ECE499/ECE590
Machine Learning for Embedded Systems (Fall 2021)

Lecture 4: Natural Langue Processing

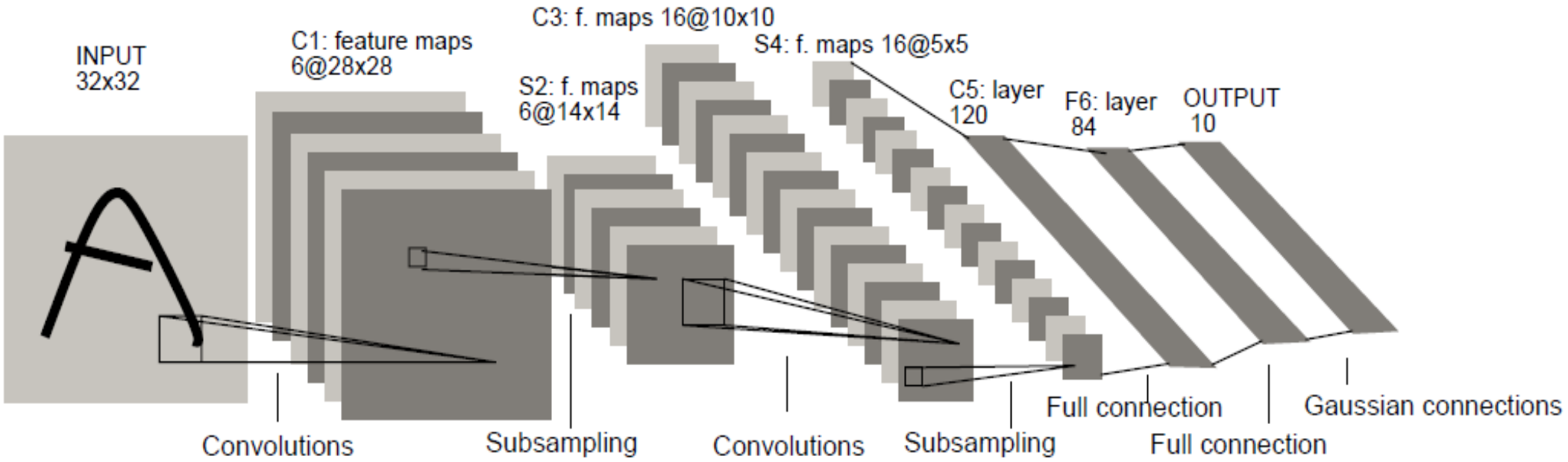
Weiwen Jiang, Ph.D.

Electrical and Computer Engineering

George Mason University

wjiang8@gmu.edu

What have we learned --- Deep CNN

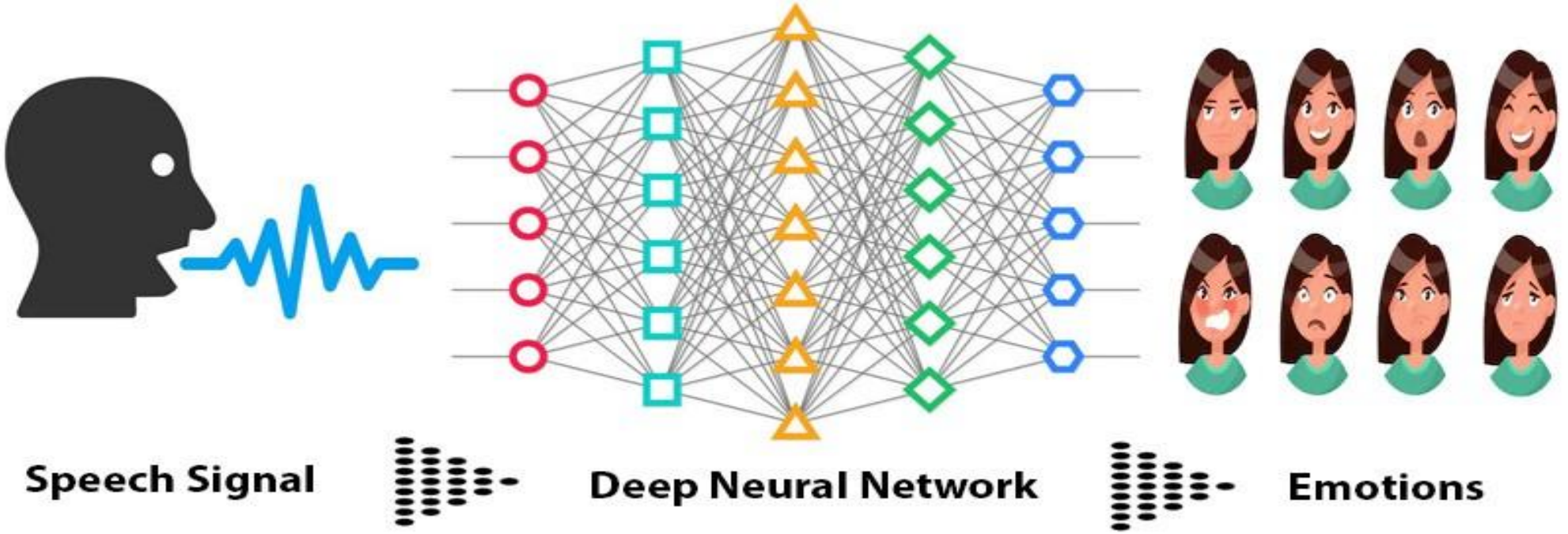


- Local receptive fields
- Shared weights
- Pooling (subsampling)

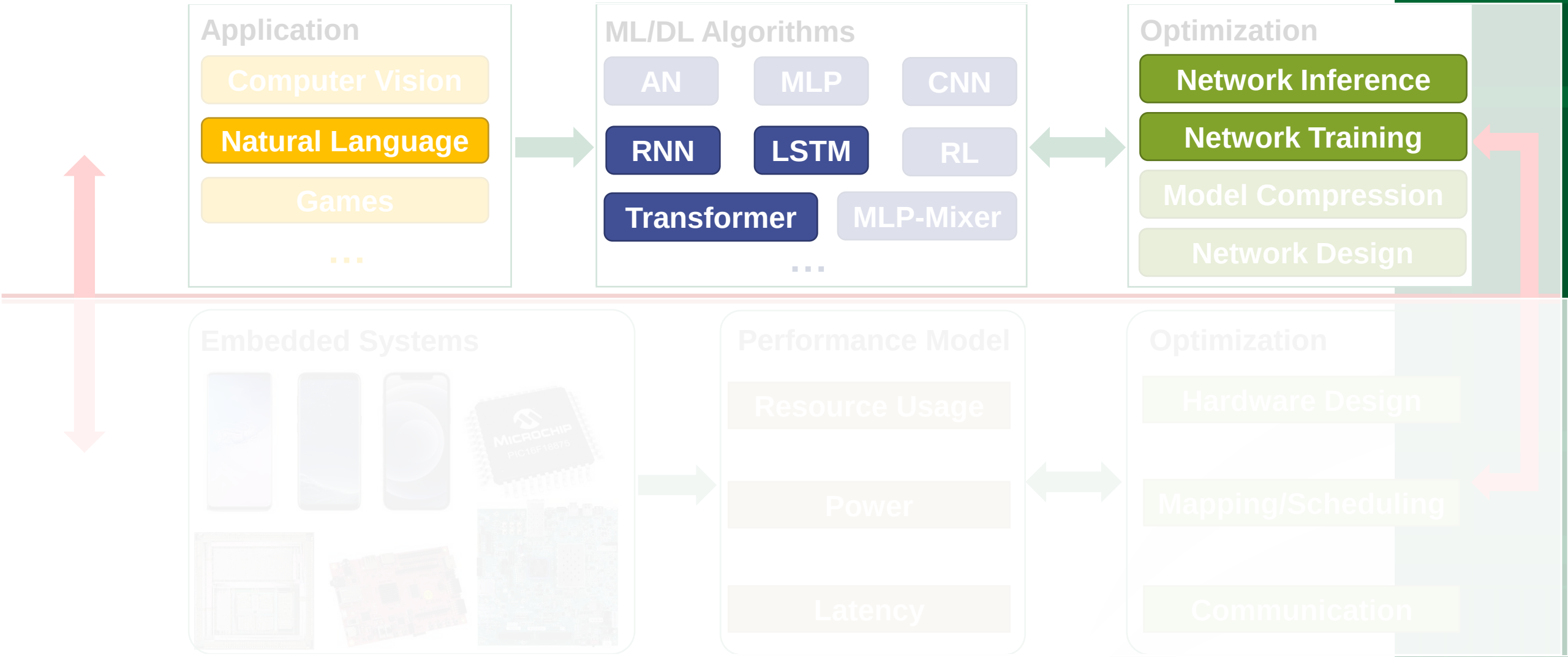
Cat?

Dog?

