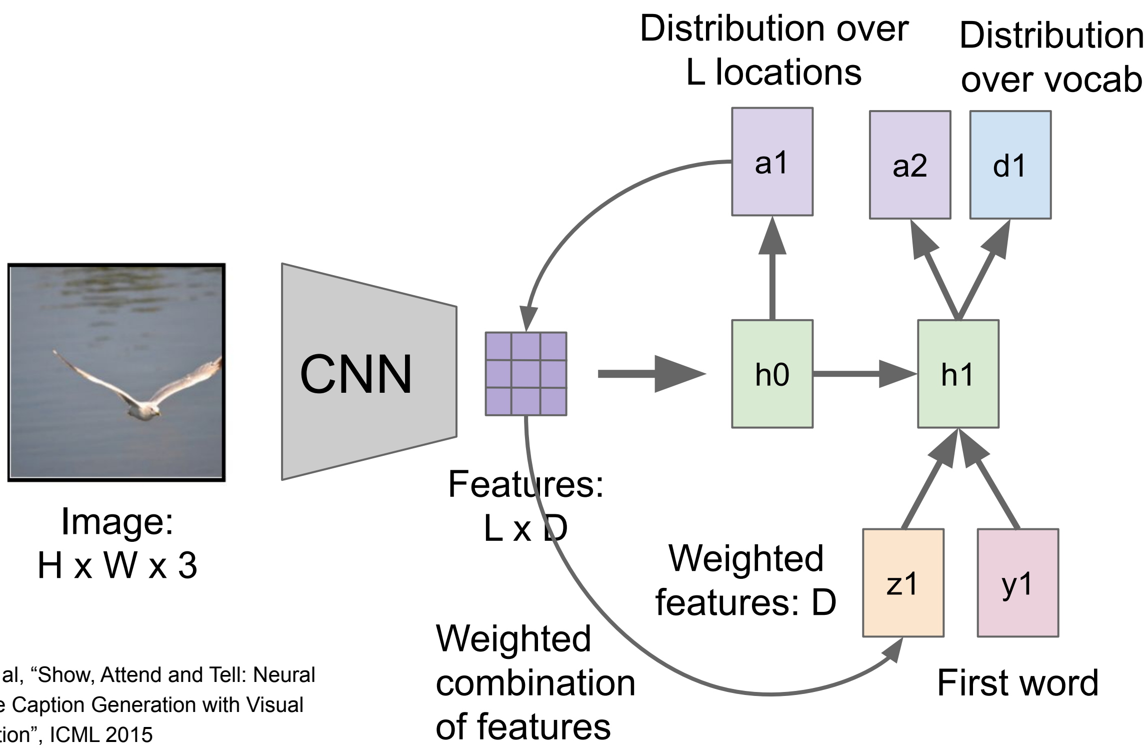
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



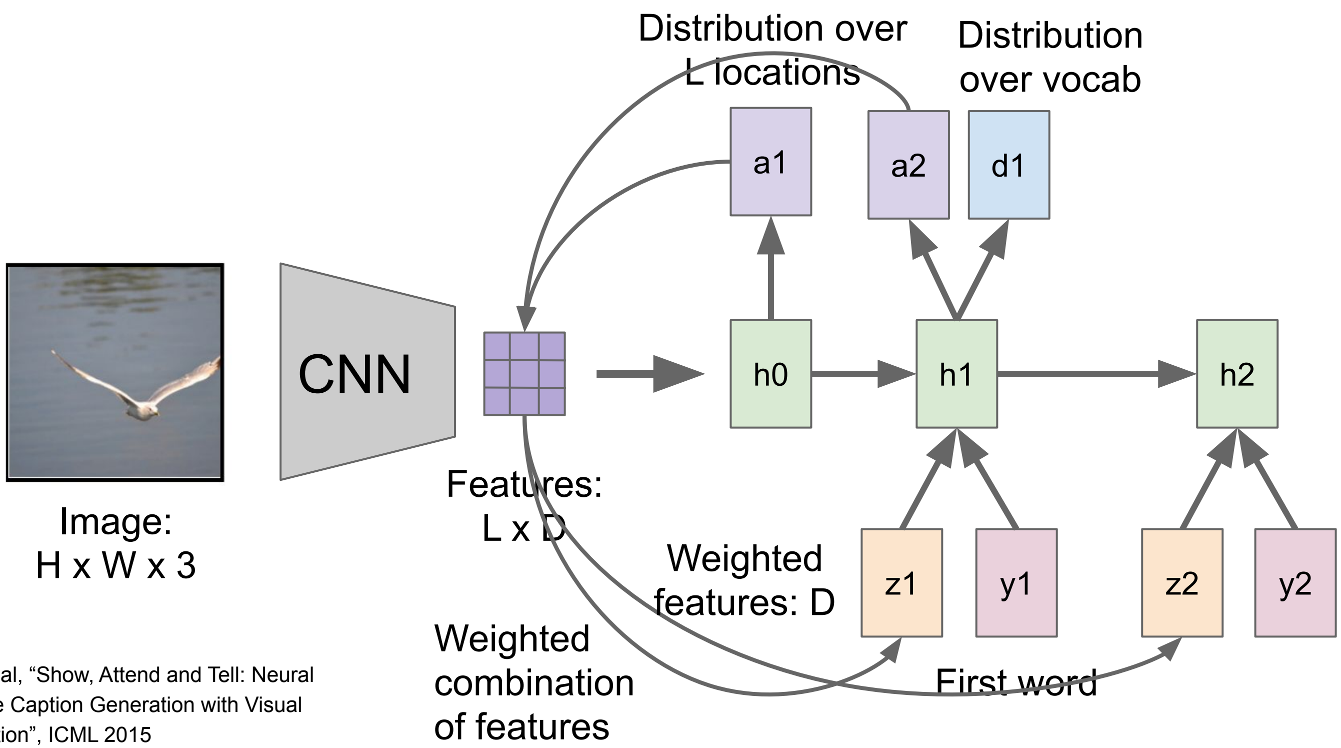
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image Captioning with Attention



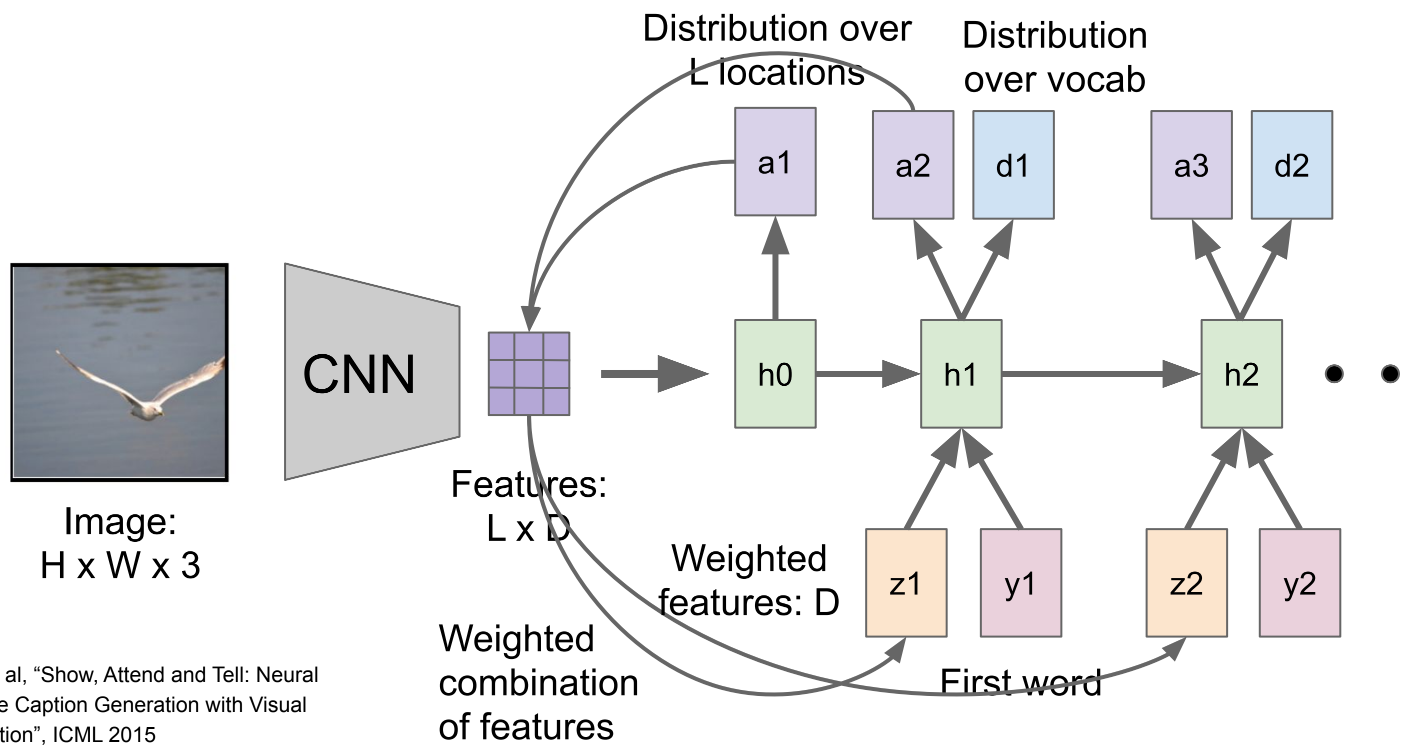
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image Captioning with Attention



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

Image Captioning with Attention



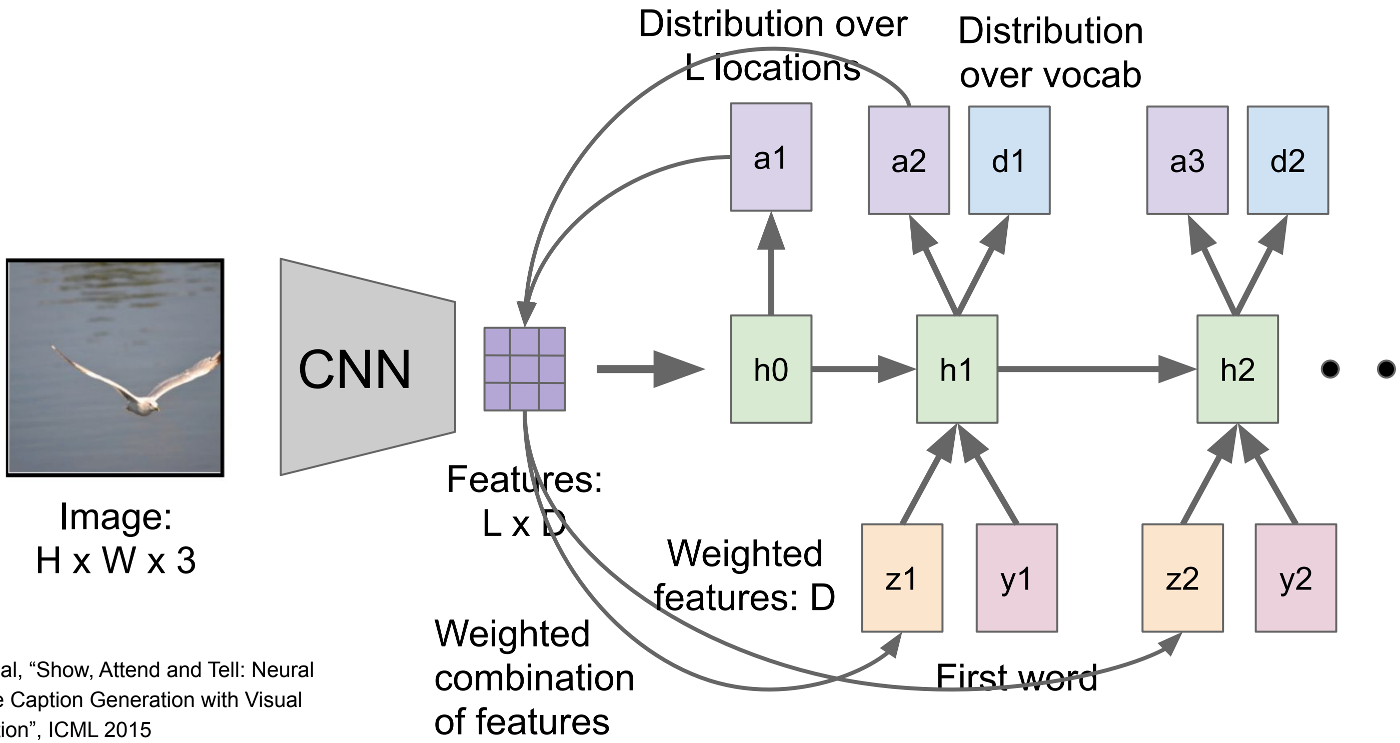
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

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.

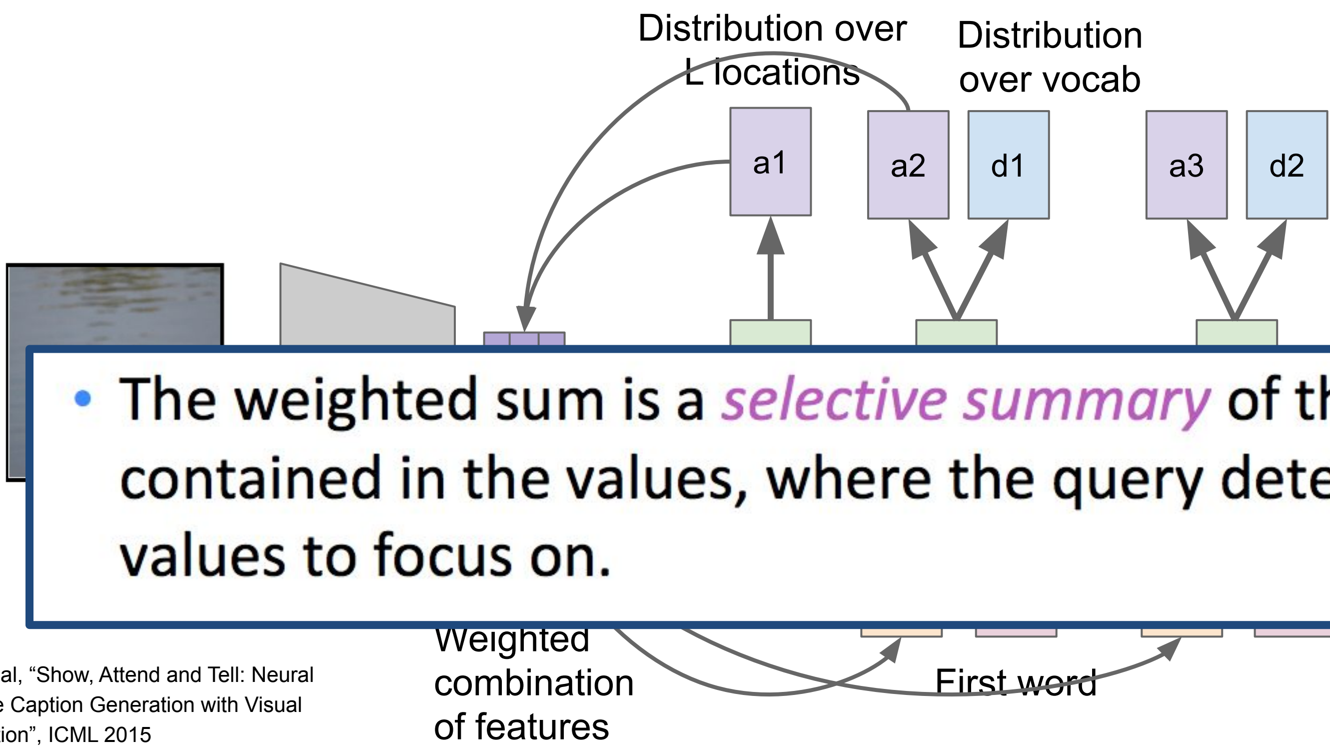
- For example, in the seq2seq + attention model, each decoder hidden state (query) *attends to* all the encoder hidden states (values).

Image Captioning with Attention



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

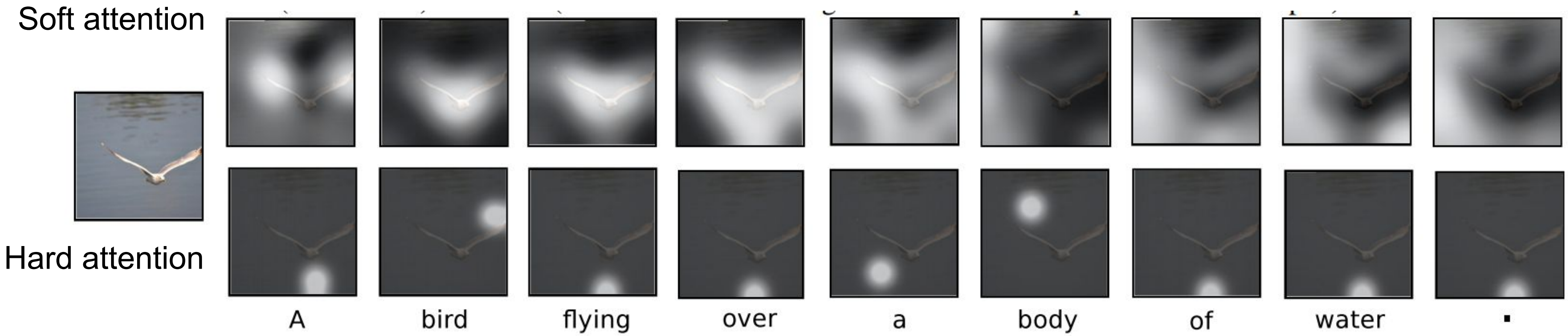
Image Captioning with Attention



- The weighted sum is a *selective summary* of the information contained in the values, where the query determines which values to focus on.

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

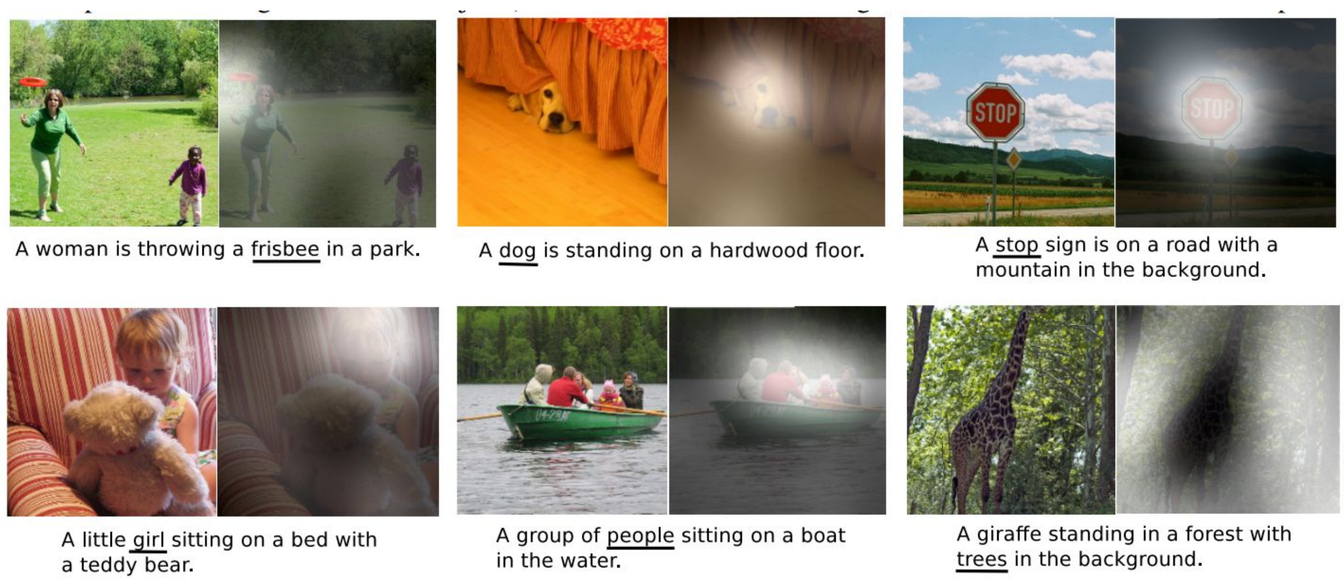
Image Captioning with Attention



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

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Benchio, 2015. Reproduced with permission.

Image Captioning with Attention



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

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Benchio, 2015. Reproduced with permission.

