CMPT 983

Grounded Natural Language Understanding

January 17, 2022

Review of deep learning models

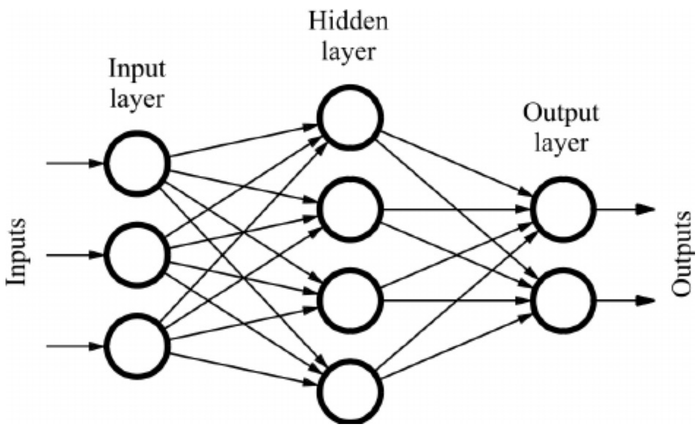
Today

- Review of basic deep learning building blocks
  - CNNs
  - RNNs
  - Attention
  - Transformers

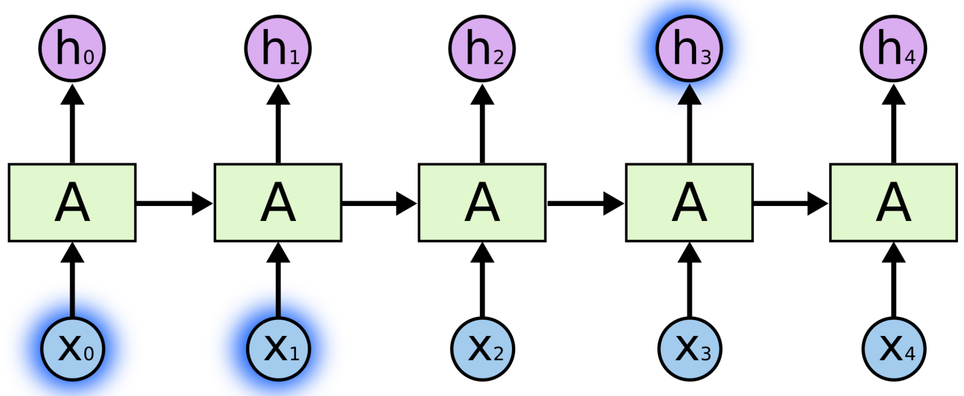
Deep learning models

Neural network architectures

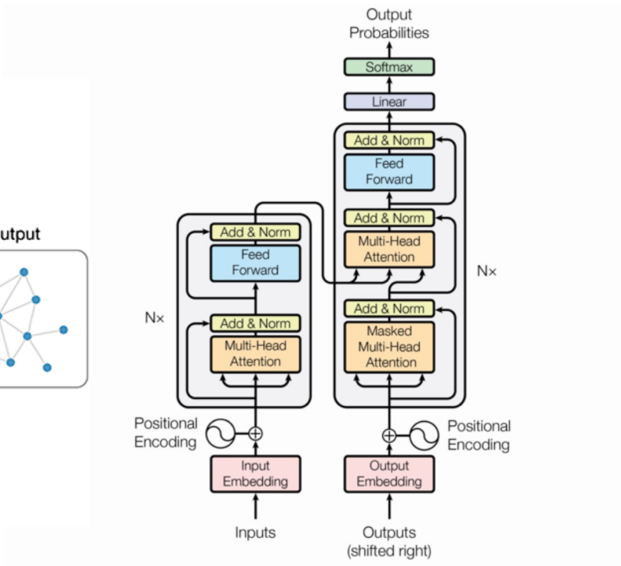
Feed-forward NNs



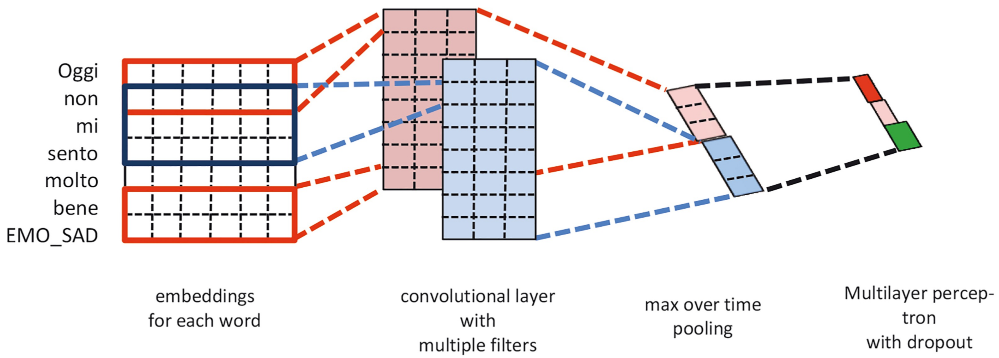
Recurrent networks (RNNs)



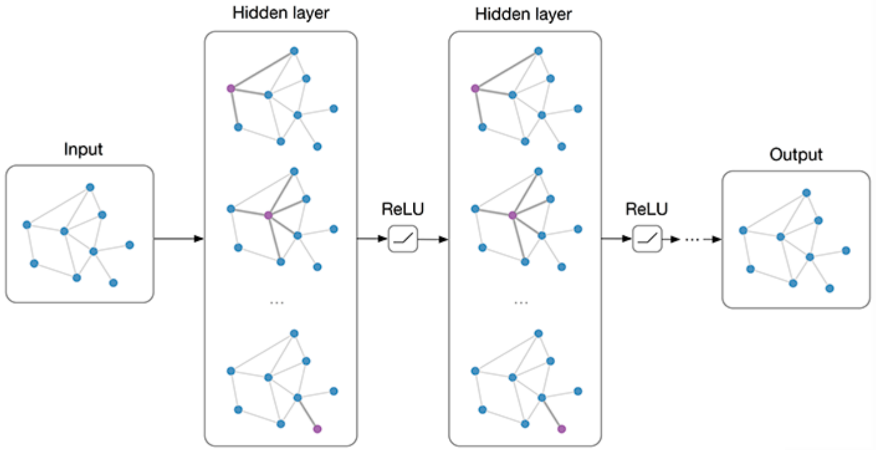
Transformers



Convolution networks (CNNs)



Graph NNs



- All network architectures can be used to model **images**, **text**, **3D representations**, etc.

- Traditionally:
  - CNNs for images – scale/translation invariance
  - RNNs for sequences (text)
  - Transformers were introduced for machine translation
    - Now used for images and 3D shapes as well
    - Currently SOTA vision+language models are all using transformers!

Modelling Images

Modelling Images

I

Image

Image Encoder

V

Useful Visual Feature

Modelling Images

Convolutional Neural Networks

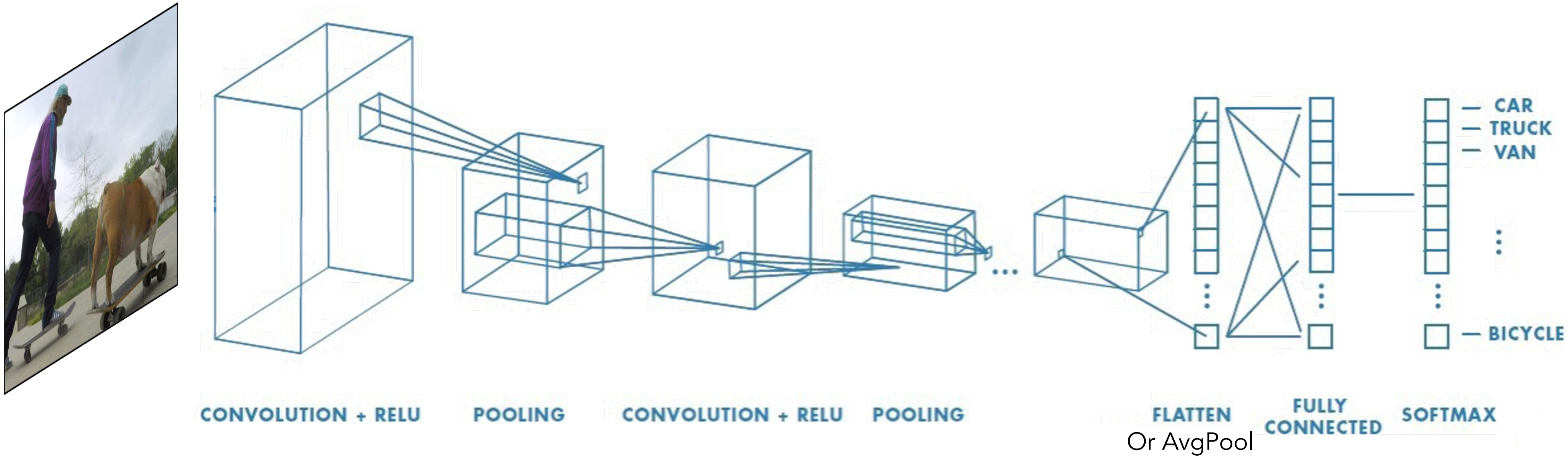
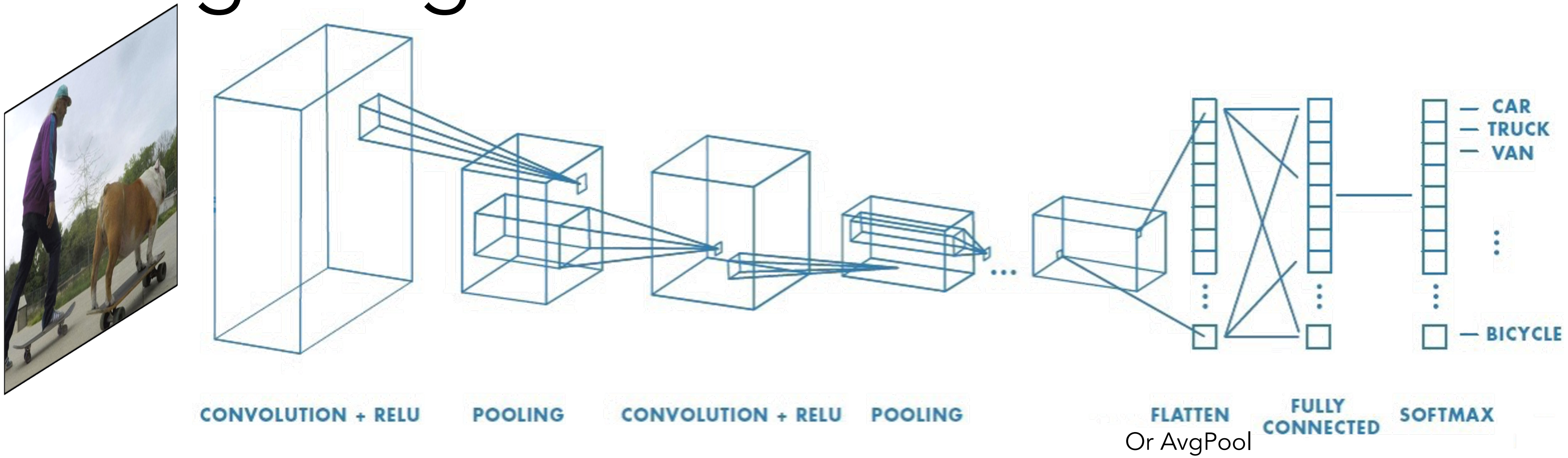


Image Credit: MathWorks

Modelling Images



Modelling Images

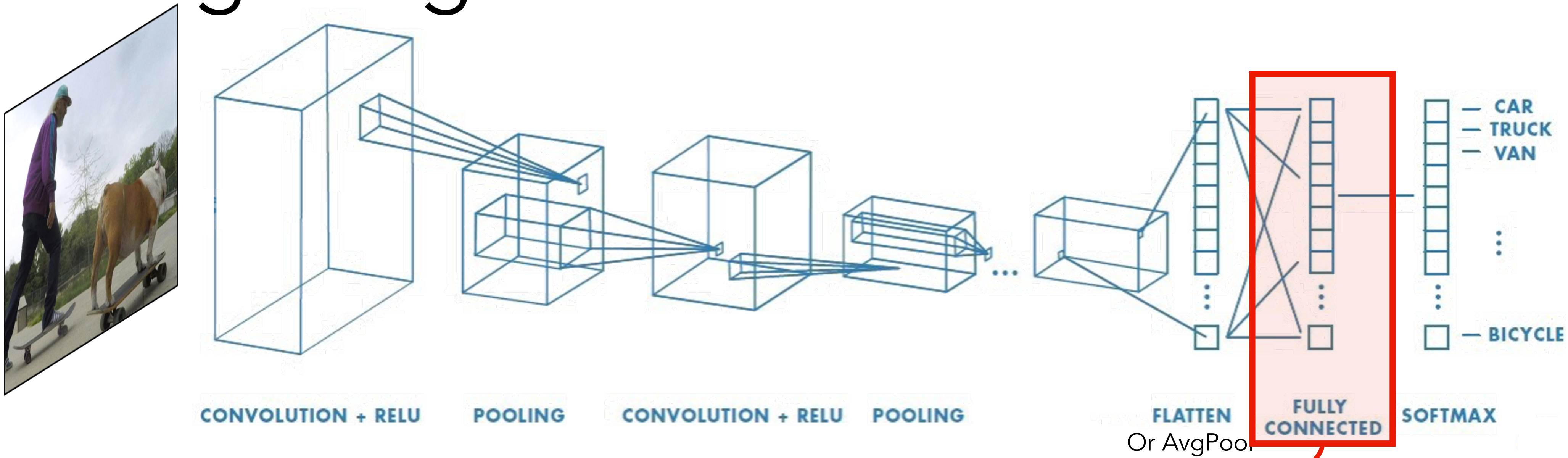
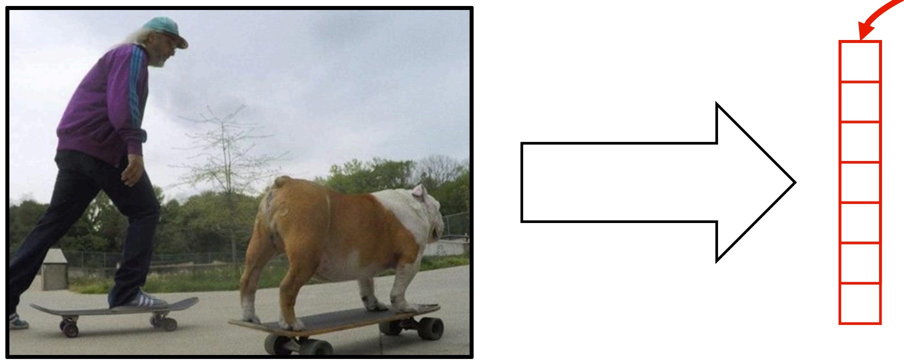