From Static Image to Sequences of Data



Week 4: From CV to NLP



Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

Modeling Sequences

- Sometimes we would like to turn an input sequence into another output sequence in a different domain

  Sound wave → word sequence

  English → Spanish

- Or we want to predict the next term in the input sequence

  The stock price for next month?

  Discover the pattern in temporal sequences for the predictions

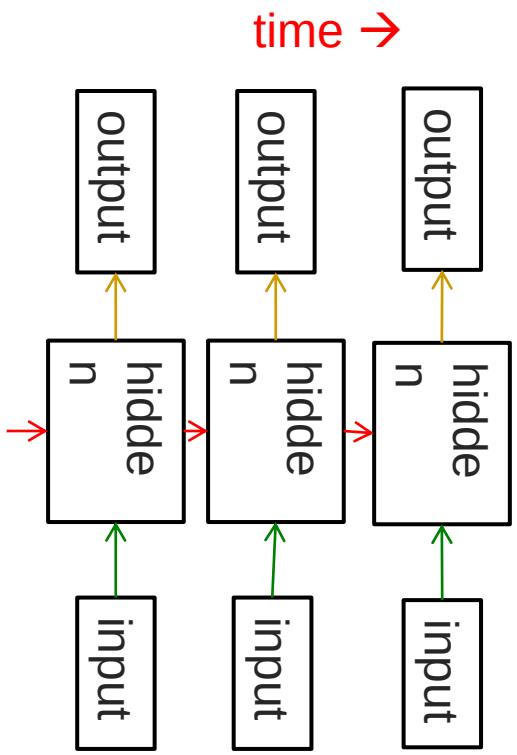
Recurrent Neural Networks (RNN)

- RNNs are very powerful, because they combine two properties:

  Distributed hidden state that allows them to store a lot of information about the past efficiently.

  State is represented by the combination of status of the hidden neurons

  Non-linear dynamics that allows them to update their hidden state in complicated ways.



Courtesy to Geff. Hinton

Different Types of RNNs

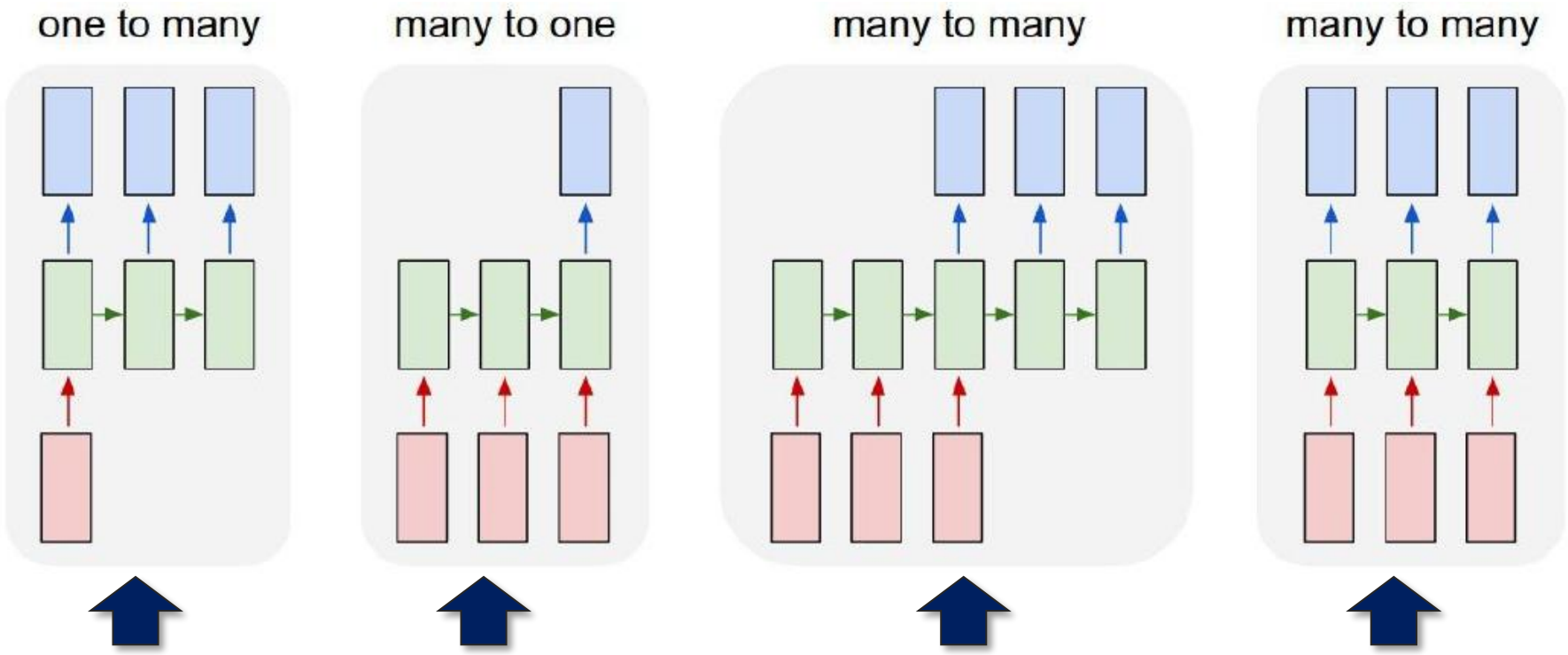


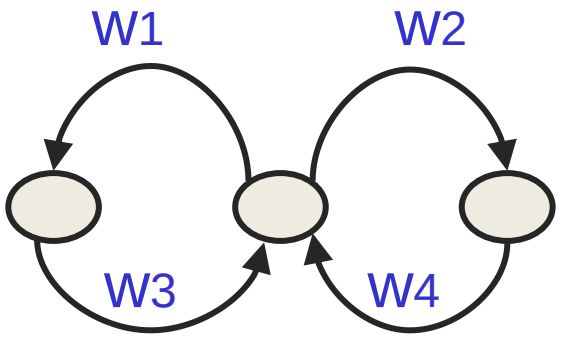
Image captioning (image → words)

Sentiment classification (words → sentiment)

Translation (words → words)

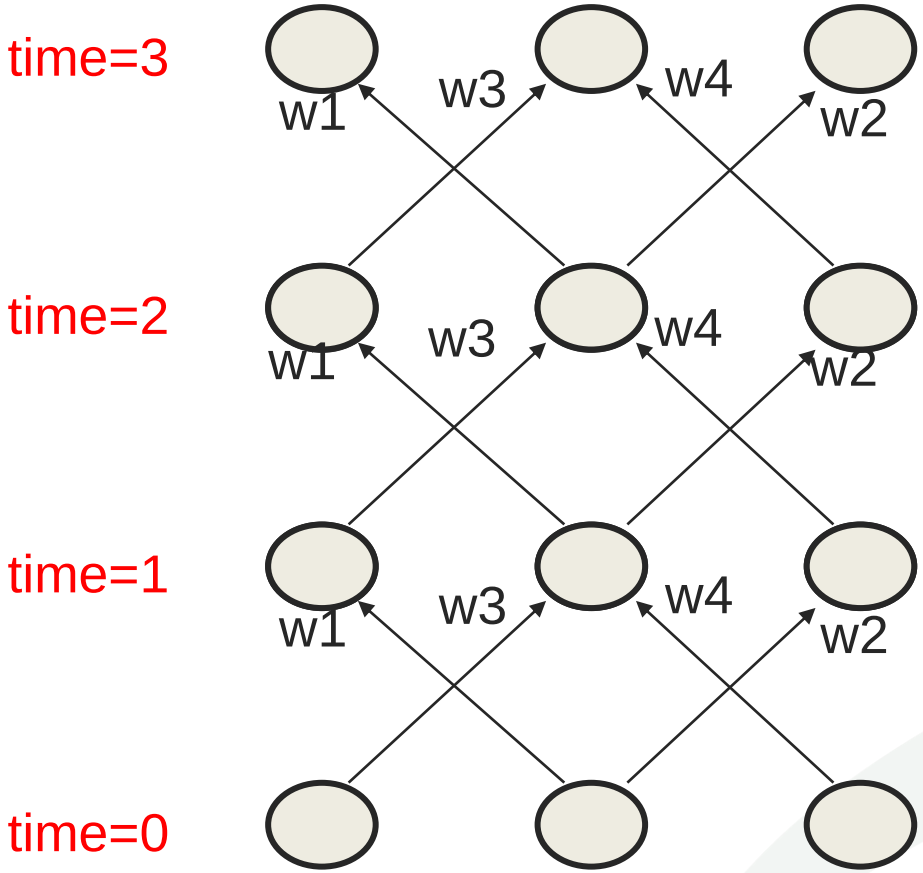
Video frame classification (frames → classes)