Plan for Today

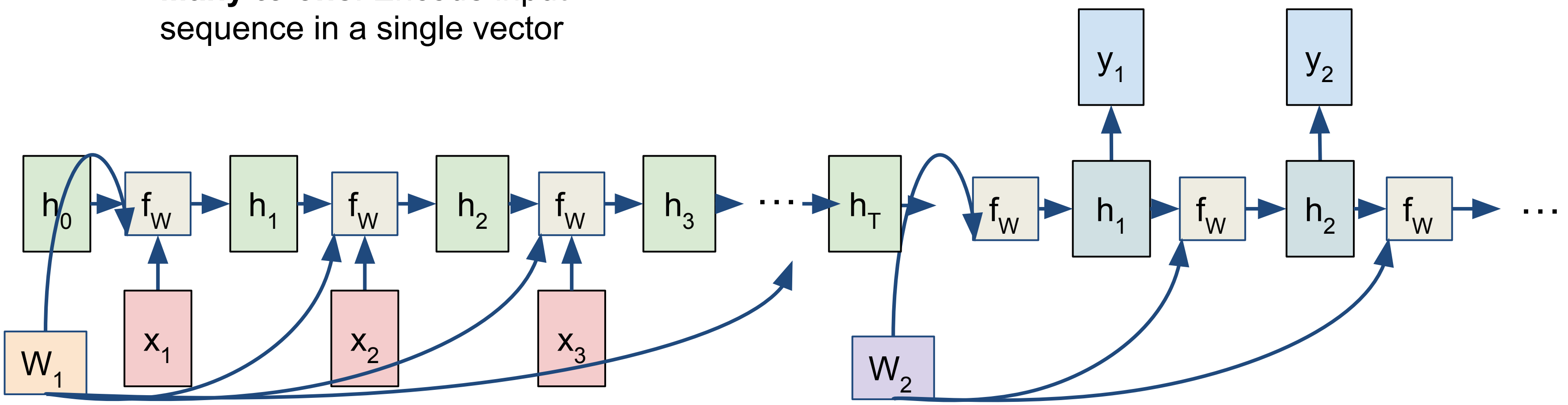
- Recap
  - RNNs, Image Captioning
- RNNs with Attention
- Machine Translation
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- More natural language applications
  - Language Modeling (ELMo, GPT)
  - BERT ("Masked Language Modeling")

Slide Credit: Sarah Wiegreffe

Sequence to Sequence: Many-to-one + one-to-many

One to many: Produce output sequence from single input vector

Many to one: Encode input sequence in a single vector



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Machine Translation

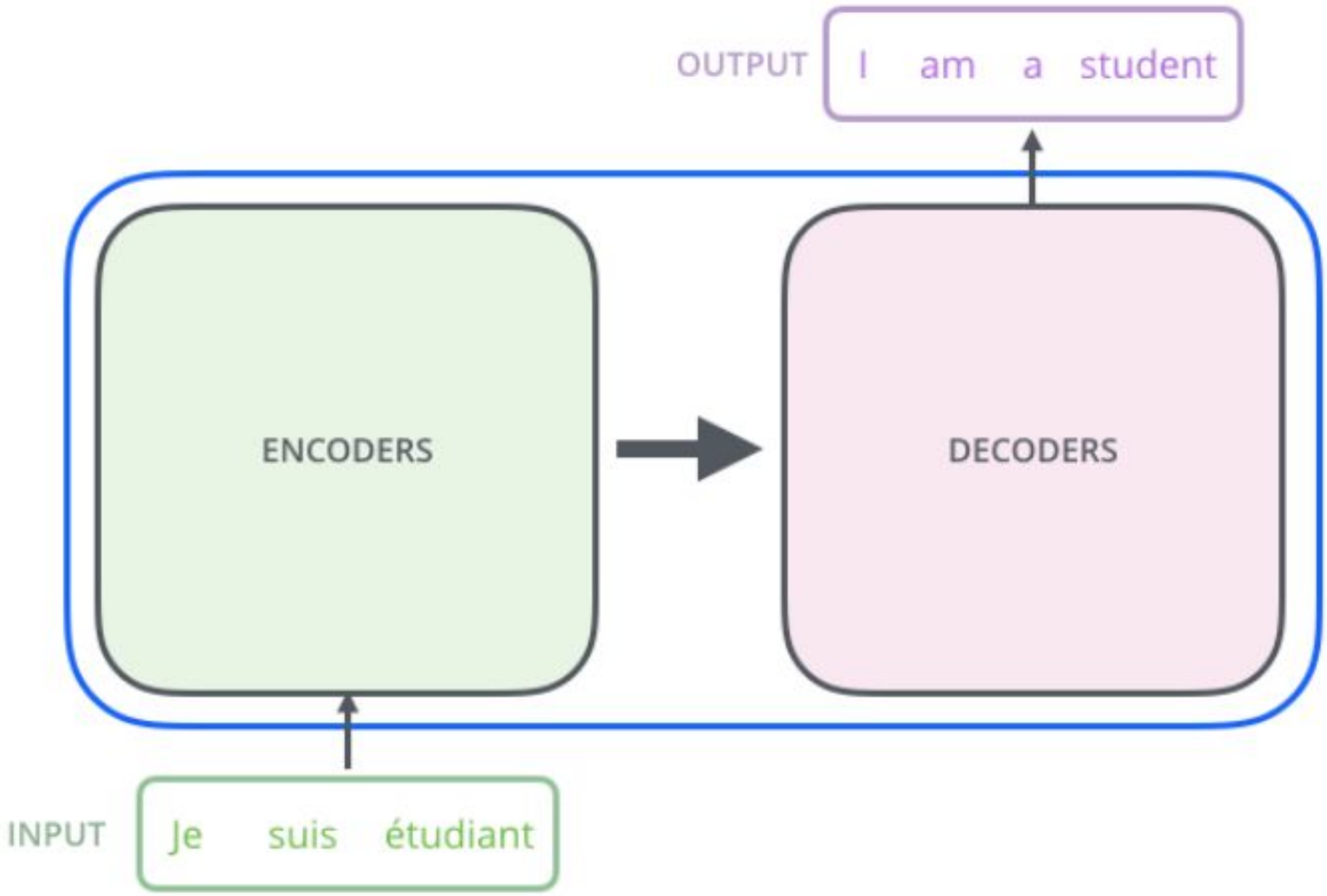
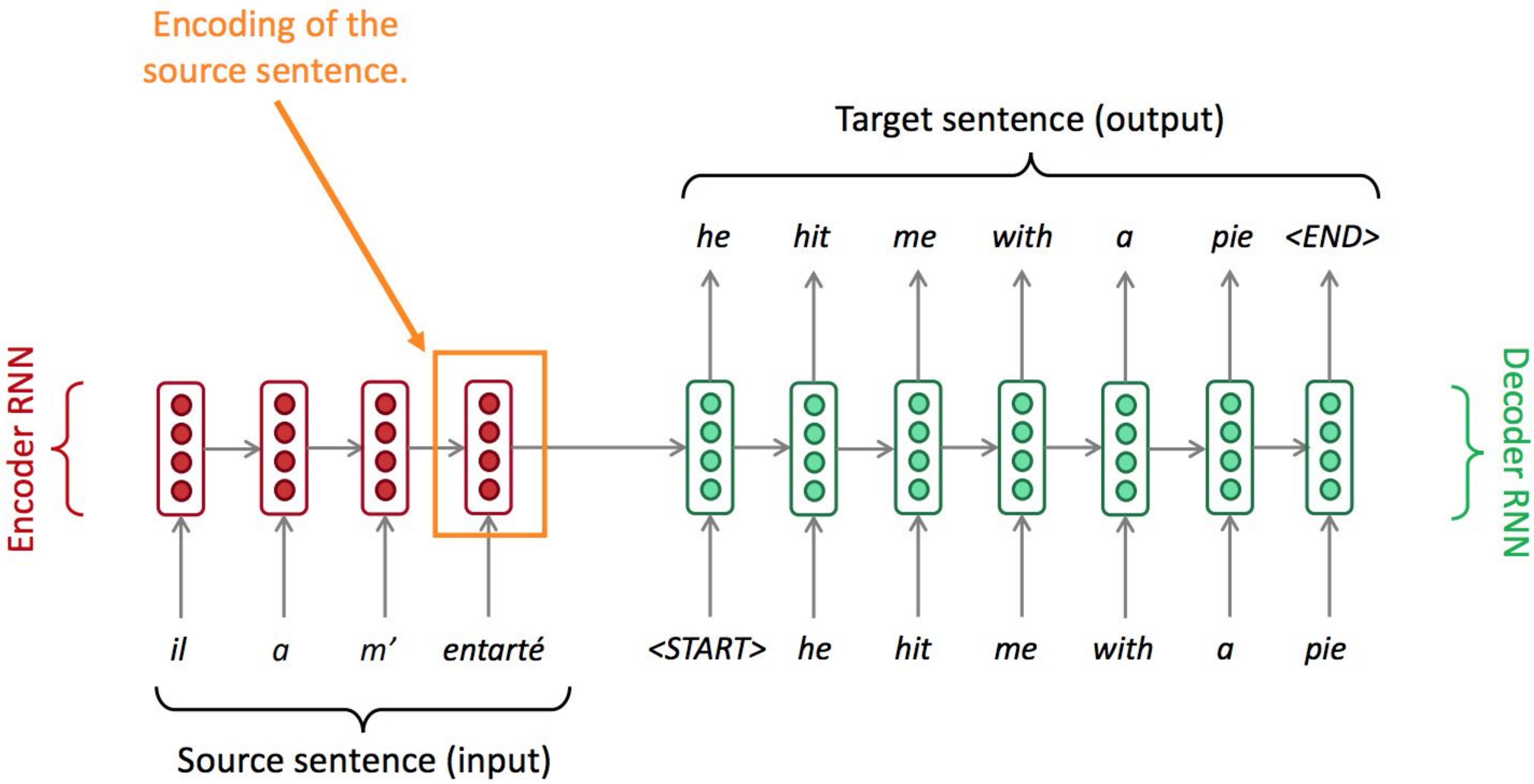


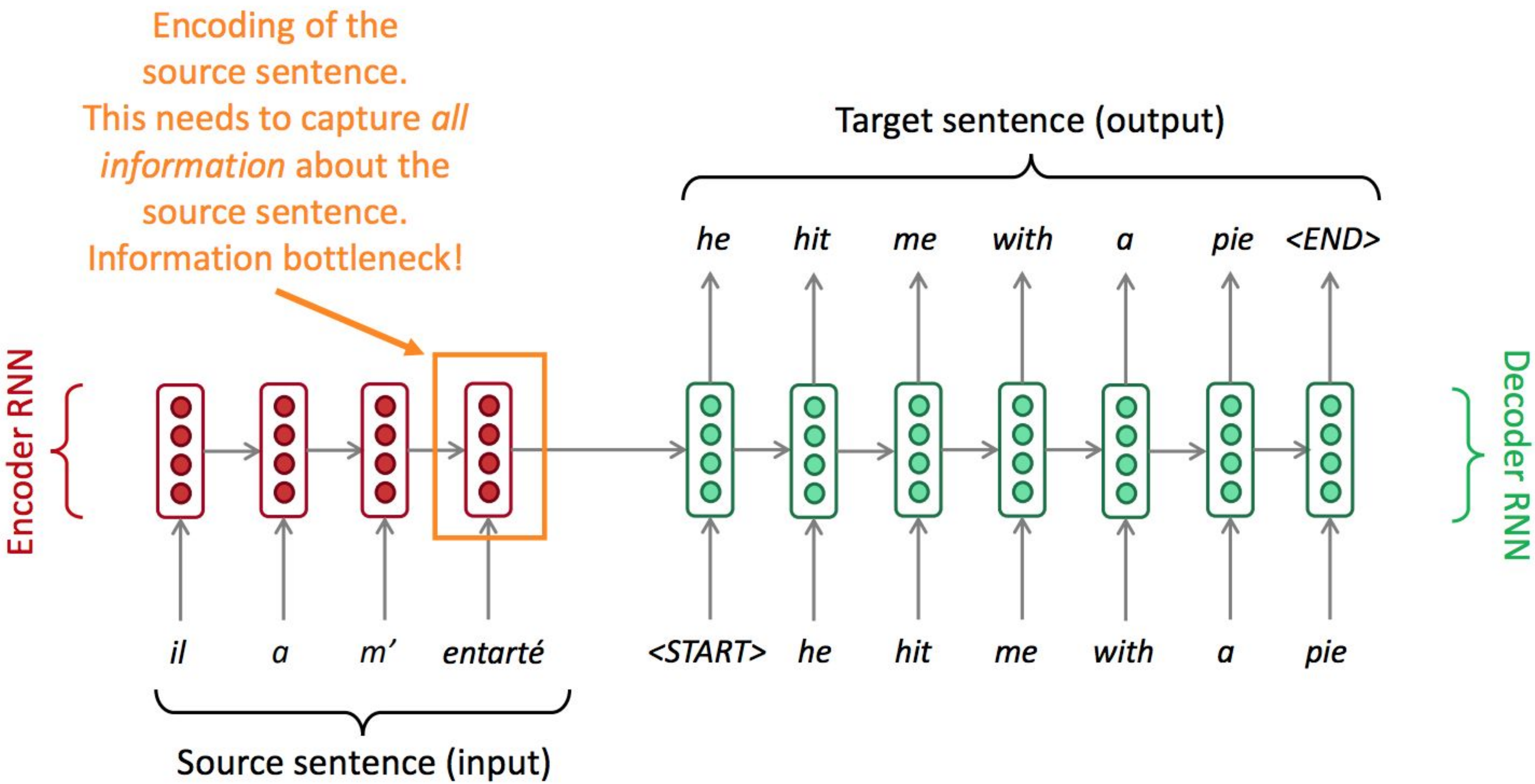
Image Credit: Jay Alammar

Machine Translation



Slide Credit: Abigail See, Stanford CS224n

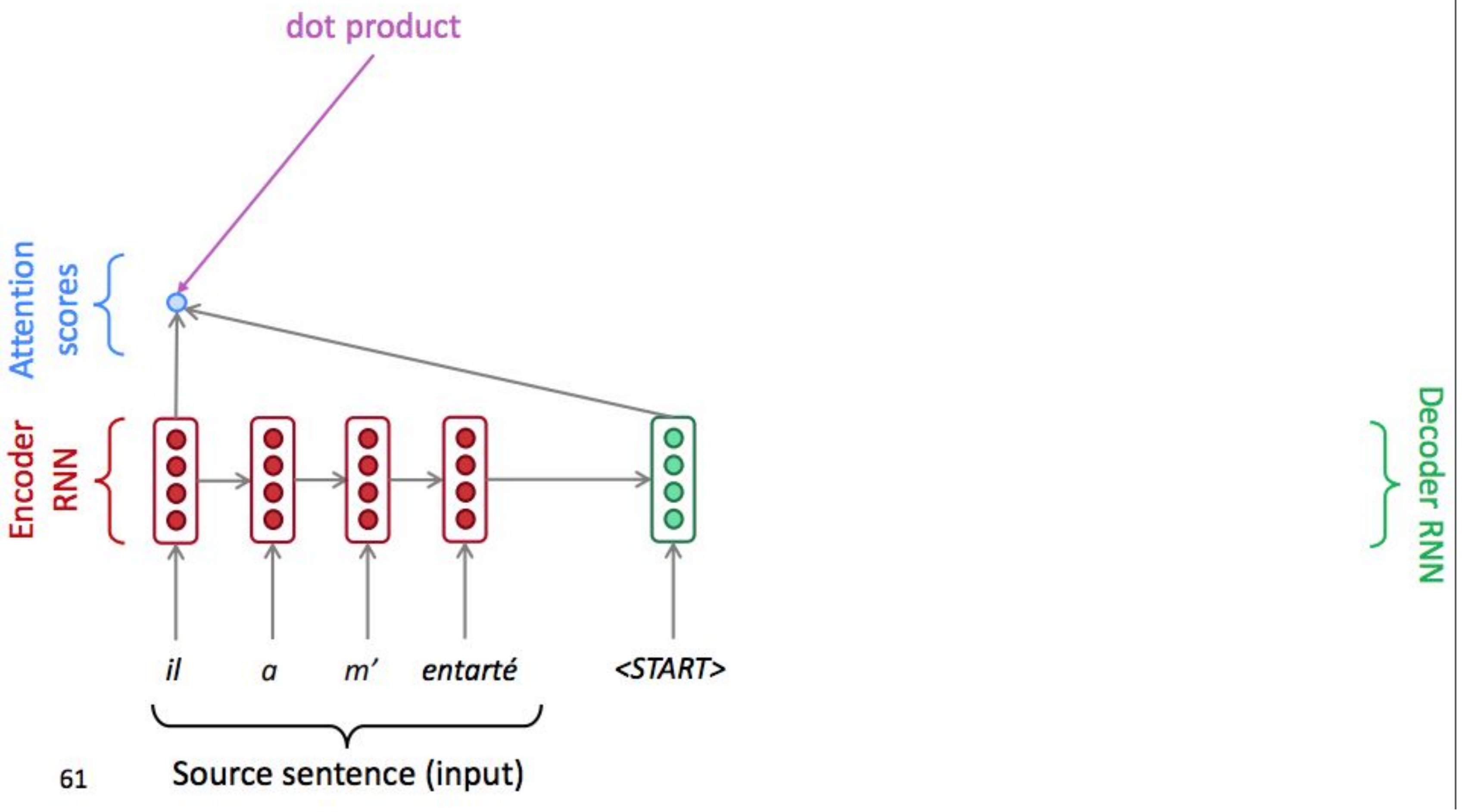
Machine Translation



Slide Credit: Abigail See, Stanford CS224n

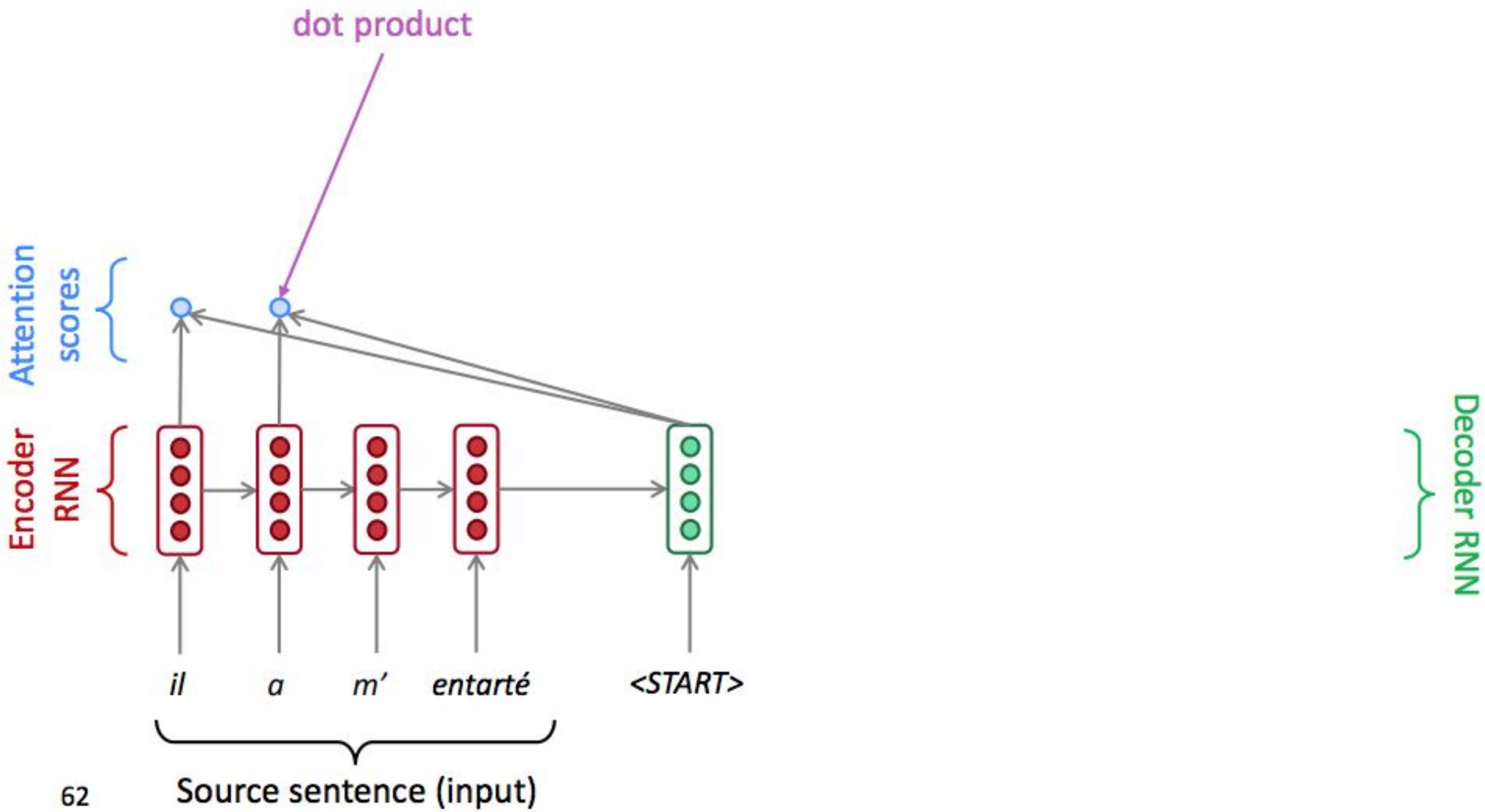
Machine Translation + Attention

Bahdanau et. al, 2014



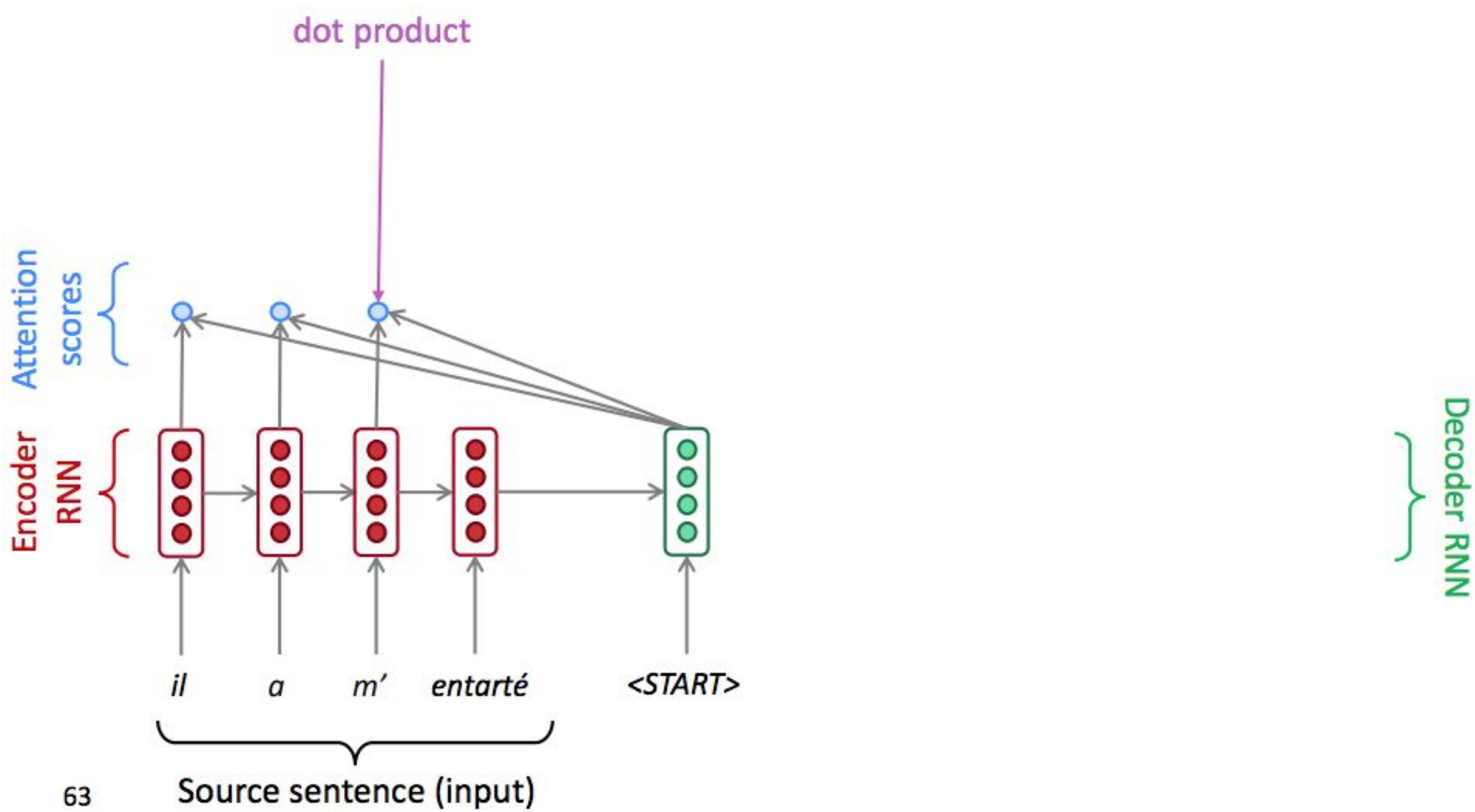
Slide Credit: Abigail See, Stanford CS224n