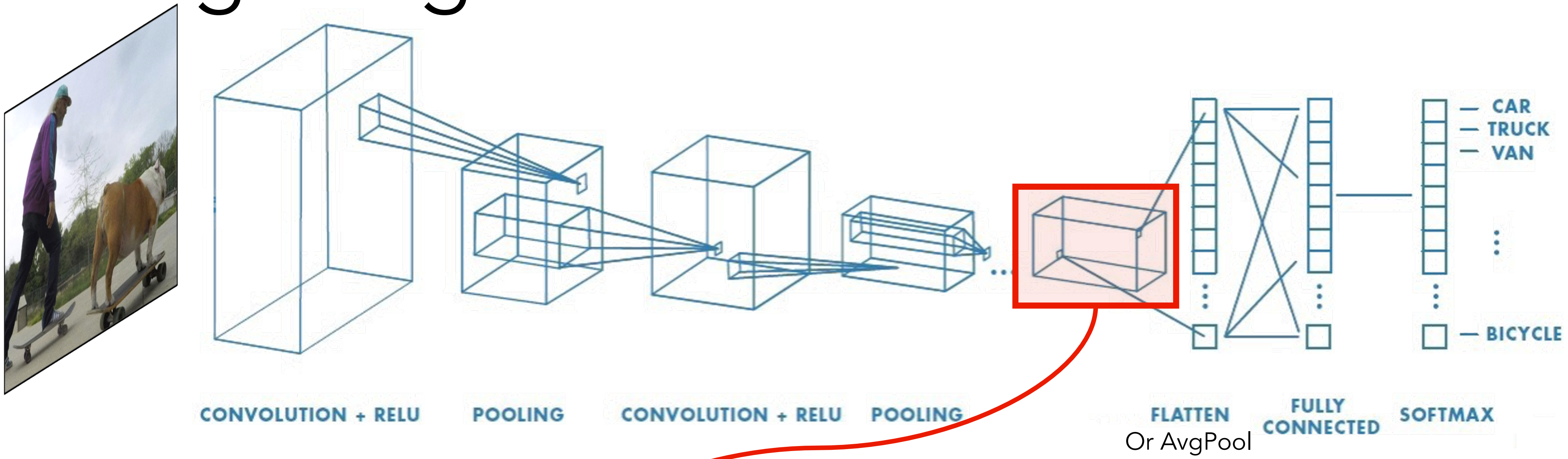
Image Level Feature



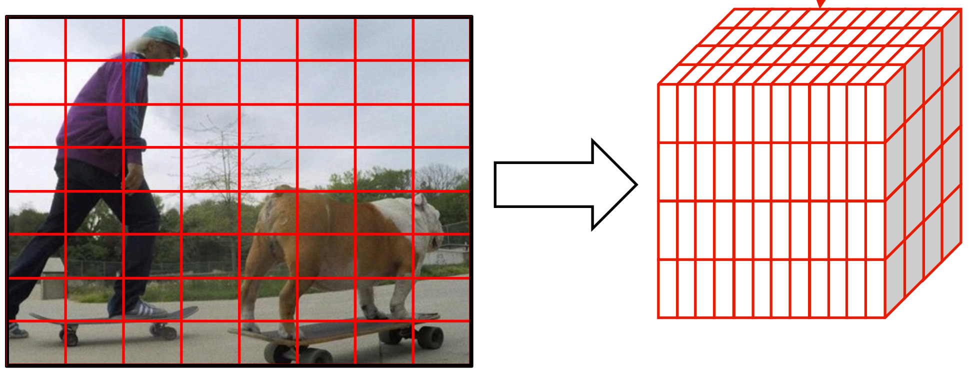
$$V \in \mathbb{R}^{1\times d}$$

- No spatial information
- Highly compressed

Modelling Images



Spatial Image Features

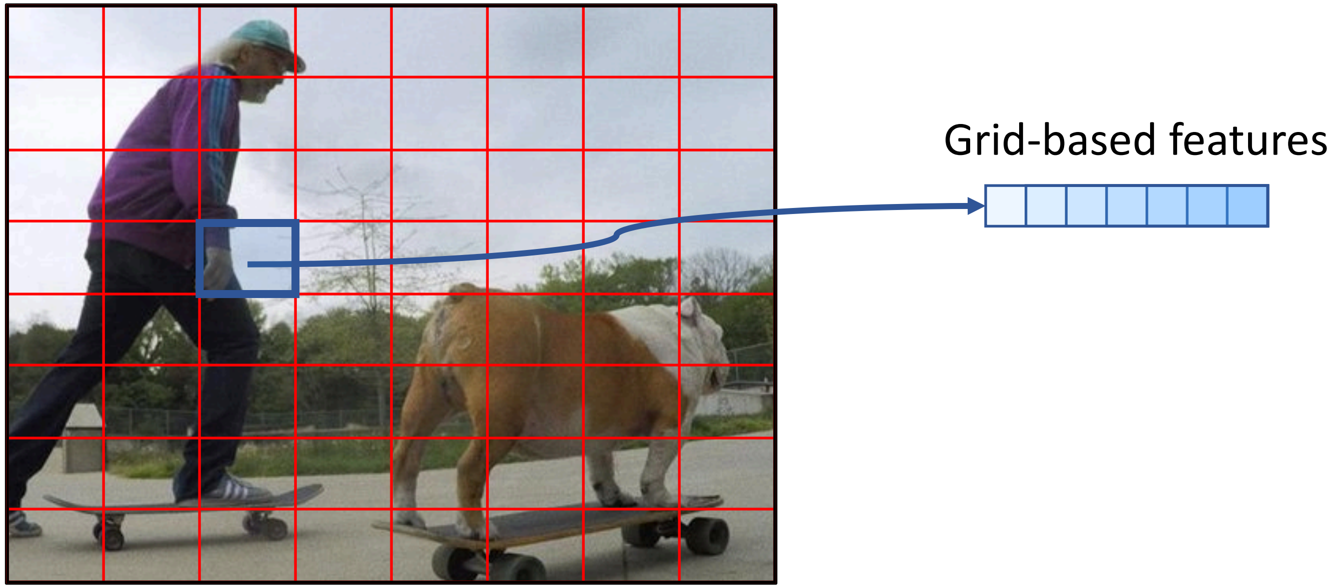


$$V \in \mathbb{R}^{w \times h \times d}$$

- Feature vector per grid cell
- Captures some spatial info
- Uniform grid…

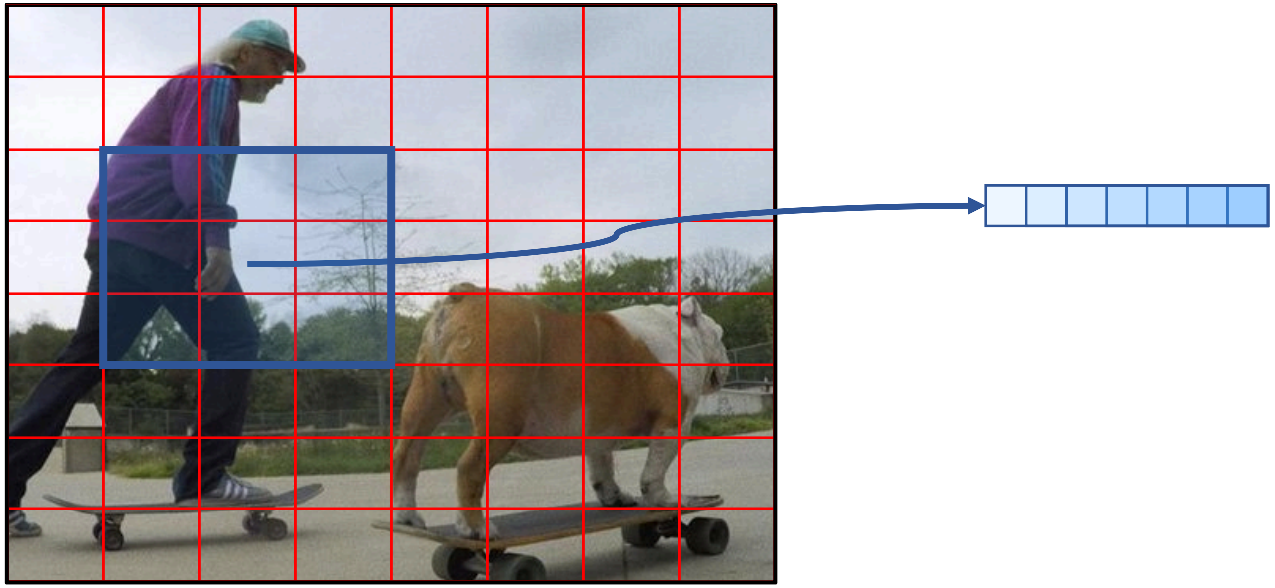
Slide credit: Stefan Lee

Modelling Images



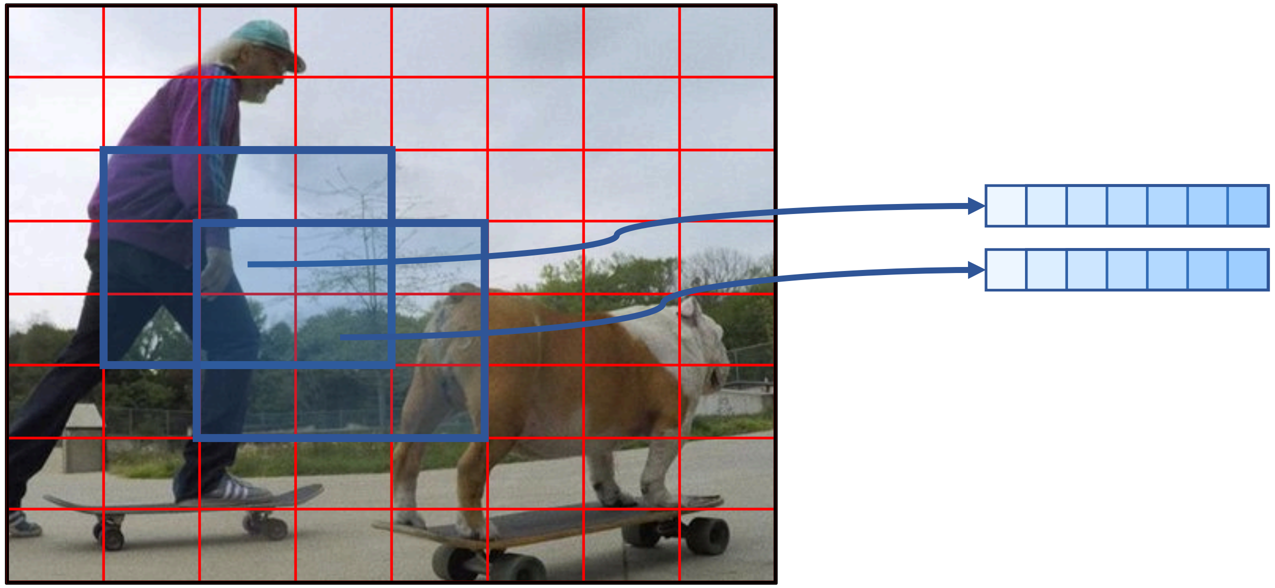
Slide credit: Stefan Lee

Modelling Images



*Considering receptive field it is actually much more like

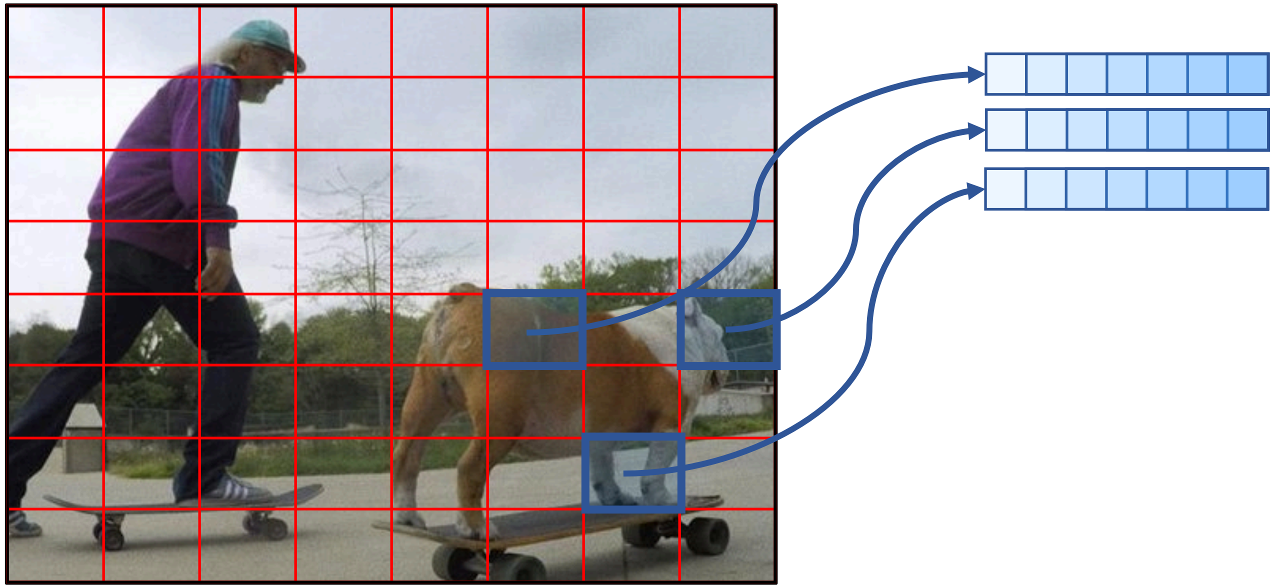
Modelling Images



*Considering receptive field it is actually much more like

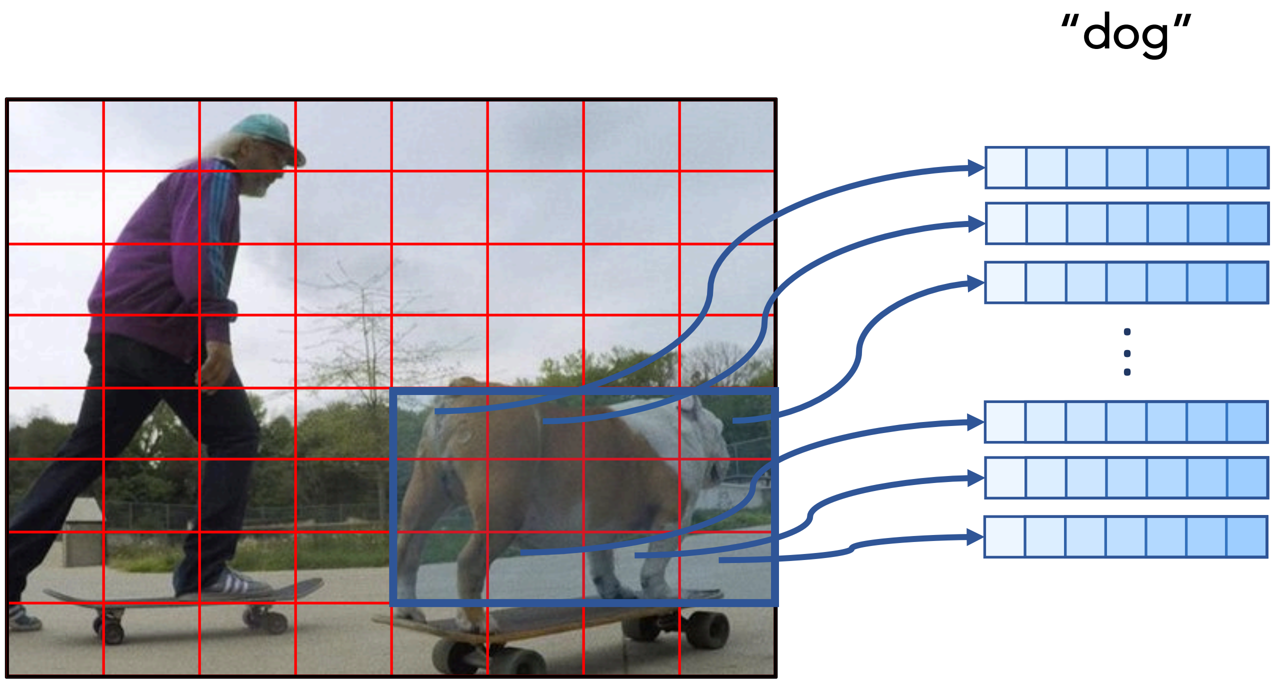
Modelling Images

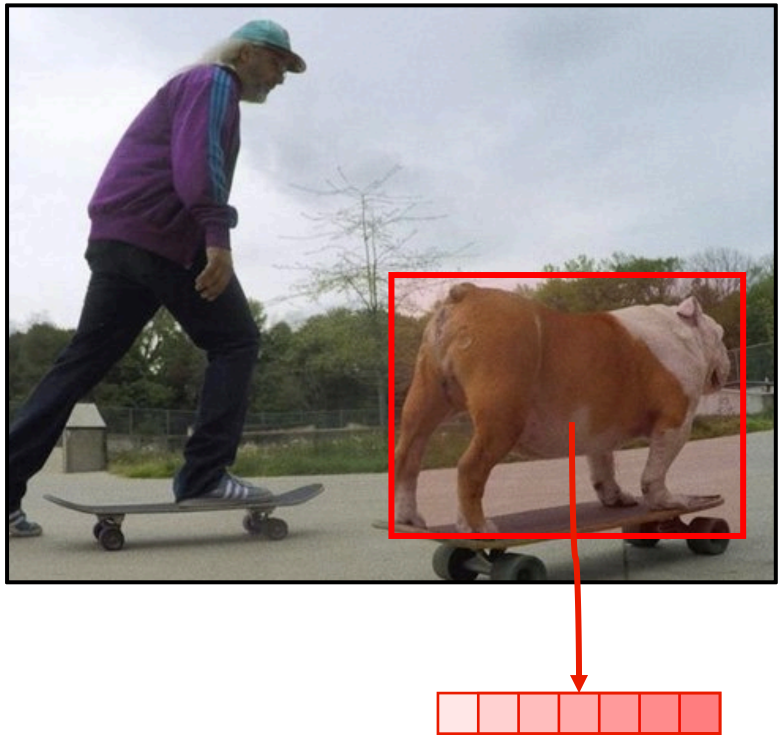
"dog"



Slide credit: Stefan Lee

Modelling Images





Modelling Images

Idea: Switch to object detection models as the backbone for image representation

- Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering arxiv.org/abs/1707.07998

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
https://arxiv.org/abs/1506.01497

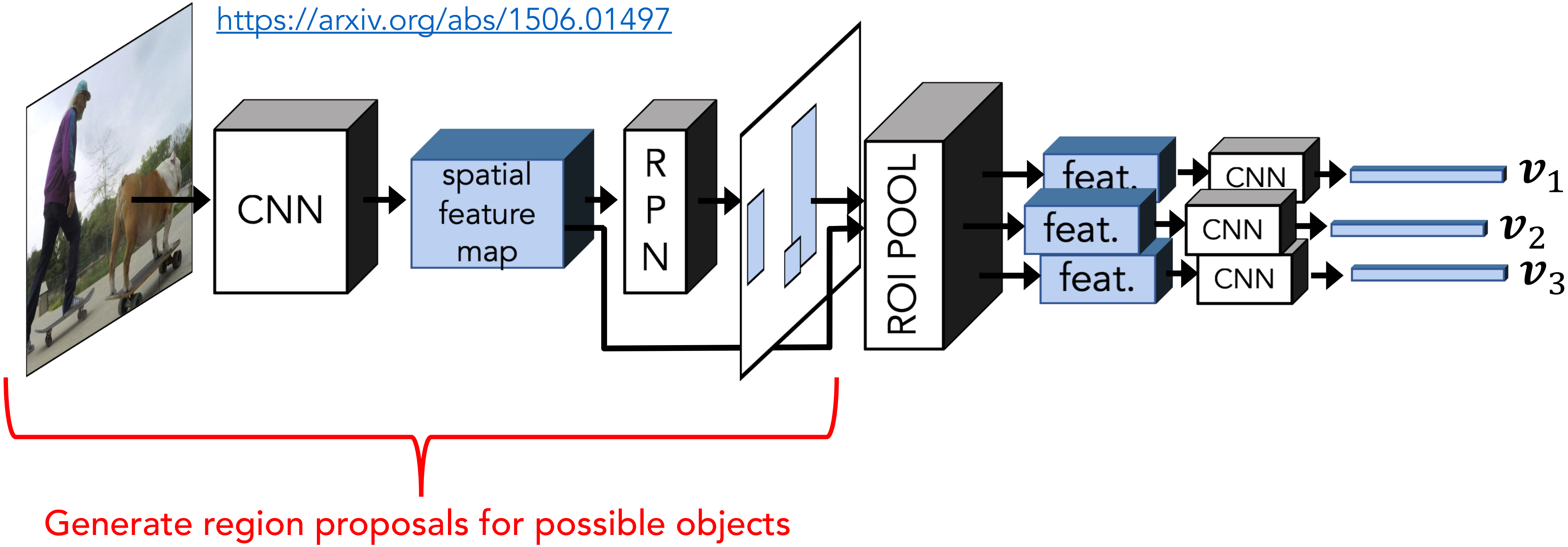
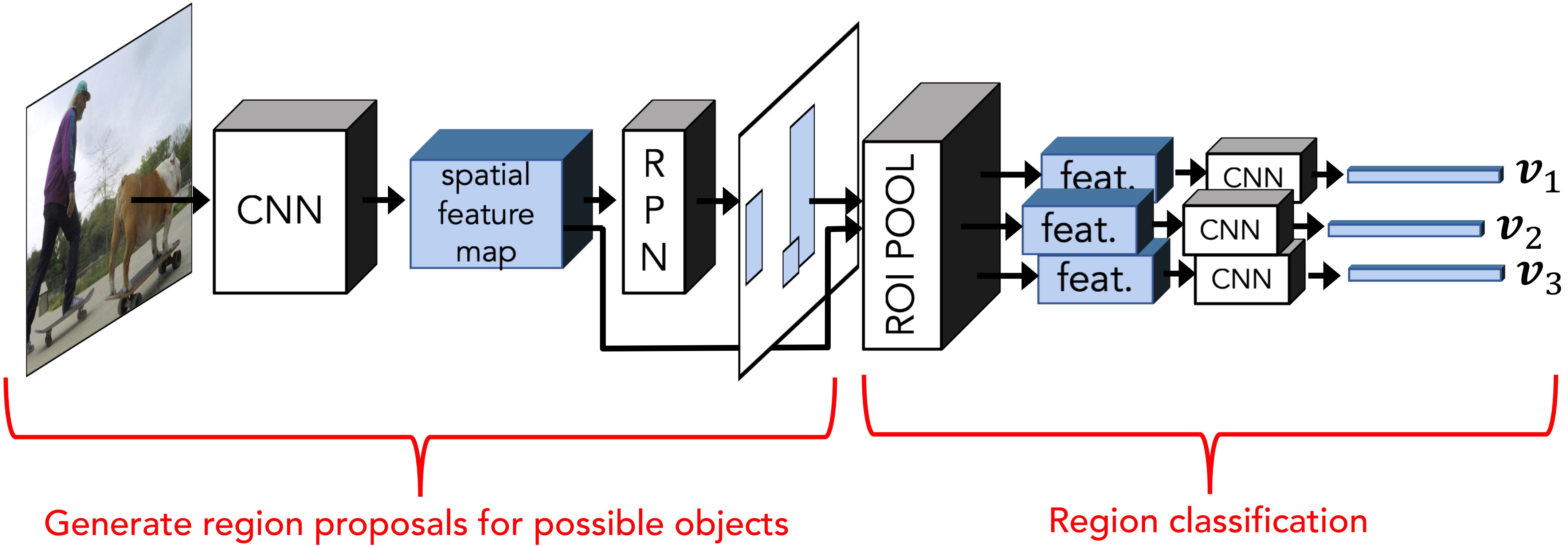


Image Credit: Peter Anderson Slide credit: Stefan Lee

Modelling Images

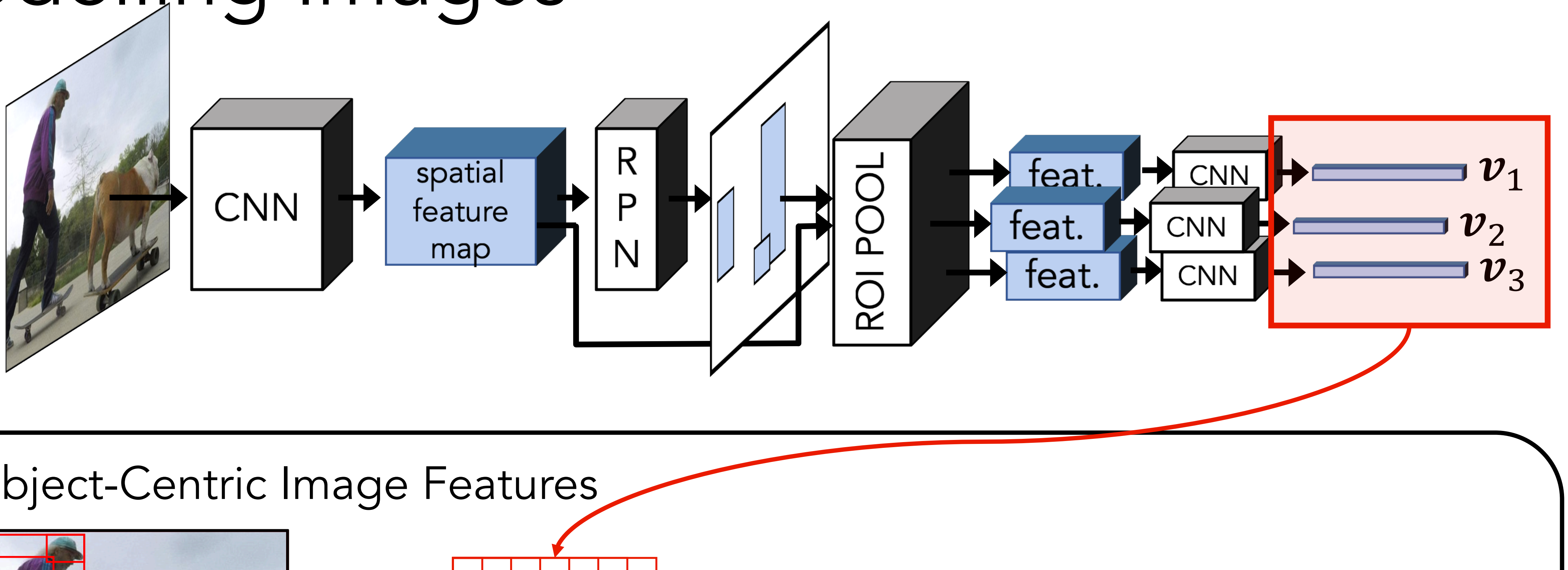
Idea: Switch to object detection models as the backbone for image representation

- Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering arxiv.org/abs/1707.07998

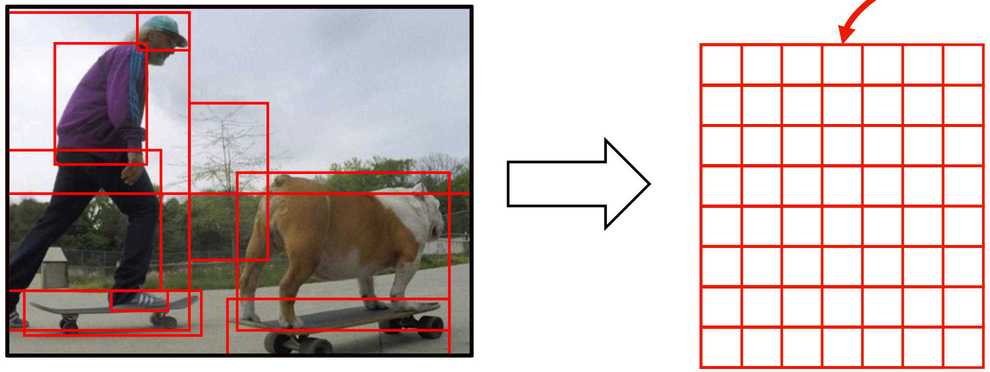


Slide credit: Stefan Lee

Modelling Images



Object-Centric Image Features



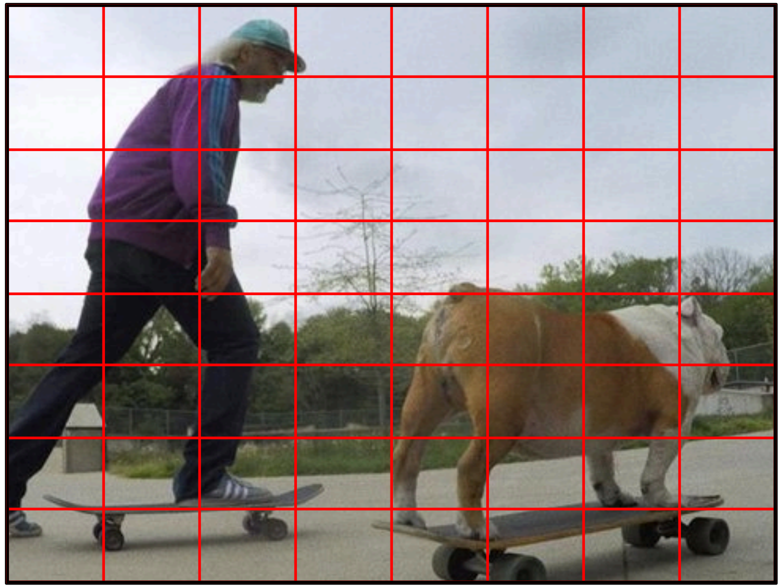
$$V \in \mathbb{R}^{k \times d}$$

- Feature vector per bounding box
- Spatial features can be added
- Object-centric

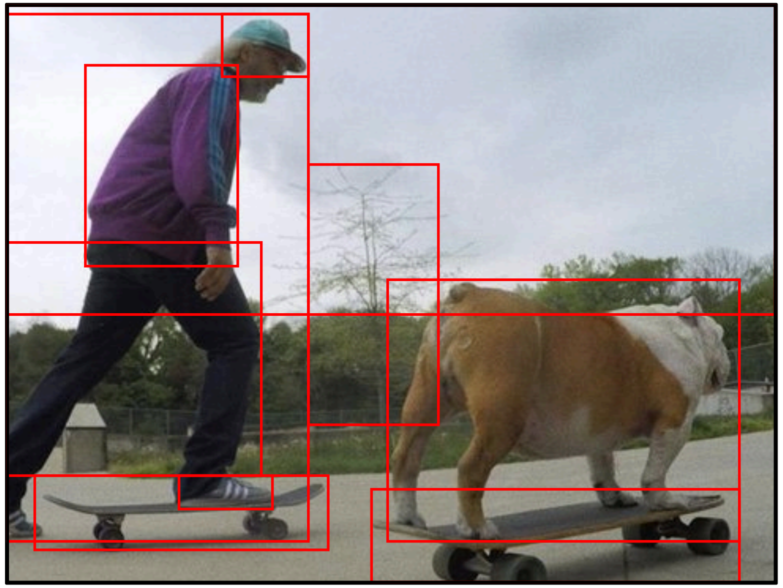
Modelling Images

Image Level Features

Spatial / Conv Features



Detection Features



ResNet 101
Trained on ImageNet

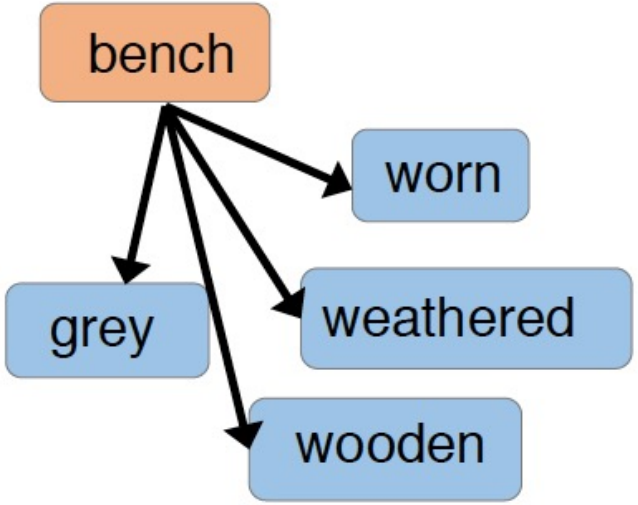
FasterRCNN - ResNet 101
Trained on Visual Genome

These are almost never fine-tuned for downstream tasks in vision-and-language.

Slide credit: Stefan Lee

Modelling Images: Pretraining

ResNet 101 Pre-training on ImageNet
- 1000 object classes (many fine-grained)



Faster R-CNN Pre-training on Visual Genome
- 1600 object classes
- 400 attribute classes

Slide Credit: Peter Anderson

Modelling Sequences

Modelling Sequences

Recurrent Neural Networks

- Ideal for processing sequential data containing possibly long-term dependencies.
- Various implementations (e.g. simple RNN, LSTM, GRU) expose the same API

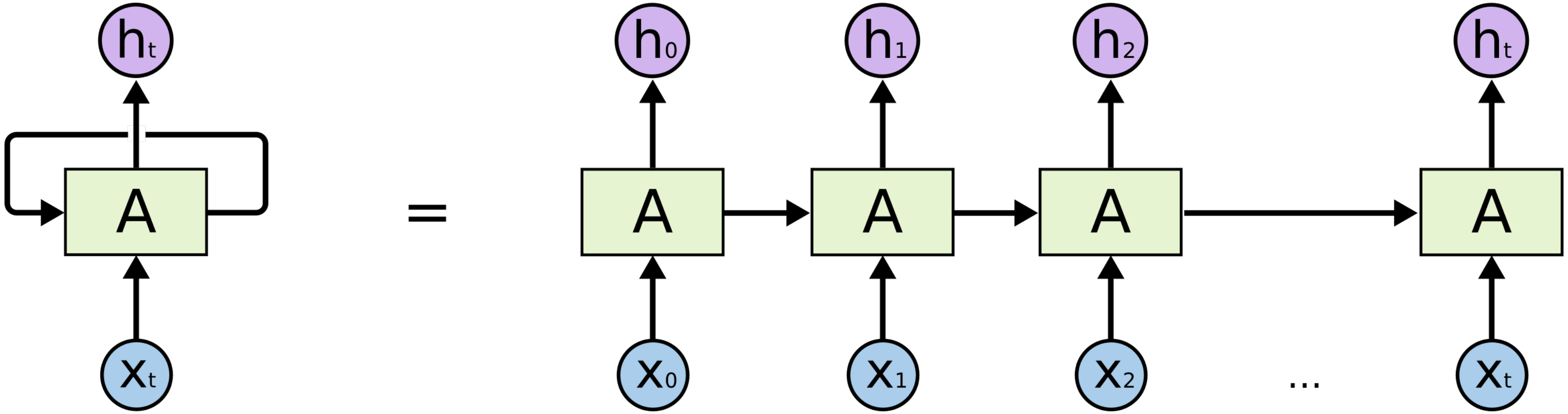


Image Credit: Christopher Olah

Modelling Sequences

Recurrent Neural Networks

- Ideal for processing sequential data containing possibly long-term dependencies.
- Various implementations (e.g. simple RNN, LSTM, GRU) expose the same API