RNN and Feedforward Network



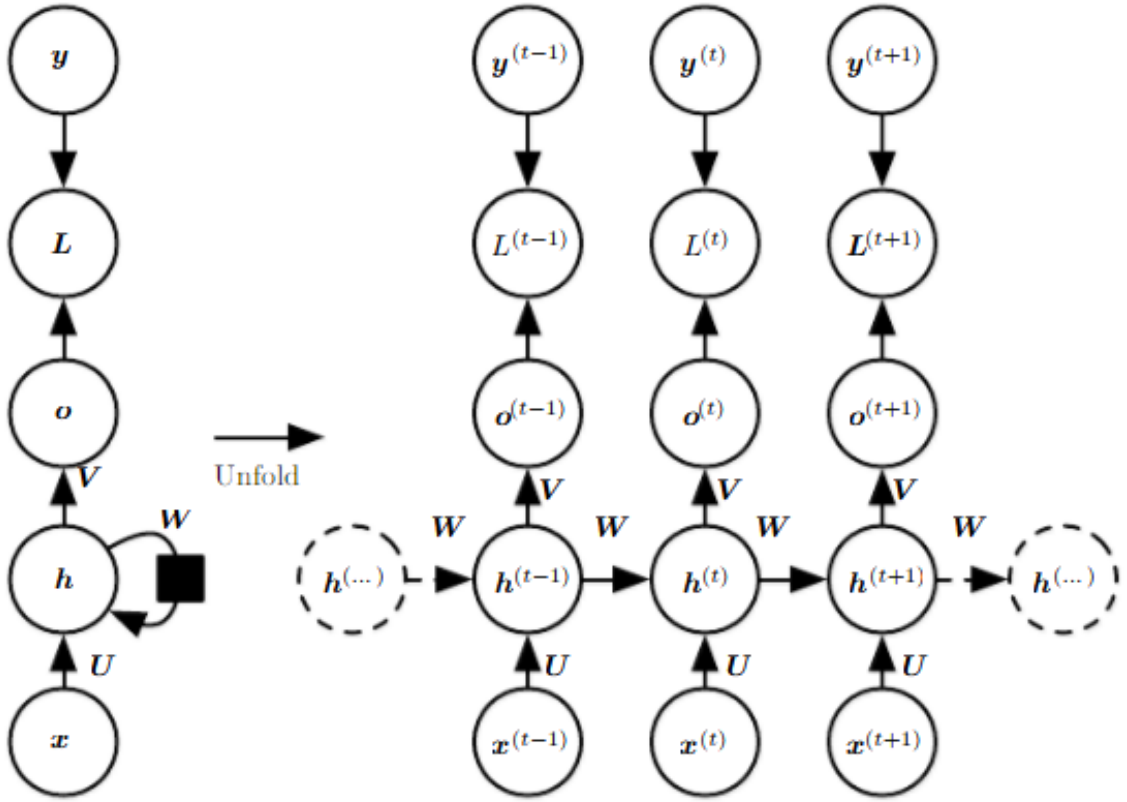
- Assume each connection has 1 unit delay
- RNN can be unrolled into feedforward networks

  Each layer keeps on reusing the same weights



Courtesy to Geff. Hinton

RNN and Feedforward Network



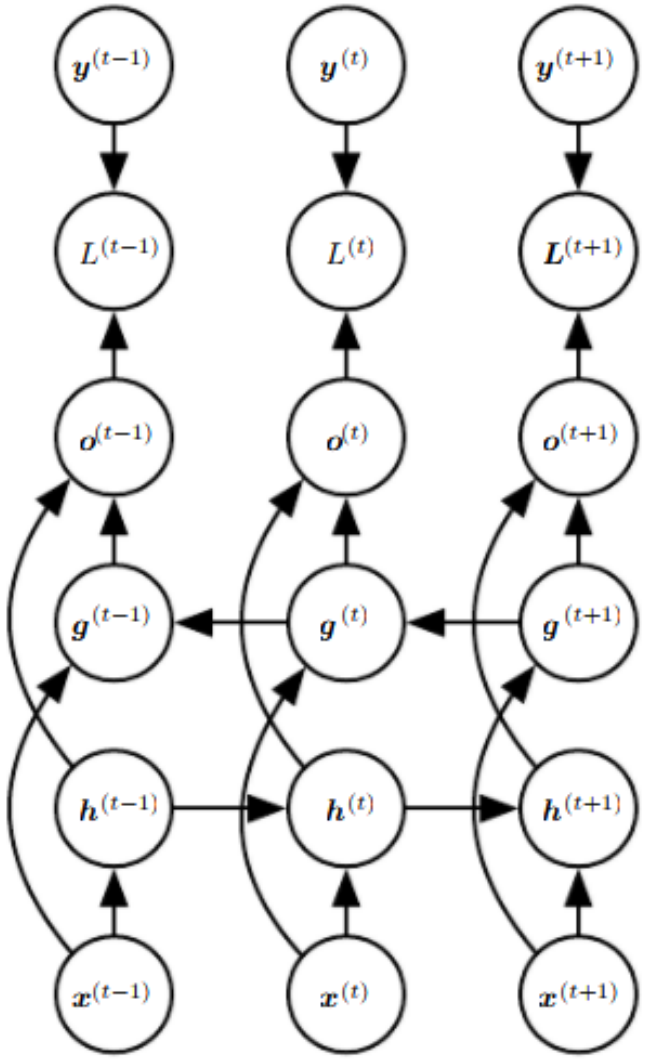
$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

From GoodFellow et al.
Deep Learning

Bidirectional RNNs (for speech recognition)



From GoodFellow et al.
Deep Learning

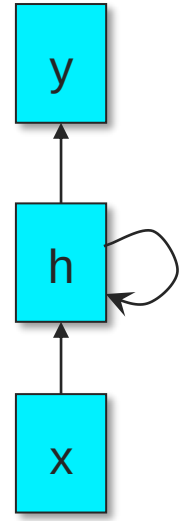
A Simple Example

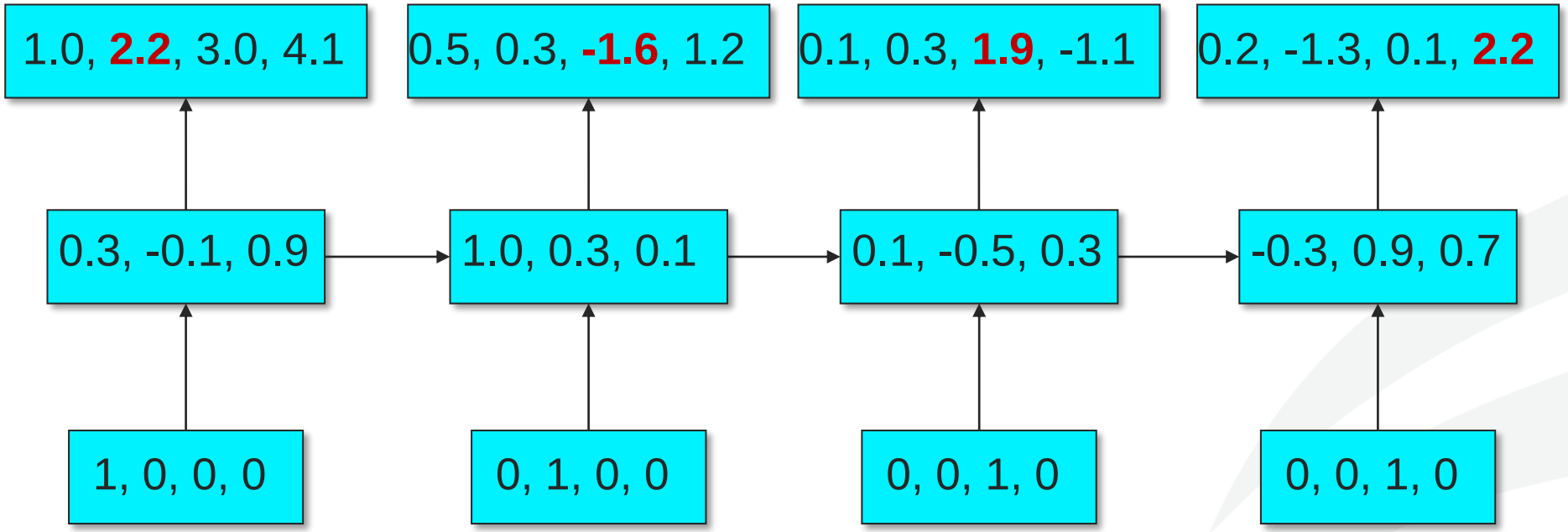
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} h_t$$

Character level
language model

Vocabulary [h, e, l, o]
Training sequence: “Hello”