Machine Translation + Attention



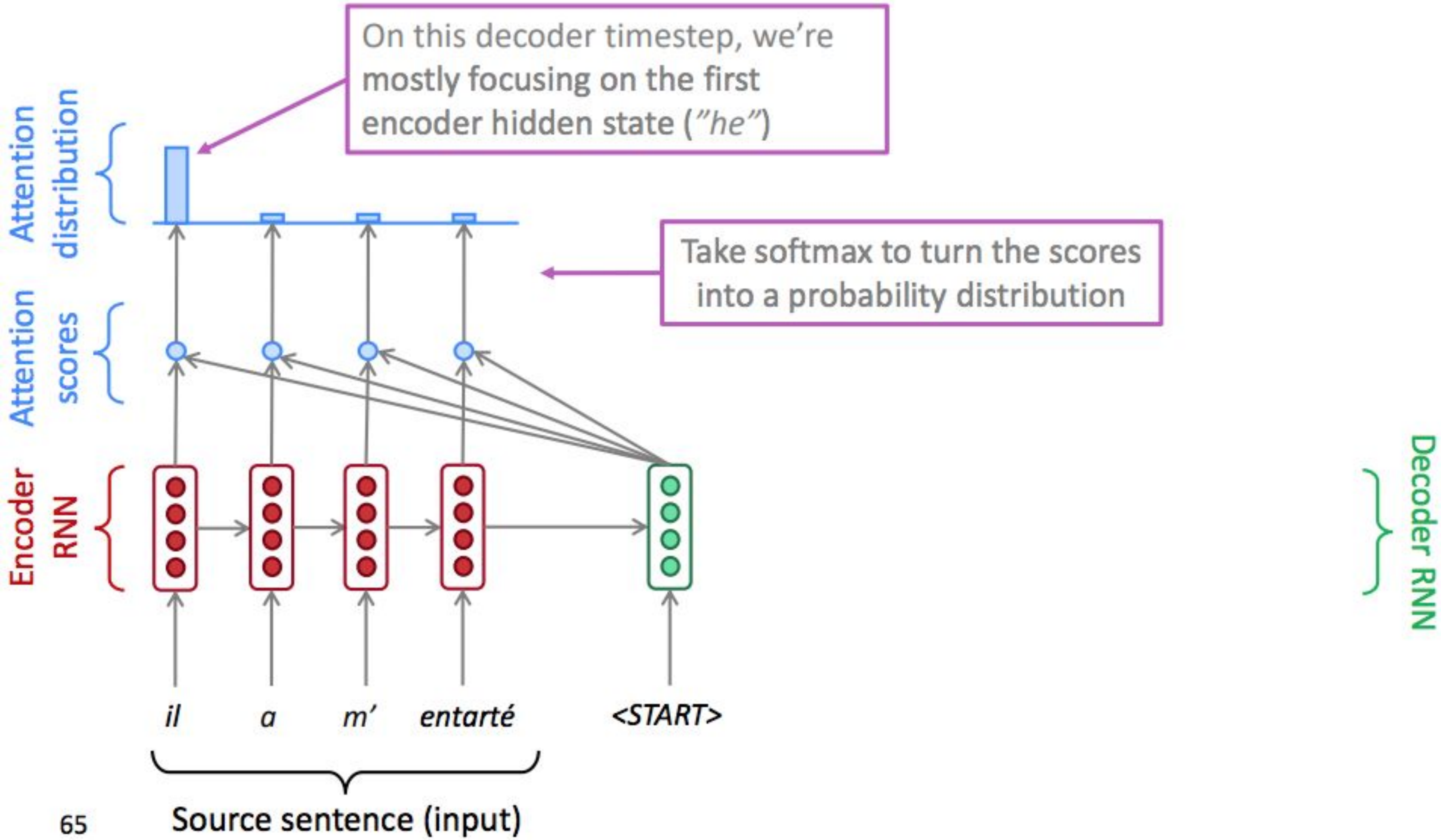
Slide Credit: Abigail See, Stanford CS224n

Machine Translation + Attention



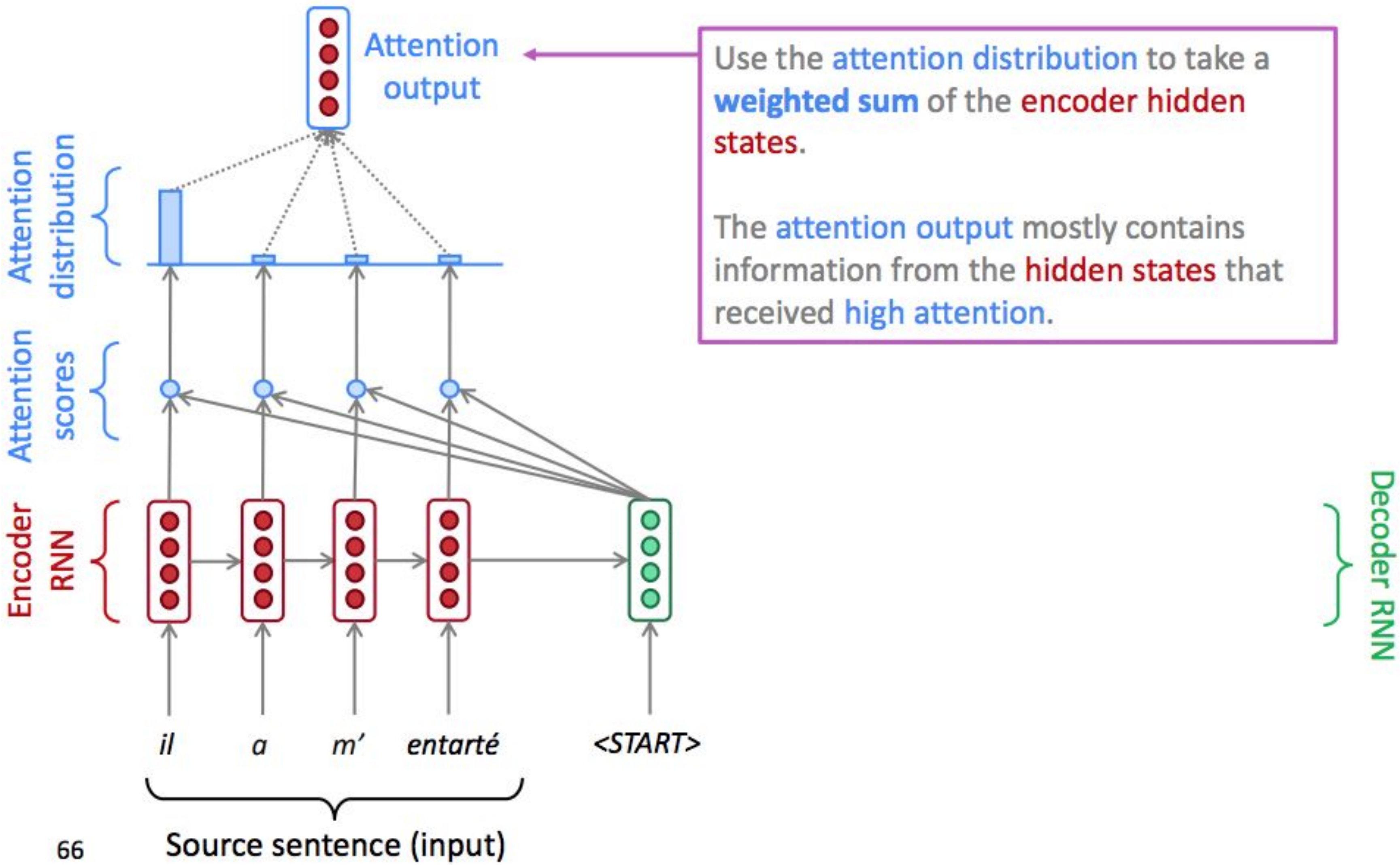
Slide Credit: Abigail See, Stanford CS224n

Machine Translation + Attention



Slide Credit: Abigail See, Stanford CS224n

Machine Translation + Attention

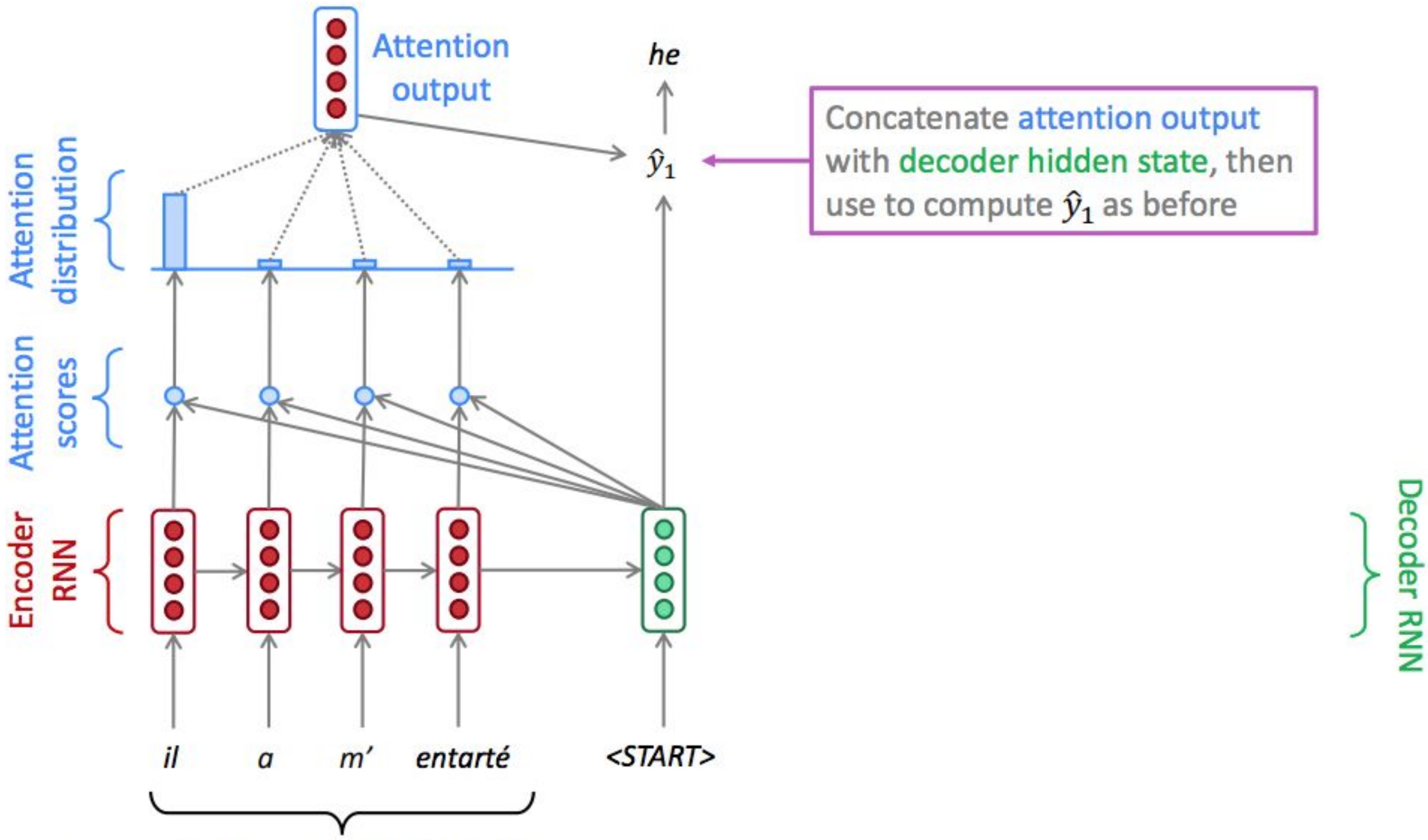


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

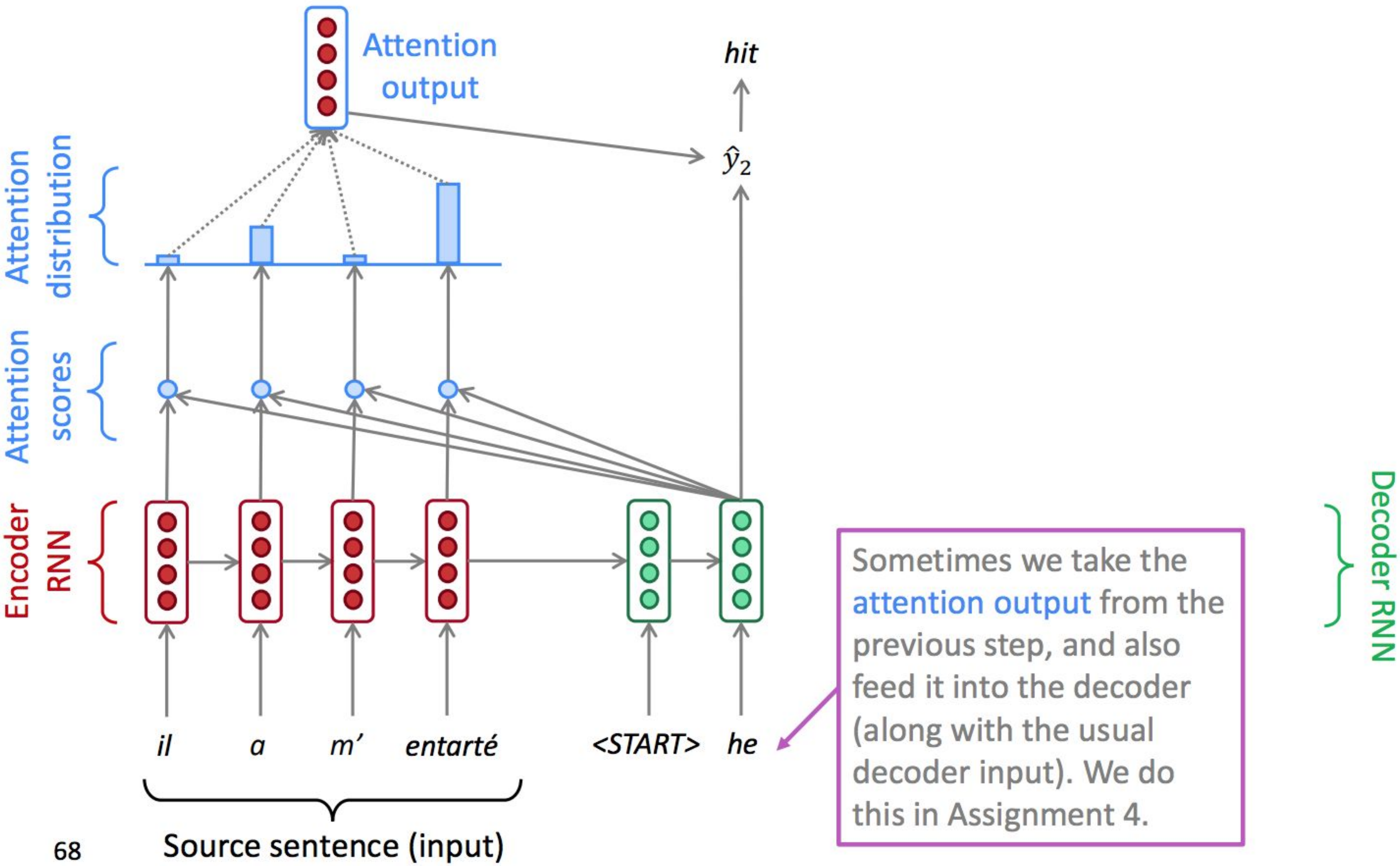
Slide Credit: Abigail See, Stanford CS224n

Machine Translation + Attention



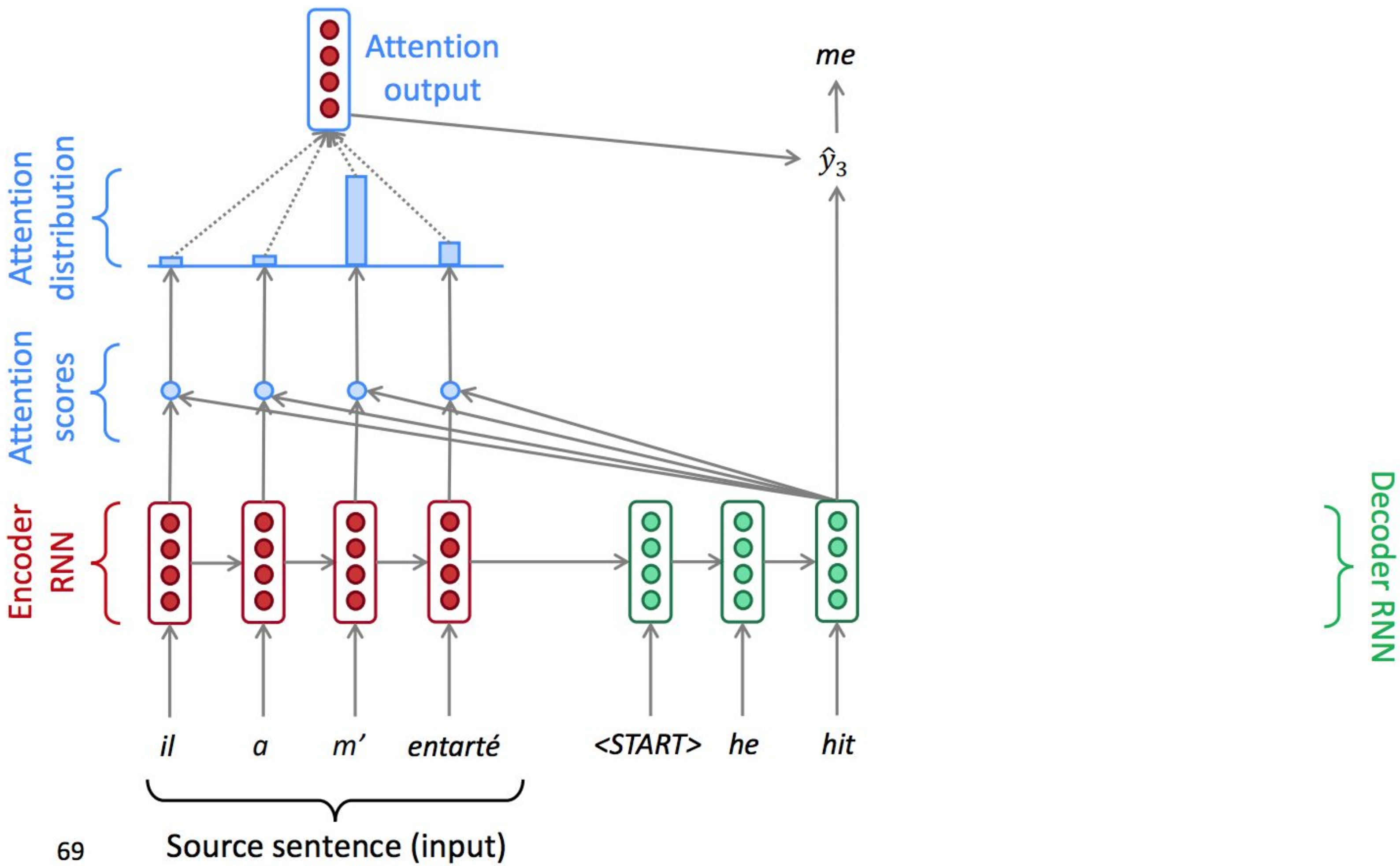
Slide Credit: Abigail See, Stanford CS224n

Machine Translation + Attention



Machine Translation + Attention

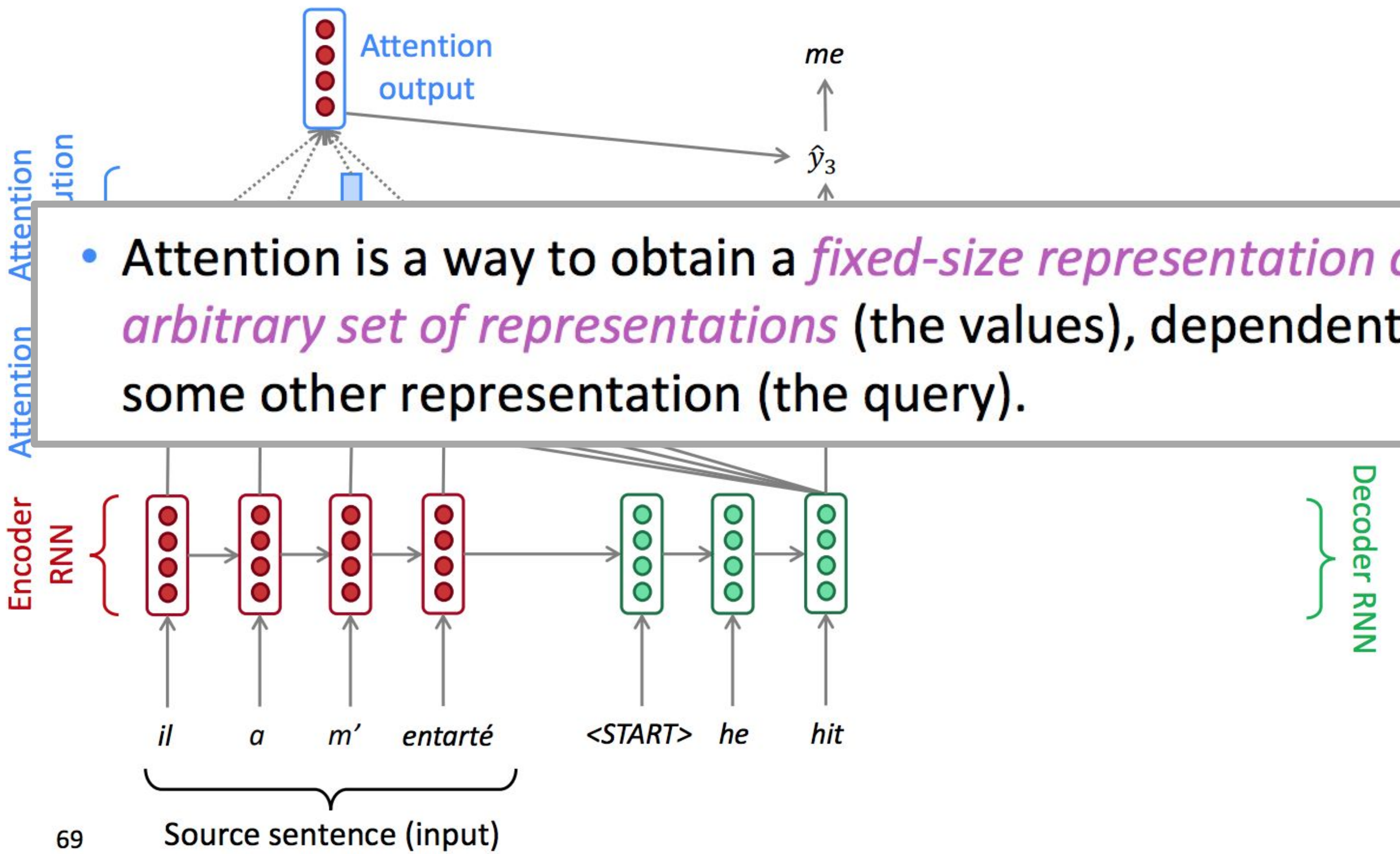
Sequence-to-sequence with attention



Slide Credit: Abigail See, Stanford CS224n

Machine Translation + Attention

Sequence-to-sequence with attention



- 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).