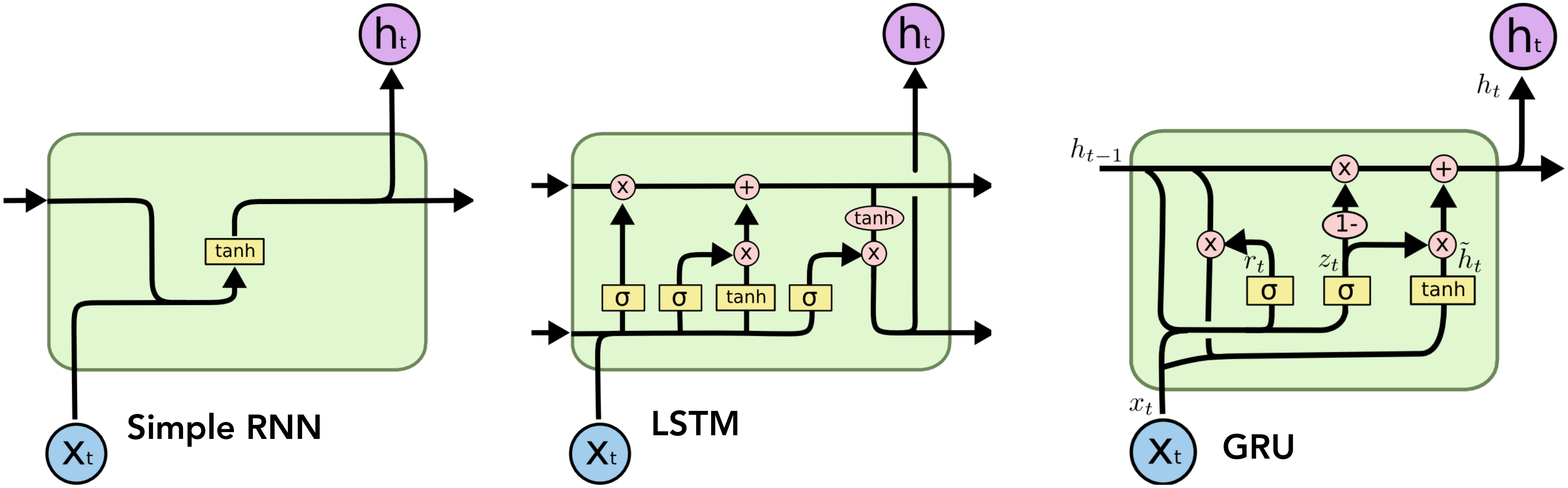
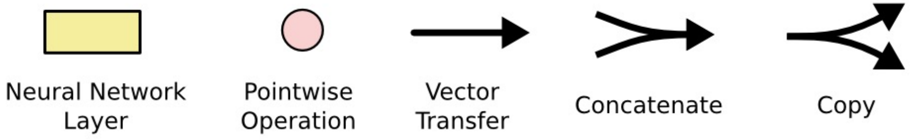


Image Credit: Christopher Olah

Slide credit: Stefan Lee



Modelling Sequences

Recurrent Neural Networks

- Ideal for processing sequential data containing possibly long-term dependencies.
- Various implementations (e.g. simple RNN, LSTM, GRU) expose the same API

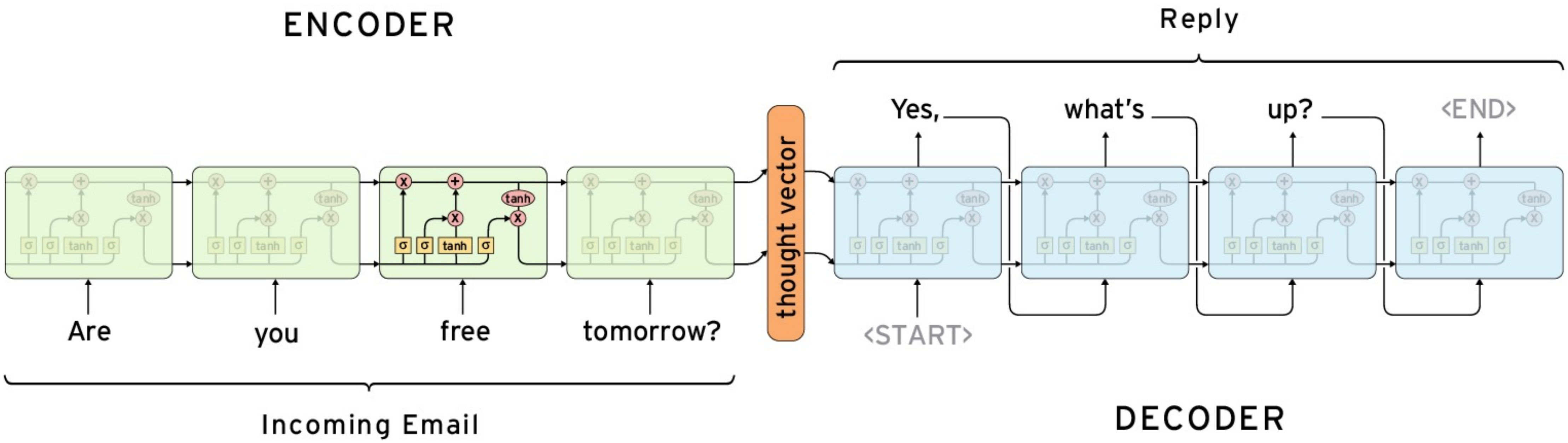


Image Credit: Christopher Olah

Multi-layer (stacked) RNNs

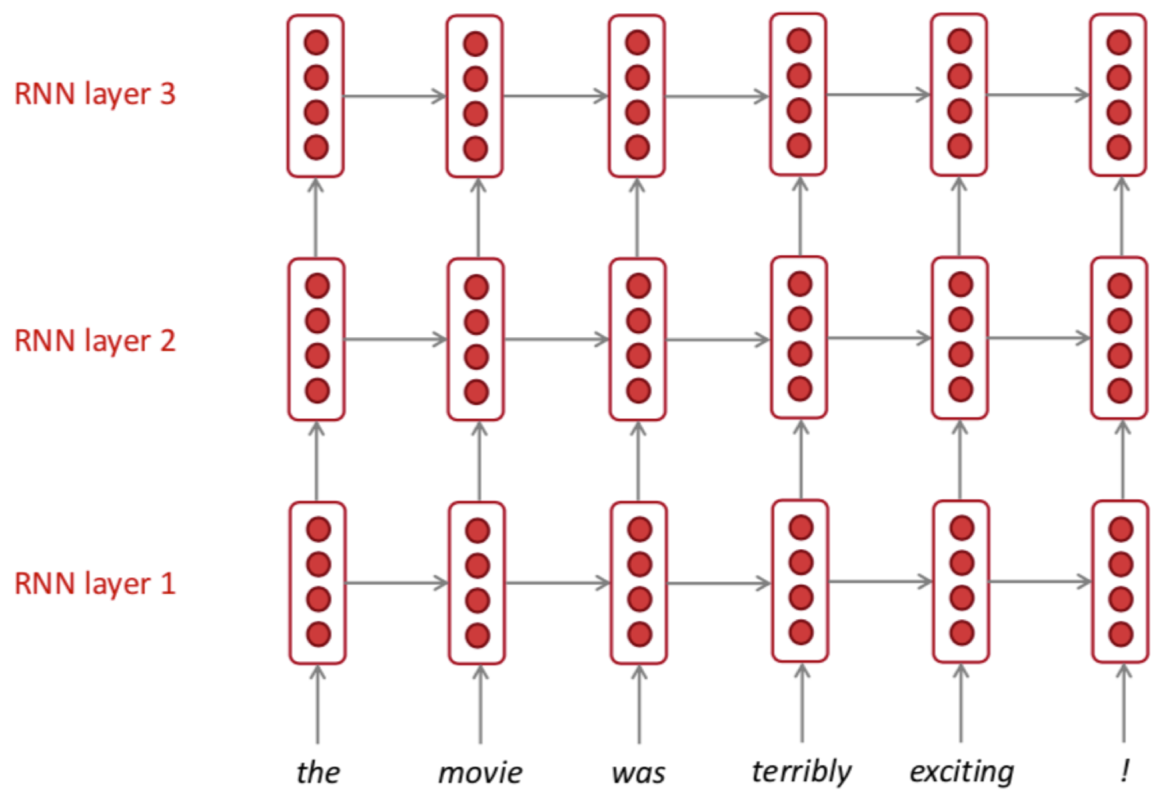


Image Credit: Abigail See

The hidden states from RNN layer i are the inputs to RNN layer $i + 1$

In practice, using 2 to 4 layers is common (usually better than 1 layer)

Transformer-based networks can be up to 24 layers with lots of skip-connections.

Bidirectional RNNs

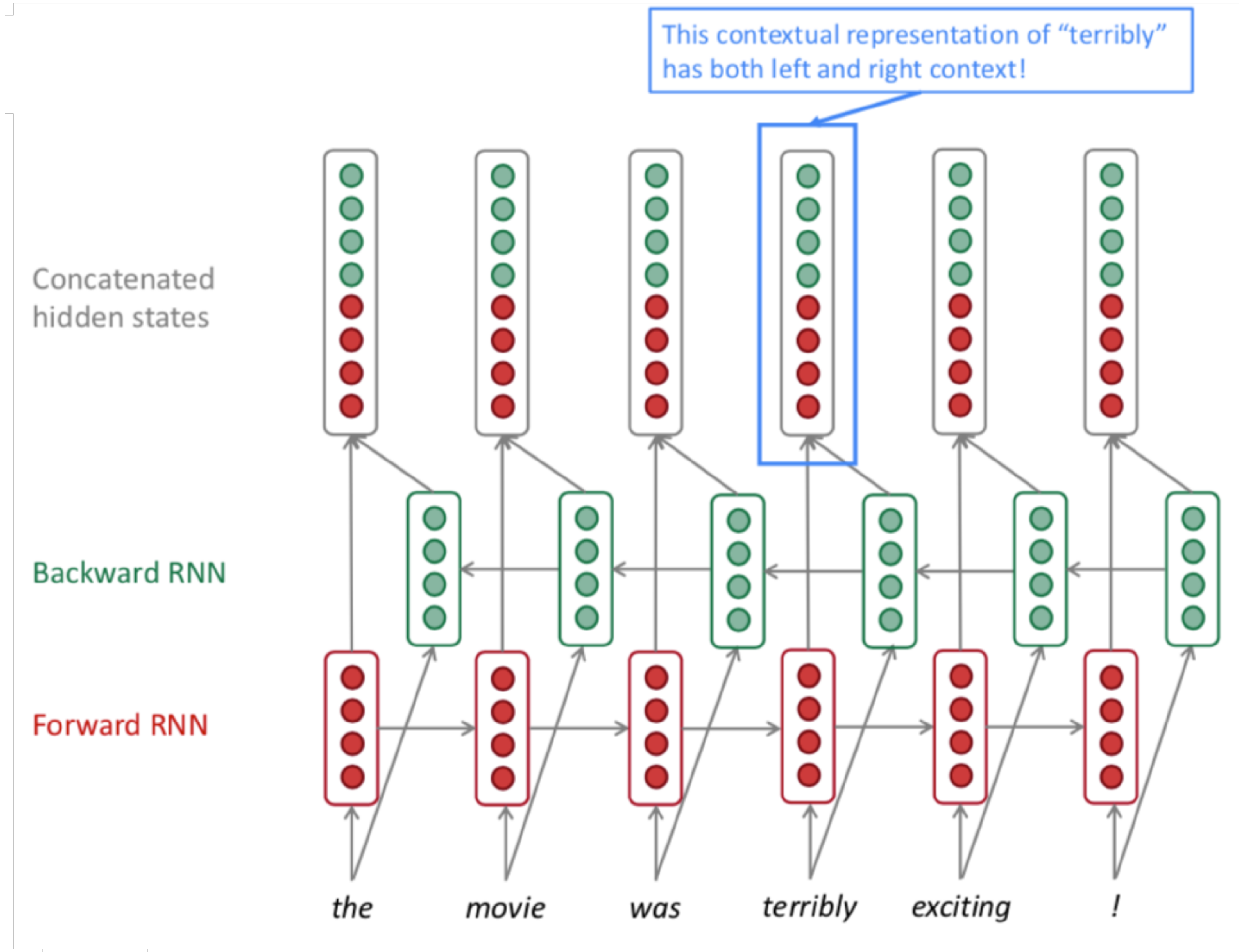


Image Credit: Abigail See

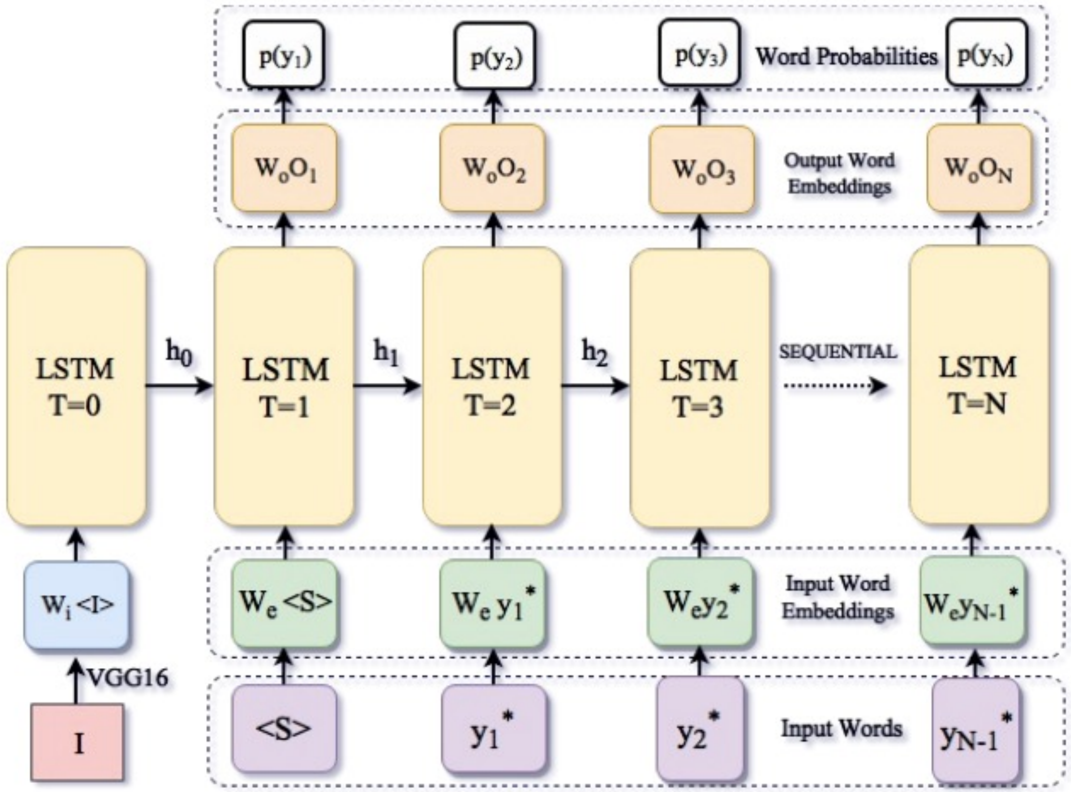
Incorporate information from both directions

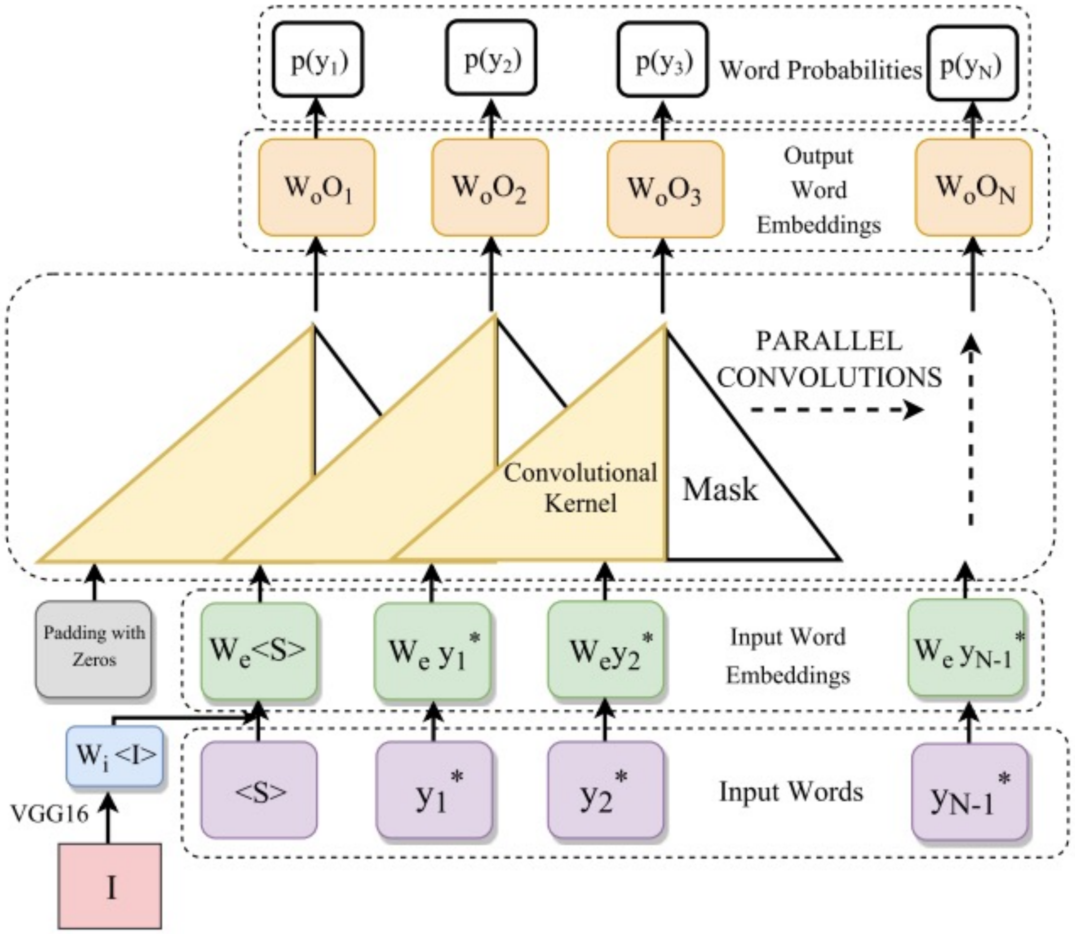
Useful in encoder

Modelling Sequences

CNNs as a fixed-time horizon alternative:

- Parallel computation!
- Tricky encoding.



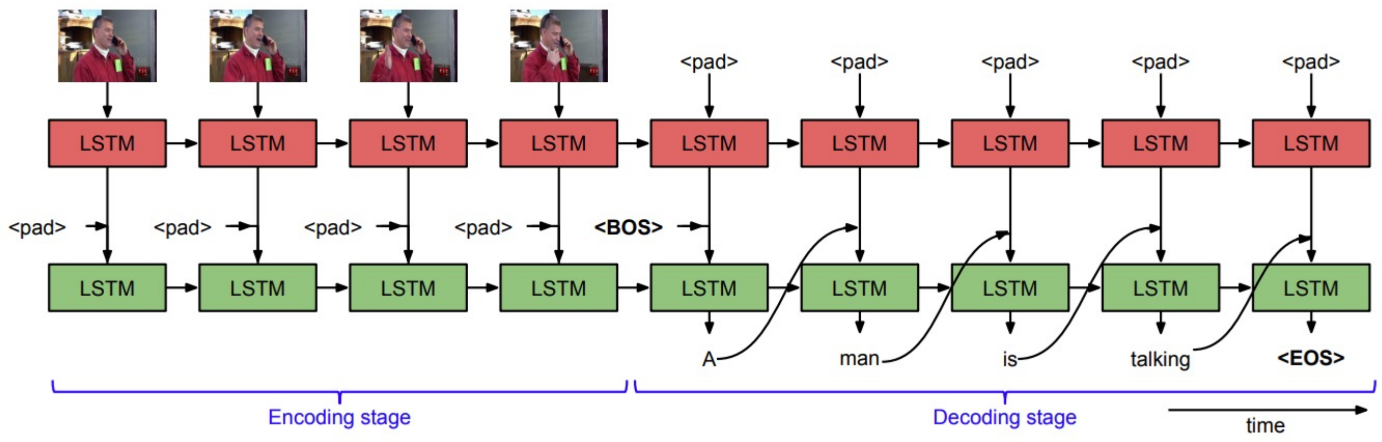


Aneja et al. CVPR 2018

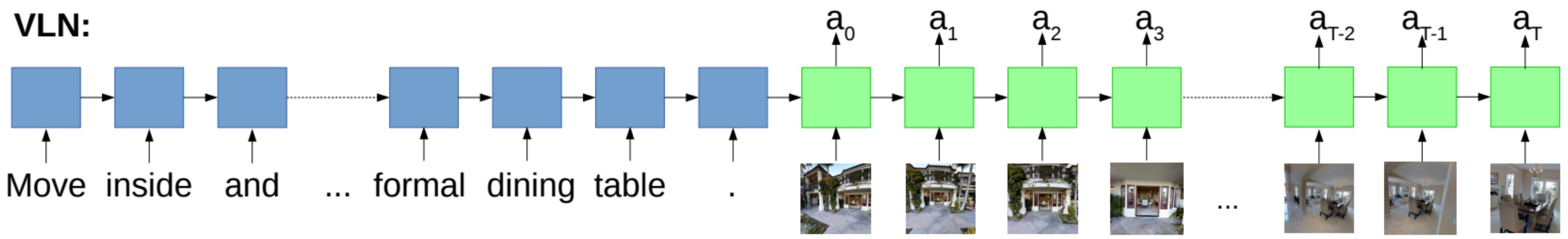
Slide credit: Stefan Lee

Multimodal seq2seq models

- Video captioning (video frames to text)

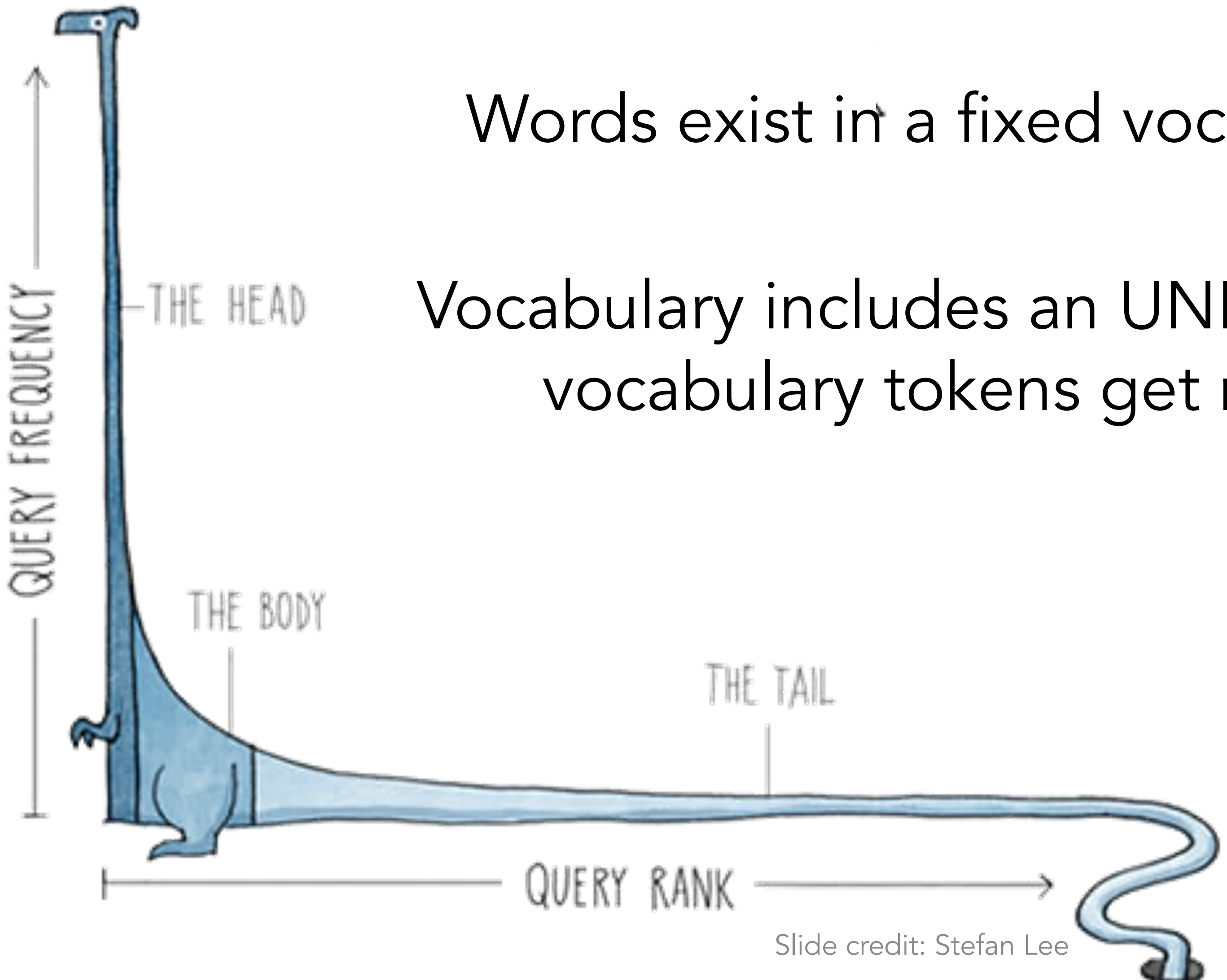


- Embodied AI (text + frames to actions)



Some Notes on Representing Text

Words and Vocabularies



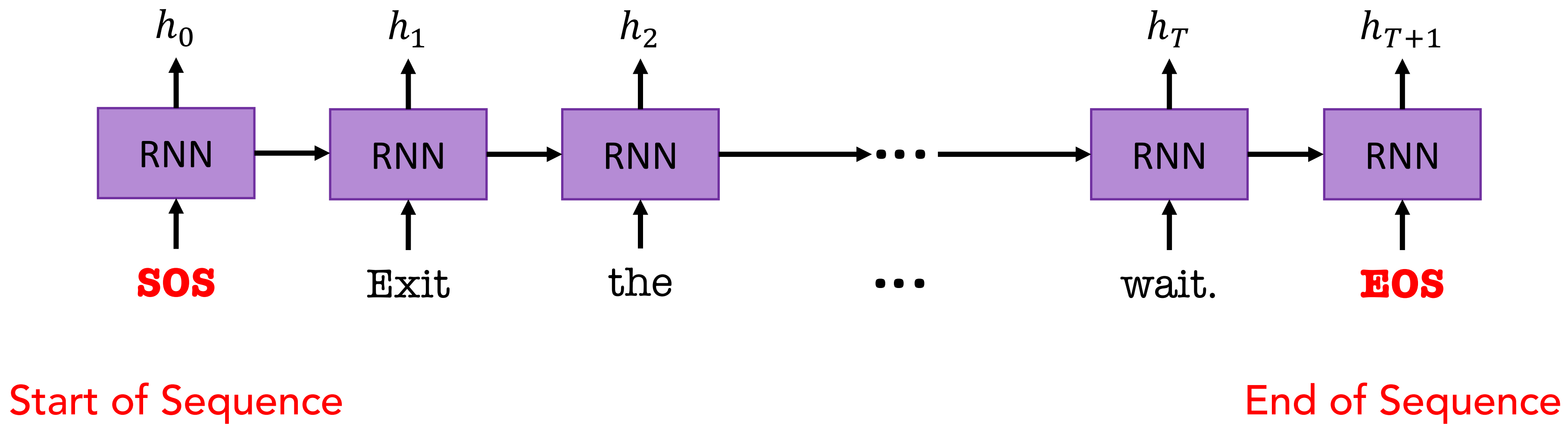
Words exist in a fixed vocabulary, i.e. $w \in V$

Vocabulary includes an UNK token – any out of vocabulary tokens get mapped to this.