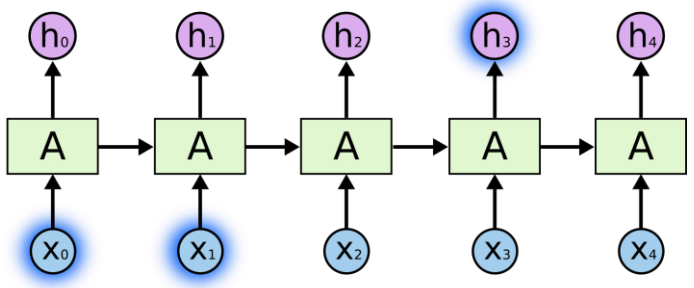


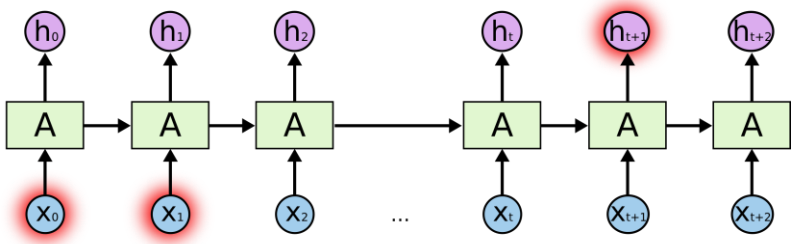


Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

Challenge of Long-Term Dependencies





Gradient explosion and gradient vanishing

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

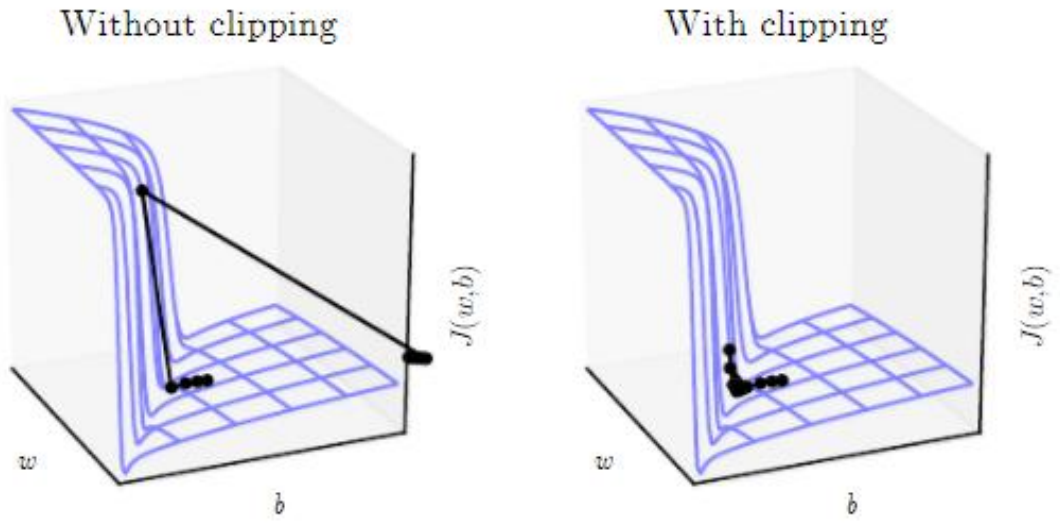
Largest singular value > 1: **Exploding gradients**

Largest singular value < 1: **Vanishing gradients**

From Colah's blog

Gradient Explosion and Vanishing

- The gradient explosion and vanishing phenomenon limits the number of steps in back propagation
  Limits the range of long-term dependency
- It can be partially resolved by gradient clipping technique
  Only applicable to gradient explosion mitigation



Gated RNN to Overcome Gradient Vanishing Problem

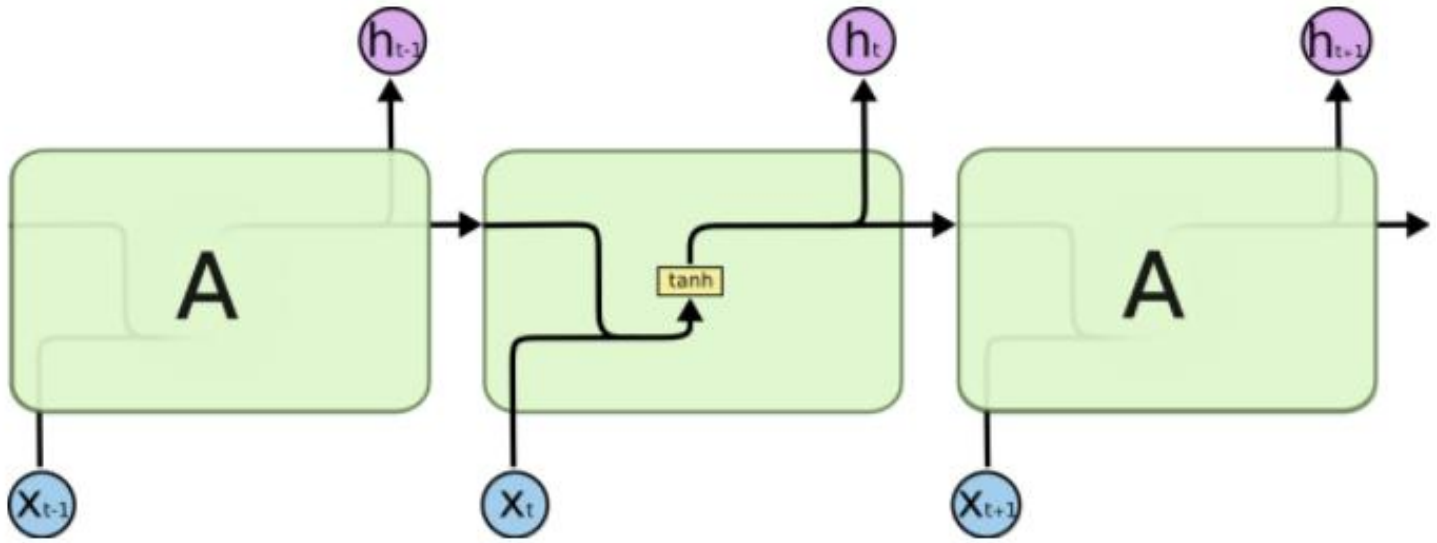
- Incorporate a “forget gate” and learn the corresponding values
  - To remember or forget long-term dependencies
  - The currently most effective way to overcome gradient vanishing problem
- Many variations of gated RNN structures
  - Long short-term memory (LSTM) network (introduced by Hochreiter & Schmidhuber in 1997)
  - Gated recurrent unit (GRU) network
  - Other variations

Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

LSTM Networks

- The most successful RNN structure of learning long-term dependencies

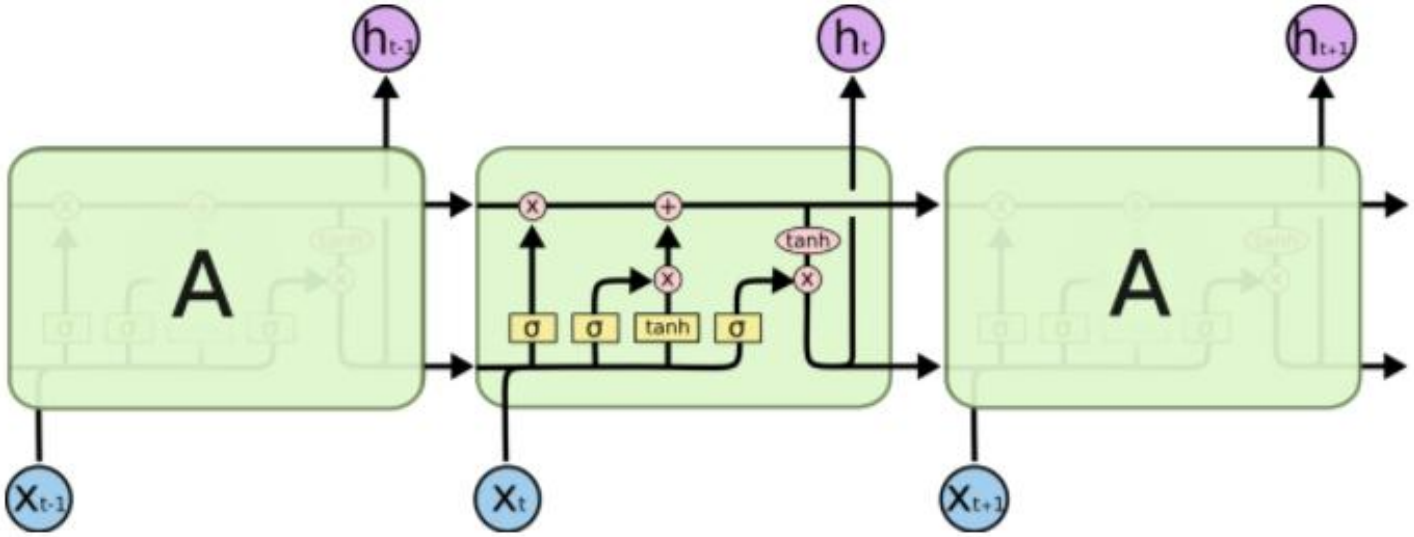


The repeating module in a standard RNN contains a single layer.