Slide Credit: Abigail See, Stanford CS224n

Attention: Formally

- For query vector $\mathbf{q}$, key vector $\mathbf{k}_i$ representing value $\mathbf{v}_i$
  - s_i is the similarity score between $\mathbf{q}$ and $\mathbf{k}_i$
- Normalize the similarity scores to sum to 1
  - p_i = Softmax(s_i)
- Compute $\mathbf{z}$ as the weighted sum of the value vectors $\mathbf{v}_i$ weighted by their scores p_i
- In Machine Translation & Image Captioning, the keys and values are the same.
  - But, they could be different.
-

$$z = \sum_{i=1}^{L} p_i v_i$$

Slide Credit: Sarah Wiegreffe

Attention: Multiple Types

- For query vector $\mathbf{q}$, key vector $\mathbf{k}_i$ representing value $\mathbf{v}_i$
  - s_i is the similarity score between $\mathbf{q}$ and $\mathbf{k}_i$
- Additive Attention (Bahdanau et. al 2014)
  - $s_i = w_3 \tanh(\mathbf{w_2}^T \mathbf{q} + \mathbf{w_1}^T \mathbf{k}_i)$
- Dot-Product Attention
  - $s_i = \mathbf{q}^T \mathbf{k}_i$

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

$$\mathbf{s}^T = \mathbf{q}^T \mathbf{K}$$

Slide Credit: Sarah Wiegreffe

Attention is great

- Attention significantly improves performance (in many applications)
  - 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

Plan for Today

- Recap
  - RNNs, Image Captioning
- RNNs with Attention
- Machine Translation
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- More natural language applications
  - Language Modeling (ELMo, GPT)
  - BERT ("Masked Language Modeling")

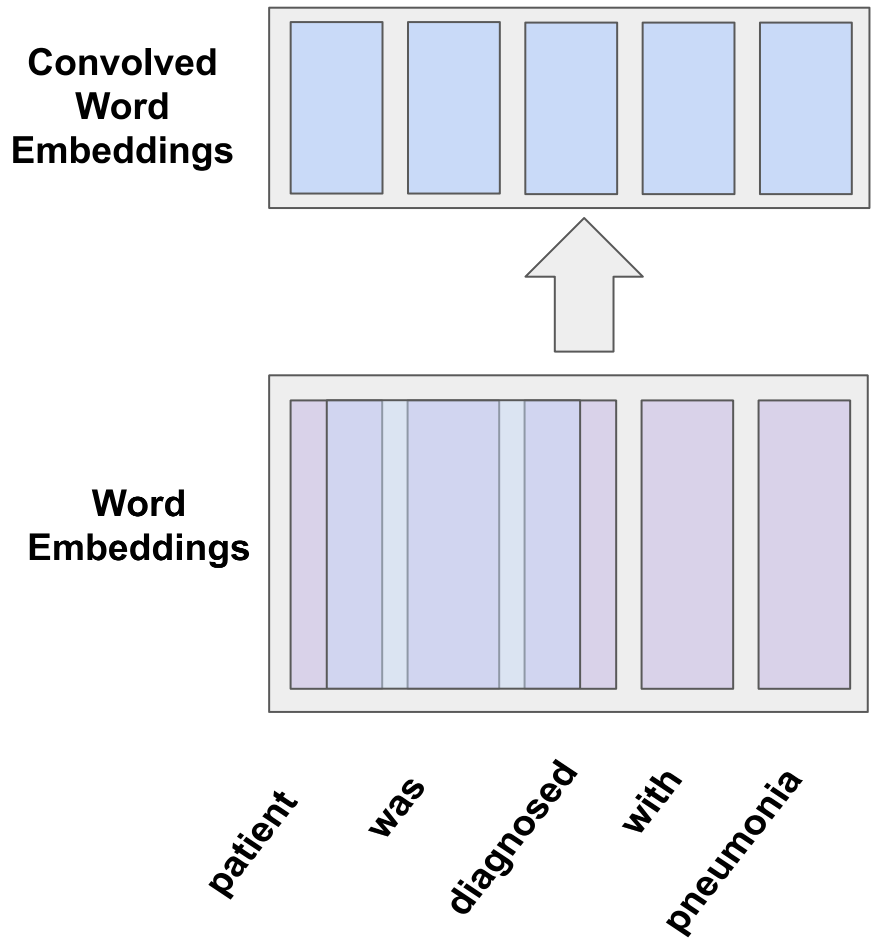
Problems with RNNs

- RNNs involve sequential computation
  - can’t parallelize = time-consuming
- RNNs “forget” past information
- No explicit modeling of long and short range dependencies

Slide Credit: Sarah Wiegreffe

Convolution?

- Trivial to parallelize (per layer)
- Exploits local dependencies; models local context
- **Long-distance dependencies require many layers**



Attention?

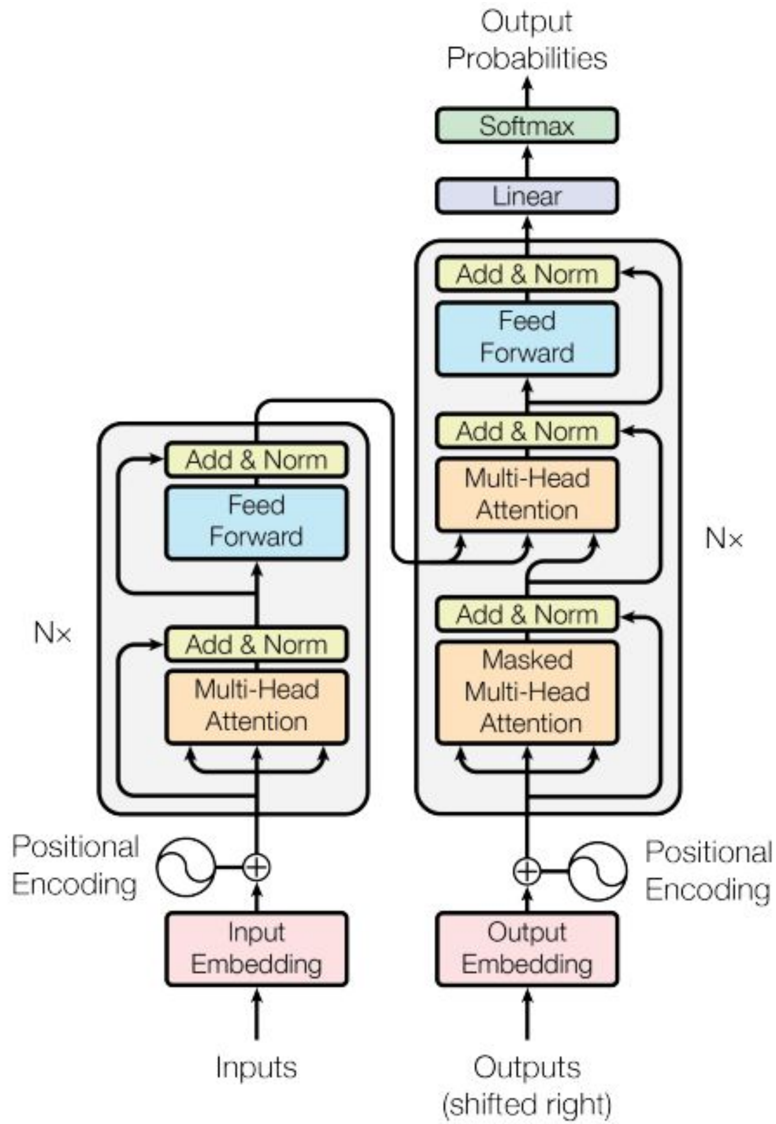
Attention between encoder and decoder is crucial in NMT.

Why not use attention for representations?

"Attention is All You Need"

(Vaswani et. al 2017)

Encoder →



← Decoder

Figure Credit: Vaswani et. al

“Attention is All You Need”

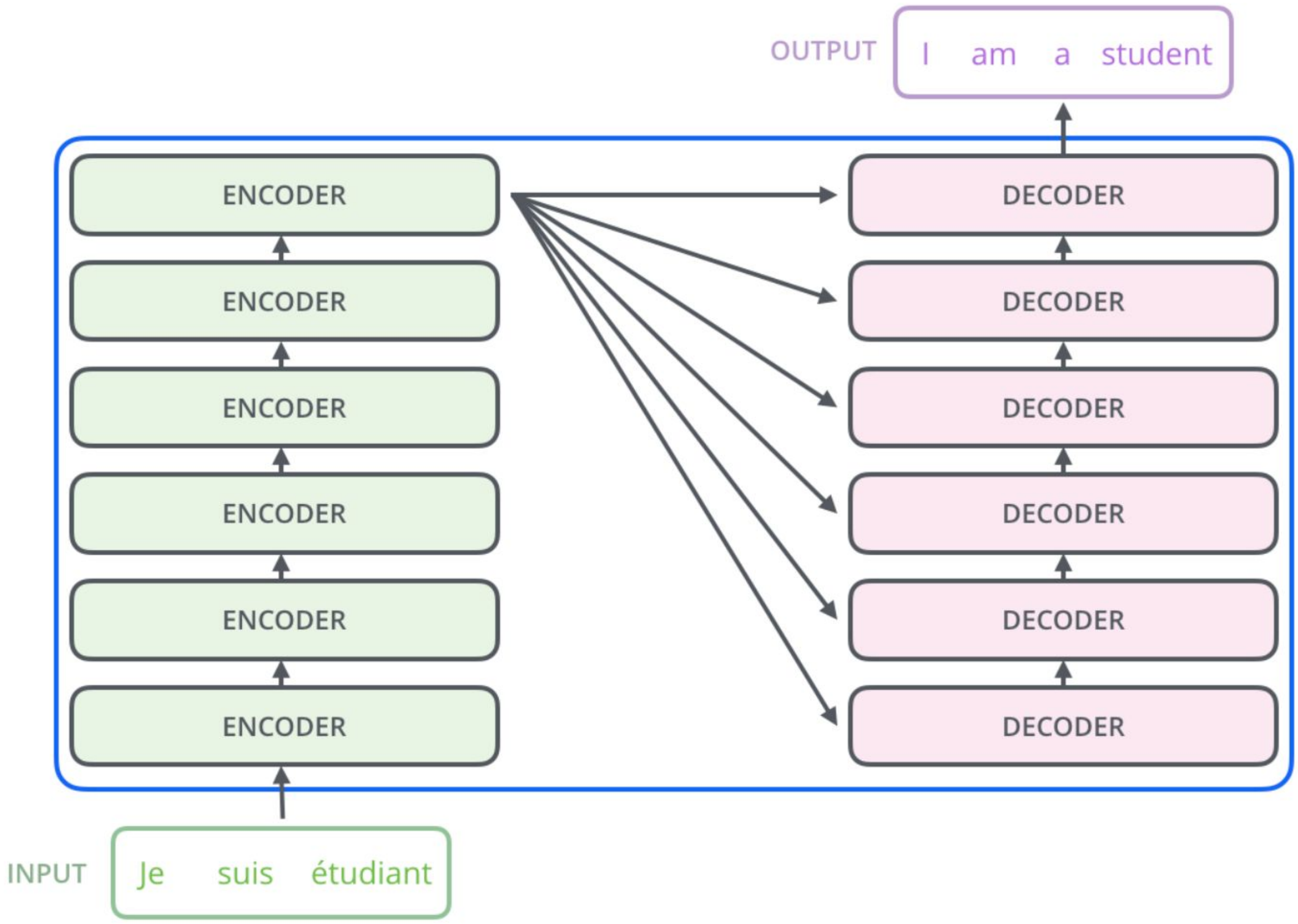


Image Credit: Jay Alammar

Components

- *Scaled Dot-Product* Attention
- Self-Attention
- *Multi-Head* Self-Attention
- Positional Encodings
- Residual Connections

A Single Block

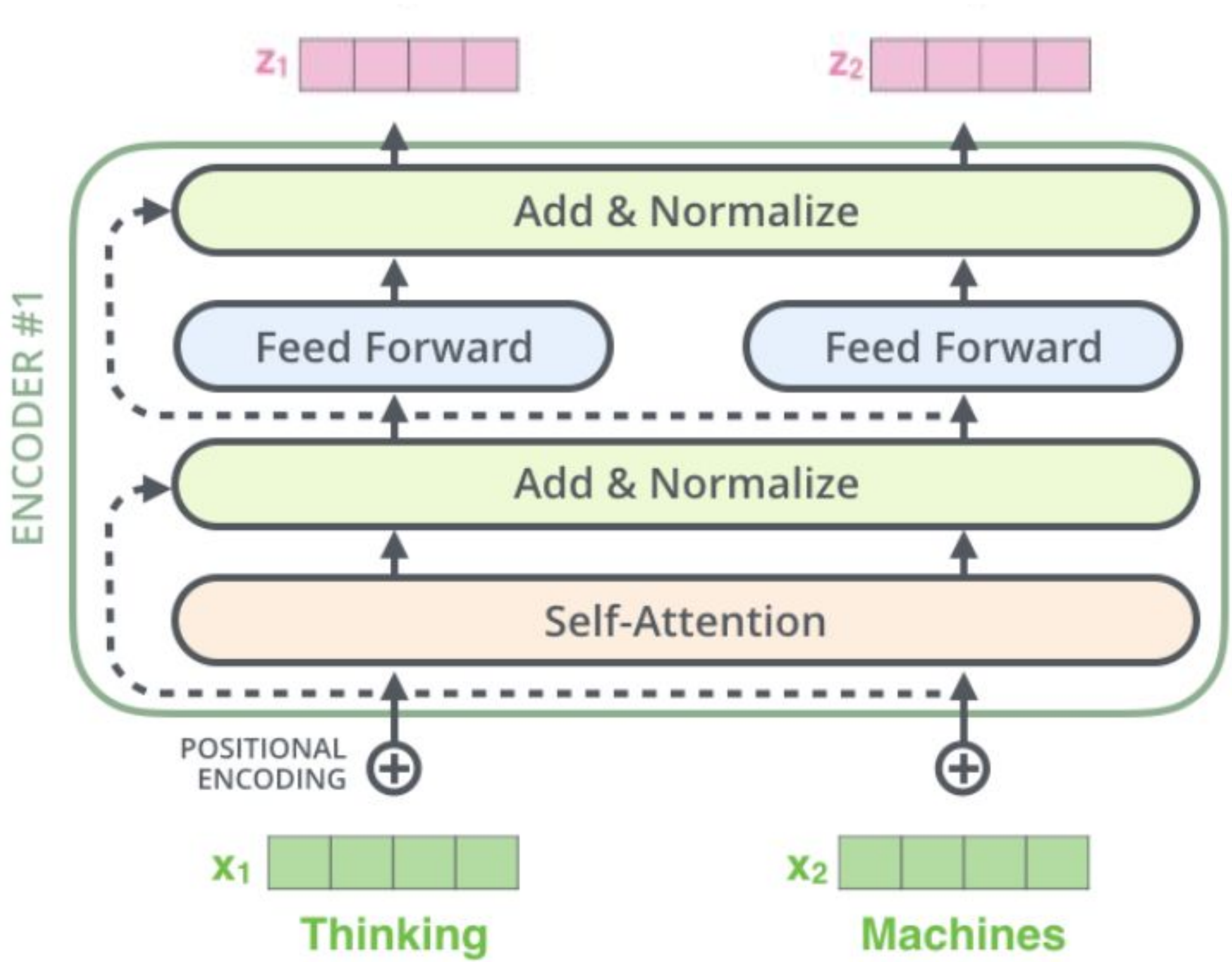


Image Credit: Jay Alammar

Components

- *Scaled Dot-Product* Attention
- Self-Attention
- *Multi-Head* Self-Attention
- Positional Encodings
- Residual Connections

Attention: Multiple Types

- For query vector $\mathbf{q}$, key vector $\mathbf{k}_i$ representing value $\mathbf{v}_i$
  - s_i is the similarity score between $\mathbf{q}$ and $\mathbf{k}_i$
- Additive Attention (Bahdanau et. al 2014)
  - $s_i = w_3 \tanh(\mathbf{w_2}^T\mathbf{q} + \mathbf{w_1}^T\mathbf{k}_i)$
- Dot-Product Attention
  - $s_i = \mathbf{q}^T\mathbf{k}_i$

Slide Credit: Sarah Wiegreffe

Attention: Multiple Types

- For query vector $\mathbf{q}$, key vector $\mathbf{k}_i$ representing value $\mathbf{v}_i$
  - s_i is the similarity score between $\mathbf{q}$ and $\mathbf{k}_i$
- Additive Attention (Bahdanau et. al 2014)
  - $s_i = w_3 \tanh(\mathbf{w_2}^T\mathbf{q} + \mathbf{w_1}^T\mathbf{k}_i)$
- Dot-Product Attention
  - $s_i = \mathbf{q}^T\mathbf{k}_i$
- *Scaled* Dot-Product Attention
  - $s_i = \mathbf{q}^T\mathbf{k}_i / \sqrt{d_k}$

Slide Credit: Sarah Wiegreffe

Components

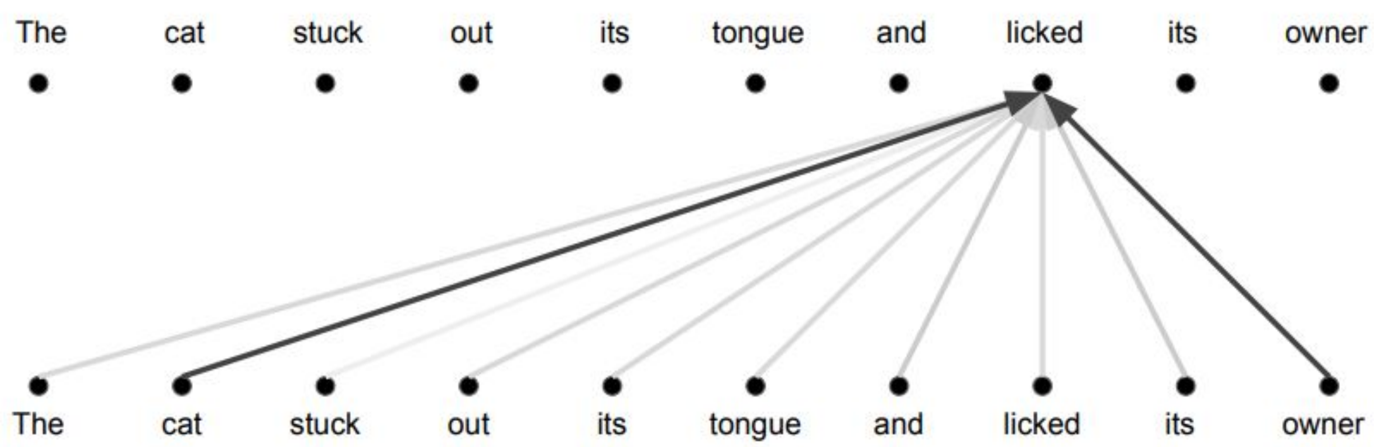
- *Scaled Dot-Product* Attention
- Self-Attention
- *Multi-Head* Self-Attention
- Positional Encodings
- Residual Connections

Slide Credit: Sarah Wiegreffe

Self-Attention

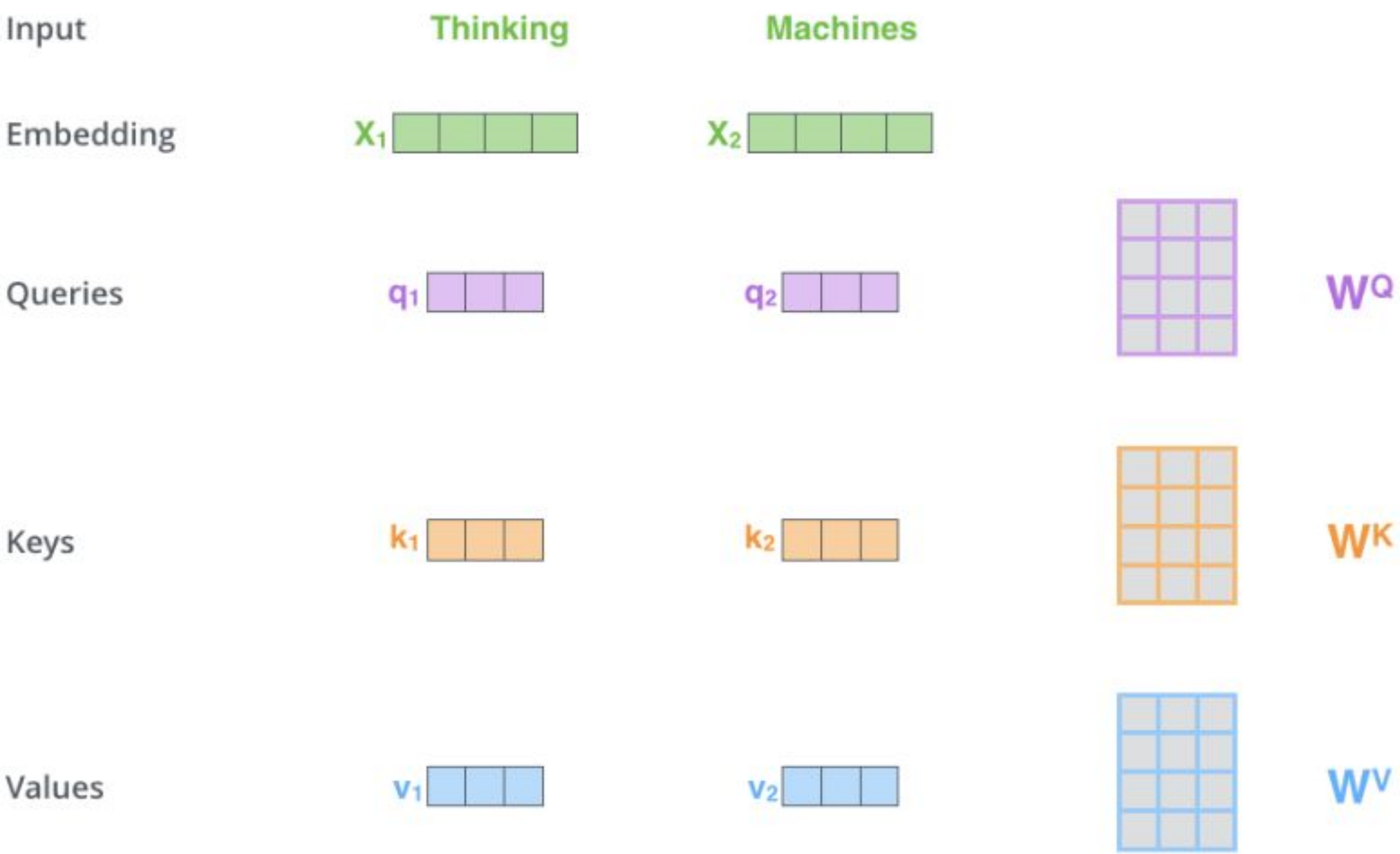
Constant 'path length' between any two positions.