Slide credit: Stefan Lee

Some Notes on Representing Text

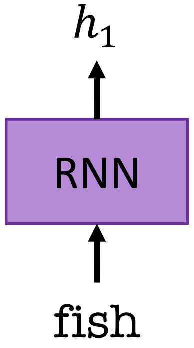
Quirks of Common Practice



Some Notes on Representing Text

What is actually input to represent tokens?

- One-hot vector → learned representation
  - For $V = \{cat, dog, fish\}$, $w_{fish} = [0\ 0\ 1]$.



$$\begin{array}{cc} \text{fish} & \mathbf{W} \\ [0\ 0\ 1] & \begin{bmatrix} 0.2 & 1 & 1.5 & 0.8 & -0.2 & 1.2 \\ -1.3 & 2 & -2 & 1.2 & 0.56 & 0.1 \\ 0.13 & 0.2 & 0.95 & 0.2 & -1.3 & 0.5 \end{bmatrix} \end{array}$$

$$w_{fish} * W = [0.13\ \ 0.2\ \ 0.95\ \ 0.2\ \ -1.3\ \ 0.5]$$

Initialize to random vectors and learn the embeddings during training

Some Notes on Representing Text

What is actually input to represent tokens?

dog
cat

fish

- Use pretrained word embeddings
  - Word2Vec
  - GloVE

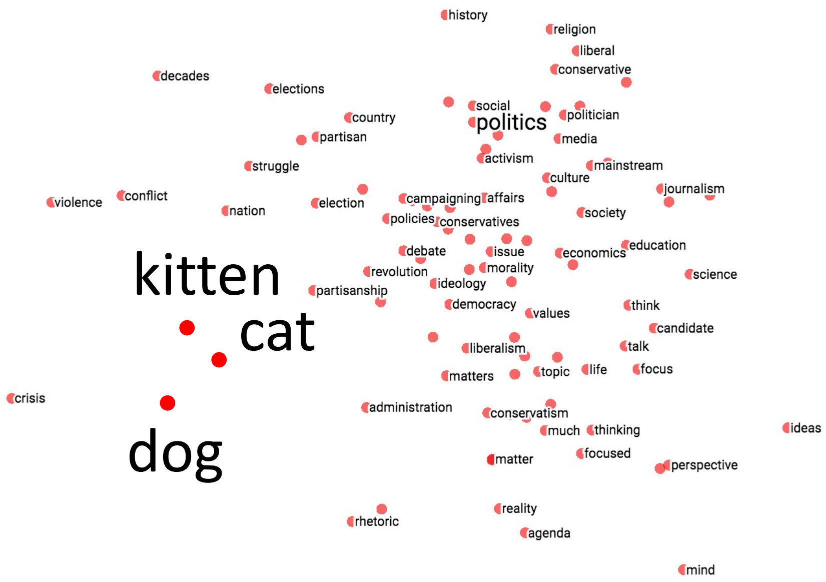
$$w_{fish} = GloVE(\text{“}fish\text{”})$$

- Can do a mix of these
  - initialize learned embeddings with pretrained values

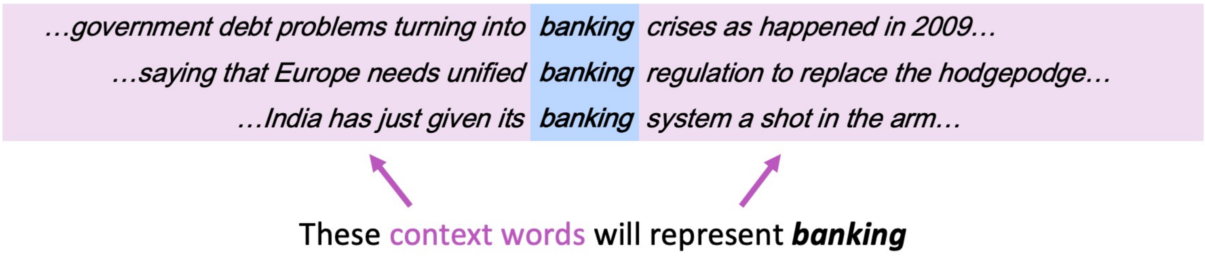
Slide credit: Stefan Lee

Some Note on Terminology

Distributed representation
- Meaning is not localized to one dimension



Distributional representation
- Meaning is learned from the context (other words) that co-occur with each word



"You shall know a word by the company it keeps"
(J.R. Firth 1957) - Distributional hypothesis

Contextual representation
- Representation changes based on context

Attention

Need for "attention"

- Uses encoding of entire input when generating each output token
- Maybe would be useful to focus on a part of the input as the output tokens are generated

Translation

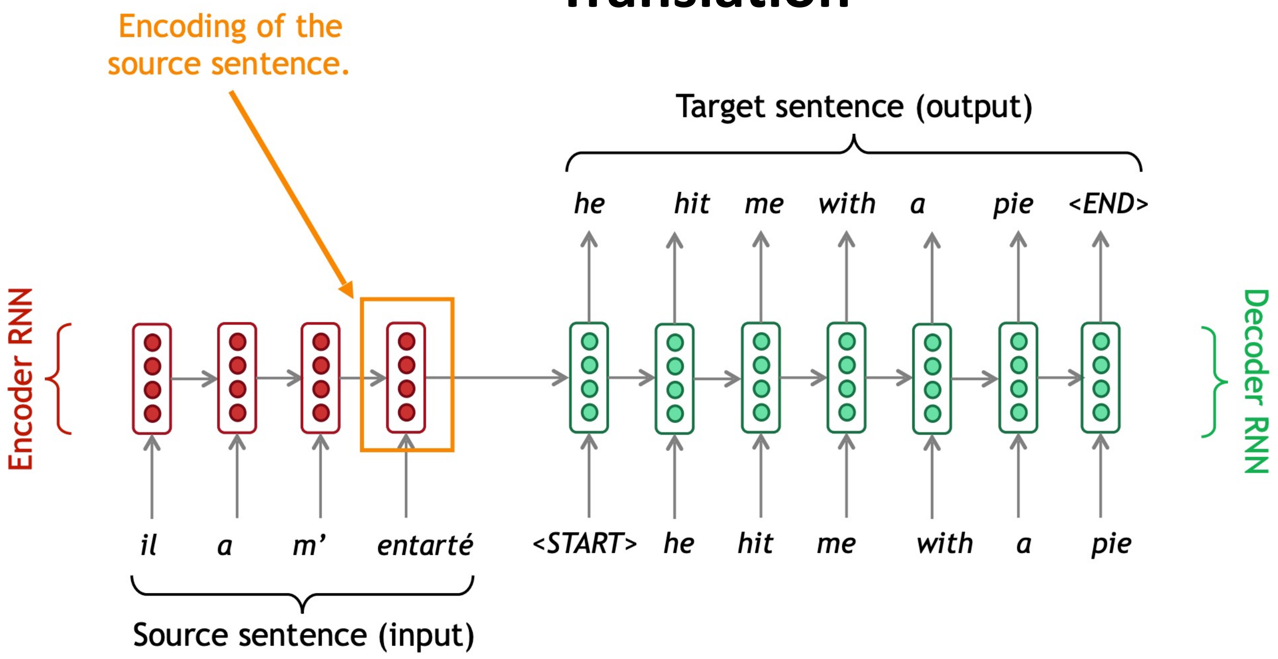


Image credit: Abigail See

Captioning

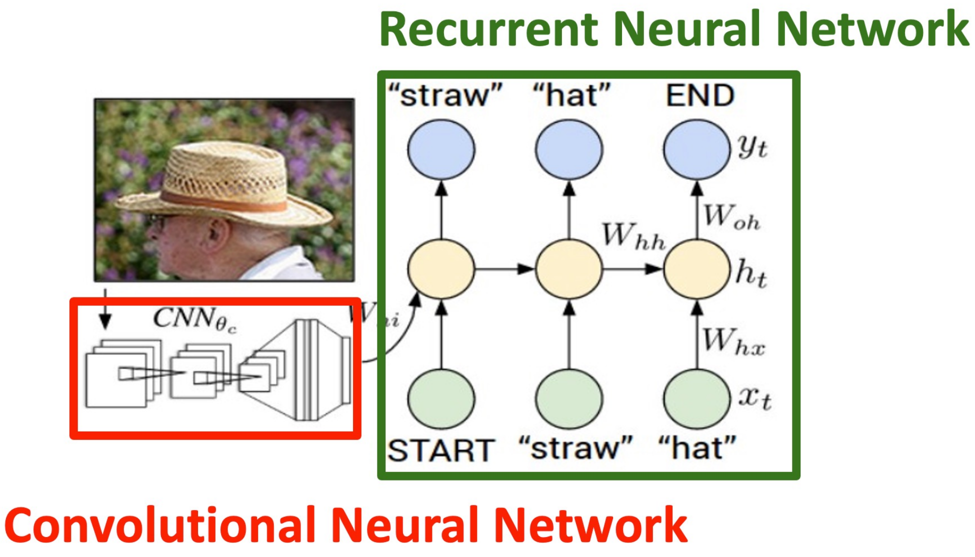


Image credit: Andrej Karpathy

Attention for VLN

Not every part of an input is important given the task context

Exit the bathroom. Turn left and exit the room using the door on the left. Wait there.

Slide credit: Stefan Lee

Attention for VLN

Not every part of an input is important given the task context

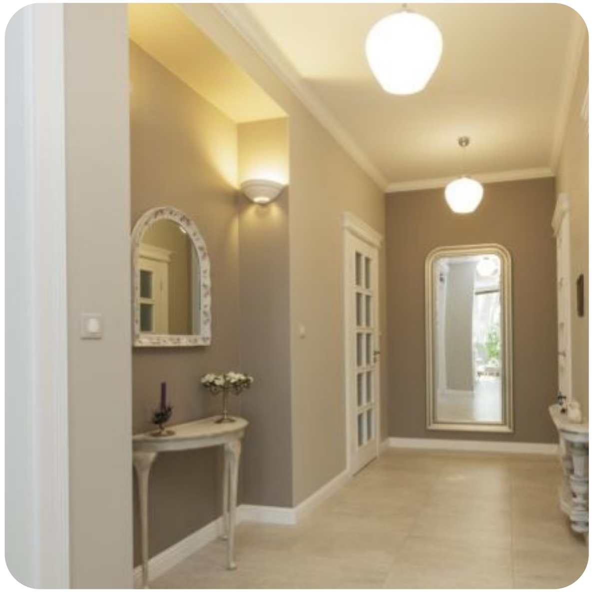
Exit the bathroom. Turn left and exit the room using the door on the left. Wait there.

Slide credit: Stefan Lee

Attention for VLN

Not every part of an input is important given the task context

Exit the bathroom. Turn left and exit the room using the door on the left. Wait there.



Slide credit: Stefan Lee

Attention for VLN

Not every part of an input is important given the task context

Take a right when you see the mirrored wardrobe.

Slide credit: Stefan Lee

Attention for VQA

Not every part of an input is important given the task context

What shape is the doormat?

Slide credit: Stefan Lee

Attention

- The concept of 'attention' has seen widespread use…
  - In many language and / or vision tasks, attention works extremely well!
  - Attention improves interpretability of neural networks

Q: What room are they in?

A: kitchen

Focus on part of input

Slides Credit: Peter Anderson

Attention mechanisms - summary

- Attention is probably one of the simplest and most effective ideas in deep learning – proven across many different domains

- Practically: focus on part of input by taking a weighted sum of different input parts

- With sufficient data, attention mechanisms can learn to ground language in visual content from ‘distant’ supervision

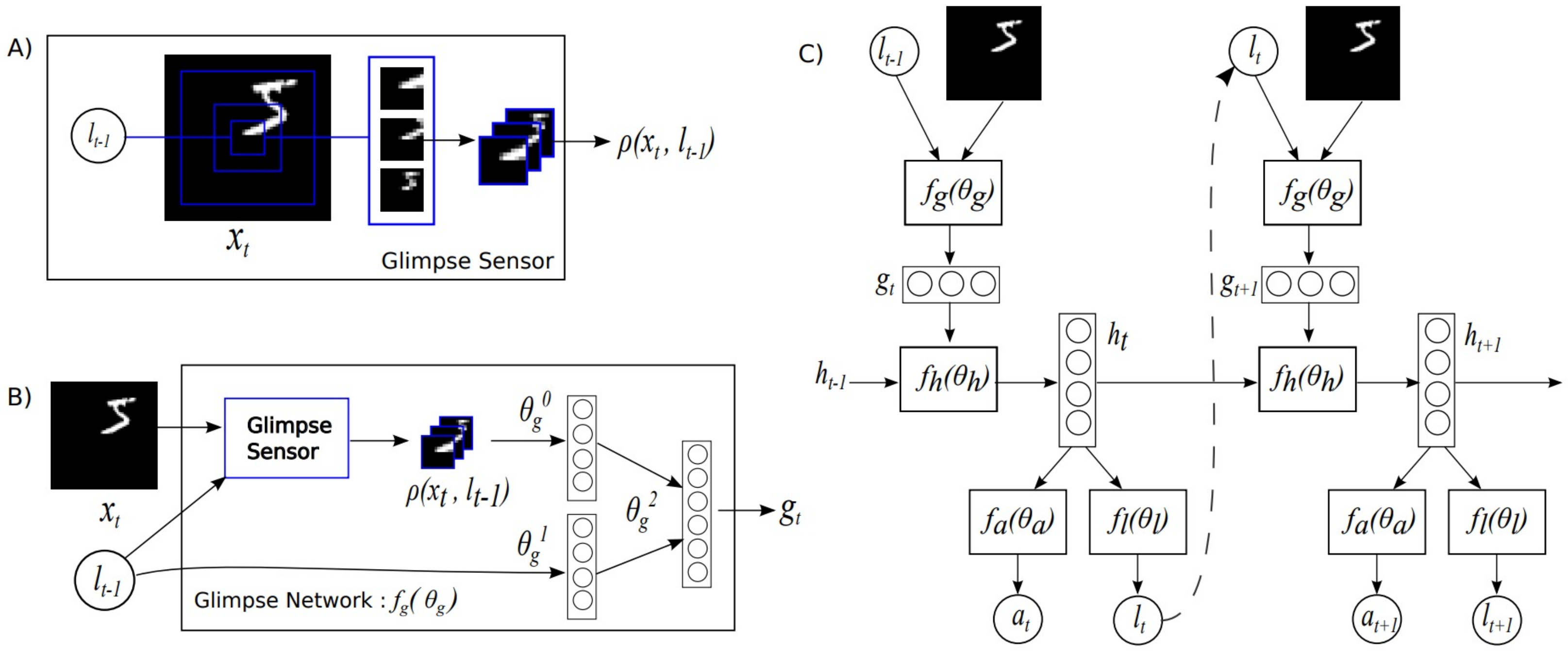
- Given the complexity of biological attention systems, assume there is still much to explore… particularly temporal aspects in context of LV&A

Slides Credit: Peter Anderson

Attention

- Major impact on computer vision and NLP from 2014/5

Recurrent Models of Visual Attention



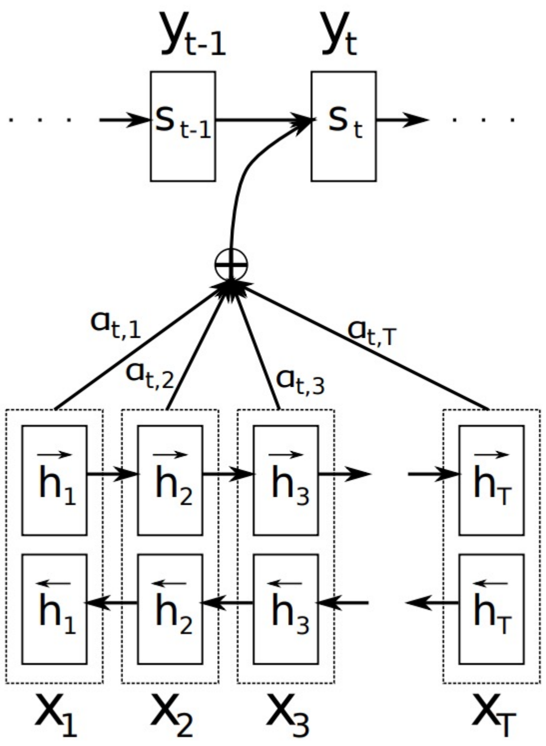
Mnih et al. NIPS 2014

Slides Credit: Peter Anderson

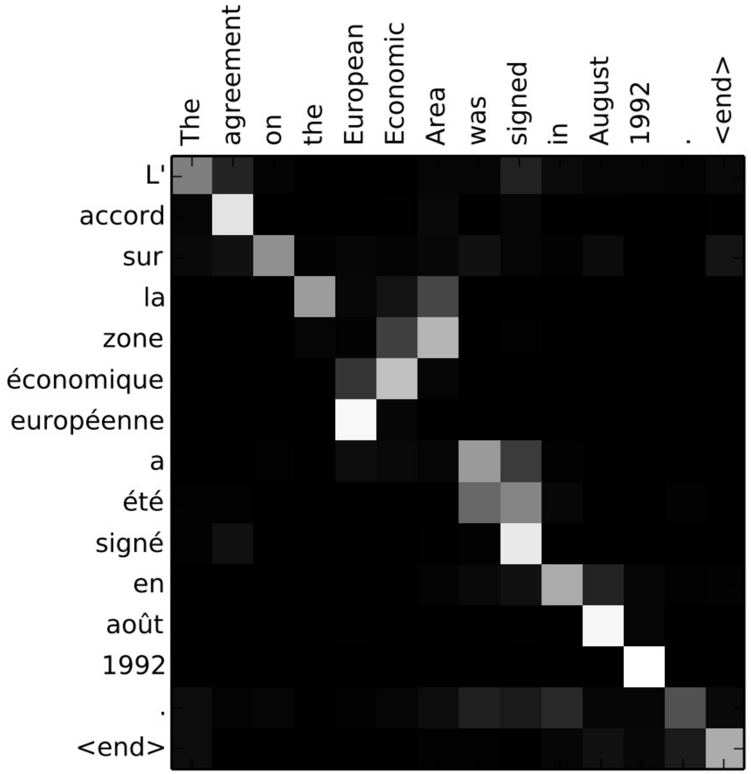
Attention

- Major impact on computer vision and NLP from 2014/5

Neural Machine Translation by Jointly Learning to Align and Translate



Bahdanau et al. ICLR 2015



generated (French)

source (English)

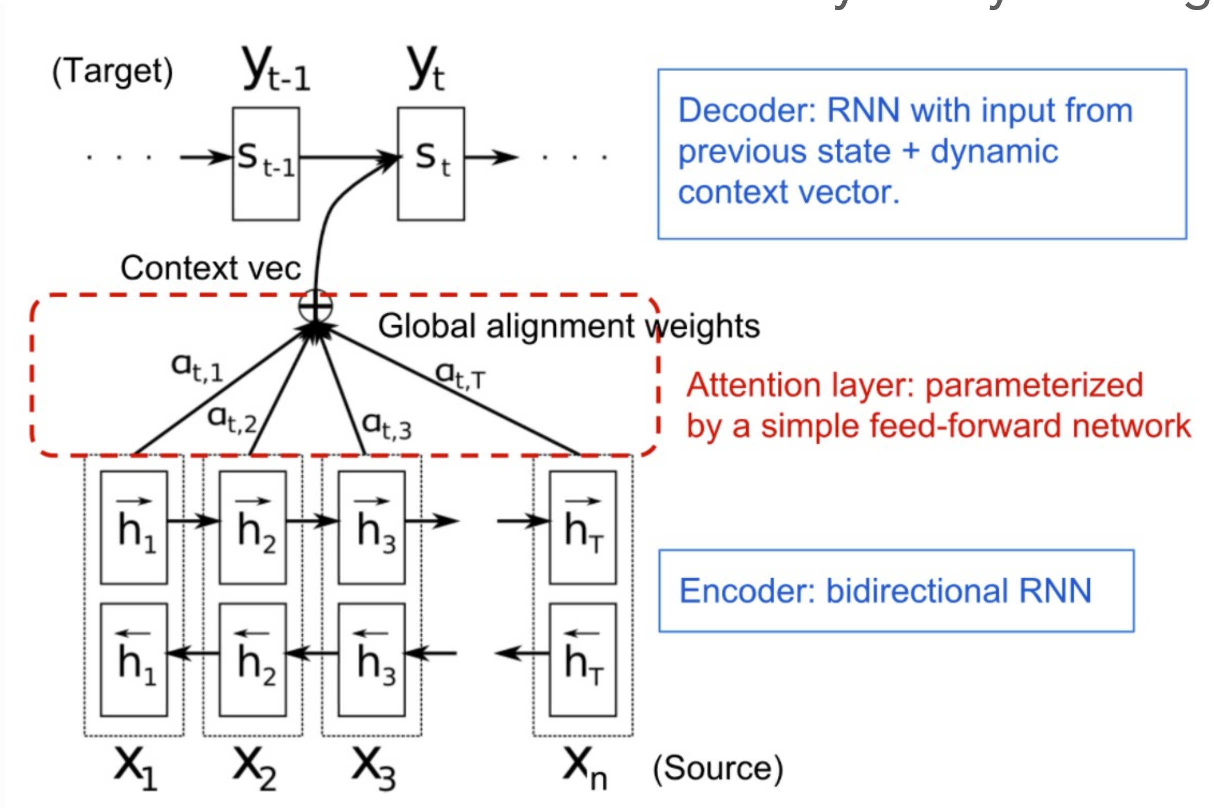
Attention weights α_i (0 = black, 1 = white)

Slides Credit: Peter Anderson

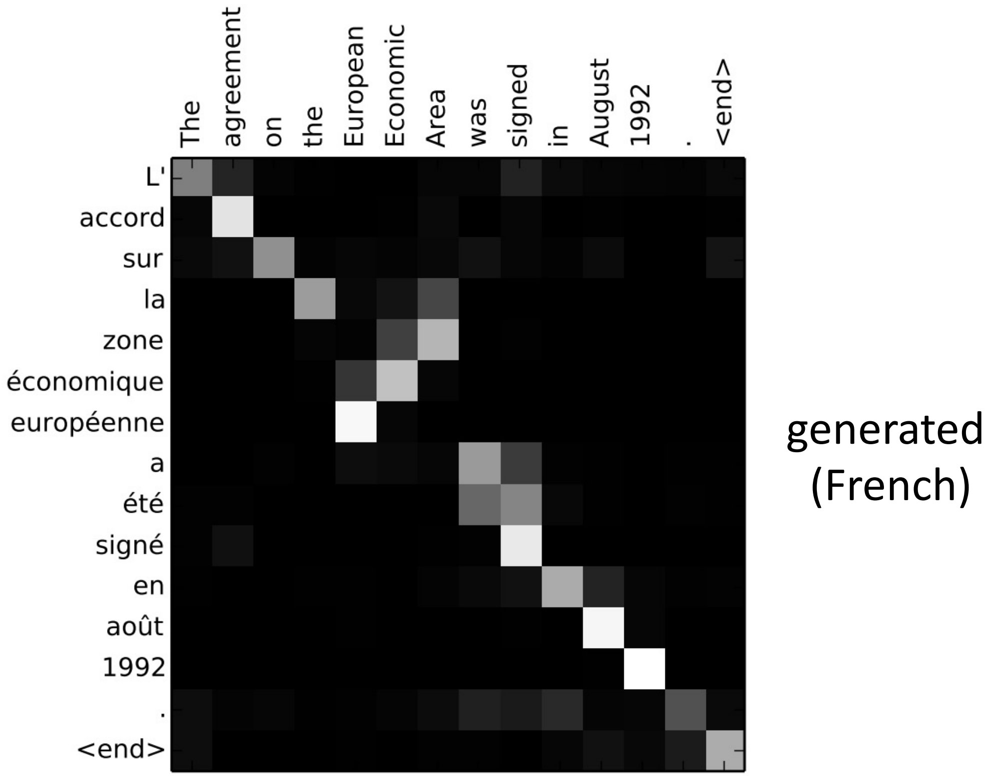
Attention

- Major impact on computer vision and NLP from 2014/5

Neural Machine Translation by Jointly Learning to Align and Translate



Bahdanau et al. ICLR 2015



Attention weights α_i (0 = black, 1 = white)

Slides Credit: Peter Anderson

Attention

Attention in Neural Networks:

A learned mechanism that **learns to focus** on a subset of the **input** that is most **relevant to the current task**.