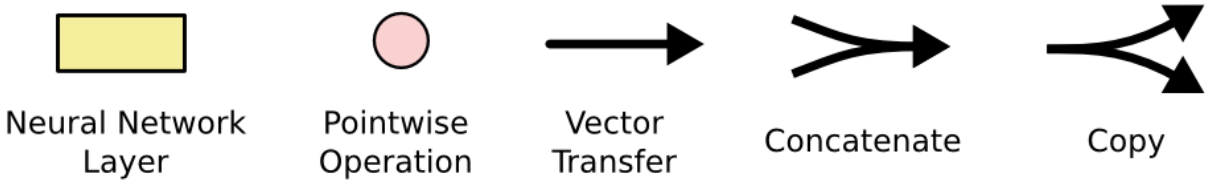
From Colah's blog

LSTM Networks

- The most successful RNN structure of learning long-term dependencies



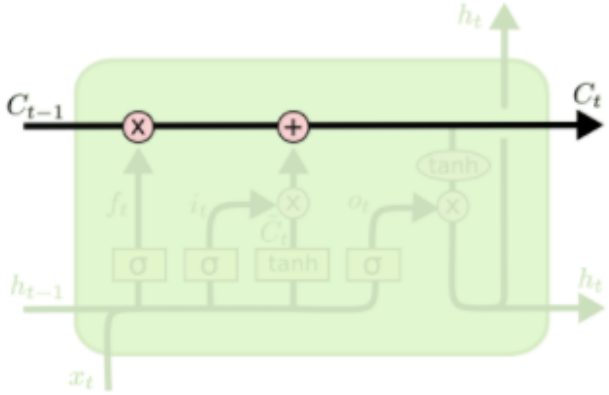
The repeating module in an LSTM contains four interacting layers.



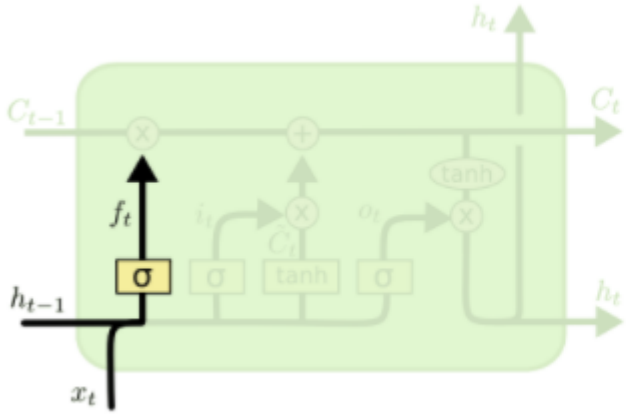
From Colah's blog

A Walk-through of LSTM Networks

- The key to LSTMs is the cell state, which is manipulated by the forget gate



- The forget gate allows us to learn to forget old states

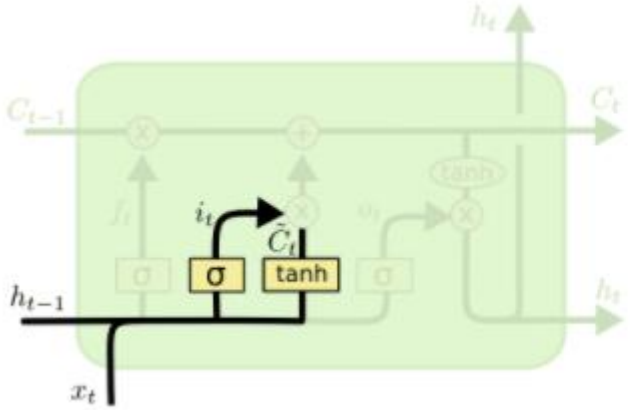


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

From Colah's blog

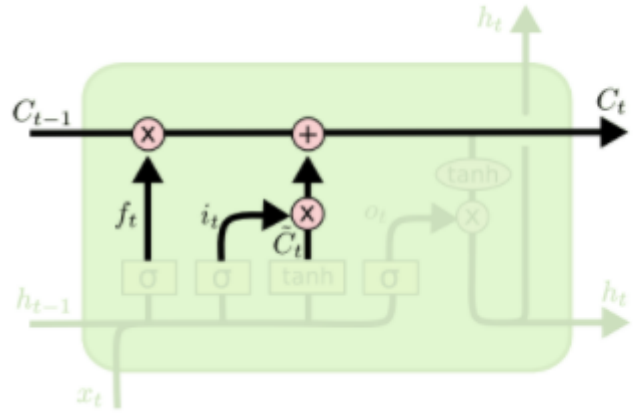
A Walk-through of LSTM Networks

- Updating the cell state: using the input gate



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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

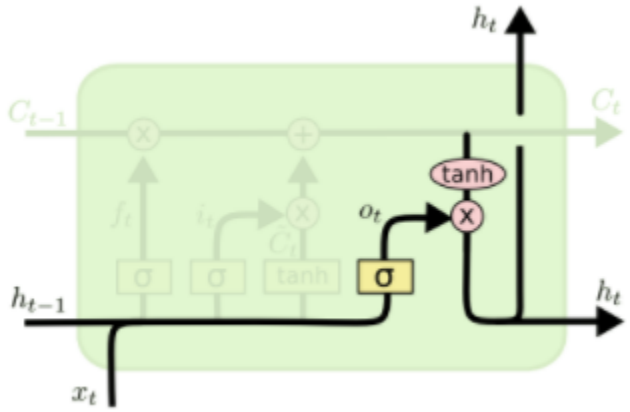


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

From Colah's blog

A Walk-through of LSTM Networks

- The output gate:
  Again non-linear transformations



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

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

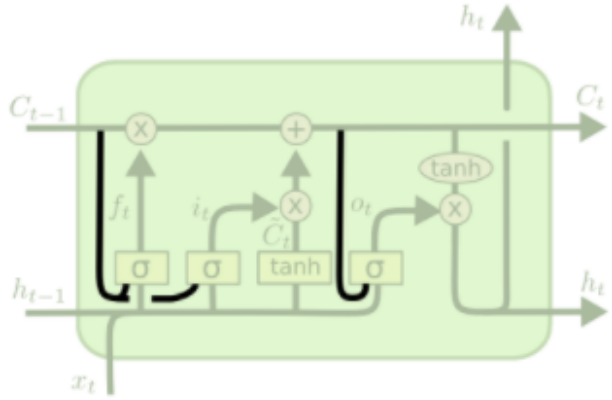
From Colah's blog

Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

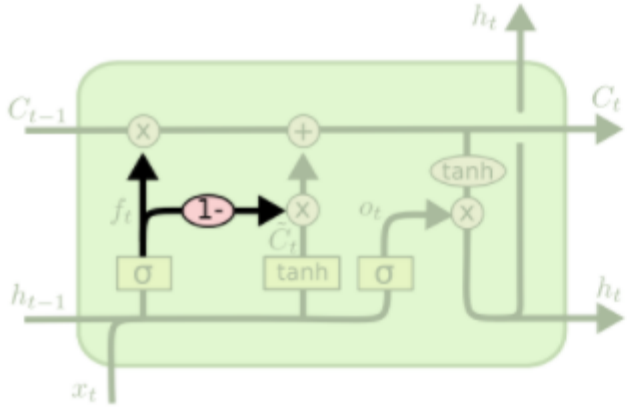
Variants on LSTM Networks

- “Peephole connections” – gate layers look at the cell state



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

- Coupling forget and input gates

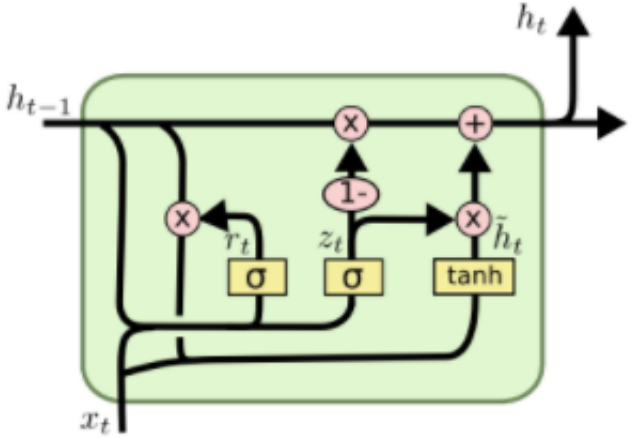


$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

From Colah’s blog

Gated Recurrent Unit (GRU) Networks

- A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit or GRU network
- It combines the forget and input gate into a single “update gate”
- It also merges the cell state and hidden state
- The resulting model is simpler than standard LSTM with slightly lower accuracy



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

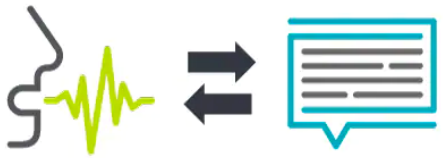
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