Trivial to parallelize (per layer).

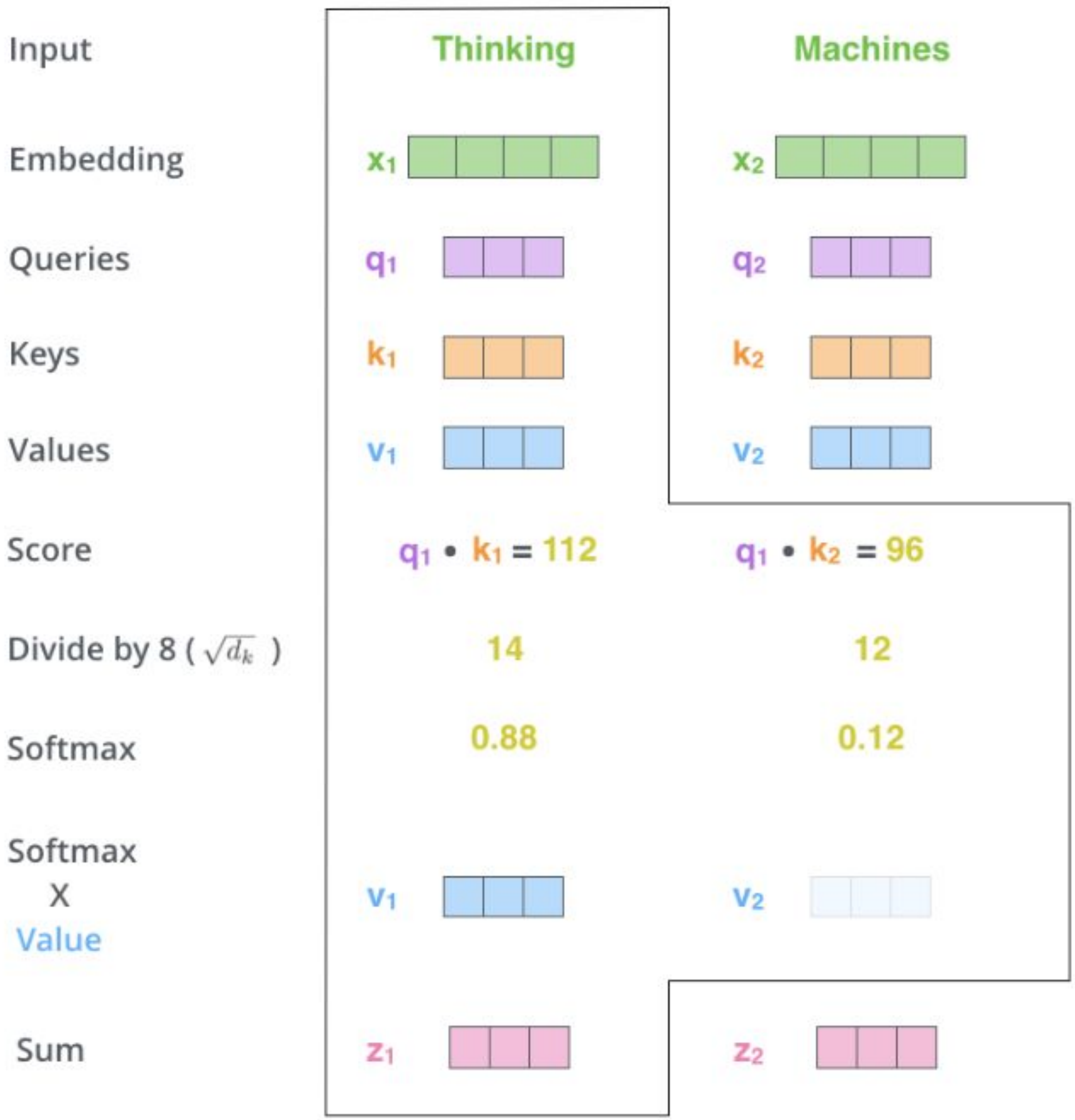


Slide Credit: Ashish Vaswani & Anna Huang, Stanford CS224n

Self (Dot-Product) Attention

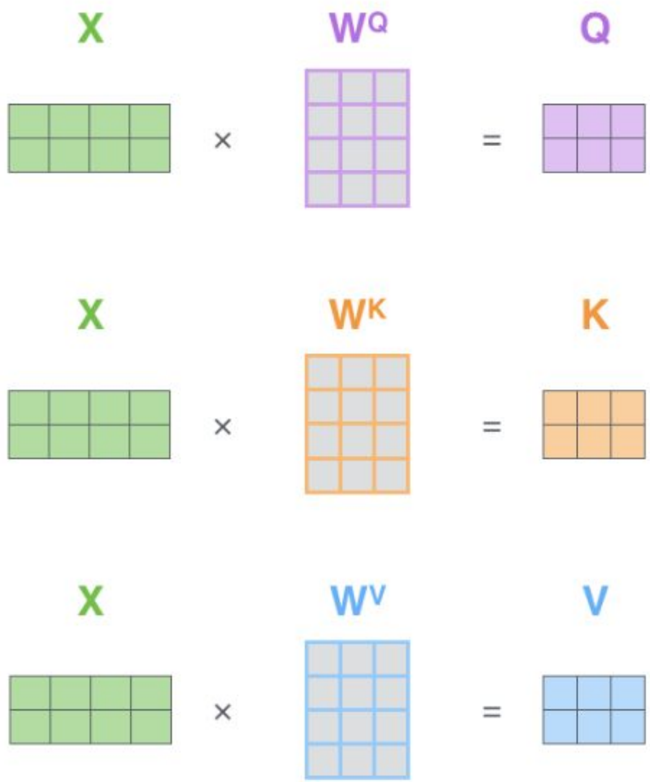


Self (Dot-Product) Attention



Slide Credit: Jay Allamar

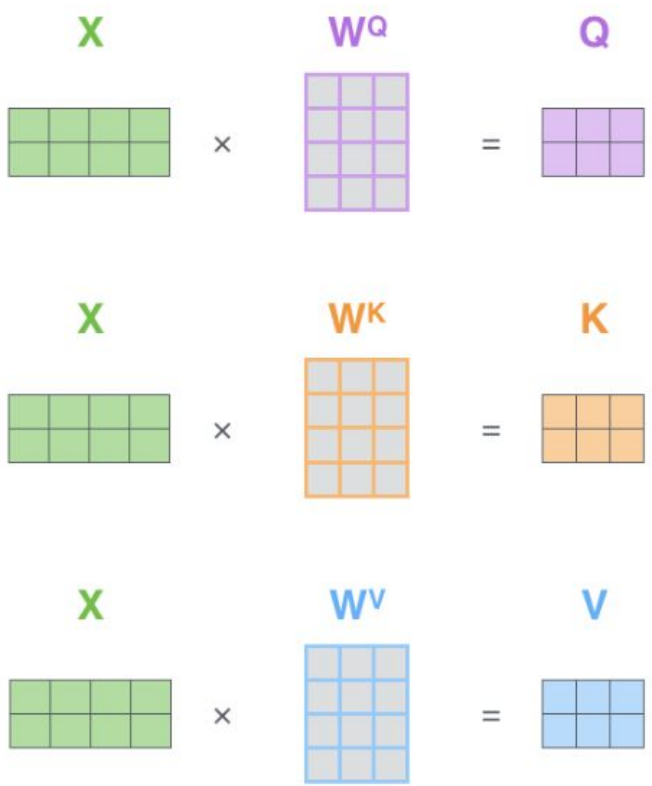
Self (Dot-Product) Attention



$$A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Image Credit: Jay Allamar

Self (Dot-Product) Attention



$$A(Q, K, V) = \boxed{softmax(\frac{QK^T}{\sqrt{d_k}})}V$$

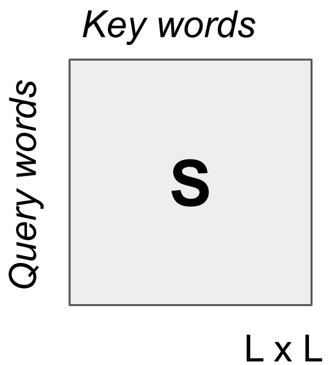
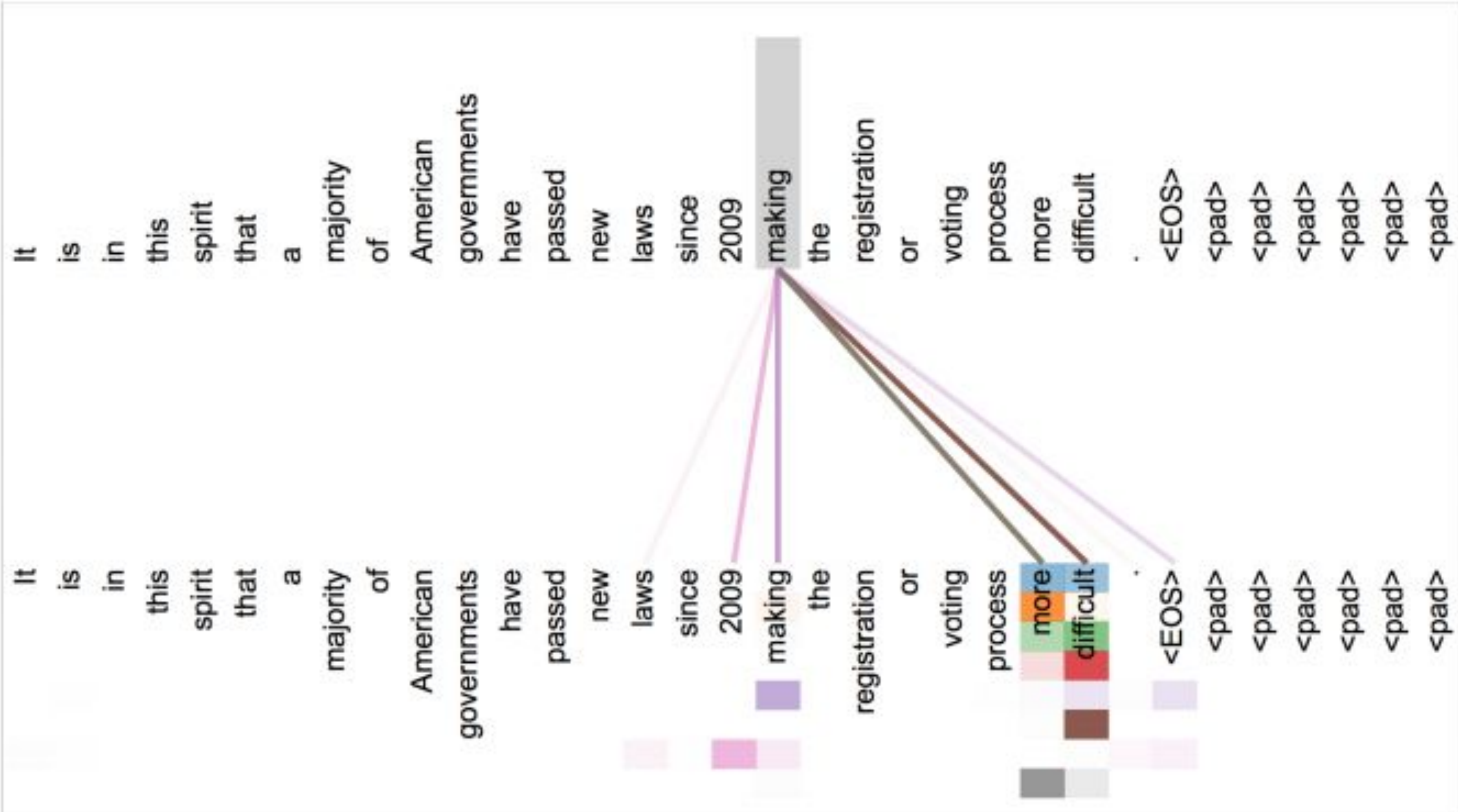


Image Credit: Jay Allamar; Slide: Sarah Wiegreffe

Attention visualization in layer 5

- Words start to pay attention to other words in sensible ways

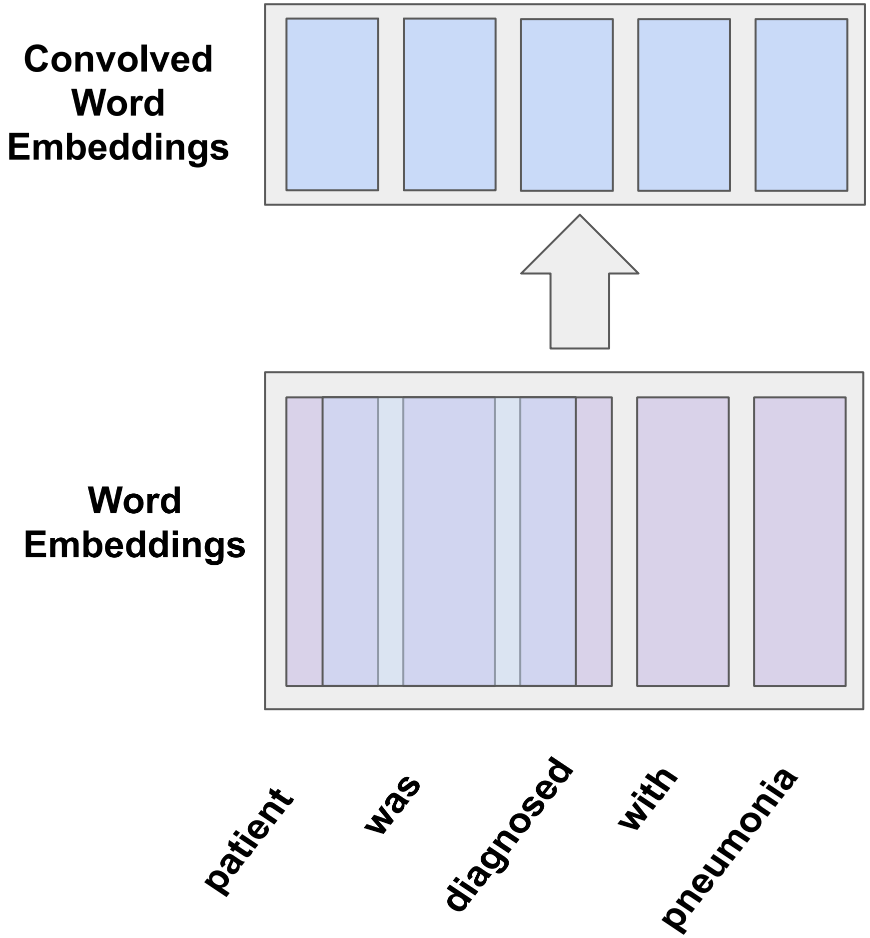


Components

- *Scaled Dot-Product* Attention
- Self-Attention
- *Multi-Head* Self-Attention
- Positional Encodings
- Residual Connections

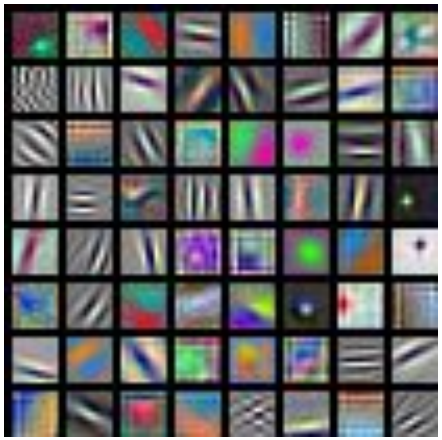
Convolutions on Text

- Often referred to “1D convolution
- Reduces dimensionality of word vectors
- Incorporates local context



Slide Credit: Sarah Wiegreffe

Visualizing filters in first layer

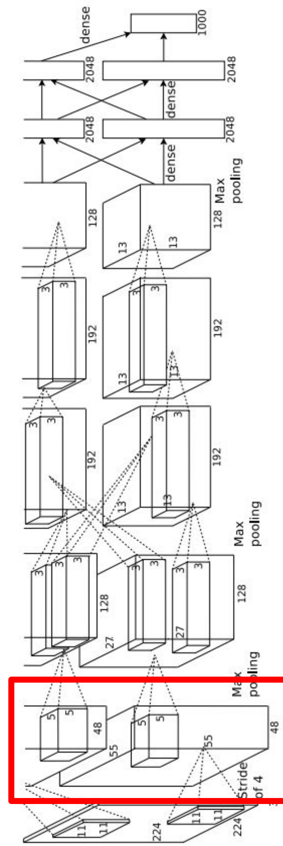


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Multiple “Filters”

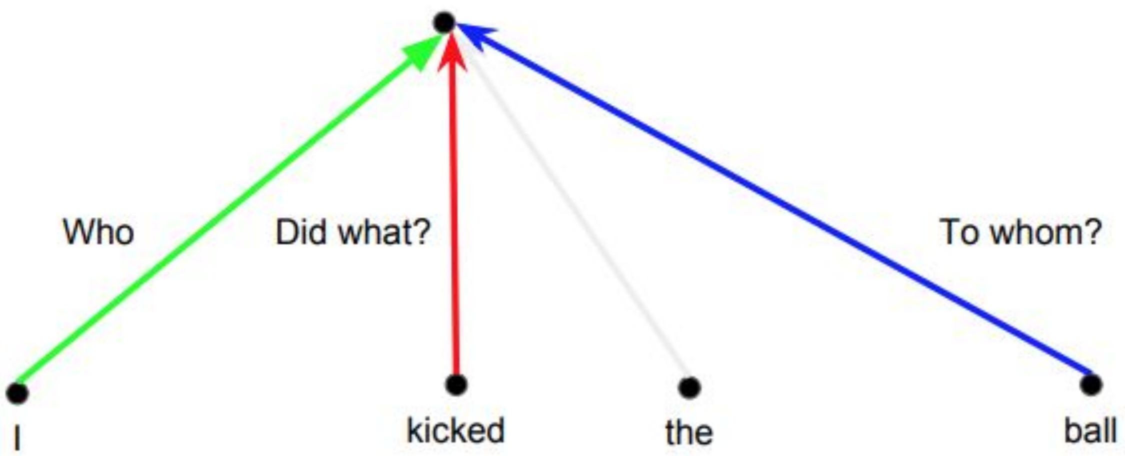
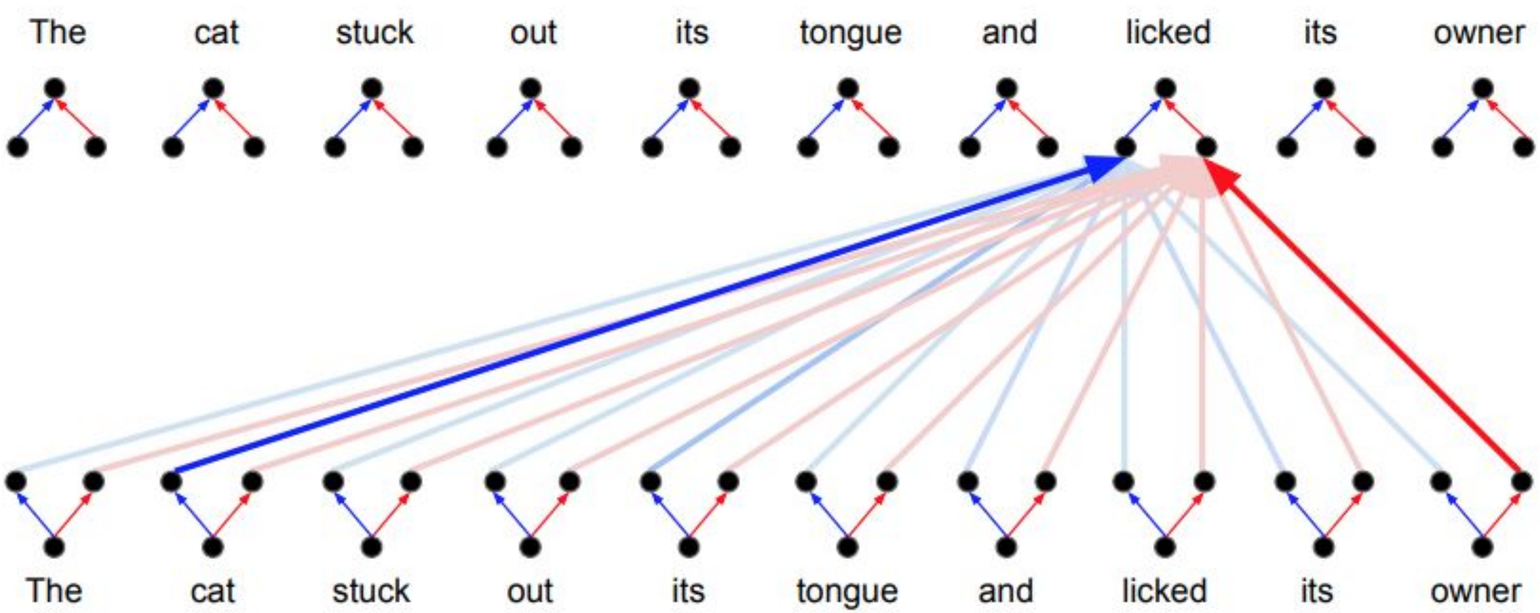


Image Credit: Ashish Vaswani & Anna Huang, Stanford CS224n

Multi-Head Attention

Parallel attention layers with different linear transformations on input and output.