task context representation

(also called query)

attended feature $\longrightarrow$ $\hat{\boldsymbol{c}} = f(\boldsymbol{z}, C)$

learned attention function (neural net)

set of attention candidates

Slides Credit: Peter Anderson

Computing attention

Attention function, f

$$a_i = g(\boldsymbol{c}_i, \boldsymbol{z})$$
$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$
$$\hat{\boldsymbol{c}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{c}_i$$

Attention scores: $\boldsymbol{a}$ (unnormalized)

Attention weights: $\boldsymbol{\alpha}$ (normalized)

Final attention output

Weighted sum of context features

Attention score $a_i = g(\boldsymbol{c}_i, \boldsymbol{z})$
how well does the attention candidate $\boldsymbol{c}_i$ match the query $\boldsymbol{z}$

- Dot-product attention:

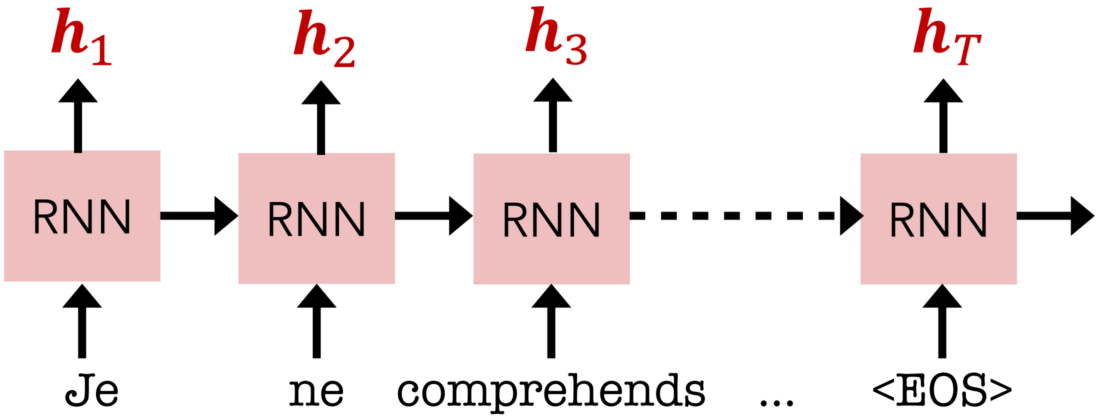
$$g(c_i, z) = z^\top c_i$$

- Neural network

$$g(c_i, z) = v^\top \tanh\left(W_1 c_i + W_2 z\right)$$

Attention over Text

Attention candidates, C typically defined by the hidden states of an encoder (e.g. one feature vector for each word in the input text)

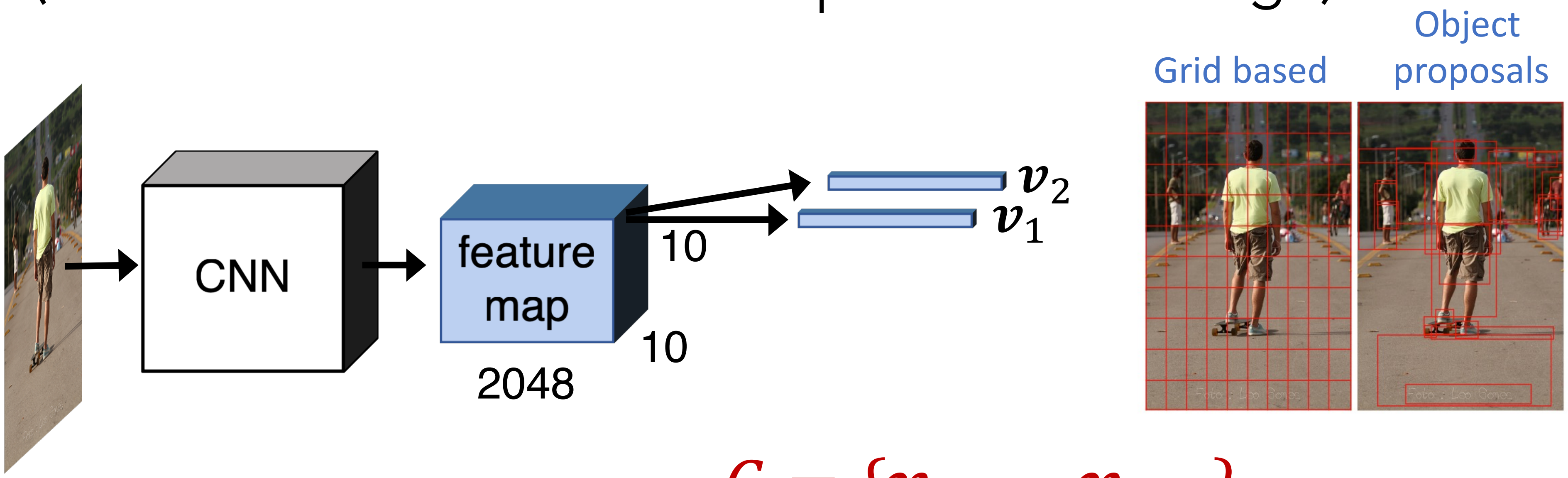


$$C = \{\boldsymbol{h}_1, \dots, \boldsymbol{h}_T\}$$

Slides Credit: Peter Anderson

Attention over Visual Features

- Attention candidates, $\boldsymbol{C}$ typically defined by the spatial output of a CNN (feature vectors for different parts of the image)

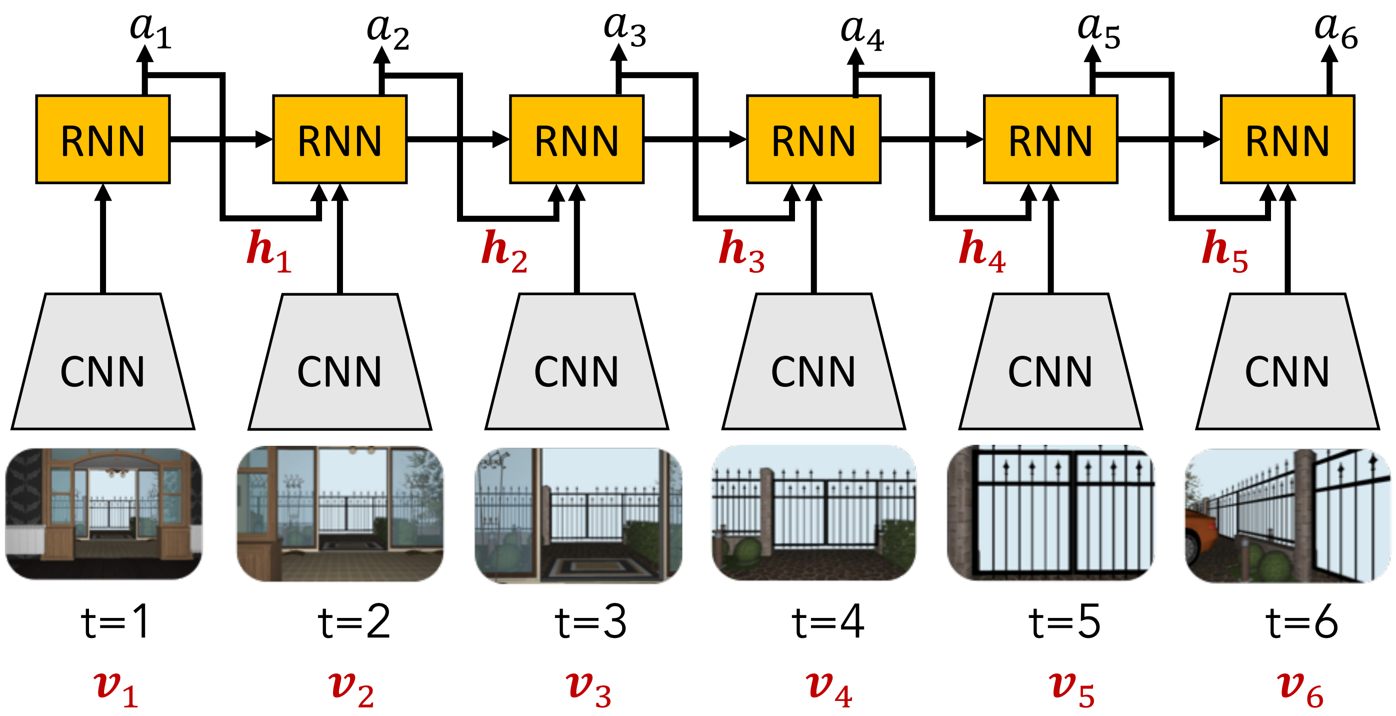


$$C = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{100}\}$$

Attention over Agent Experience

Embodied AI (visual language navigation)

- Attention candidates, $\boldsymbol{C}$ as agent hidden state or visual vectors



$$C = \{\boldsymbol{h}_1, \dots, \boldsymbol{h}_5\}$$

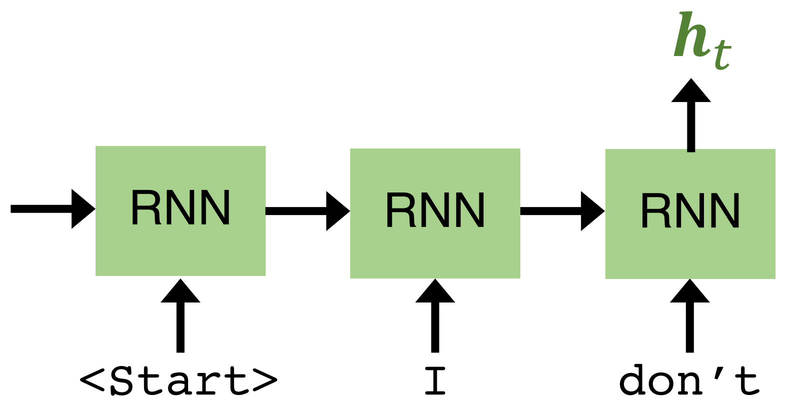
or

$$C = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_6\}$$

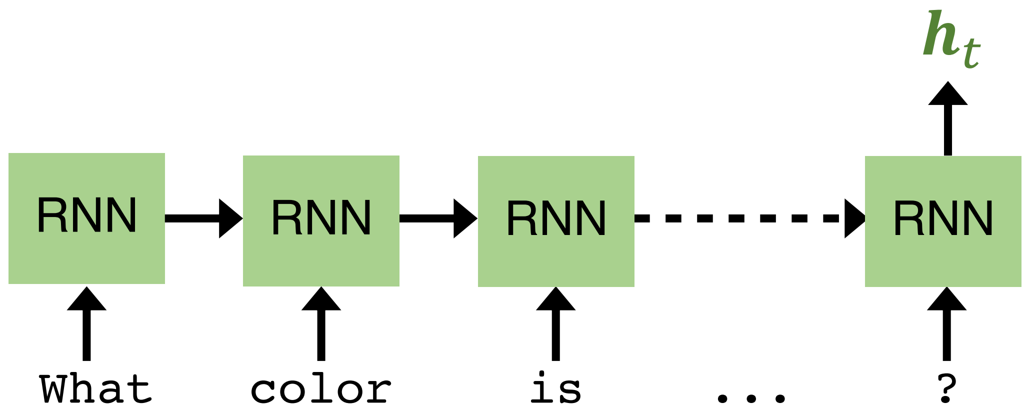
Task context for Attention

- Task context representation, z, is often an RNN encoding

Machine translation / image captioning:
Decoder hidden state



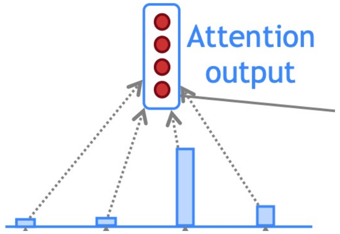
VQA: Question encoding
(final encoder hidden state)