From Colah’s blog

Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

Core Applications of Natural Language Processing



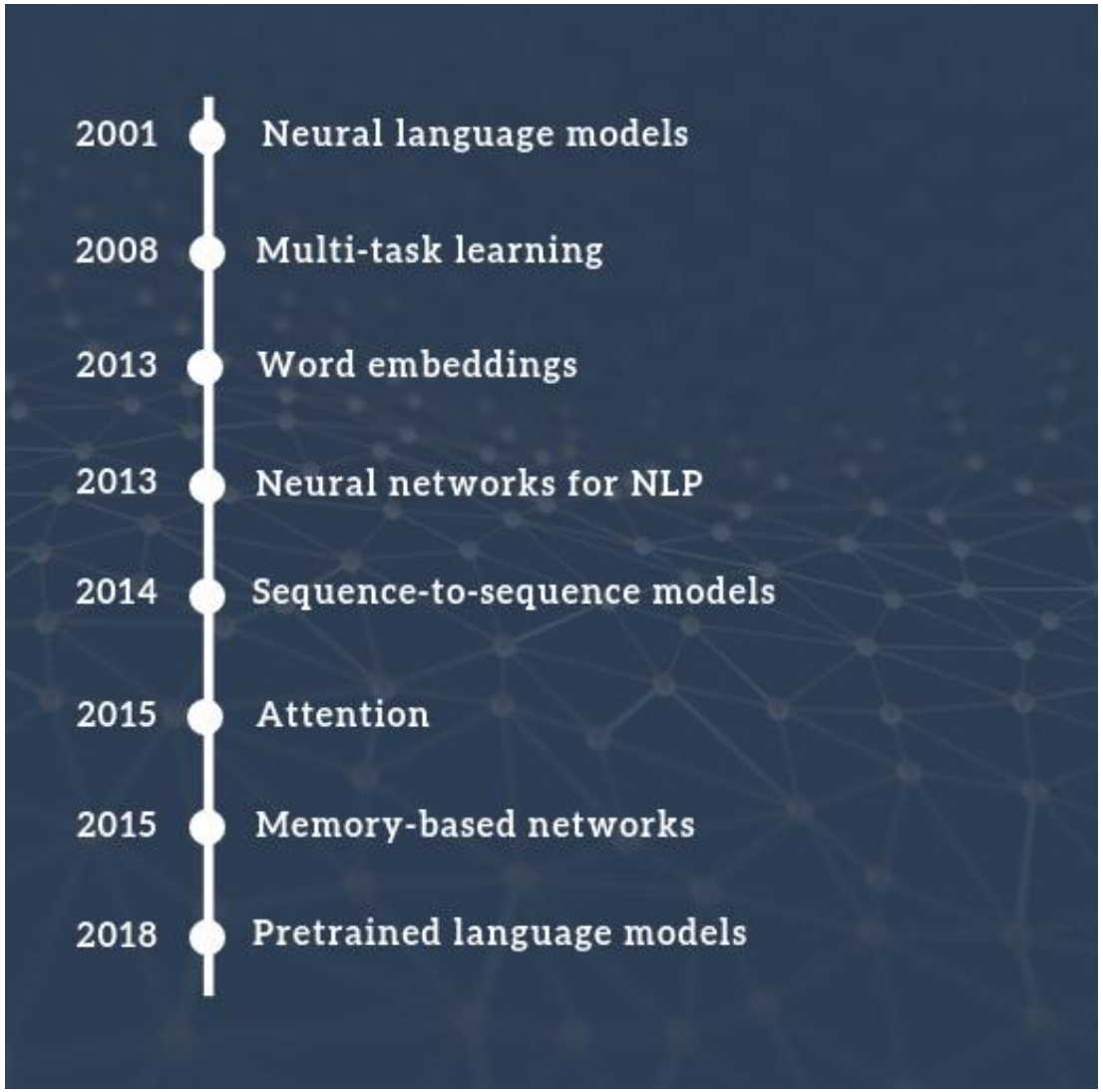
Speech to text/ text to speech (e.g., voice bots)

Text processing and language generation (e.g., text analytics, chatbots)

Machine translation

Source: Deloitte analysis.

Natural Language Processing (NLP) method History



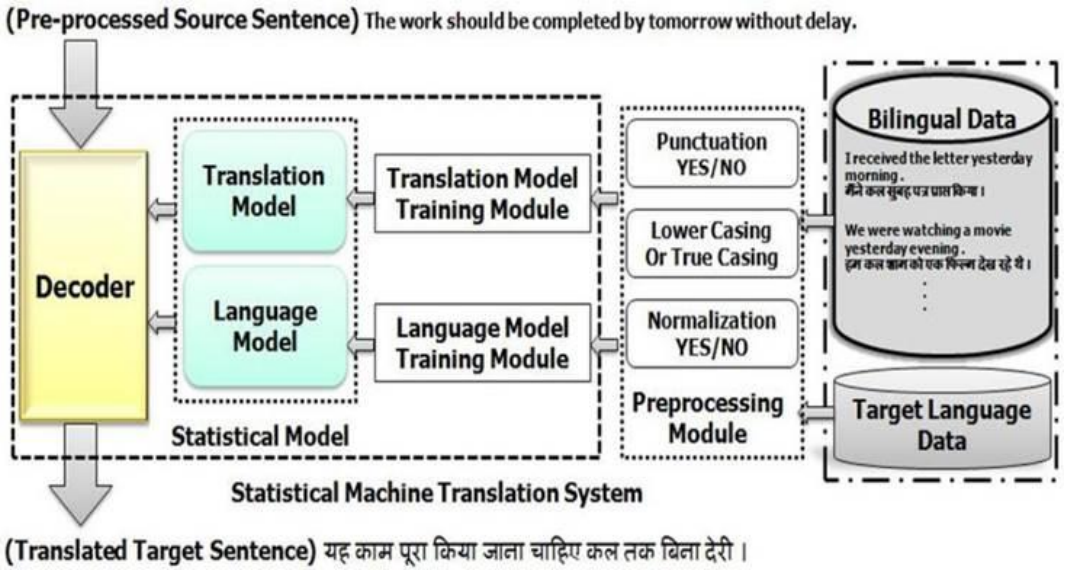
https://ruder.io/a-review-of-the-recent-history-of-nlp/

Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

Statistical Machine Translation (SMT)