Slide Credit: Ashish Vaswani & Anna Huang, Stanford CS224n

Multi-Head Attention

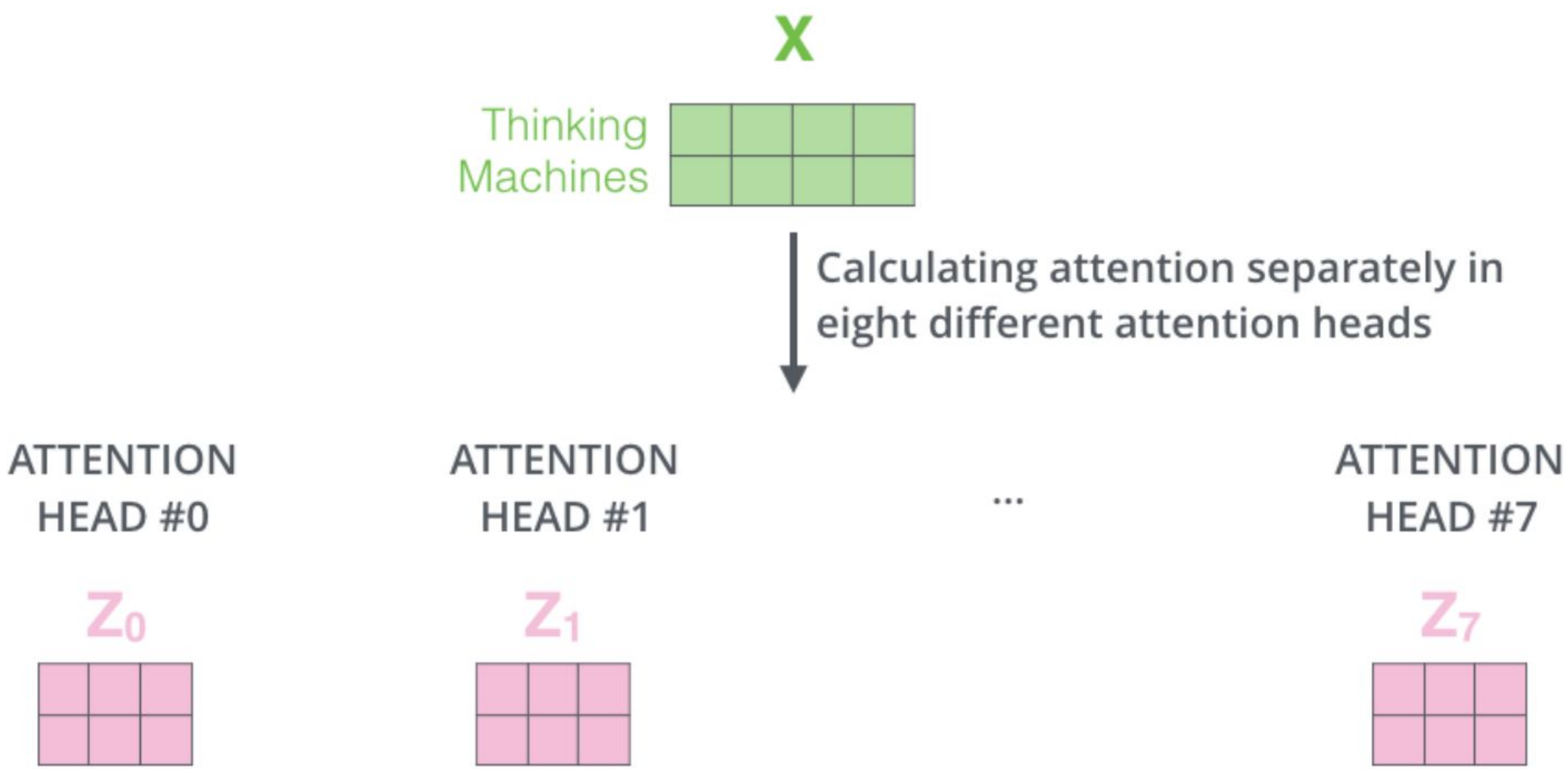


Image Credit: Jay Alammar

Retaining Hidden State Size

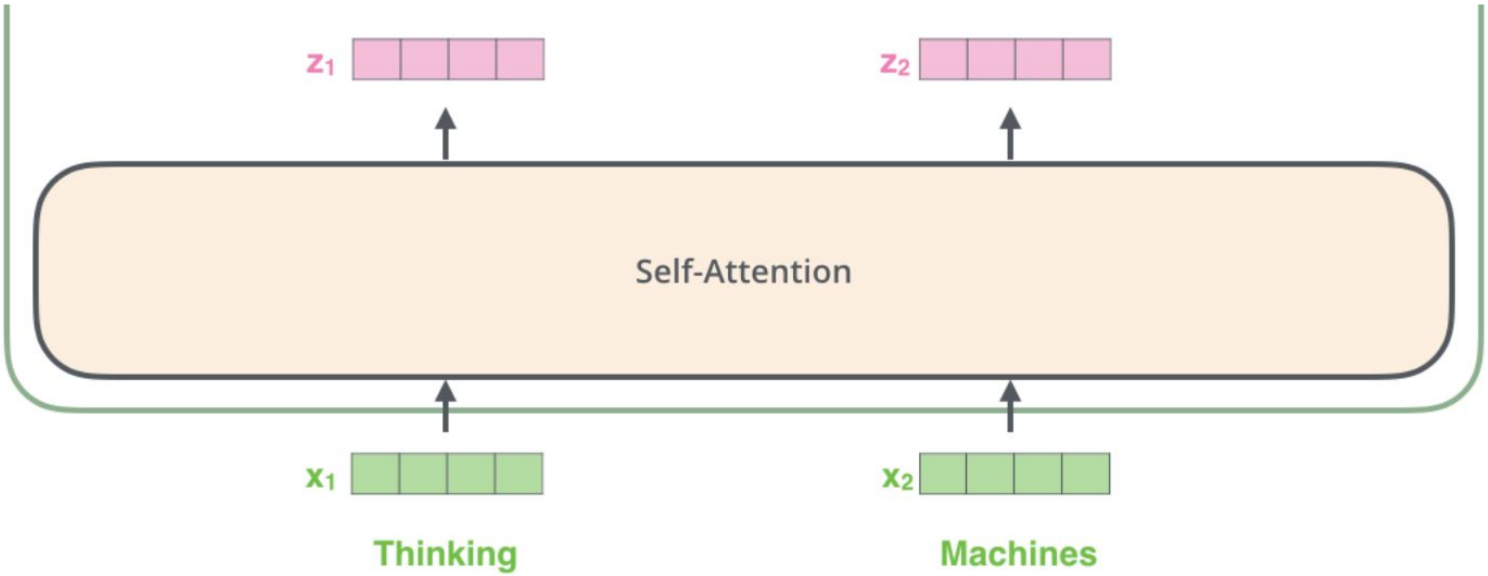


Image Credit: Jay Alammar

Details

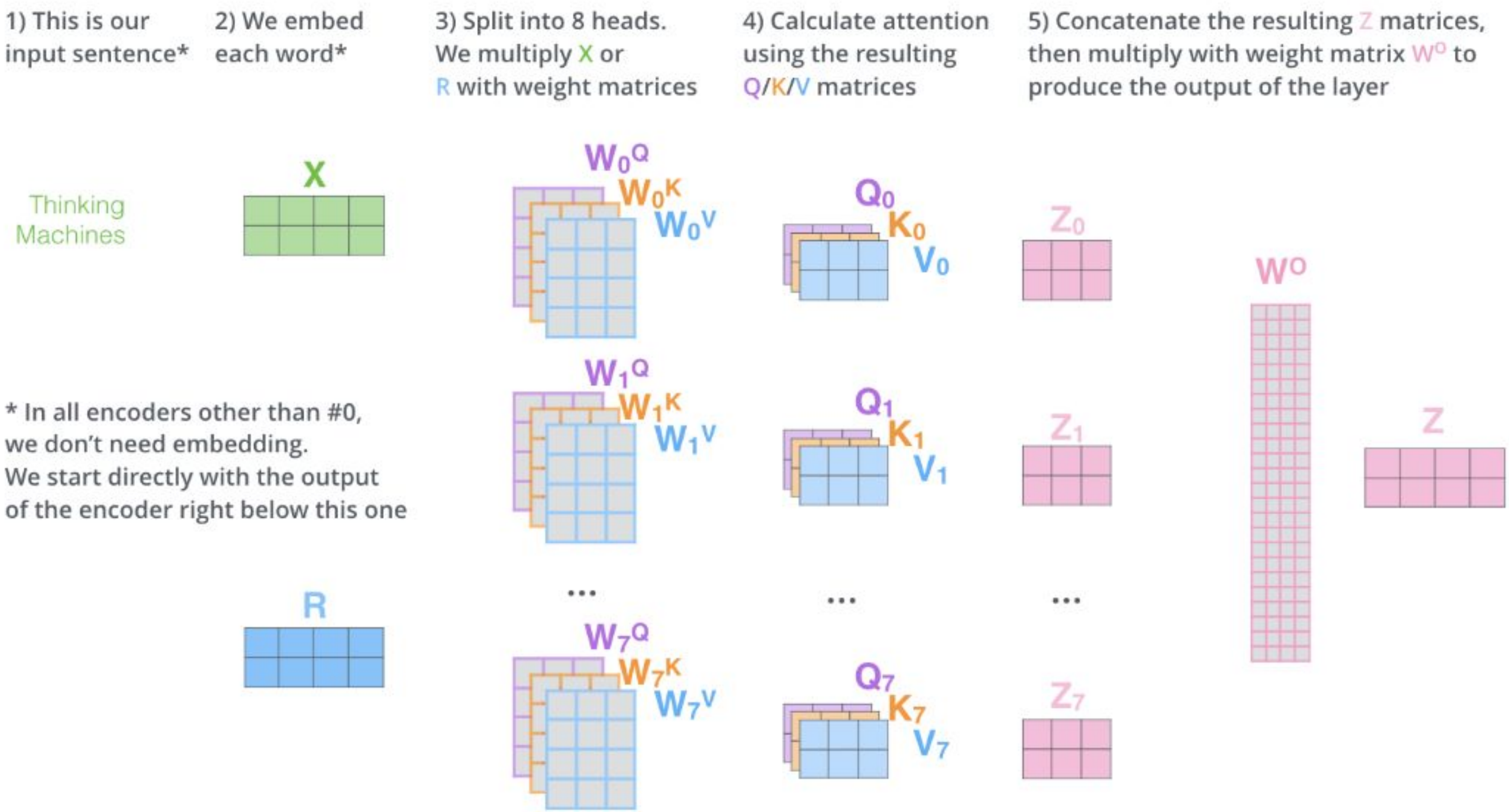


Image Credit: Jay Alammar

Components

- *Scaled Dot-Product* Attention
- Self-Attention
- *Multi-Head* Self-Attention
- Positional Encodings
- Residual Connections

A Single Block

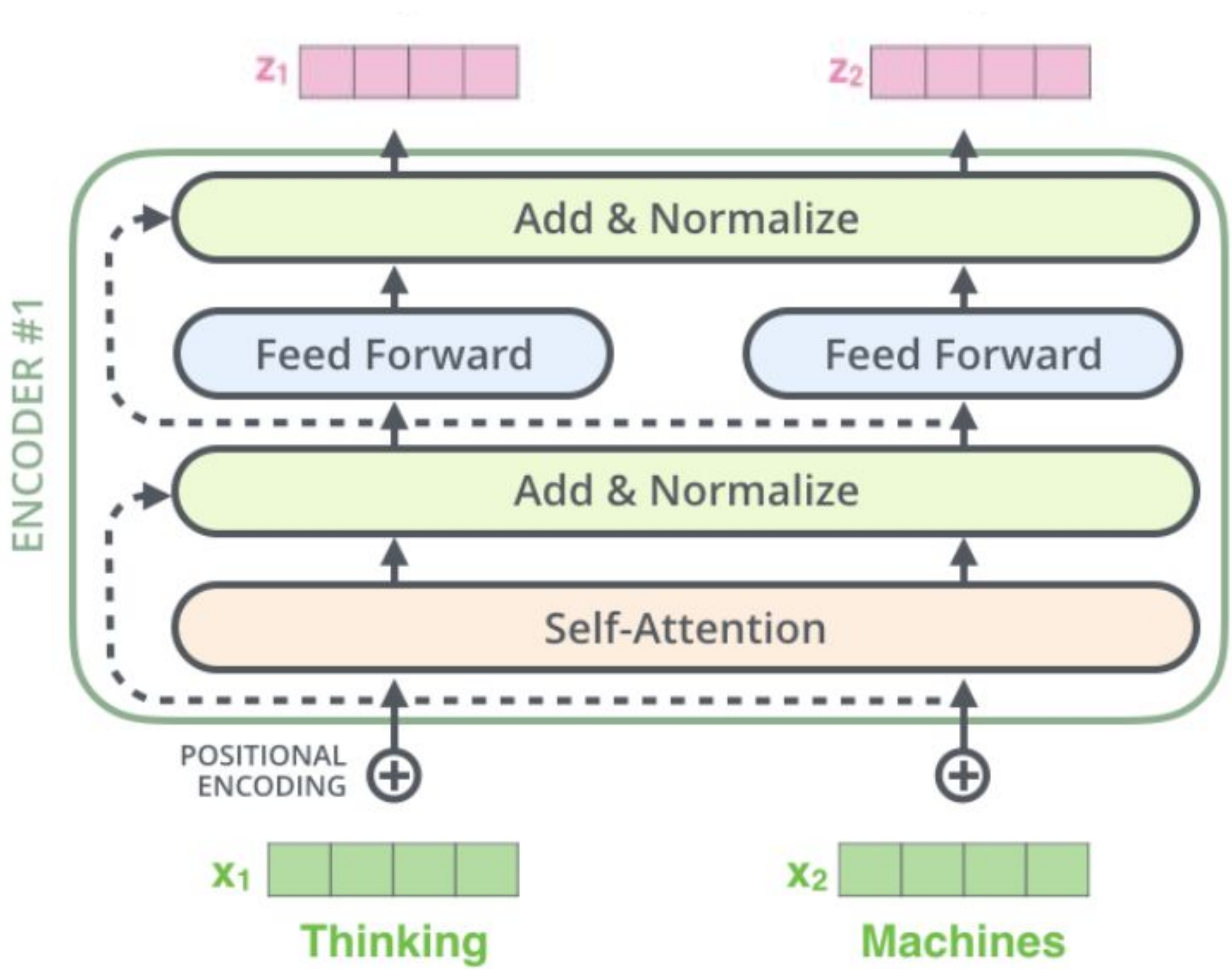


Image Credit: Jay Alammar

A Single Block

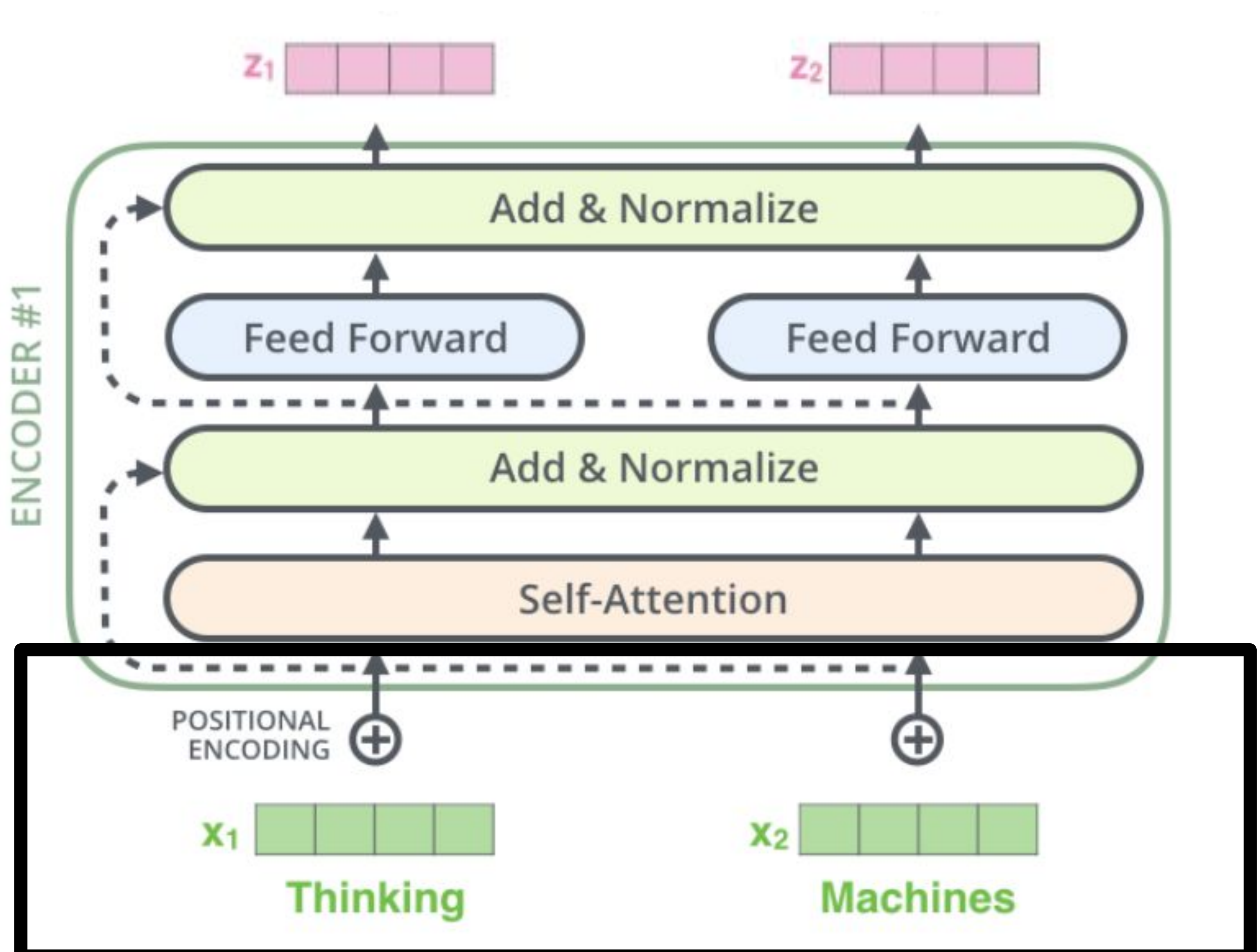


Image Credit: Jay Alammar

Why do we need them?

- It is not the same for a word to be at the beginning, vs. at the end or the middle, of a sentence.
- Both convolution & recurrent models process words in a particular order.
- However, the Transformer doesn’t (everything happens in parallel)
- We want to inject some temporal information so that the model knows the *relationships between words based on their context* (distance & word order is important)

Slide Credit: Sarah Wiegreffe

Positional Encodings

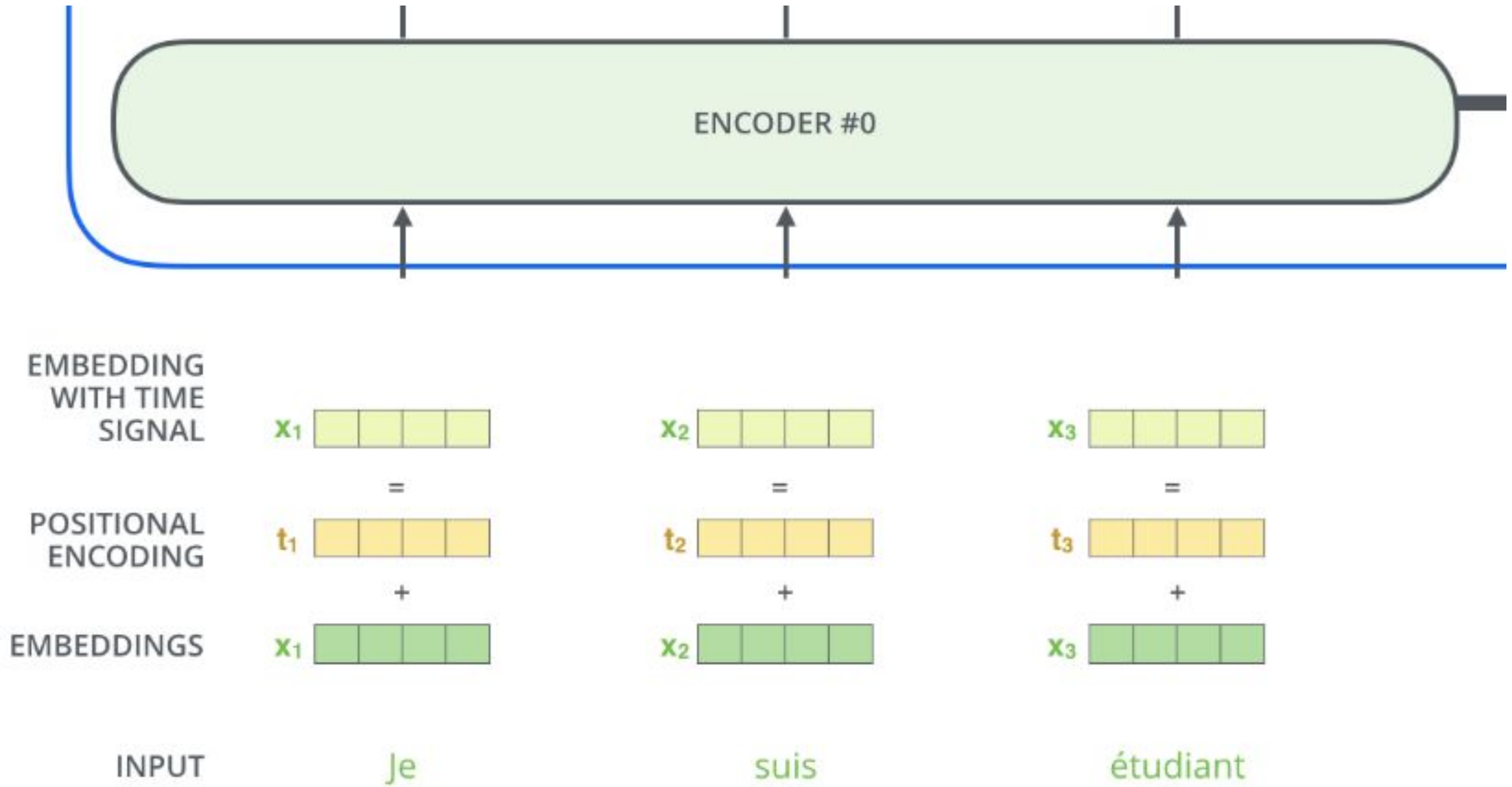


Image Credit: Jay Alammar

Positional Encodings: Two Types

- Sinusoid
  - can extrapolate beyond max. sequence length at test-time
  - represent periodicity of positions: a continuous way of binary encoding of position

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- Learned
  - rather straightforward
  - can't extrapolate
- Perform comparably, so learned embeddings are most often used in practice.