Slides Credit: Peter Anderson

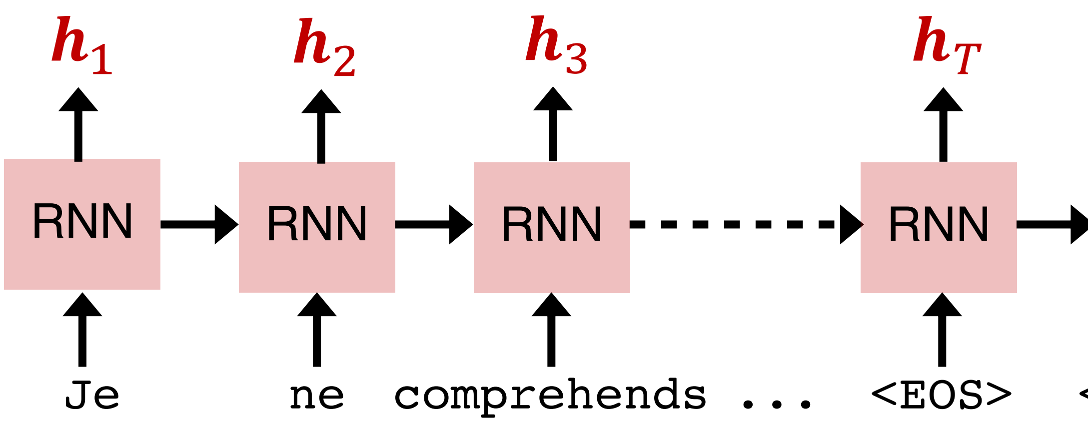
Putting it all together

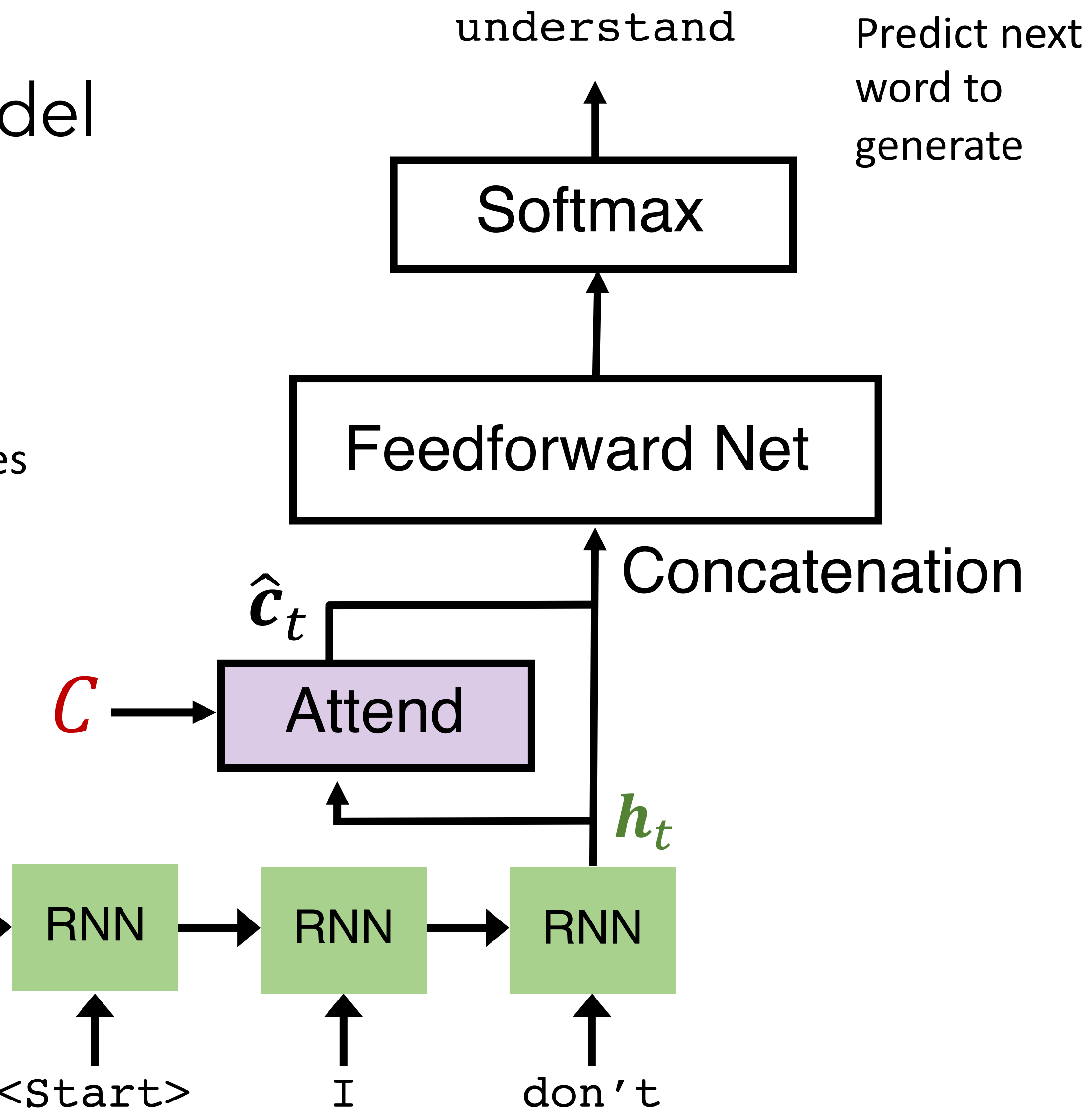
- Attentive machine translation model



$\hat{c}_t$ is the weighted sum of encoder hidden states

$$C = \{c_1, \ldots, c_T\}$$

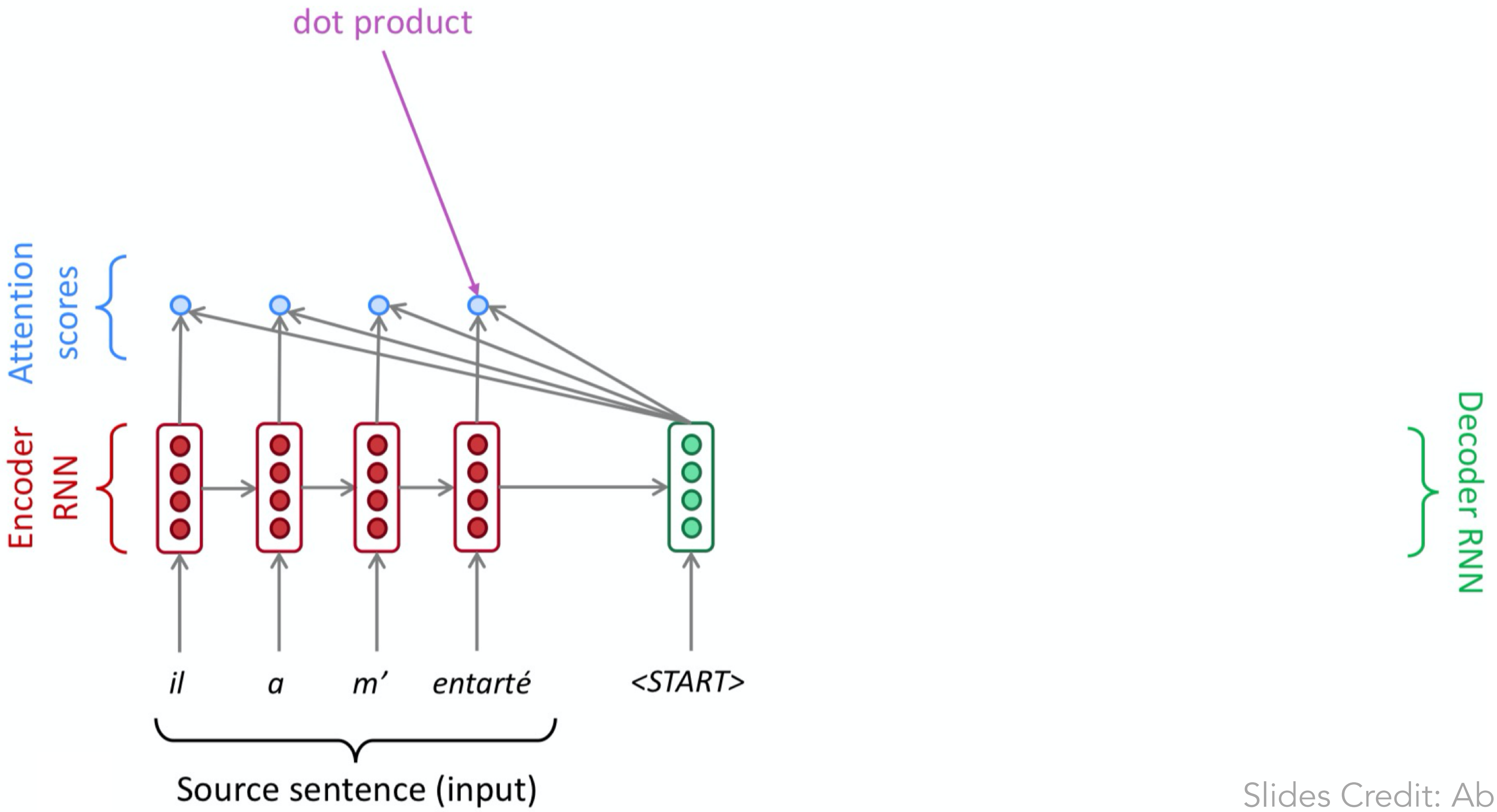




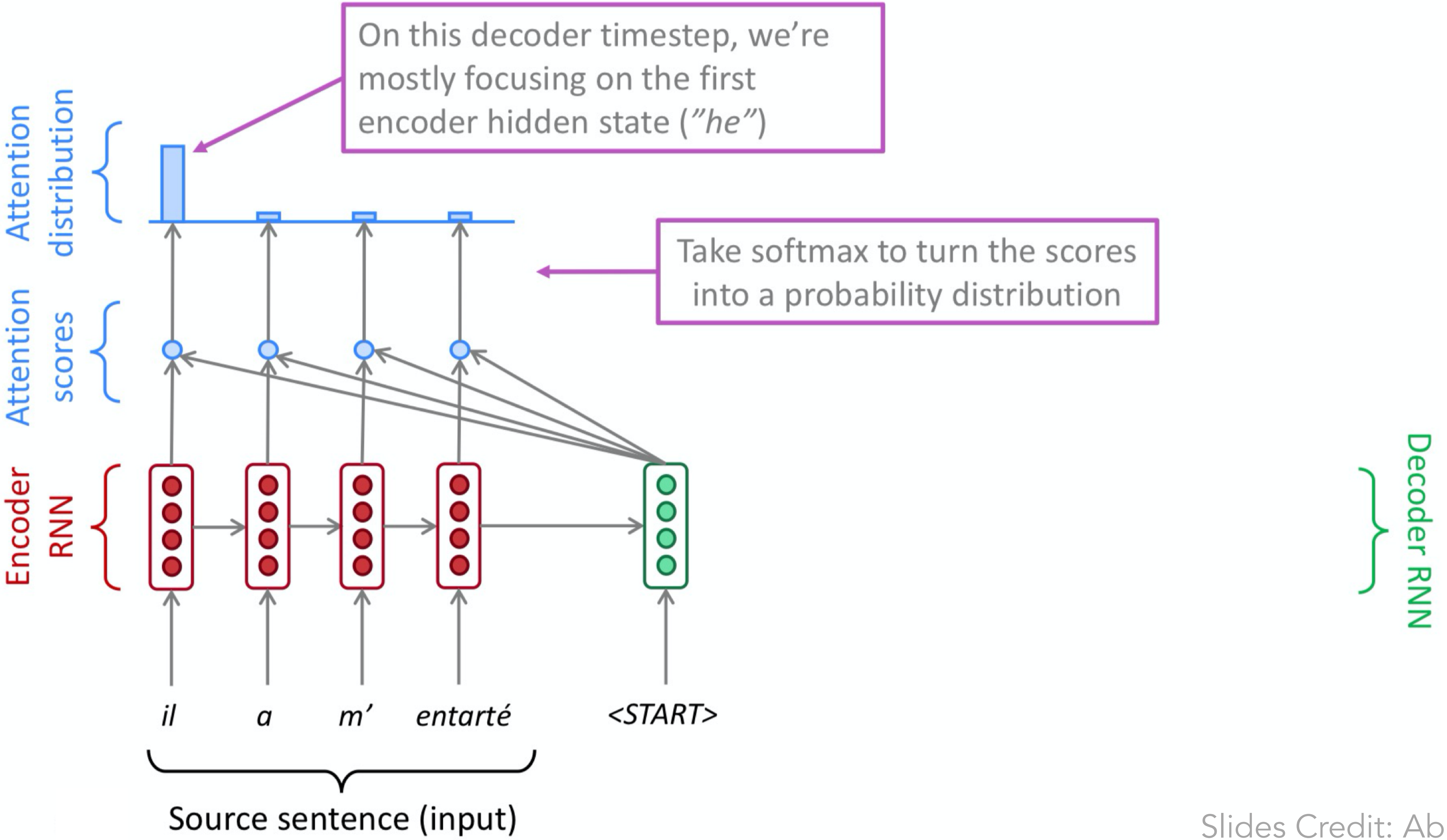
Slides Credit: Peter Anderson

Interlude: Attention for machine translation

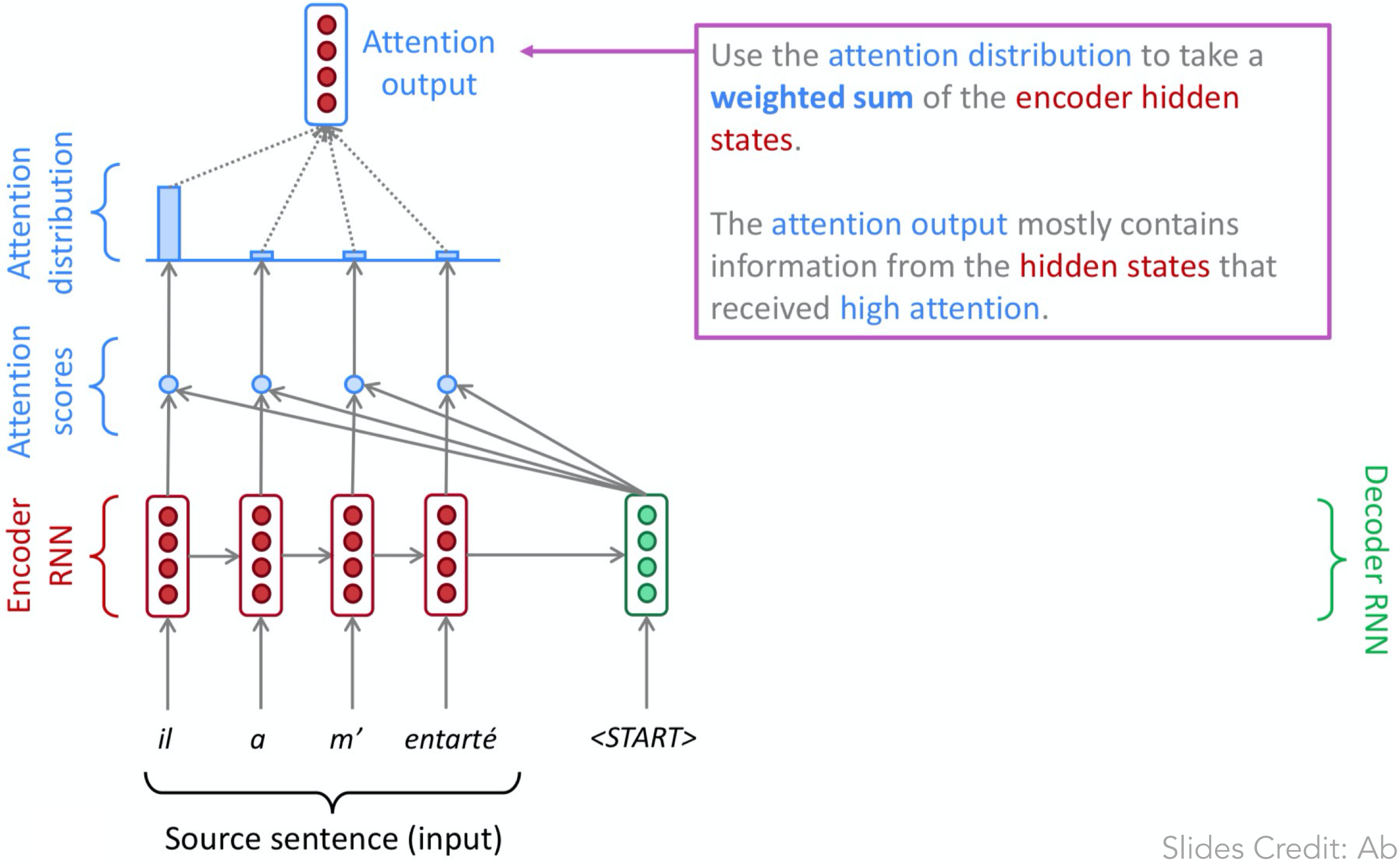
Attention for machine translation



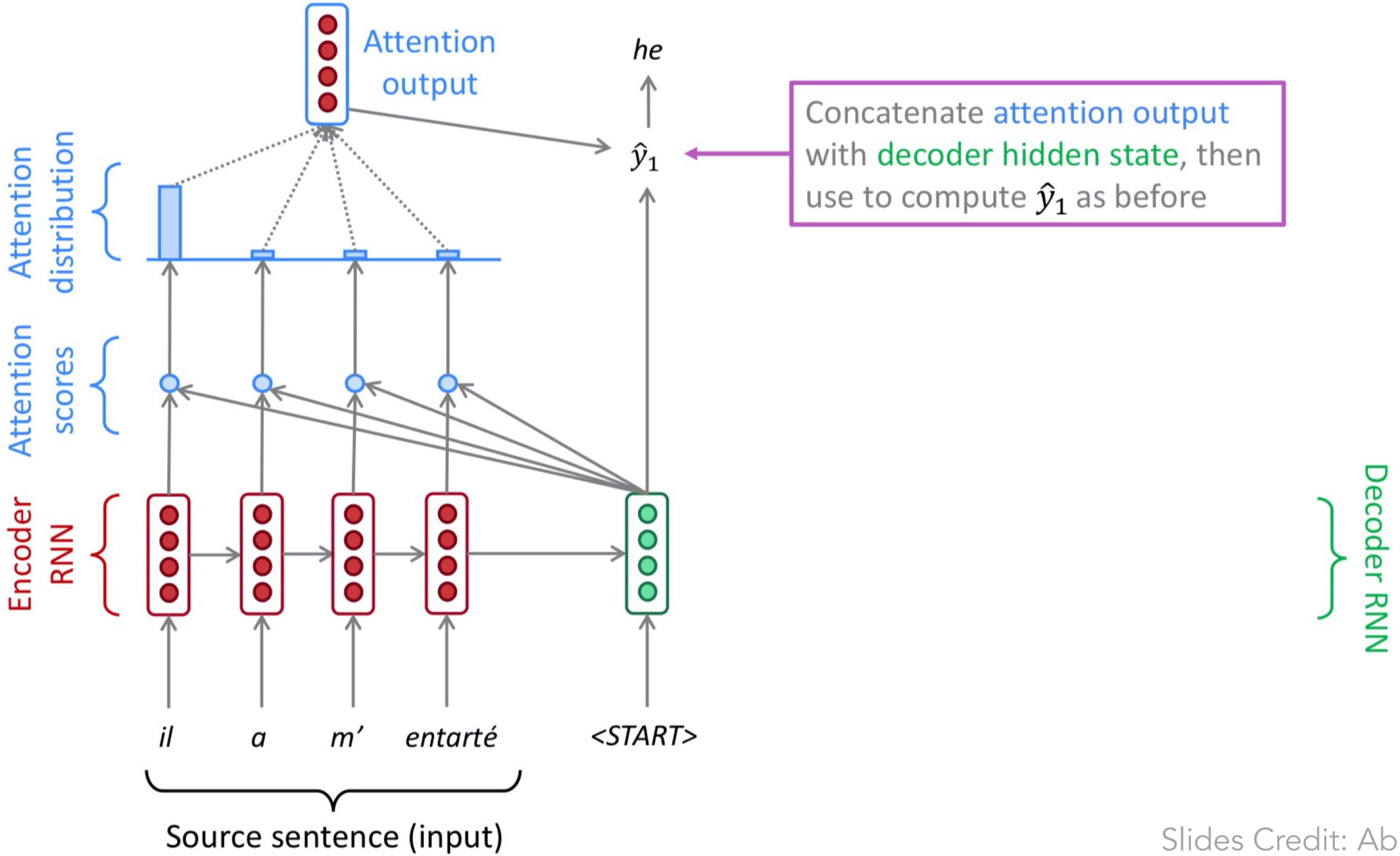
Slides Credit: Abigail See



Slides Credit: Abigail See

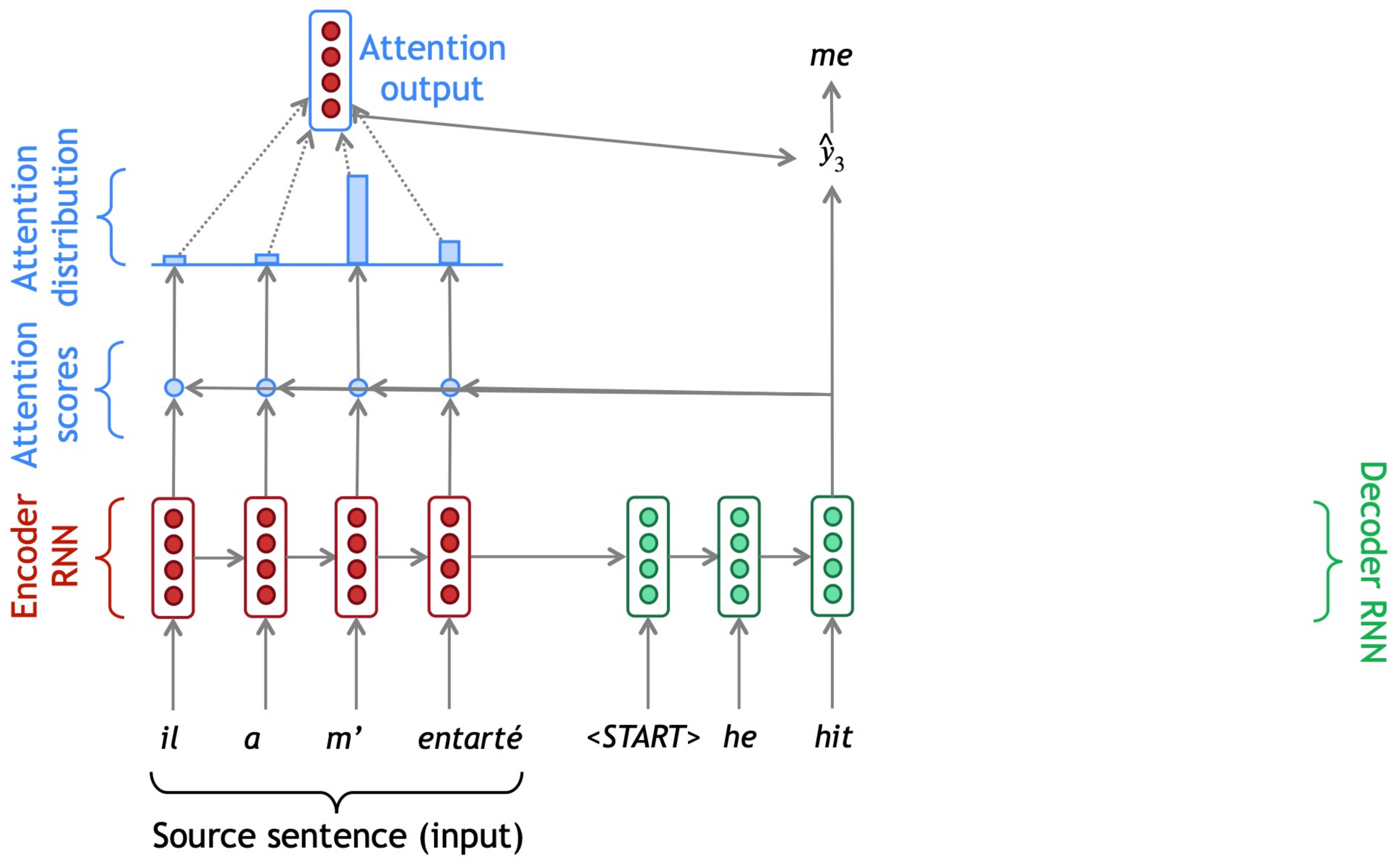


Slides Credit: Abigail See

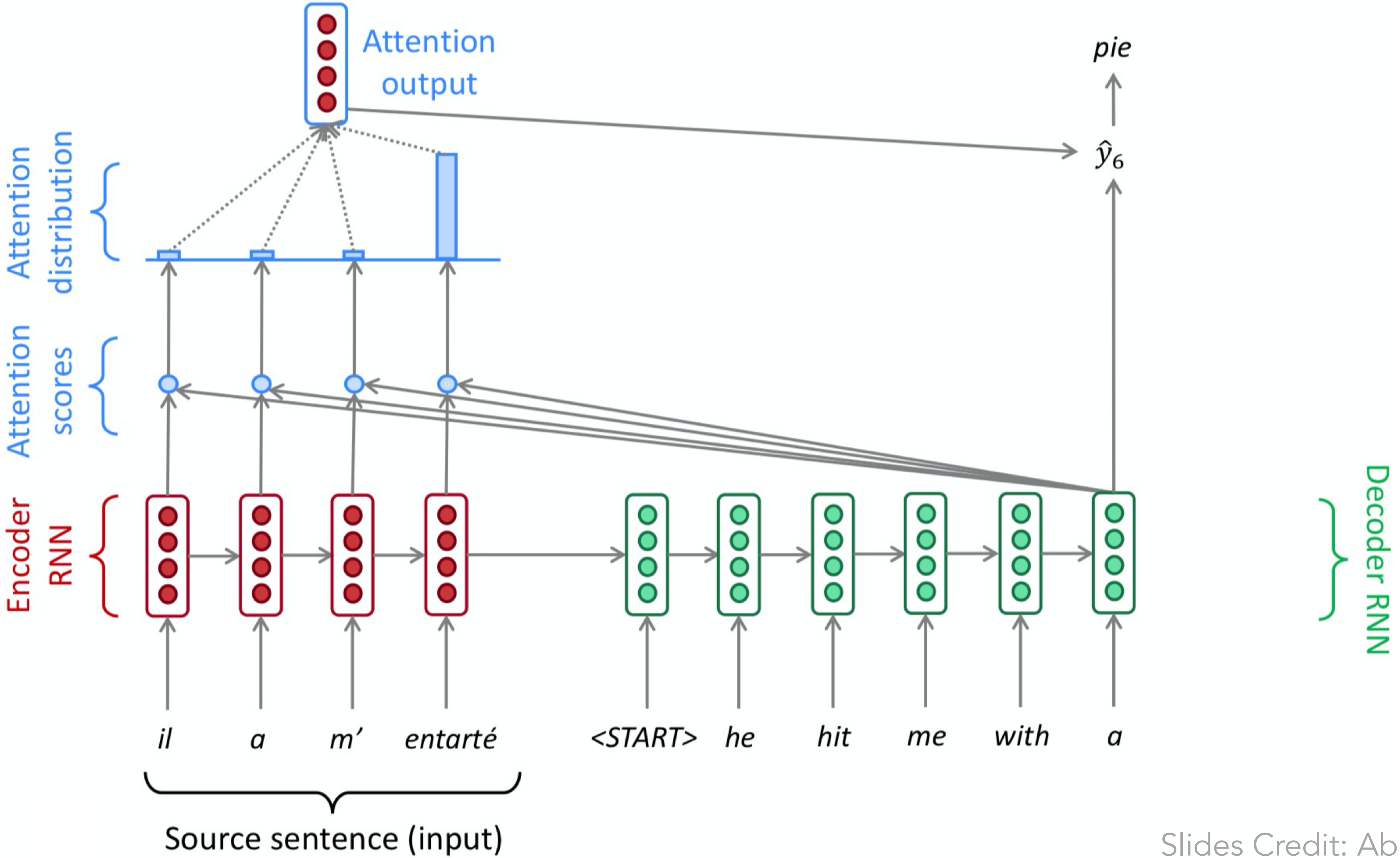


Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Slides Credit: Abigail See



Attention output
me
$\hat{y}_3$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
Source sentence (input)



Slides Credit: Abigail See

Putting it all together

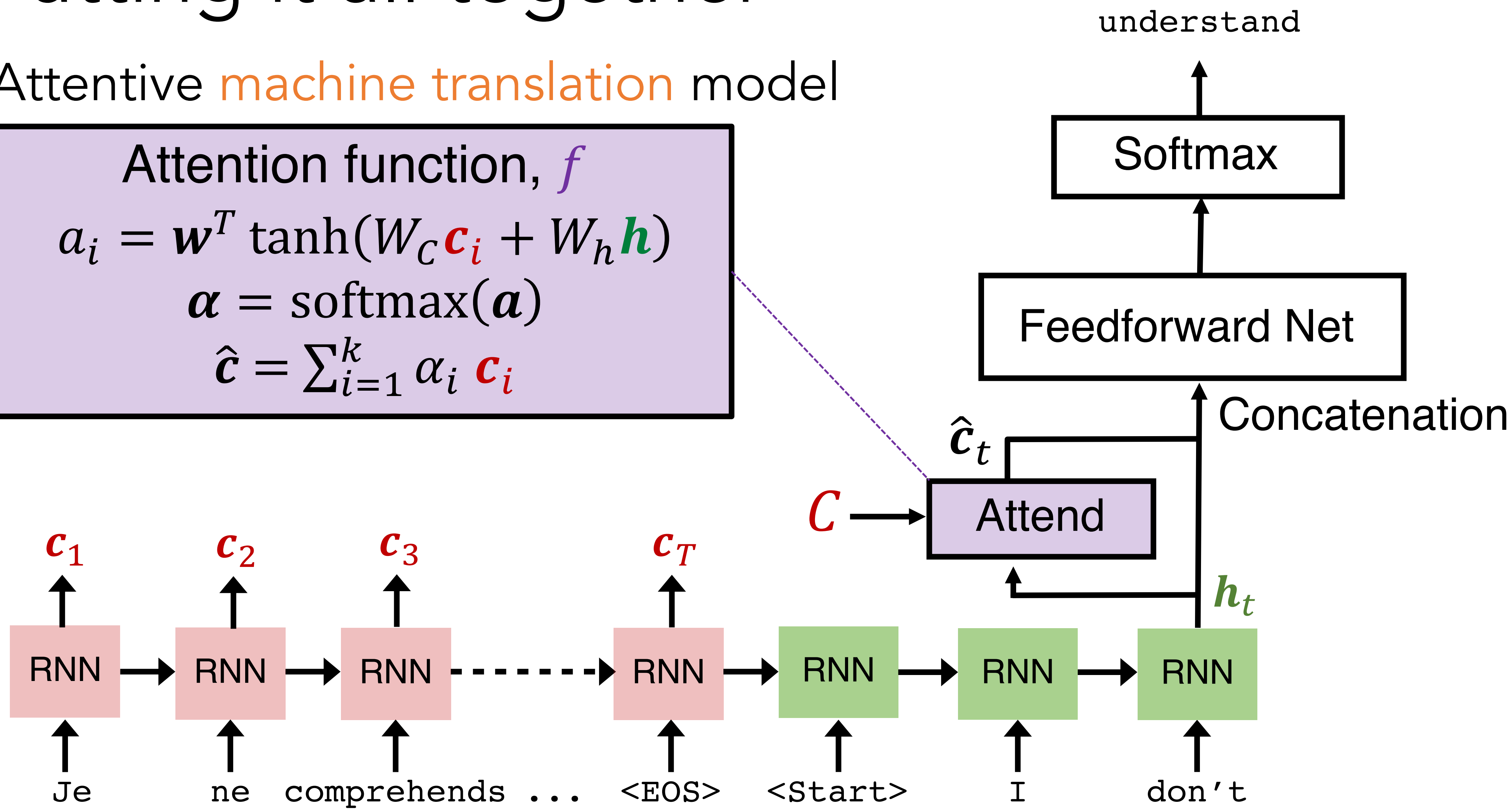
- Attentive machine translation model

Attention function, f

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

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

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



Slides Credit: Peter Anderson

Putting it all together

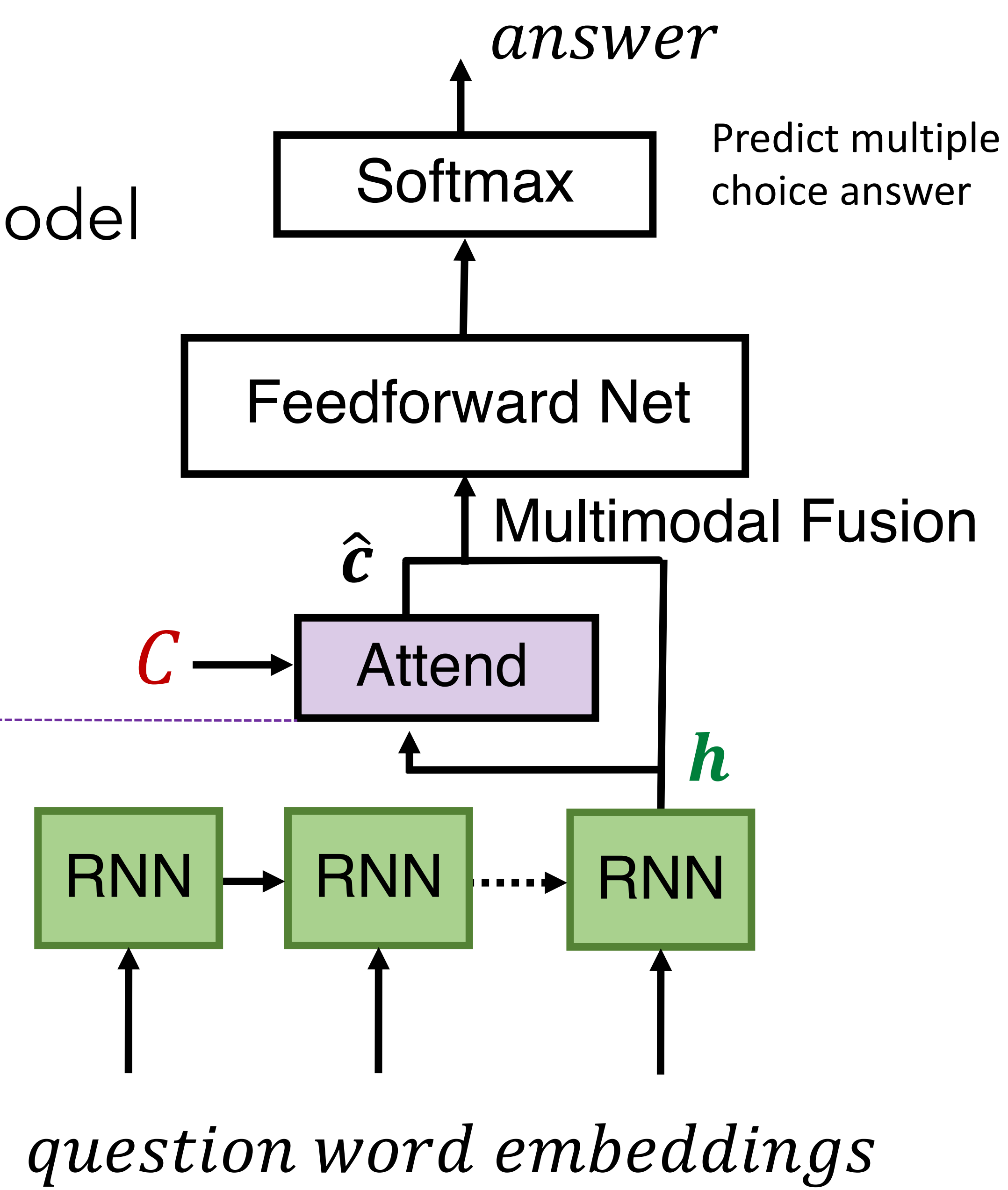
- Attentive Visual Question Answering model

Attention over visual regions!

$$C = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{100}\}$$

Attention function, f

$$a_i = \boldsymbol{w}^T \tanh(W_C \boldsymbol{c}_i + W_h \boldsymbol{h})$$
$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$
$$\hat{\boldsymbol{c}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{c}_i$$



Slides Credit: Peter Anderson

Putting it all together

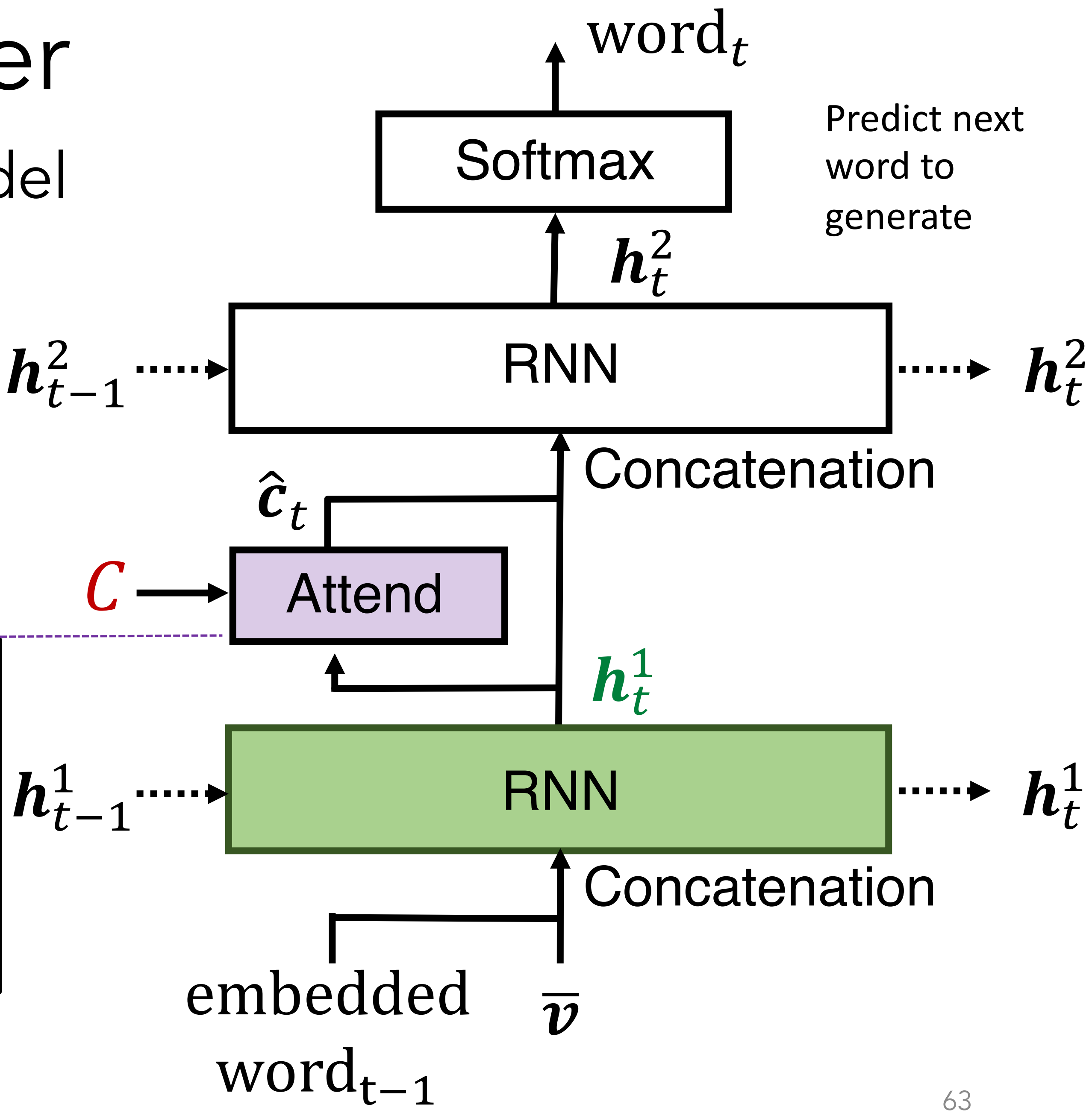
- Attentive Image Captioning model

Attention over visual regions!