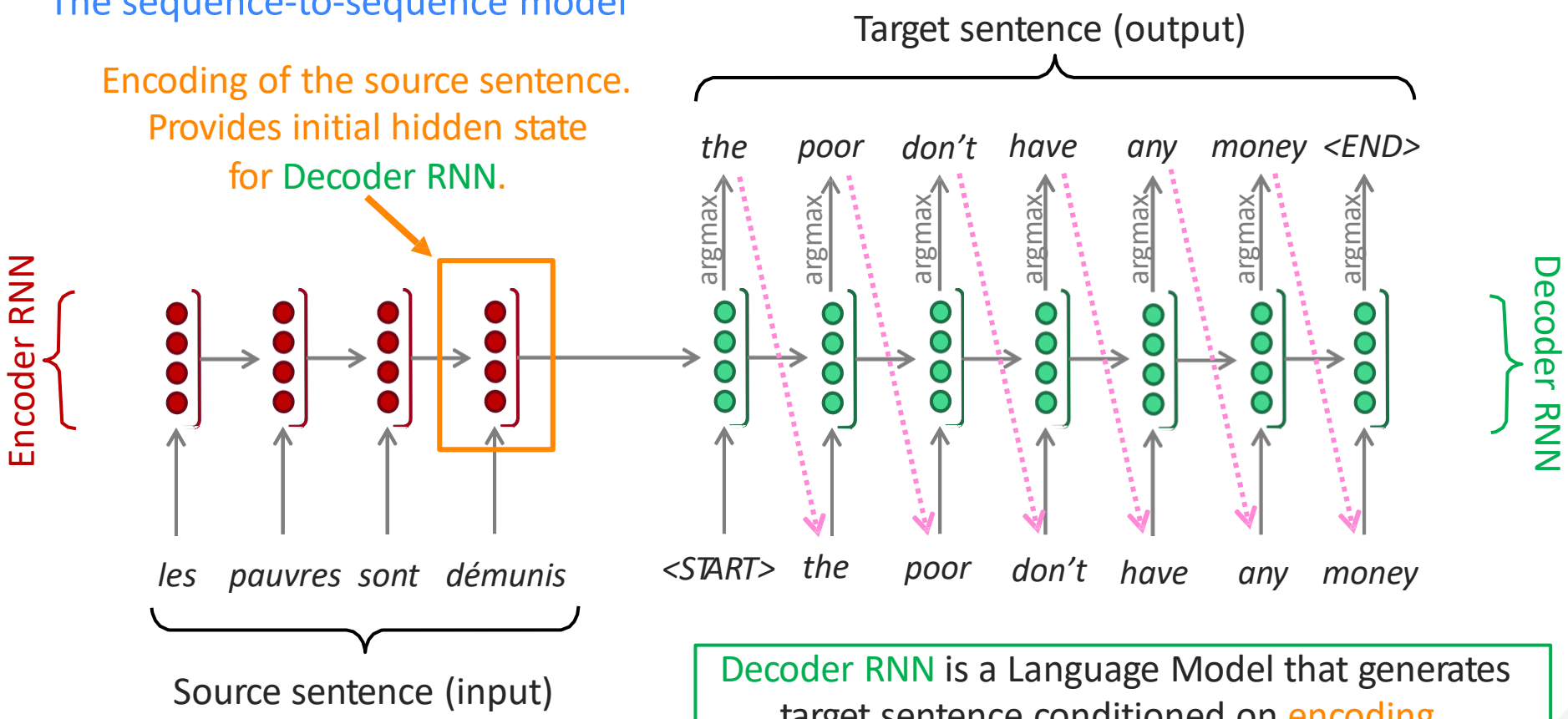
- Built on pioneering work at IBM in the 1990s
  - P. Brown & al. The mathematics of statistical machine translation: parameter estimation (1993)
  - Bayesian framework, formalized word alignment concept, etc.
- Models later extended to phrases
  - P. Koehn & al. Statistical **phrase-based translation** (2003)
  - Largely used in academia and industry since then




```
" التطورات في ||| " developments in the ||| 1 0.336365 0.461538 0.129703 2.718 ||| ||| 6 13
" التطورات في ||| " developments in ||| 1 0.336365 0.538462 0.349129 2.718 ||| ||| 7 13
" التطورات في ميدان ||| " developments in the field of ||| 0.857143 0.170757 0.5 0.017437 2.718 ||| ||| 7 12
" التطورات في ميدان ||| " developments in the field ||| 0.857143 0.170757 0.5 0.0880919 2.718 ||| ||| 7 12
" التطورات في ميدان المعلومات ||| " developments in the field of information ||| 0.857143 0.0726128 1 0.0158491 2.718 ||| ||| 7 6
" التطورات في ميدان ||| " developments in the field of ||| 0.142857 0.000431254 0.5 0.0172767 2.718 ||| ||| 7 2
" التطورات في ميدان ||| " developments in the field ||| 0.142857 0.000431254 0.5 0.0872822 2.718 ||| ||| 7 2
" التطورات في ميدان المعلومات ||| " developments in the field of information ||| 0.142857 0.000183387 1 0.0157034 2.718 ||| ||| 7 1
" التعاون ||| " cooperation ||| 0.965517 0.619446 0.982456 0.894784 2.718 ||| ||| 58 57
" التعاون ||| " enhanced ||| 1 0.0100982 0.0175439 0.00079395 2.718 ||| ||| 1 57
" التعاون الإقليمي " ||| " enhanced regional " ||| 1 0.00315024 1 0.000574123 2.718 ||| ||| 1 1
" تدرج كلمة التعاون الإقليمي " ||| " enhanced regional " , insert " ||| 1 1.28402e-05 1 3.54351e-06 2.718 ||| ||| 1 1
" تدرج كلمة التعاون الإقليمي ||| " enhanced regional " , insert ||| 1 1.3527e-05 1 3.6022e-06 2.718 ||| ||| 1 1
" التعاون الإقليمي ||| " enhanced regional ||| 1 0.00331873 1 0.000583632 2.718 ||| ||| 1 1
" التعاون الدولي ||| " international cooperation ||| 0.5 0.280061 1 0.842491 2.718 ||| ||| 2 1
" التعاون الدولي من ||| " international cooperation ||| 0.5 0.0285172 1 0.842491 2.718 ||| ||| 2 1
" التعاون الدولي من أجل ||| " international cooperation to ||| 1 0.0004685 1 0.261179 2.718 ||| ||| 1 1
" التعاون الدولي من أجل الحد من ||| " international cooperation to reduce ||| 1 8.01874e-06 1 0.0115687 2.718 ||| ||| 1 1
" التعاون بين ||| " cooperation between the ||| 0.97619 0.573927 0.42268 0.166976 2.718 ||| ||| 42 97
" التعاون بين ||| " cooperation between ||| 0.965517 0.573927 0.57732 0.449459 2.718 ||| ||| 58 97
" التعاون بين الأمم المتحدة ||| " cooperation between the united nations and ||| 0.9 0.498652 0.397059 0.00836641 2.718 ||| ||| 30 68
" التعاون بين الأمم المتحدة ||| " cooperation between the united nations ||| 0.931818 0.498652 0.602941 0.1609 2.718 ||| ||| 44 68
" التعاون بين الأمم المتحدة والمنظمة ||| " cooperation between the united nations and ||| 0.0666667 0.000262789 0.5 0.00928672 2.718 ||| ||| 30 4
" التعاون بين الأمم المتحدة والمنظمة ||| " cooperation between the united nations ||| 0.0454545 0.000262789 0.5 0.178599 2.718 ||| ||| 44 4
" التعاون بين منظومة ||| " cooperation between ||| 0.0172414 5.217e-05 1 0.449459 2.718 ||| ||| 58 1
" التعـــاون ||| " cooperation ||| 0.0172414 0.00164891 1 0.73778 2.718 ||| ||| 58 1
```

Neural Machine Translation (NMT)

The sequence-to-sequence model

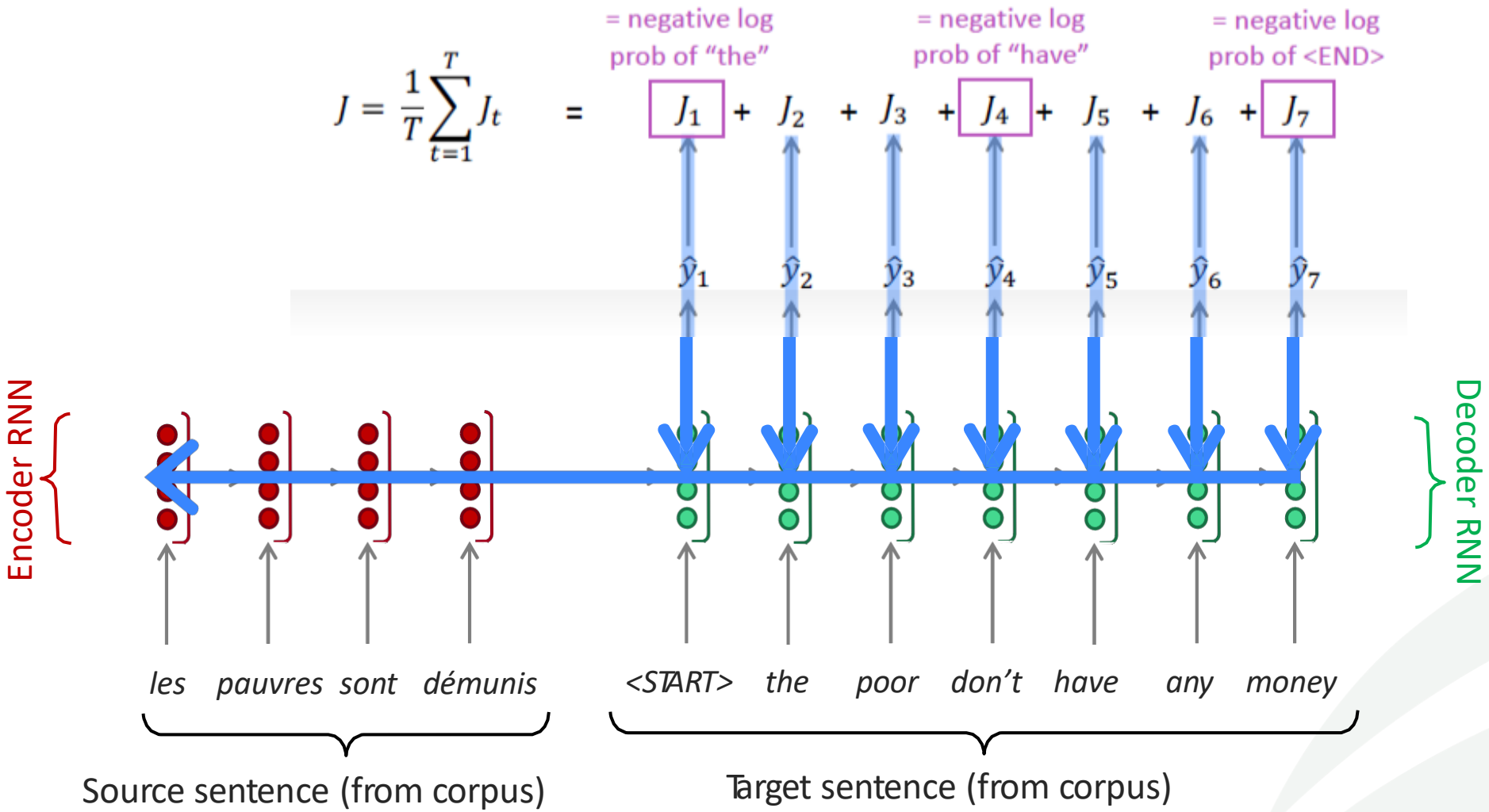


Decoder RNN is a Language Model that generates target sentence conditioned on encoding.

Encoder RNN produces an encoding of the source sentence.