Components

- *Scaled Dot-Product* Attention
- Self-Attention
- *Multi-Head* Self-Attention
- Positional Encodings
- Residual Connections

$$x + \mathrm{Sublayer}(x)$$

Slide Credit: Sarah Wiegreffe

A Single Block

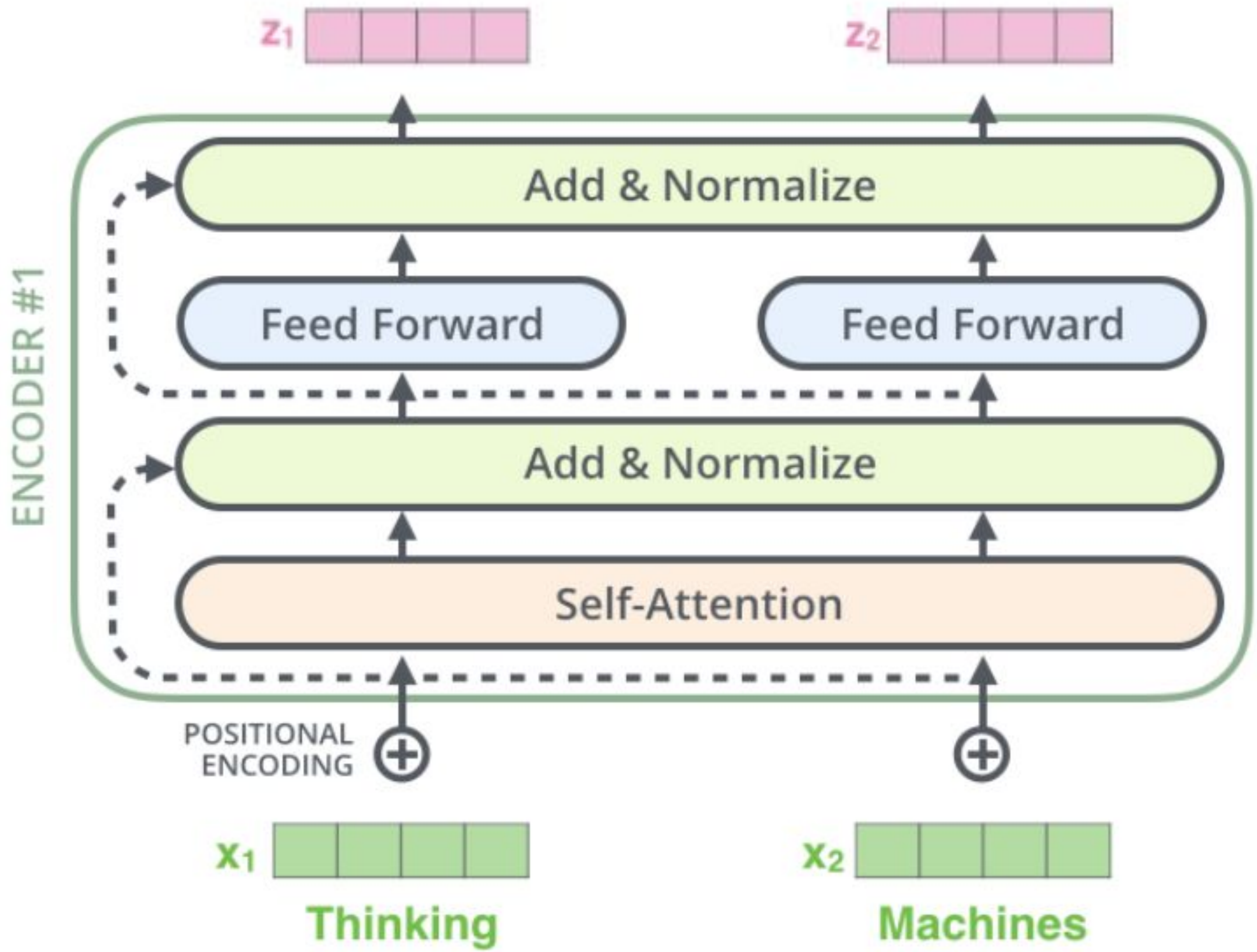
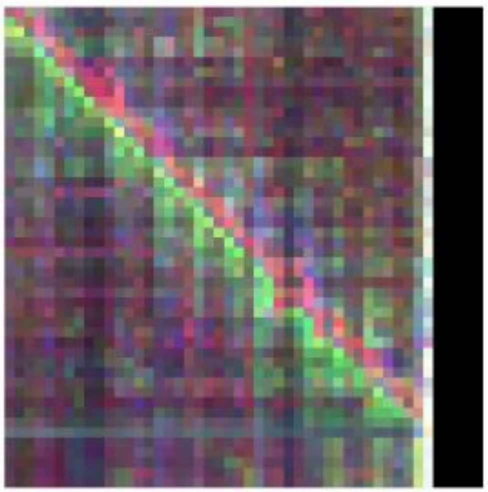


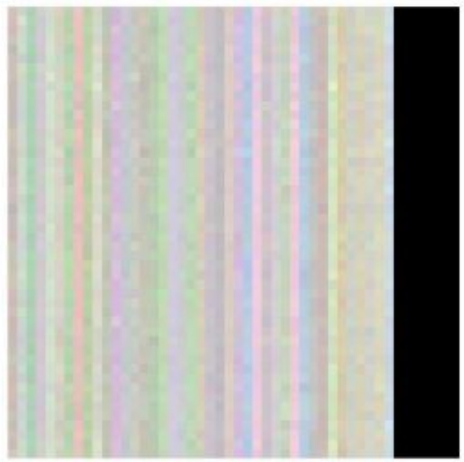
Image Credit: Jay Alammar

Importance of Residuals

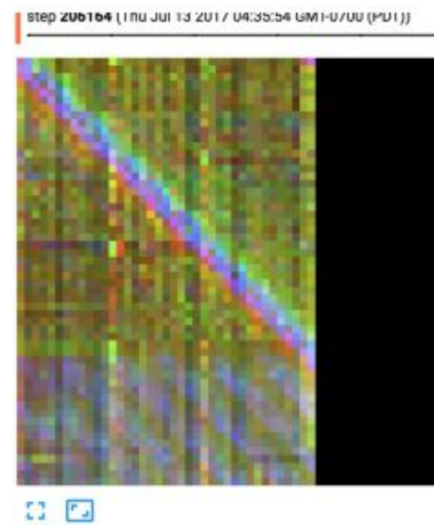
Residuals carry positional information to higher layers, among other information.



With residuals



Without residuals

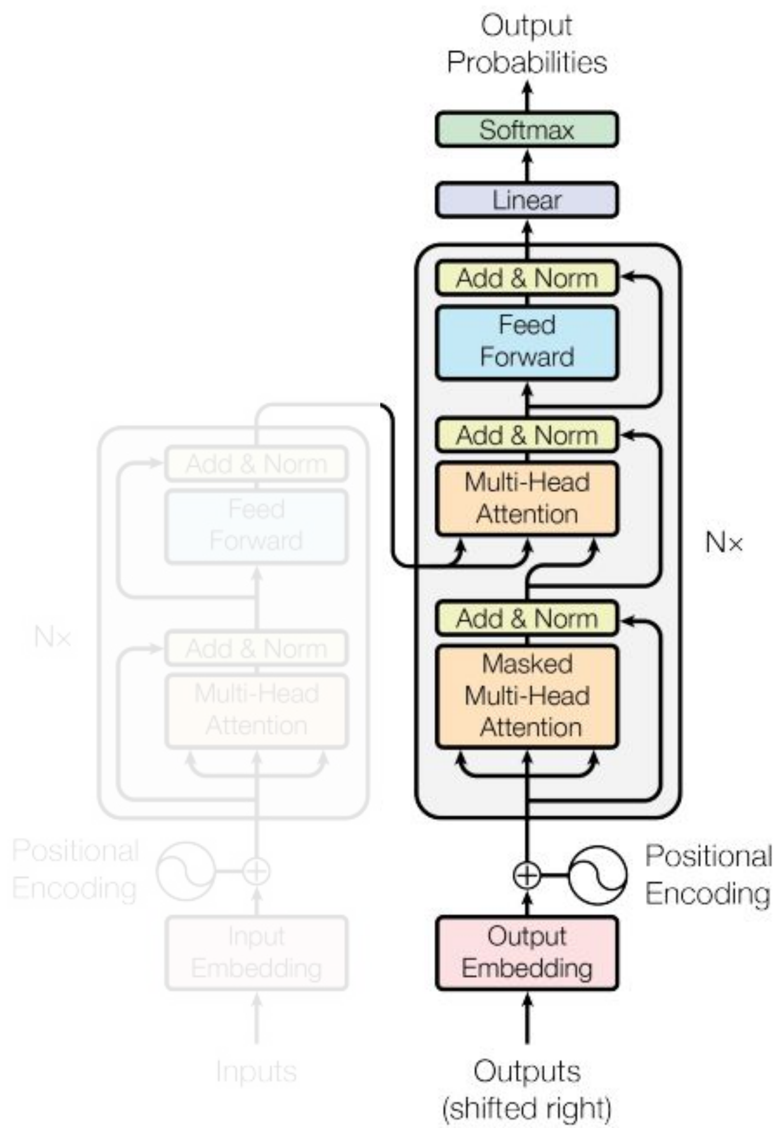


Without residuals,
with timing signals

Plan for Today

- Recap
  - RNNs (for machine translation)
  - Attention
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- Natural Language Applications
  - Language Modeling (ELMo, GPT)
  - BERT ("Masked Language Modeling")

“Attention is All You Need”



Encoder →

← Decoder

Figure Credit: Vaswani et. al

Decoder- A Single Block

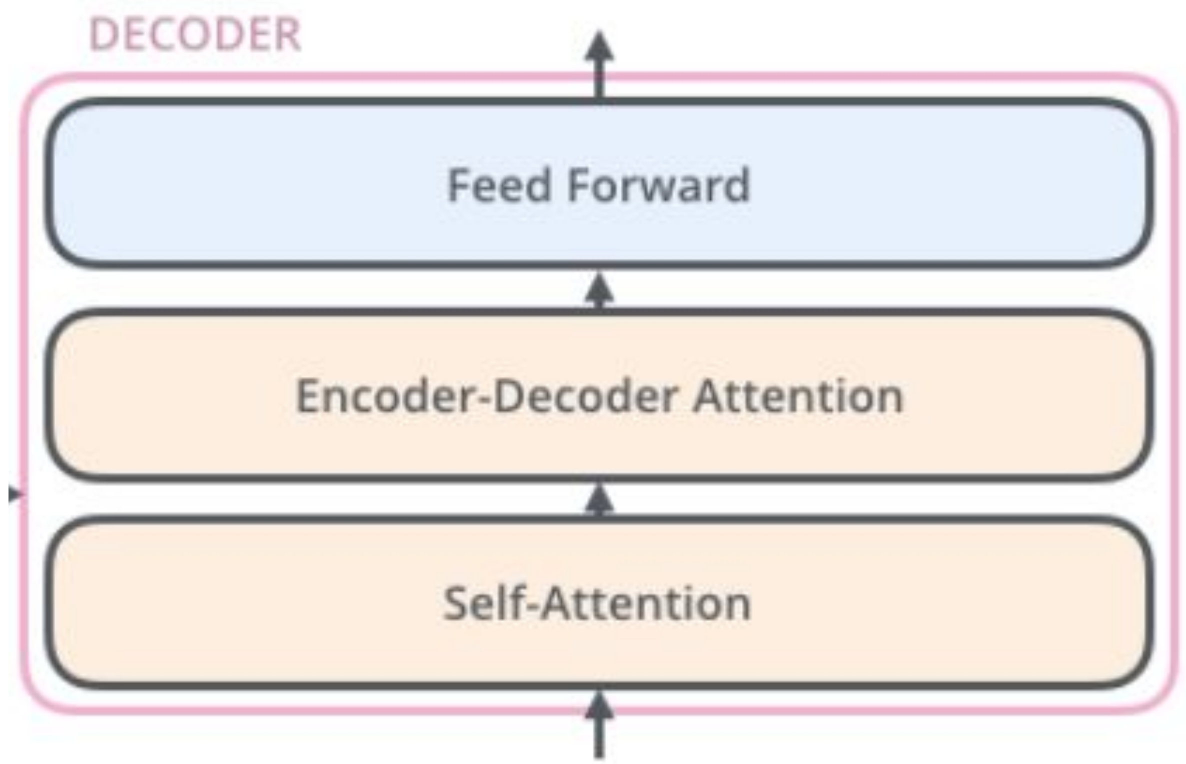


Image Credit: Jay Alammar

Problem with Transformer Encoder

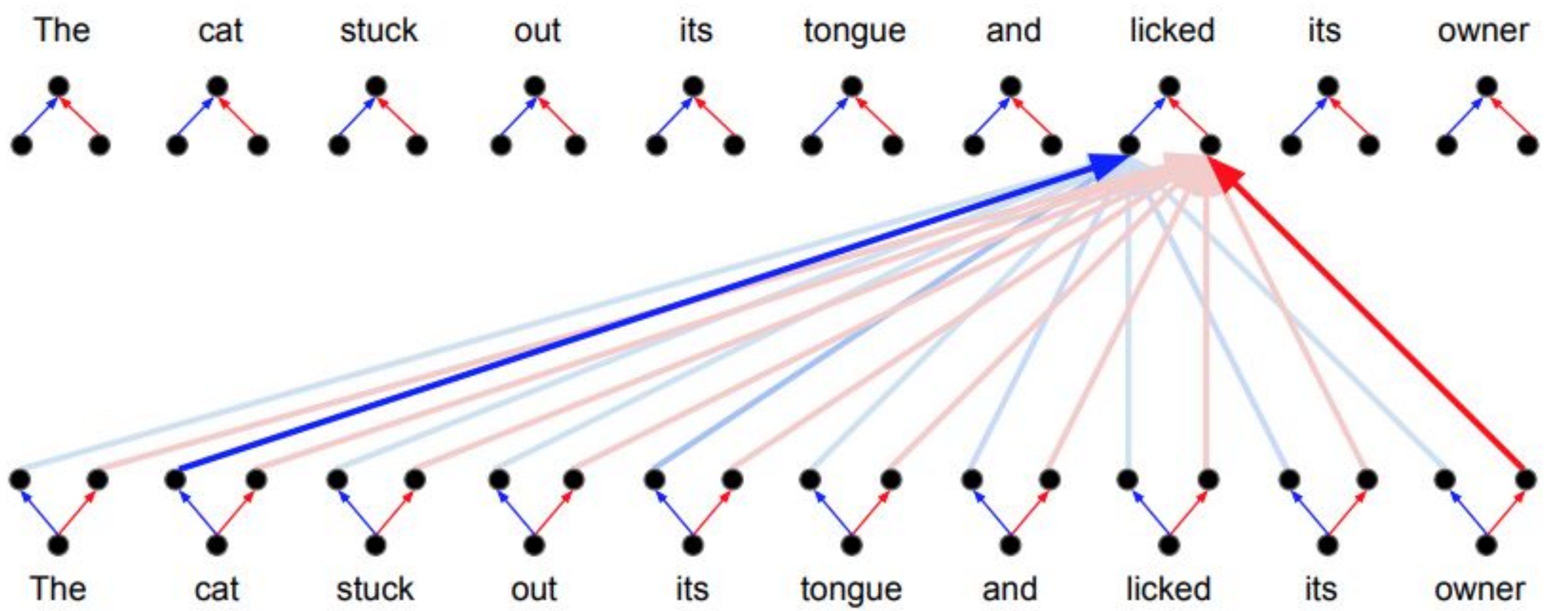


Image Credit: Ashish Vaswani & Anna Huang, Stanford CS224n

We can't see the future

P("he hit me") =
P("he")P("hit"|"he")P("me"|"he hit")

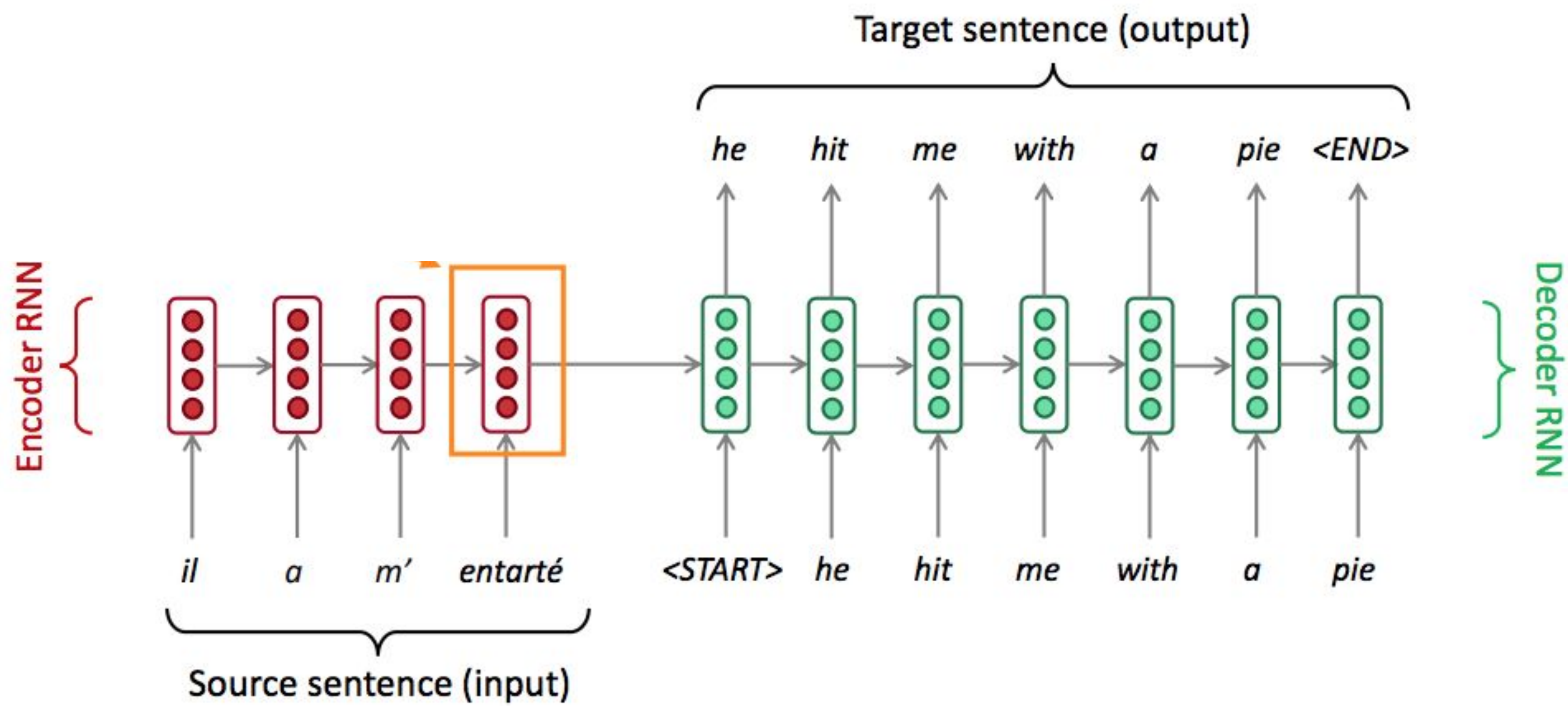


Image Credit: Abigail See, Stanford CS224n

Masked Multi-Head Attention

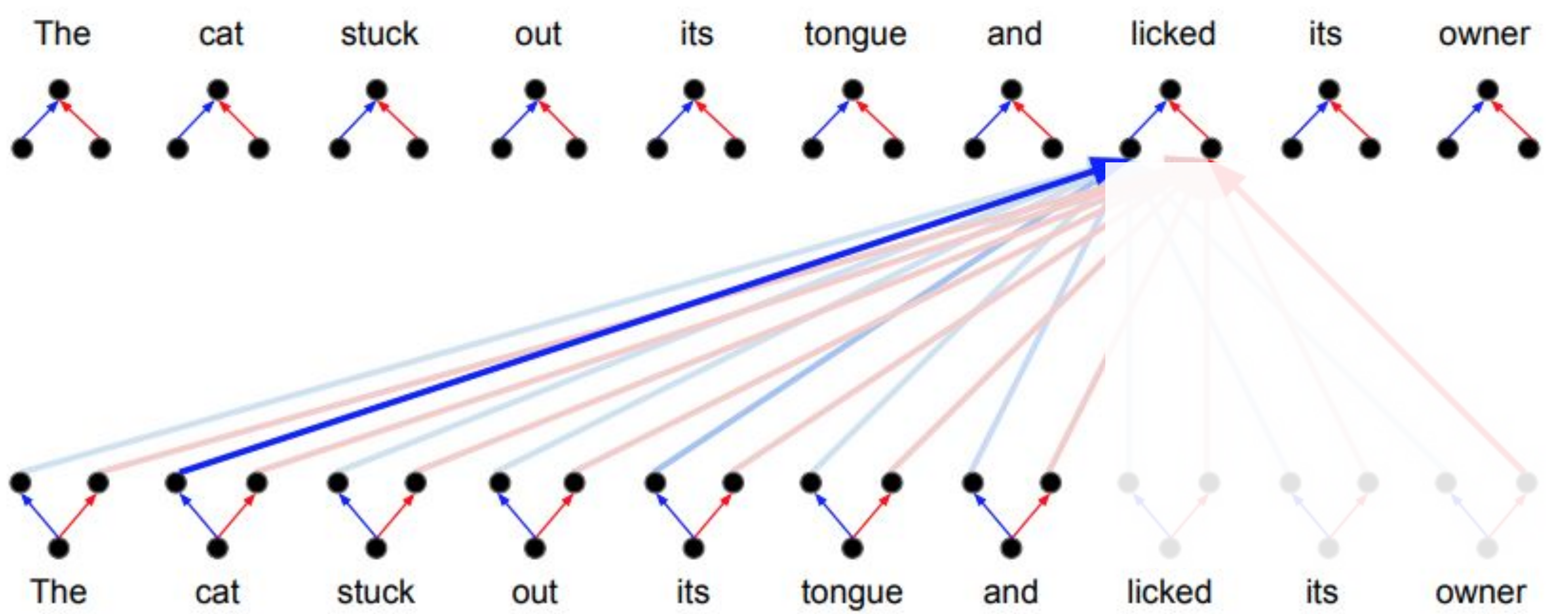


Image Credit: Ashish Vaswani & Anna Huang, Stanford CS224n

Plan for Today

- Recap
  - RNNs (for machine translation)
  - Attention
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- Natural Language Applications
  - Language Modeling (ELMo, GPT)
  - BERT ("Masked Language Modeling")

Attention is Cheap

Attention is Cheap

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

- Cheaper when n < d
- No long-range dependencies!
- No sequential operations!