$$C = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_{100}\}$$

Attention function, f

$$a_i = \boldsymbol{w}^T \tanh(W_C \boldsymbol{c}_i + W_h \boldsymbol{h})$$
$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$
$$\hat{\boldsymbol{c}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{c}_i$$



Types of attention scores

Attention function, f

$$a_i = g(\boldsymbol{c}_i, \boldsymbol{z})$$

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

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

- Dot-product attention:
  $$g(c_i, z) = z^\top c_i$$
- Scaled dot-product attention:
  $$g(c_i, z) = z^\top c_i / \sqrt{d}$$
- Bilinear / multiplicative attention:
  $$g(c_i, z) = z^\top W c_i \in \mathbb{R}$$
  where W is a weight matrix
- Additive attention (essentially MLP):
  $$g(c_i, z) = v^\top \tanh\left(W_1 c_i + W_2 z\right)$$
  where W_1, W_2 are weight matrices and v is a weight vector

Query-key-value view of attention

Attention function, f

$$a_i = g(\boldsymbol{c}_i, \boldsymbol{z})$$
$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$
$$\hat{\boldsymbol{c}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{c}_i$$

→

Attention function, f

$$a_i = g(\boldsymbol{k}_i, \boldsymbol{q})$$
$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$
$$\hat{\boldsymbol{c}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{v}_i$$

Projected query,key,value →

$$\boldsymbol{q} = W_Q \, \boldsymbol{z}$$
$$\boldsymbol{k}_i = W_K \, \boldsymbol{c}_i$$
$$\boldsymbol{v}_i = W_V \, \boldsymbol{c}_i$$

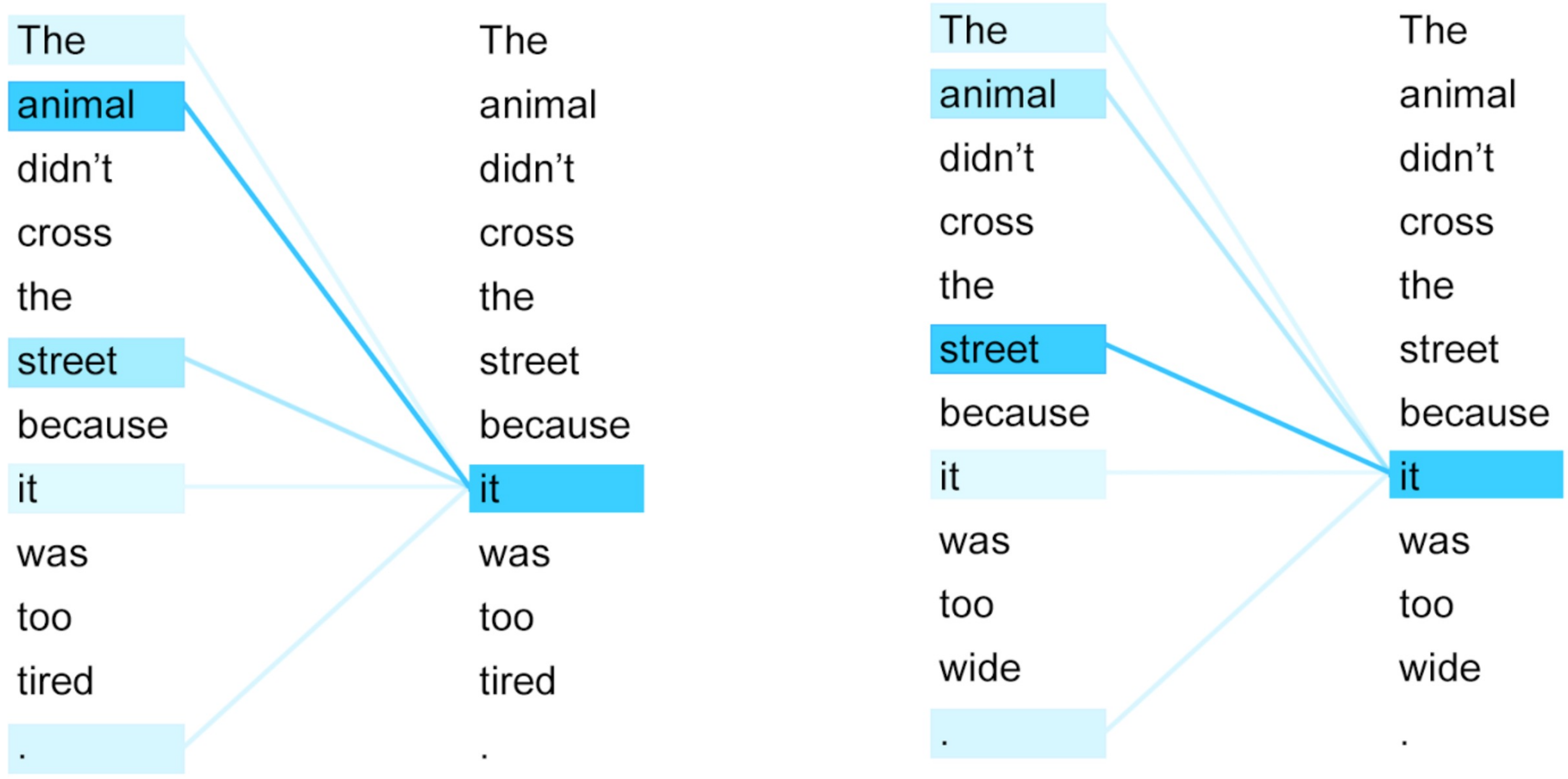
→

Matrix form

$$\boldsymbol{q} = W_Q \, \boldsymbol{z}$$
$$K = W_K \, C^T$$
$$V = W_V \, C^T$$

Self-attention

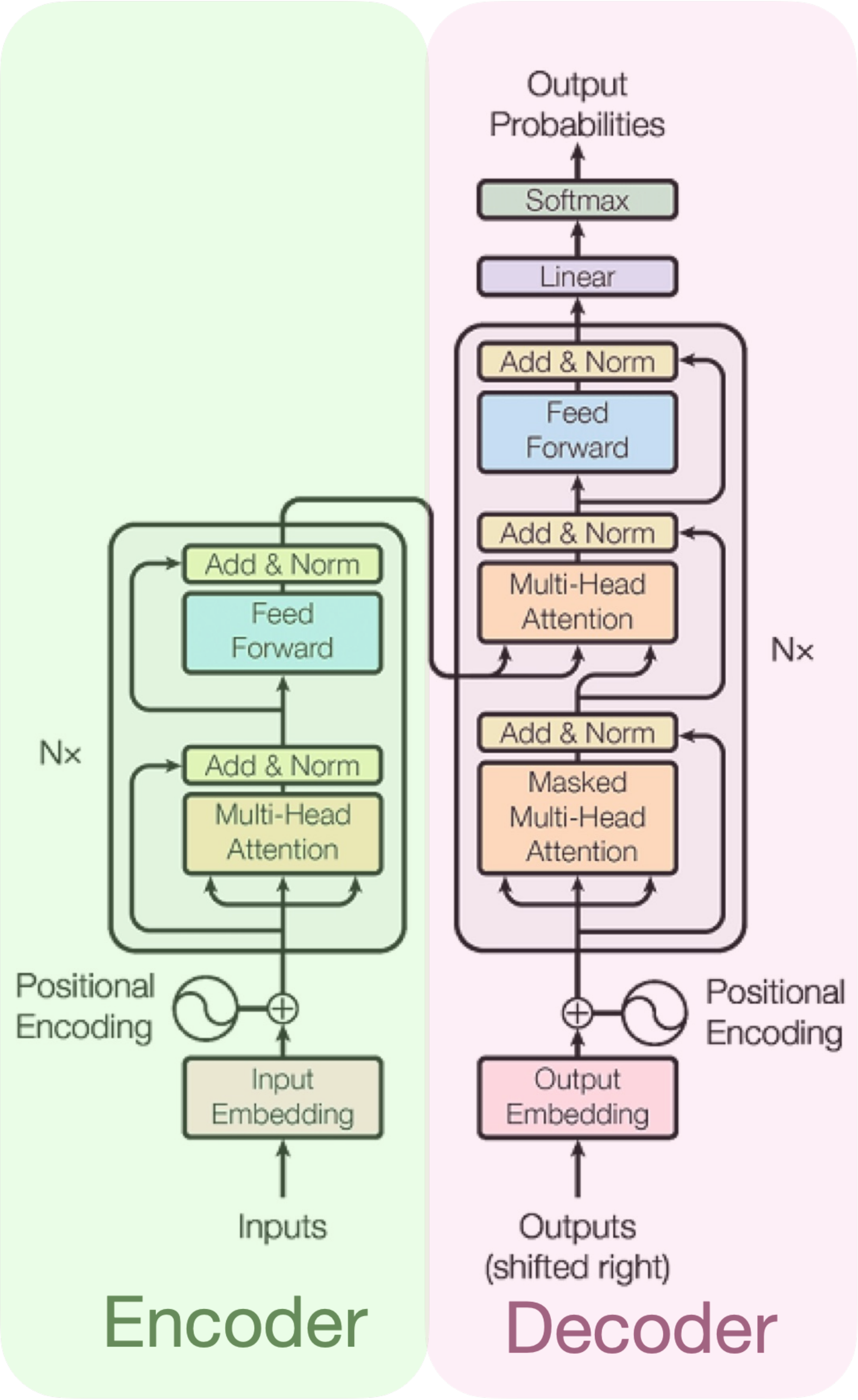
- Attention (correlation) with different parts of itself



- Transformers: modules with scaled dot-product self-attention

Transformers

- NIPS'17: Attention is All You Need
- Originally proposed for NMT (encoder-decoder framework)
- Key idea: **Multi-head self-attention**
- No recurrence structure so training can be parallelized



Modelling Sequences -- Transformers

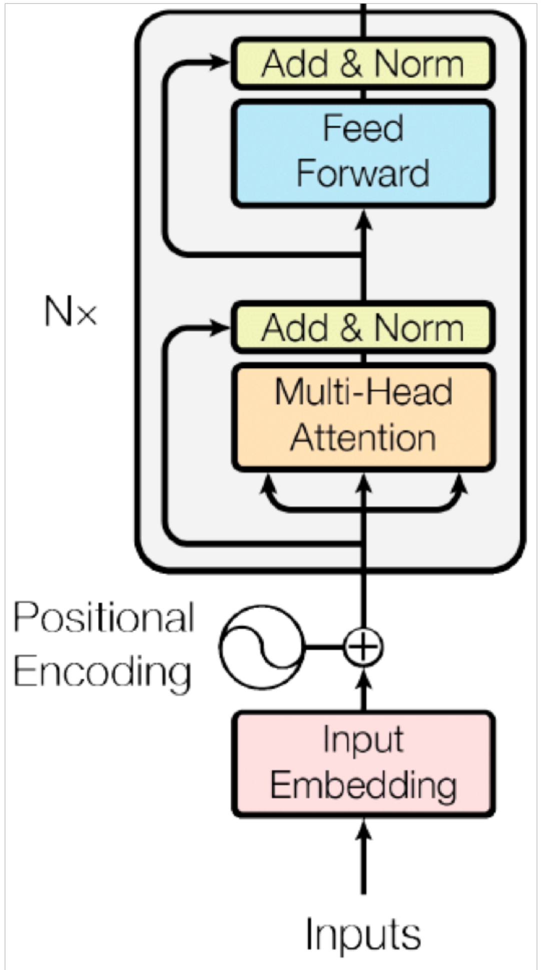
- Each Transformer block has two sub-layers
  - Multi-head attention
  - 2-layer feedforward NN (with ReLU)

- Each sublayer has a residual connection and a layer normalization

$$\mathrm{LayerNorm}(x + \mathrm{SubLayer}(x))$$

Helps the training process!

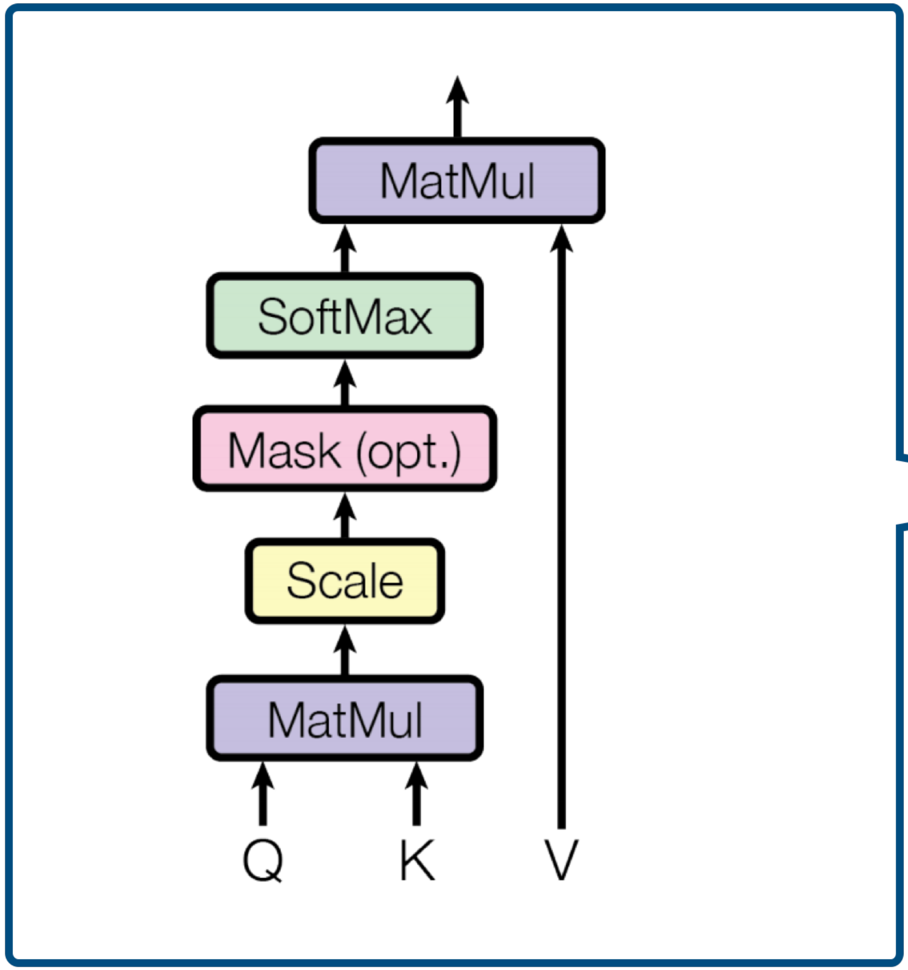
- Input layer has a positional encoding

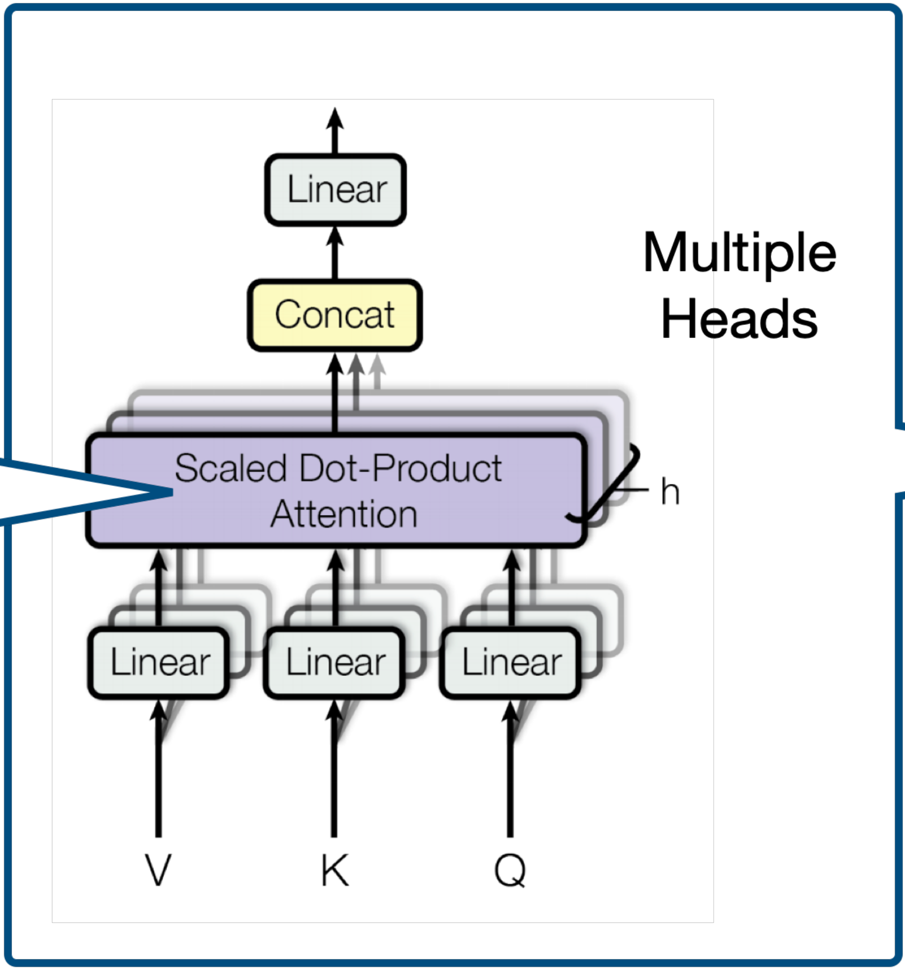


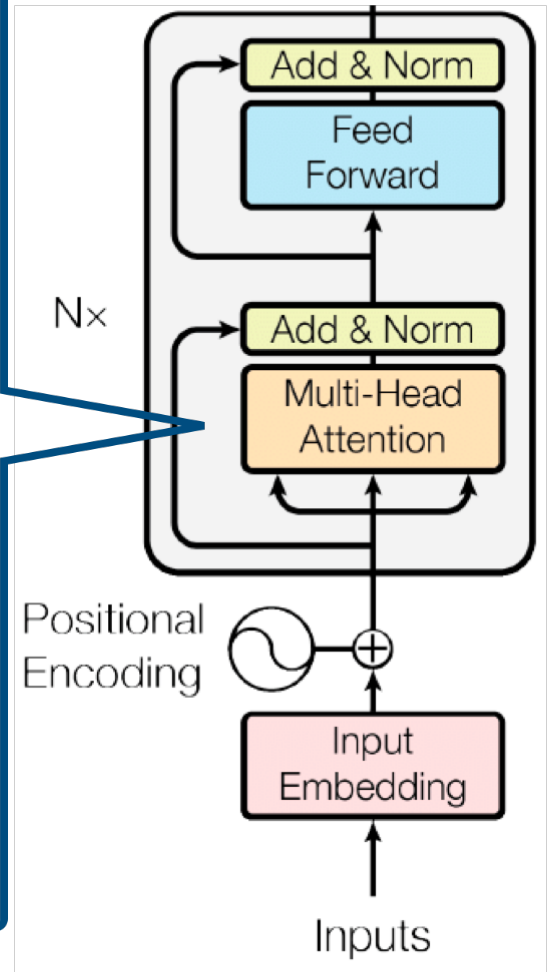
Modelling Sequences -- Transformers

Scaled Dot-Product Attention

self-attention

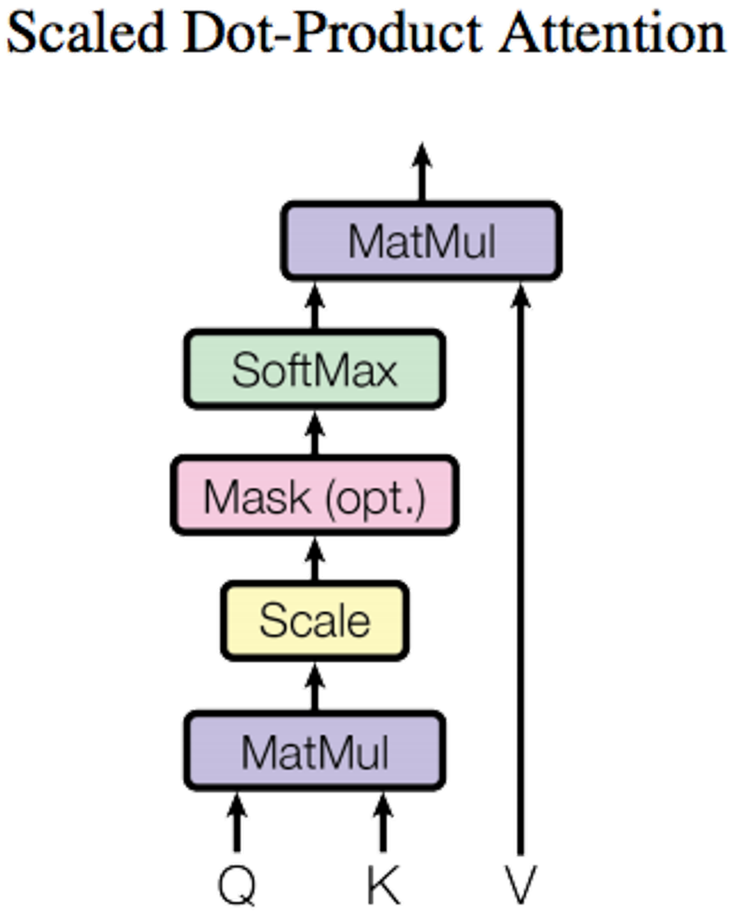






Scaled Dot Product Attention

Efficient, stable training



Let $X \in \mathbb{R}^{M \times d_X}$ be a matrix of task context vectors to attend to

Let $C \in \mathbb{R}^{N \times d_C}$ be a matrix of input vectors to attend over

$\boldsymbol{SDPAttention}(\boldsymbol{X}, \boldsymbol{C})$:

$$Q = W_Q X^T \qquad \mathrm{W_Q} \in \mathbb{R}^{d_h \times d_X}$$

$$K = W_K C^T \qquad \mathrm{W_K} \in \mathbb{R}^{d_h \times d_C}$$

$$V = W_V C^T \qquad \mathrm{W_K} \in \mathbb{R}^{d_V \times d_C}$$

$$Return \quad \hat{V} = softmax\left(\frac{Q^T K}{\sqrt{d_h}}\right) V$$

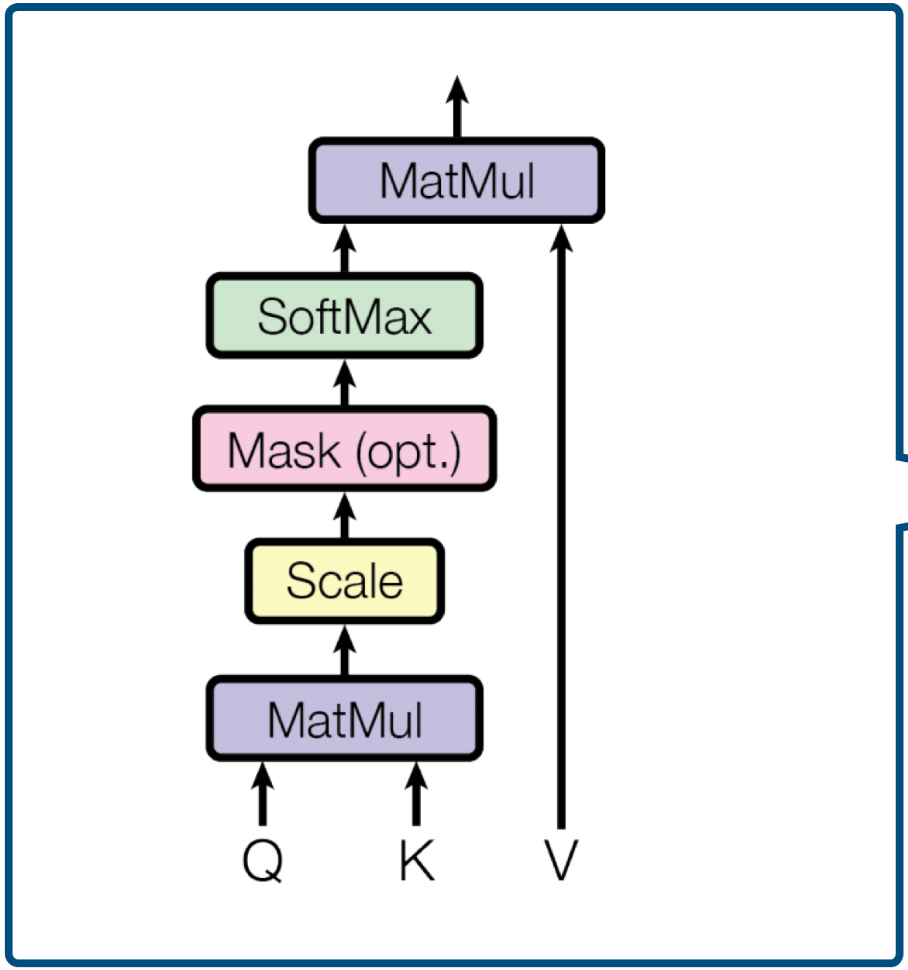
$\hat{V} \in \mathbb{R}^{M \times d_V}$ be a matrix of attended values

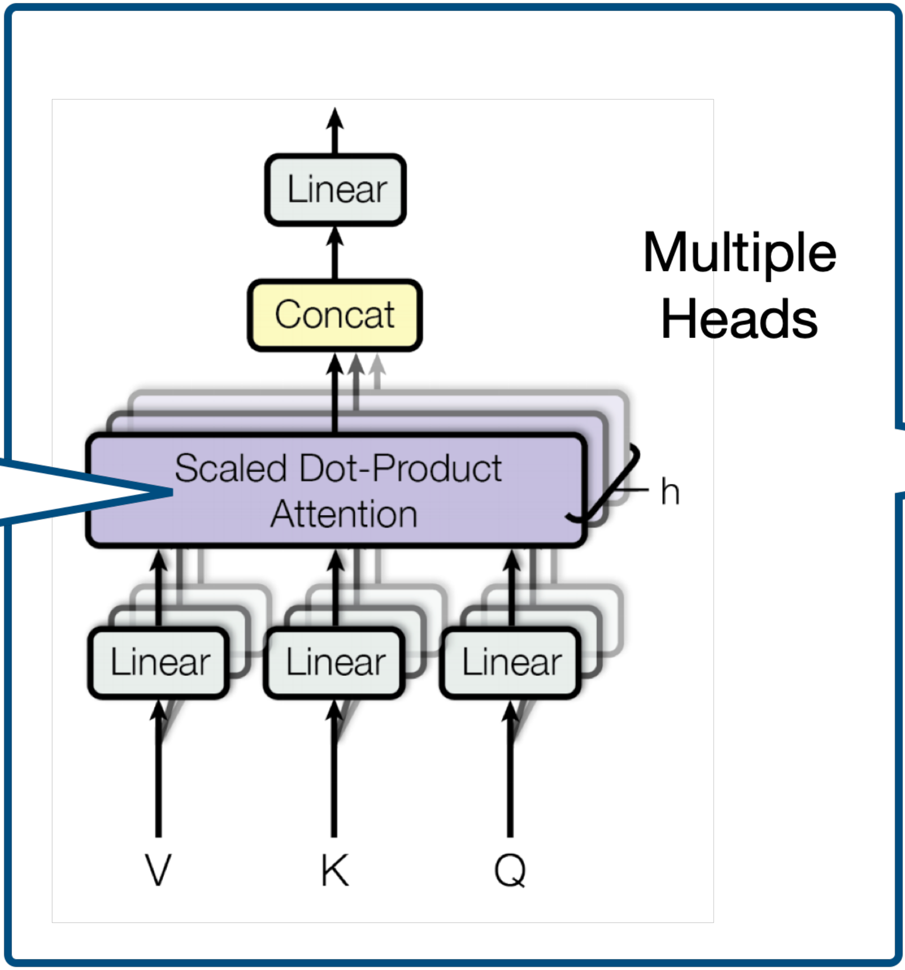
Attention Is All You Need https://arxiv.org/pdf/1706.03762.pdf

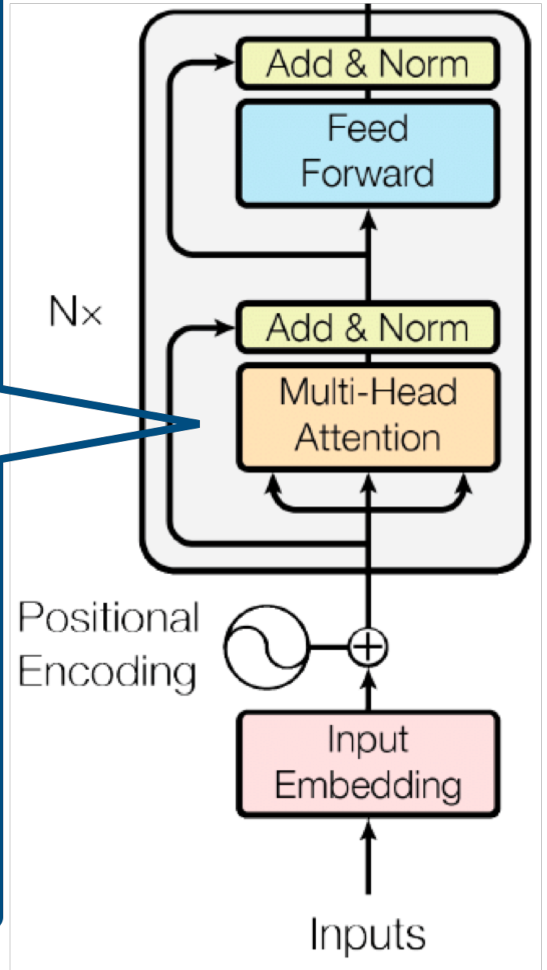
Modelling Sequences -- Transformers

Scaled Dot-Product Attention

self-attention $SDPAttention(C, C)$:







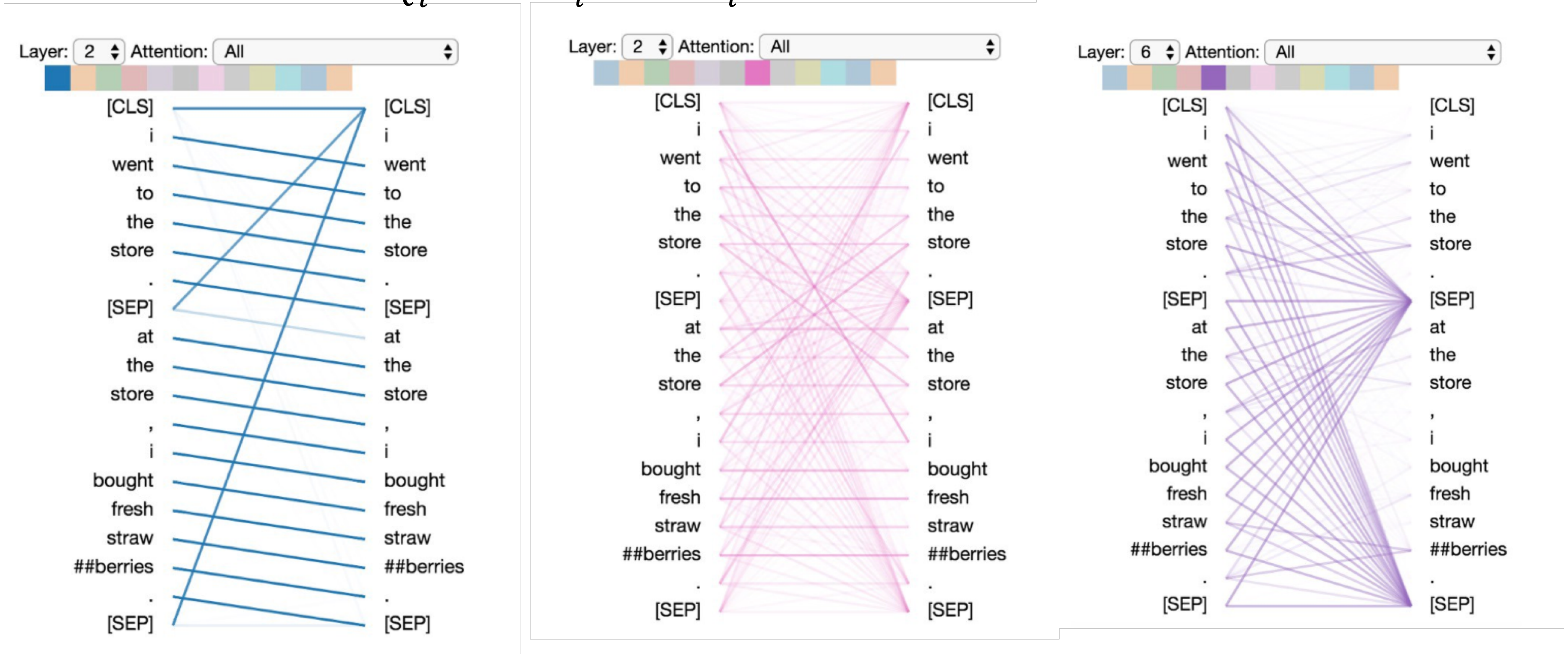
Multi-head attention

One head is not expressive enough. Let's have multiple heads!

$$A(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W_O$$

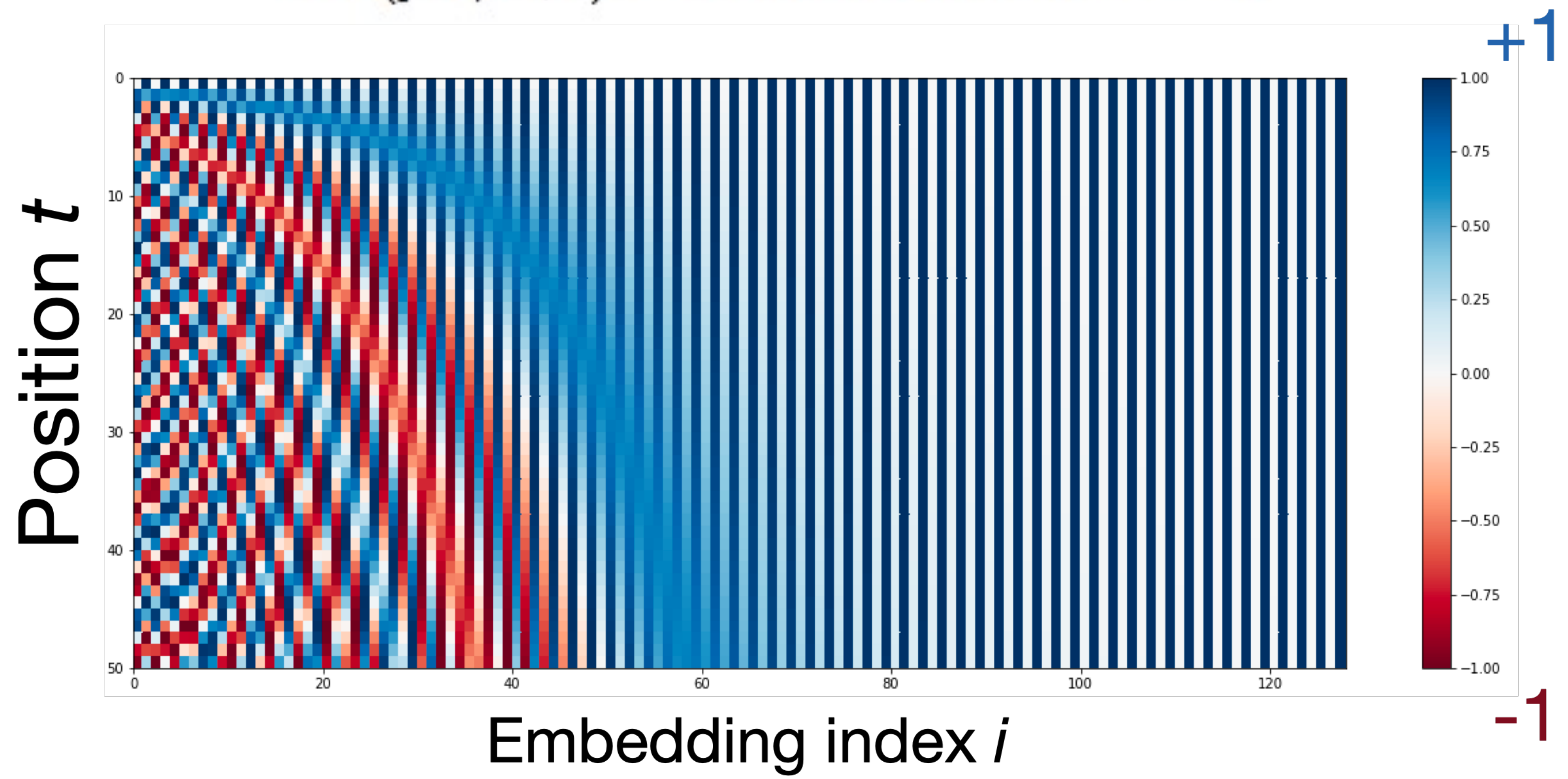
$$\text{head}_i = A(W_{Q_i} X^T, W_{K_i} X^T, W_{V_i} X^T)$$

In practice, $h = 8$,
$d = d_{out}/h, W_O \in \mathbb{R}^{d_{out} \times d_{out}}$

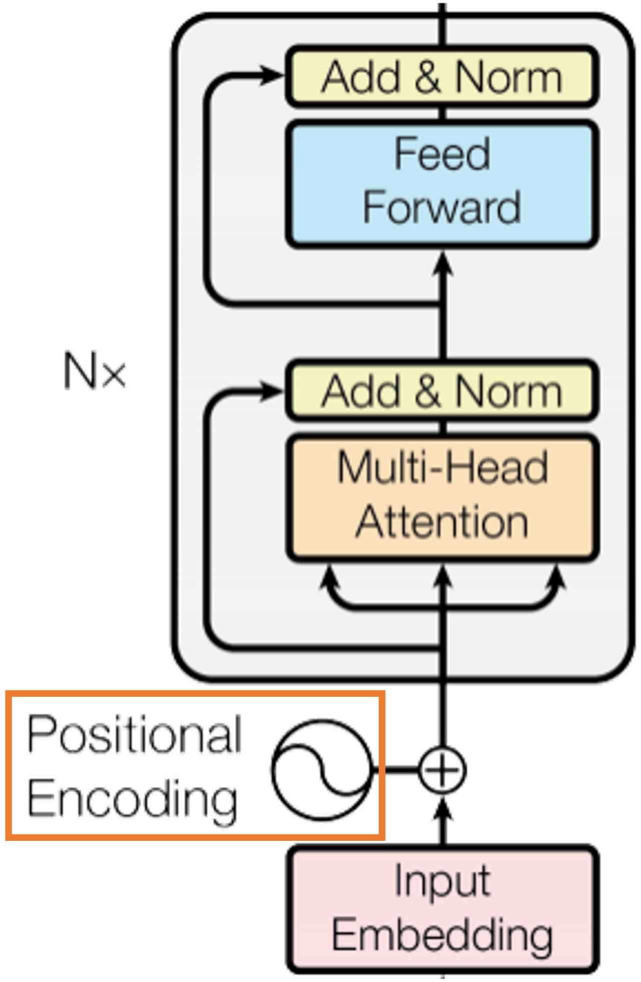


Transformers: Encoding position

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



$SDPAttention(Y, Y)$:



Attention Is All You Need https://arxiv.org/pdf/1706.03762.pdf

Modelling Sequences -- Transformers

- Each Transformer block has two sub-layers
  - Multi-head attention
  - 2-layer feedforward NN (with ReLU)

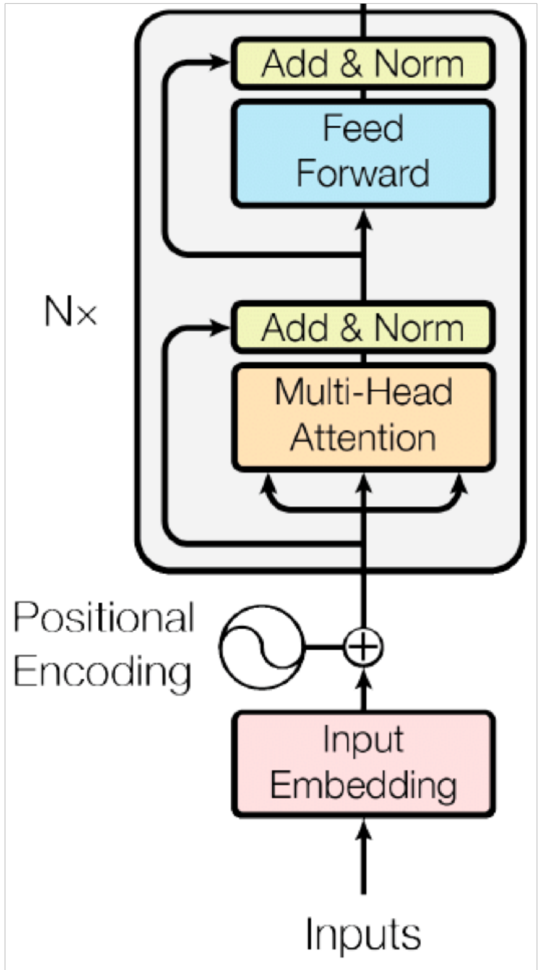
    Provides non-linearity

- Each sublayer has a residual connection and a layer normalization

$$\text{LayerNorm}(x + \text{SubLayer}(x))$$

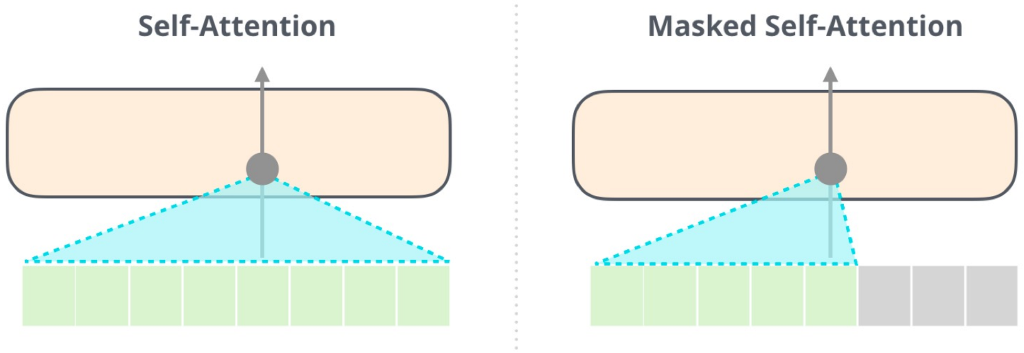
Helps the training process!

- Input layer has a positional encoding

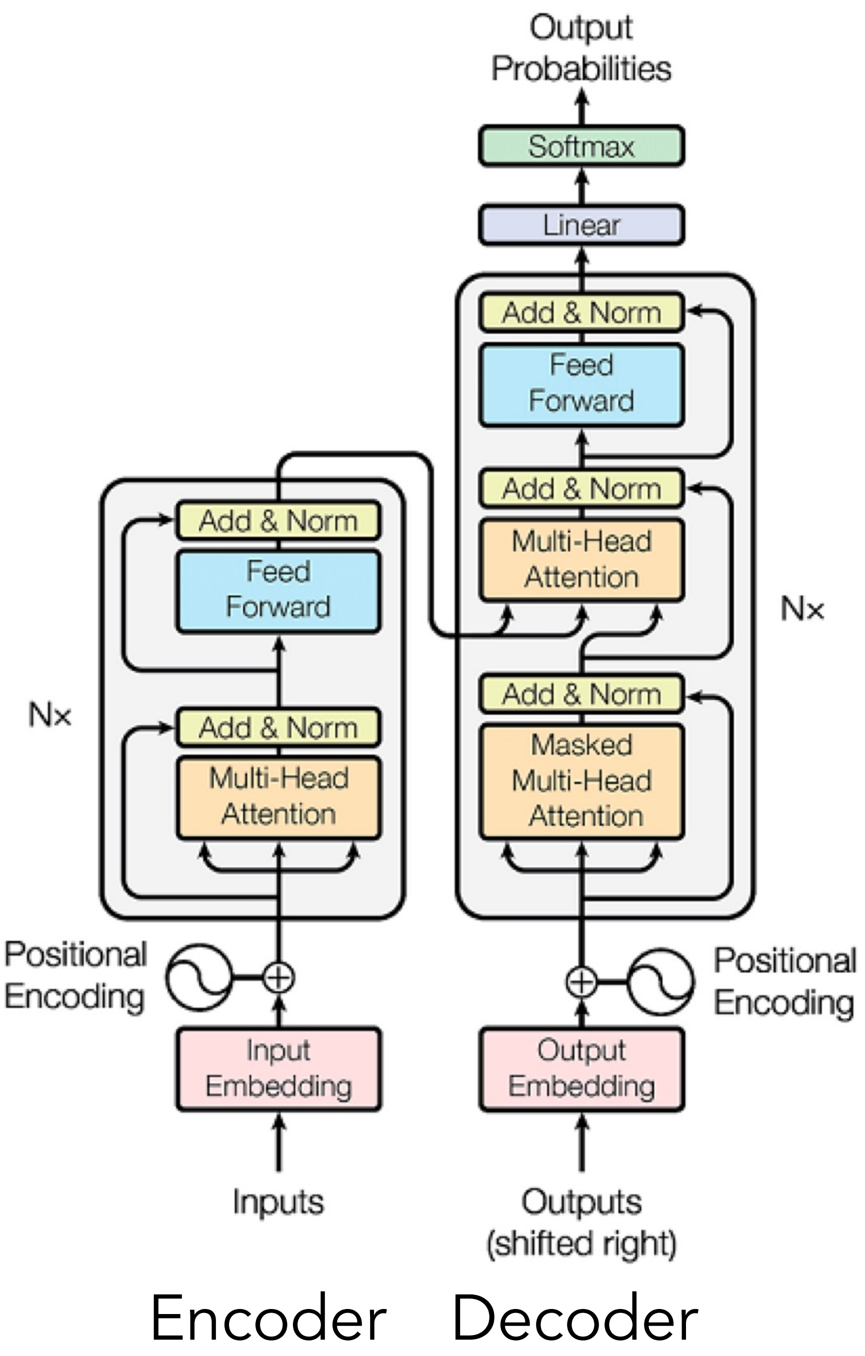


Transformers

- Encoder: Multi-headed self-attention
- Decoder
  - Masked self-attention

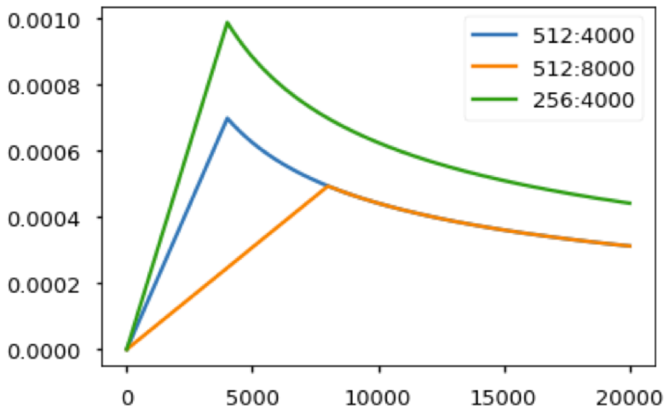


  - Cross attention
    - queries: previous decoder layer
    - keys/values: output of encoder
  - Autoregressive decoding

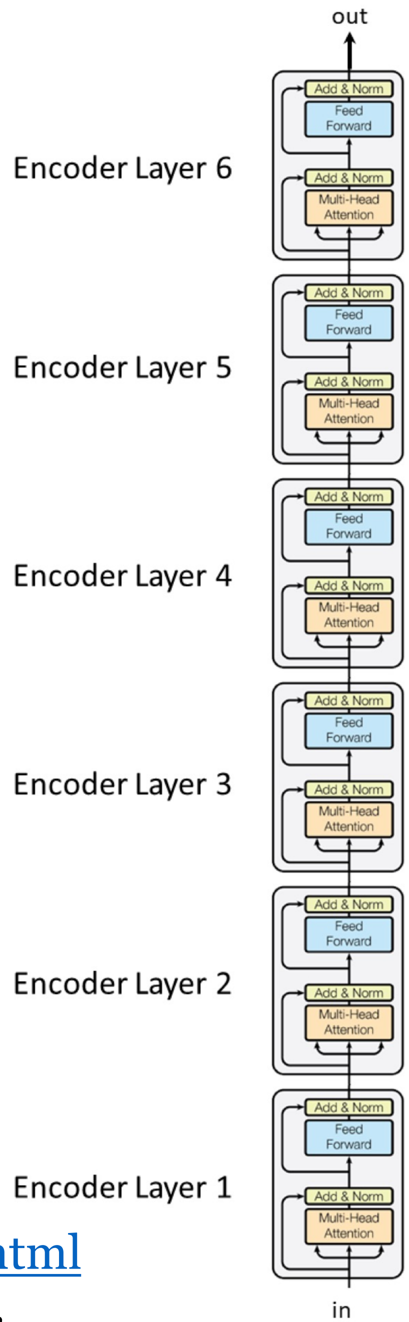


Transformers

- Stacked into multi-layers
- For language, input embedding is subwords
  - Byte-pair encoding (BPE) / Word pieces
- Other training details:
  - Learning rate with warmup and decay



  - Label smoothing: one-hot vector + noise



The Annotated Transformer http://nlp.seas.harvard.edu/2018/04/03/attention.html

A Jupyter notebook which explains how Transformer works line by line in PyTorch!

Transformers are used for everything!

10 Novel Applications using Transformers [DL]

Transformers have had a lot of success in training neural language models. In the past few weeks, we've seen several trending papers with code applying Transformers to new types of task:

🏞 **Transformer for Image Synthesis** - 🔗 Esser et al. (2020)
🔵 **Transformer for Multi-Object Tracking** - 🔗 Sun et al. (2020)
🎶 **Transformer for Music Generation** - 🔗 Hsiao et al. (2021)
💃 **Transformer for Dance Generation with Music** - 🔗 Huang et al. (2021)
🔮 **Transformer for 3D Object Detection** - 🔗 Bhattacharyya et al. (2021)
🗿 **Transformer for Point-Cloud Processing** - 🔗 Guo et al. (2020)
⏰ **Transformer for Time-Series Forecasting** - 🔗 Lim et al. (2020)
👁 **Transformer for Vision-Language Modeling** - 🔗 Zhang et al. (2021)
🛣 **Transformer for Lane Shape Prediction** - 🔗 Liu et al. (2020)
🌠 **Transformer for End-to-End Object Detection** - 🔗 Zhu et al. (2021)

PapersWithCode newsletter (1/20/2021)
https://paperswithcode.com/newsletter/3

Next time

- Multimodal representations