Table Credit: Vaswani et. al 2017

Plan for Today

- Recap
  - RNNs (for machine translation)
  - Attention
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- Natural Language Applications
  - Language Modeling (ELMo, GPT)
  - BERT (“Masked Language Modeling”)

Useful Resources

- Jay Allamar's blogposts: http://jalammar.github.io/
- Lecture on Transformers: https://www.youtube.com/watch?v=5vcj8kSwBCY&feature=youtu.be
- Stanford CS224n slides/resources http://web.stanford.edu/class/cs224n/
- The Annotated Transformer https://nlp.seas.harvard.edu/2018/04/03/attention.html
- Jacob Eisenstein's Intro to NLP textbook https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf
- Yoav Goldberg's Deep Learning for NLP textbook https://www.morganclaypool.com/doi/abs/10.2200/S00762ED1V01Y201703HLT037

Lecture ends here.

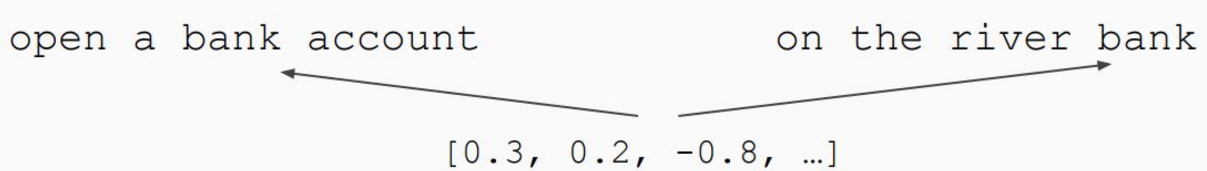
Plan for Today

- Recap
  - RNNs (for machine translation)
  - Attention
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- Natural Language Applications
  - Language Modeling (ELMo, GPT)
  - BERT (“Masked Language Modeling”)

Word Embeddings

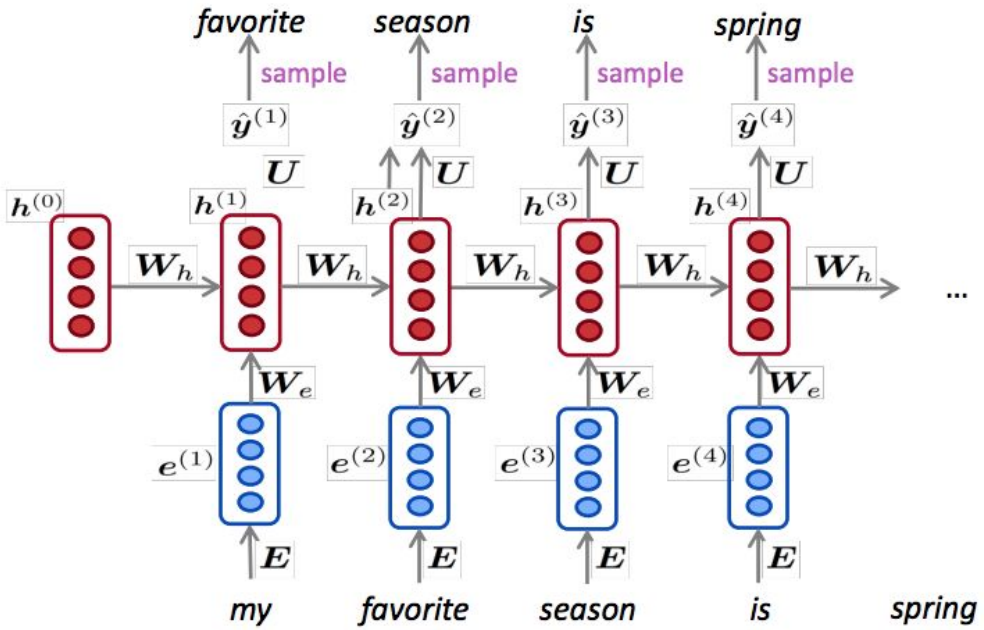
- Word embeddings (`word2vec`, `GloVe`) are often *pre-trained* on text corpus from co-occurrence statistics

- **Problem**: Word embeddings are applied in a context free manner



Did we all along have a solution to this problem?

- In, an NLM, we immediately stuck word vectors (perhaps only trained on the corpus) through LSTM layers
- Those LSTM layers are trained to predict the next word
- But those language models are producing context-specific word representations at each position!



Slide Credit: Chris Manning, CS224n

Transformers Do This Even Better

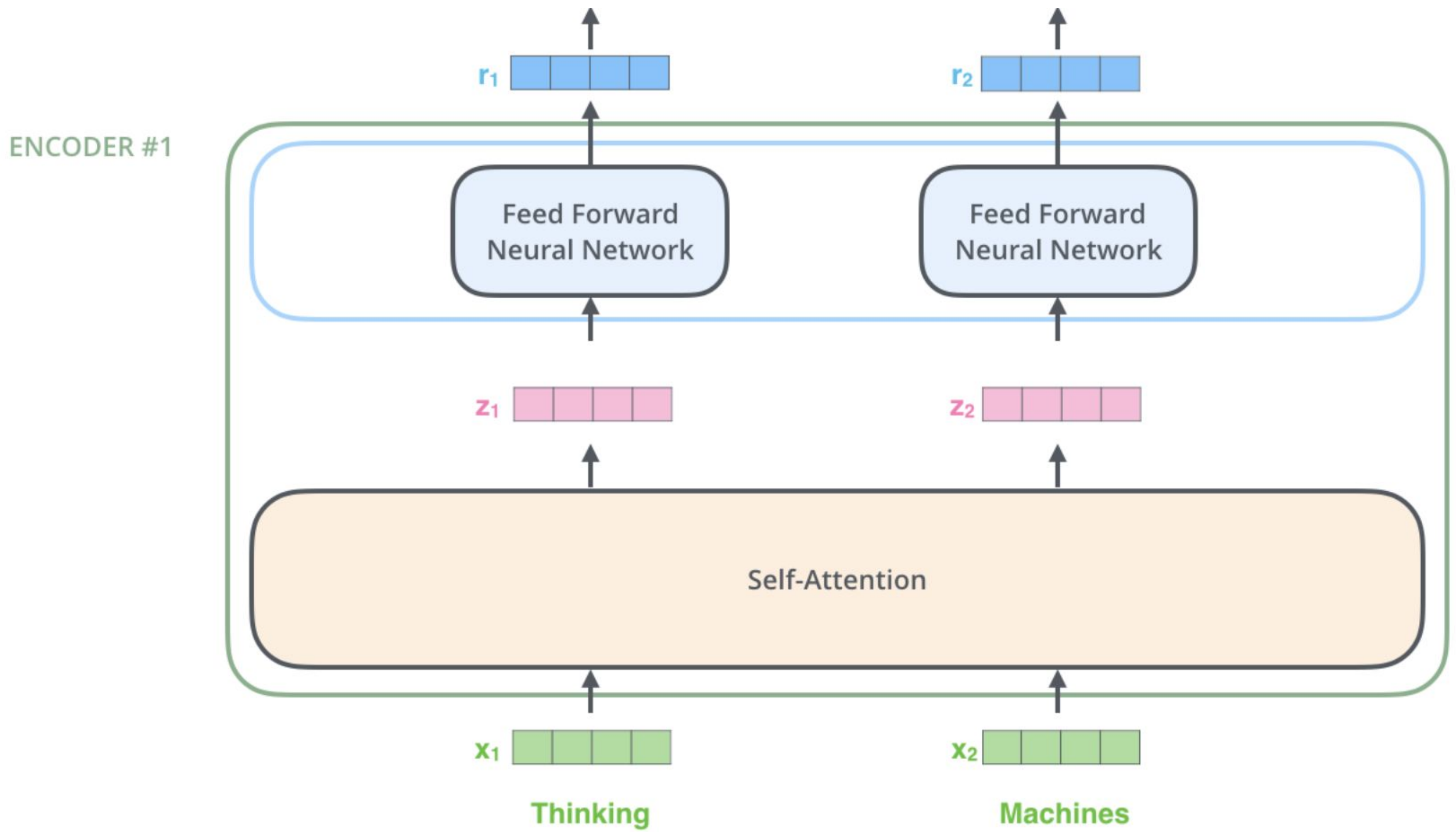


Image Credit: Jay Alammar

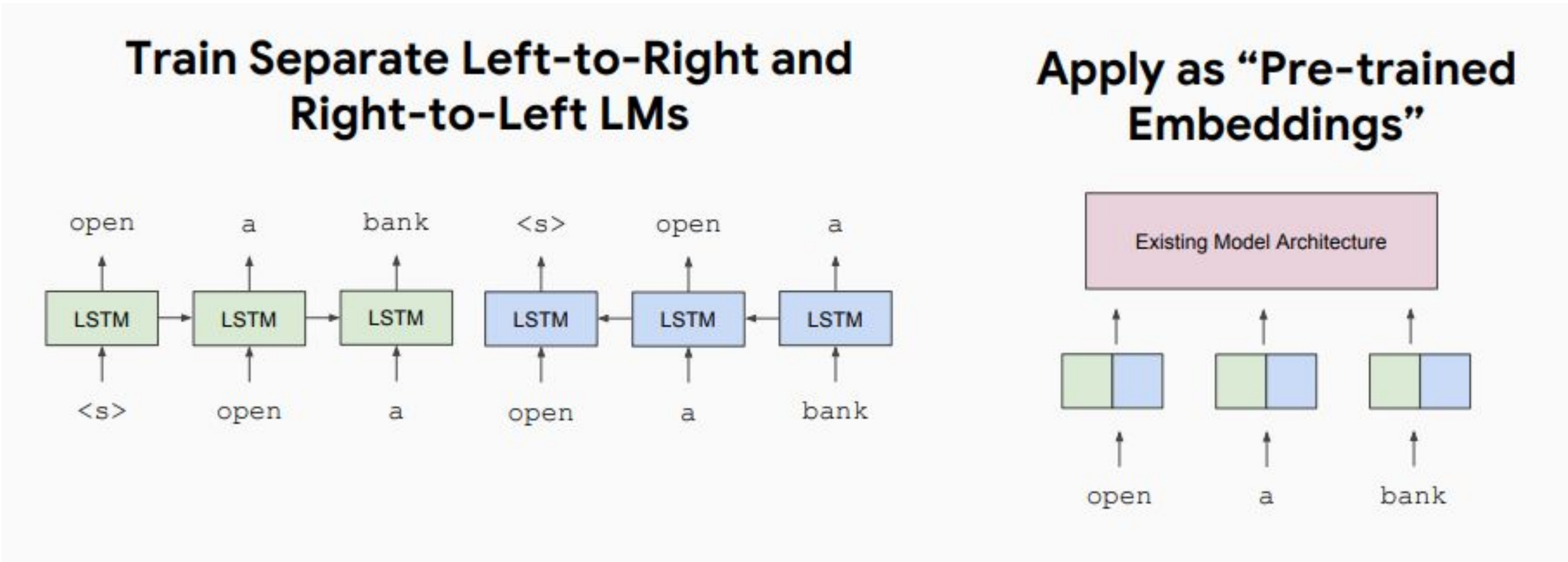
Contextualized Embeddings

- Embeddings from Language Models = **ELMo** (Peters et. al 2017)
- **OpenAI GPT** (Radford et al. 2018)
- Bidirectional Encoders from Transformers = **BERT** (Devlin et. al 2018)

OpenAI

Slide Credit: Sarah Wiegreffe

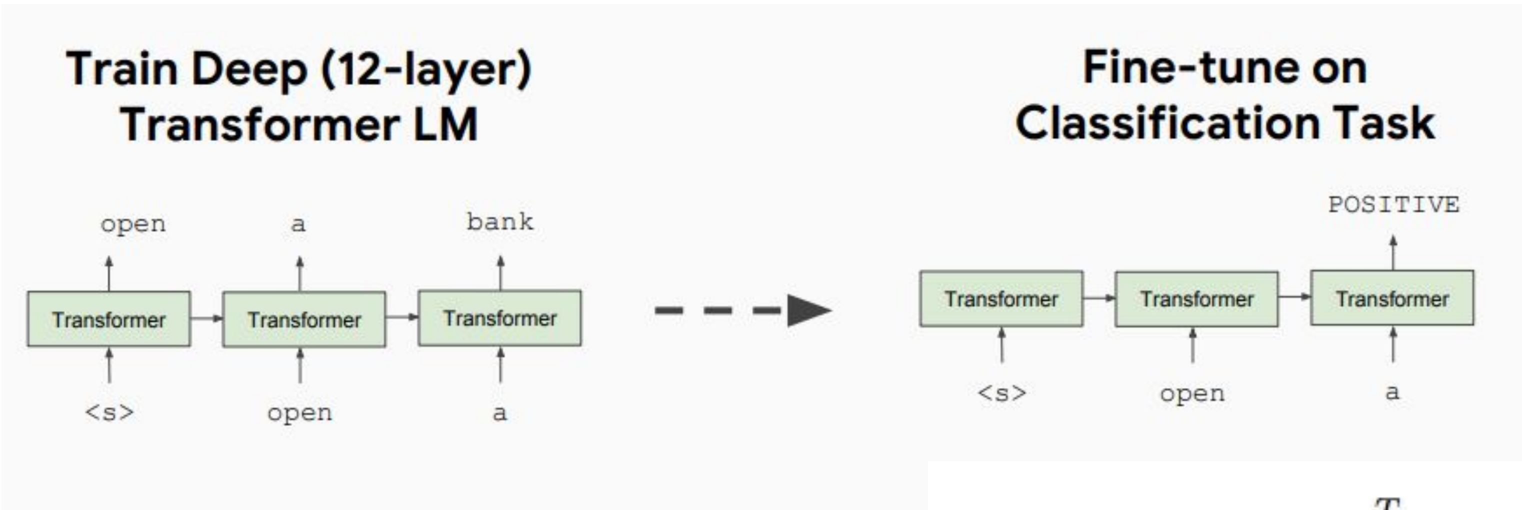
ELMo



- Why can't you use a bi-directional RNN?

Image Credit: Jacob Devlin

OpenAI GPT



$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t})$$

- Use a Transformer **decoder** to "predict the next word"
- Why can't you use a bi-directional Transformer?

Image Credit: Jacob Devlin

Answer

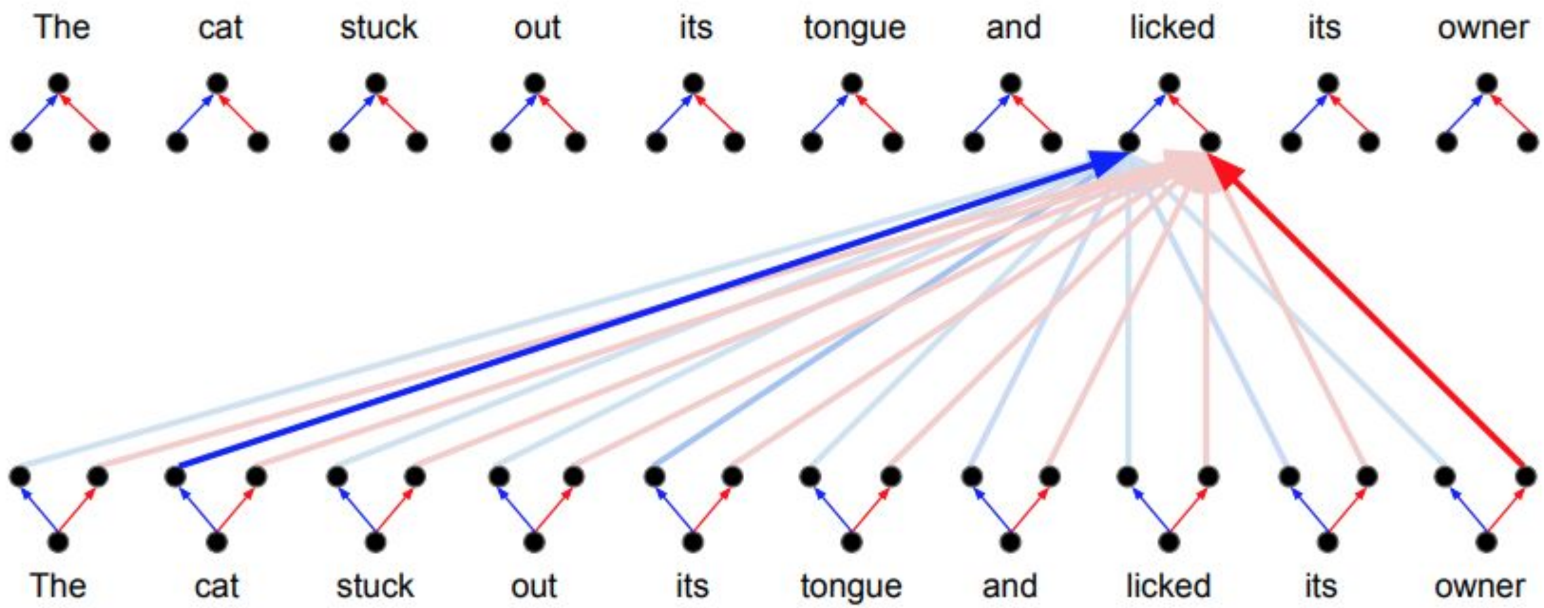
- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
  - We don't care about this.
- Reason 2: Words can "see themselves" in a bidirectional encoder.

- We want to use a Transformer encoder!

Image Credit: Jacob Devlin

Answer

Bidirectional context
Words can "see themselves"



Plan for Today

- Recap
  - RNNs (for machine translation)
  - Attention
- Transformers: an alternative to RNNs
  - Architecture
  - Decoding
  - Efficiency
- Natural Language Applications
  - Language Modeling (ELMo, GPT)
  - BERT (“Masked Language Modeling”)

BERT

Solution: Mask out *k*% of the input words, and then predict the masked words

- We always use $k = 15\%$

$$\max_{\theta} \quad \log p_{\theta}(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_{\theta}(x_t \mid \hat{\mathbf{x}})$$

```
                        store              gallon
                          ↑                  ↑
the man went to the [MASK] to buy a [MASK] of milk
```

- `BERT-Base`: 12-layer, 768-hidden, 12-head
- `BERT-Large`: 24-layer, 1024-hidden, 16-head

Slide Credit: Jacob Devlin, XLNET (2019)

BERT

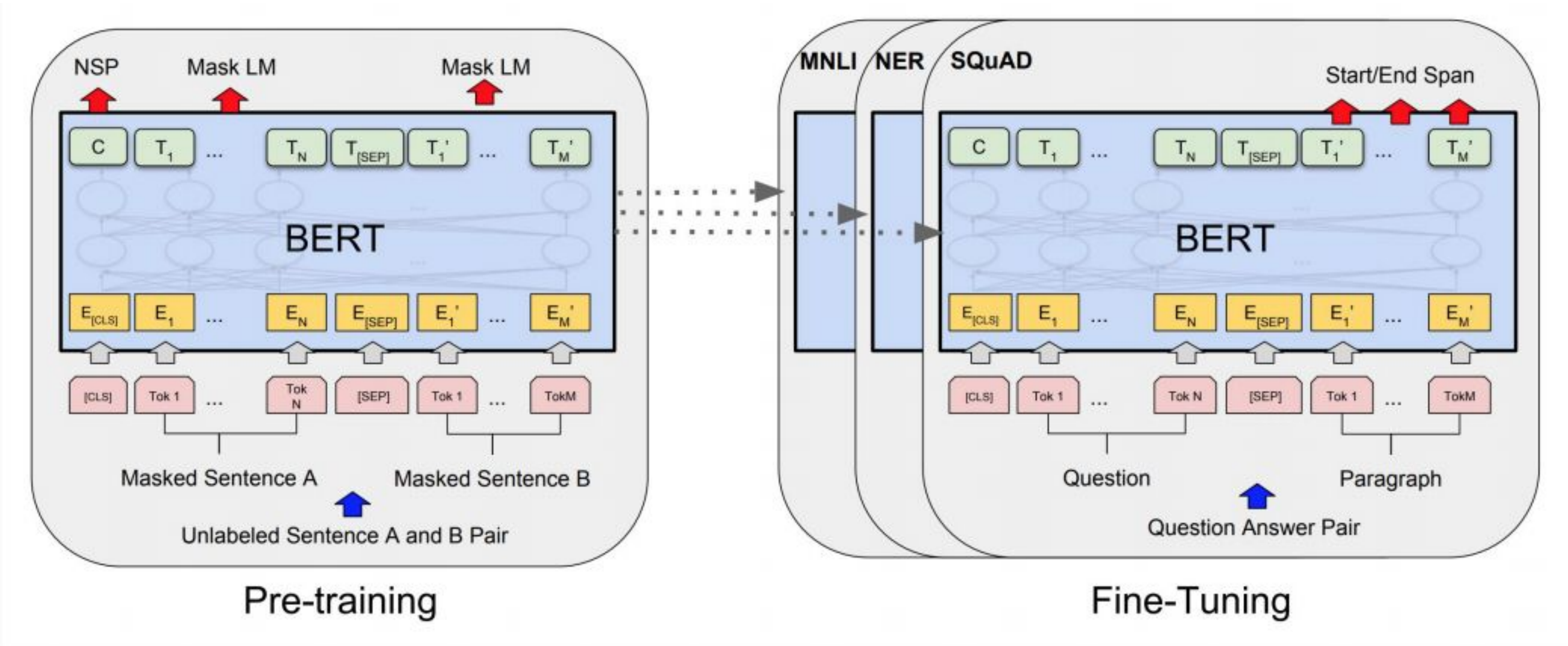


Image Credit: Jacob Devlin

Recap: Contextualized Word Representations

Trained on:

	Unidirectional Context	Bidirectional Context
RNN	**ELMo** (and others)	others
Transformer	**GPT and GPT-2**	**BERT**