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

Training an NMT System



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

Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

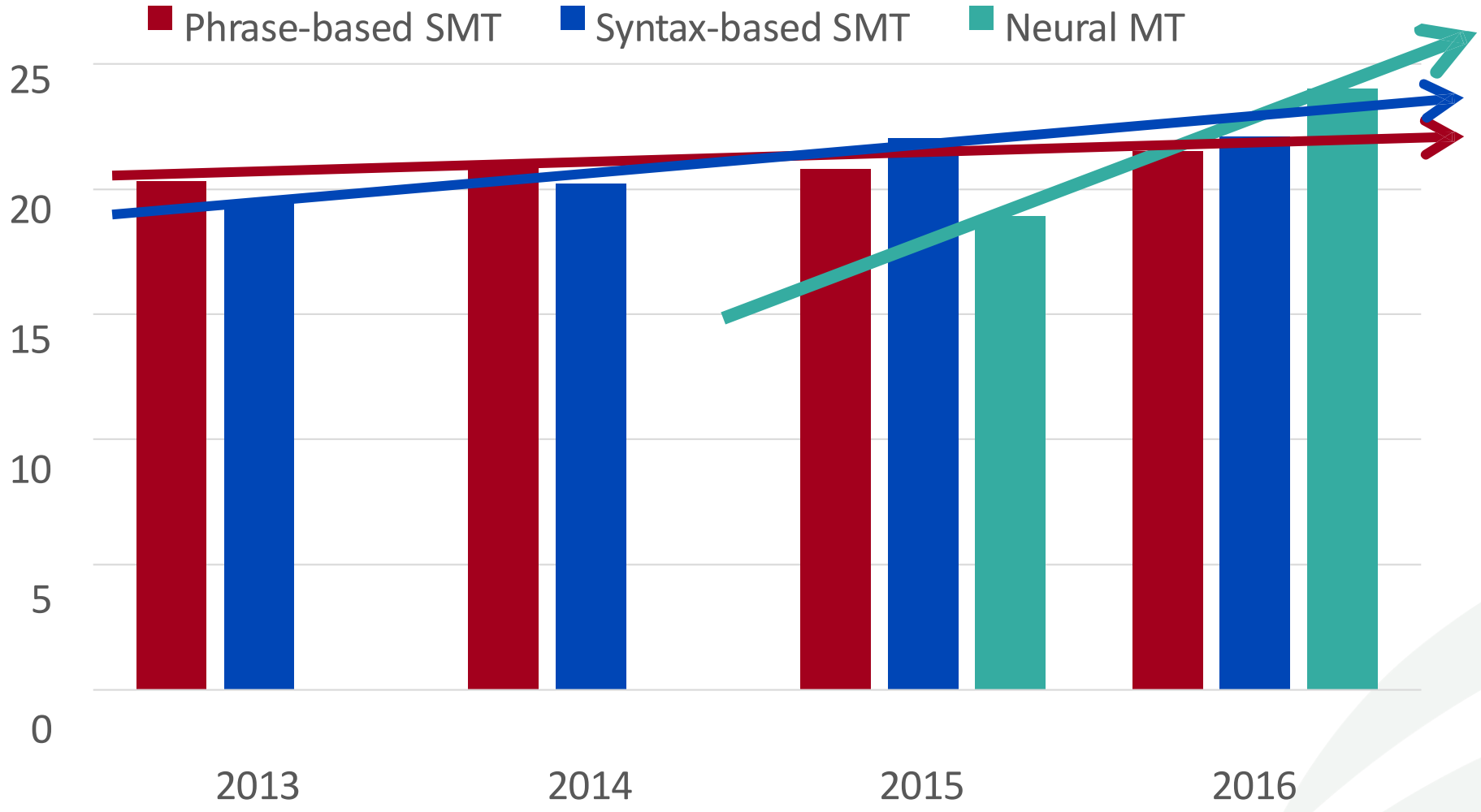
Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs

MT progress over time

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]



Source: http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf

NMT: the Biggest Success Story of NLP Deep Learning

Neural Machine Translation went from a fringe research activity in **2014** to the leading standard method in **2016**

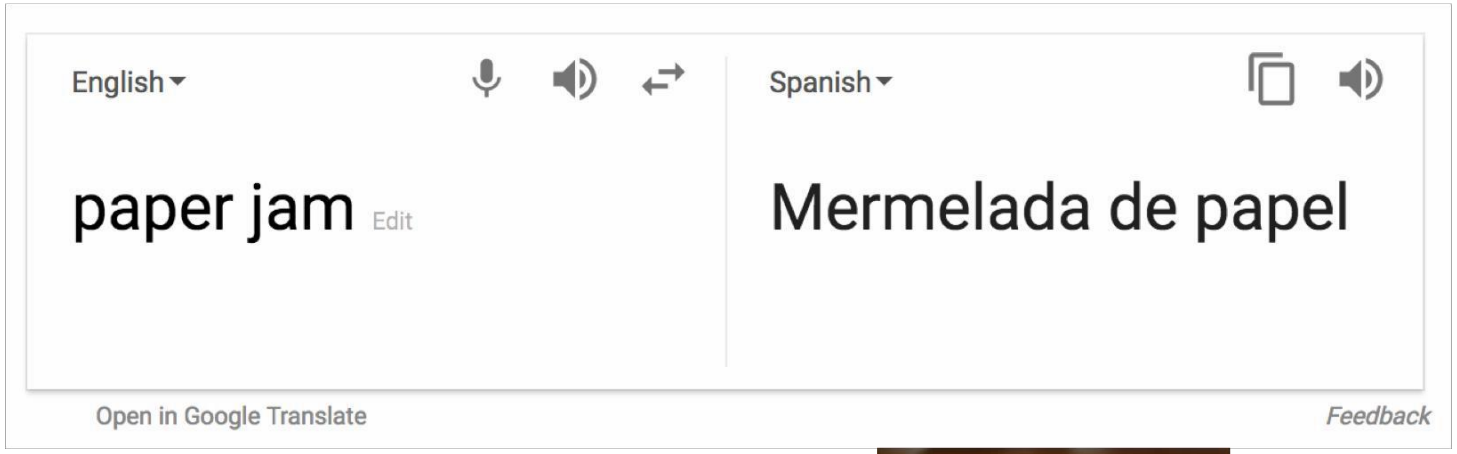
- **2014**: First seq2seq paper published
- **2016**: Google Translate switches from SMT to NMT
- This is amazing!
  - **SMT** systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a handful of engineers in a few months

So is Machine Translation Solved?

- **Nope!**
- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over longer text
  - Low-resource language pairs

So is Machine Translation Solved?

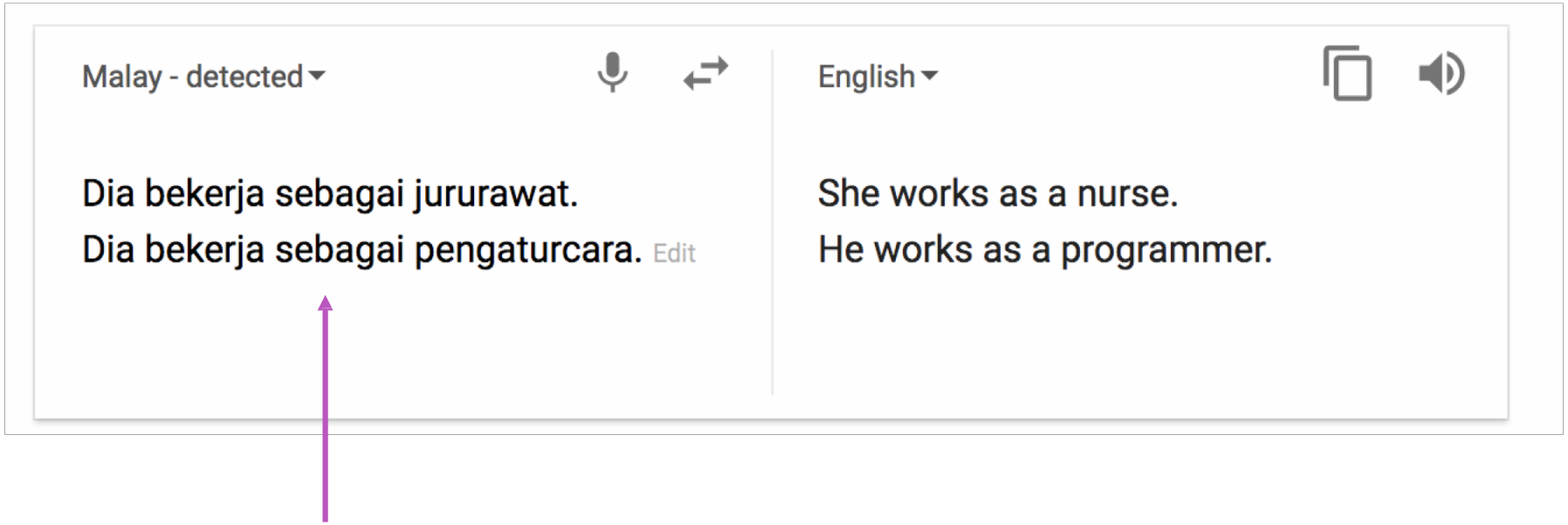
- **Nope!**
- Using common sense is still hard



?

So is Machine Translation Solved?

- **Nope!**
- NMT picks up biases in training data

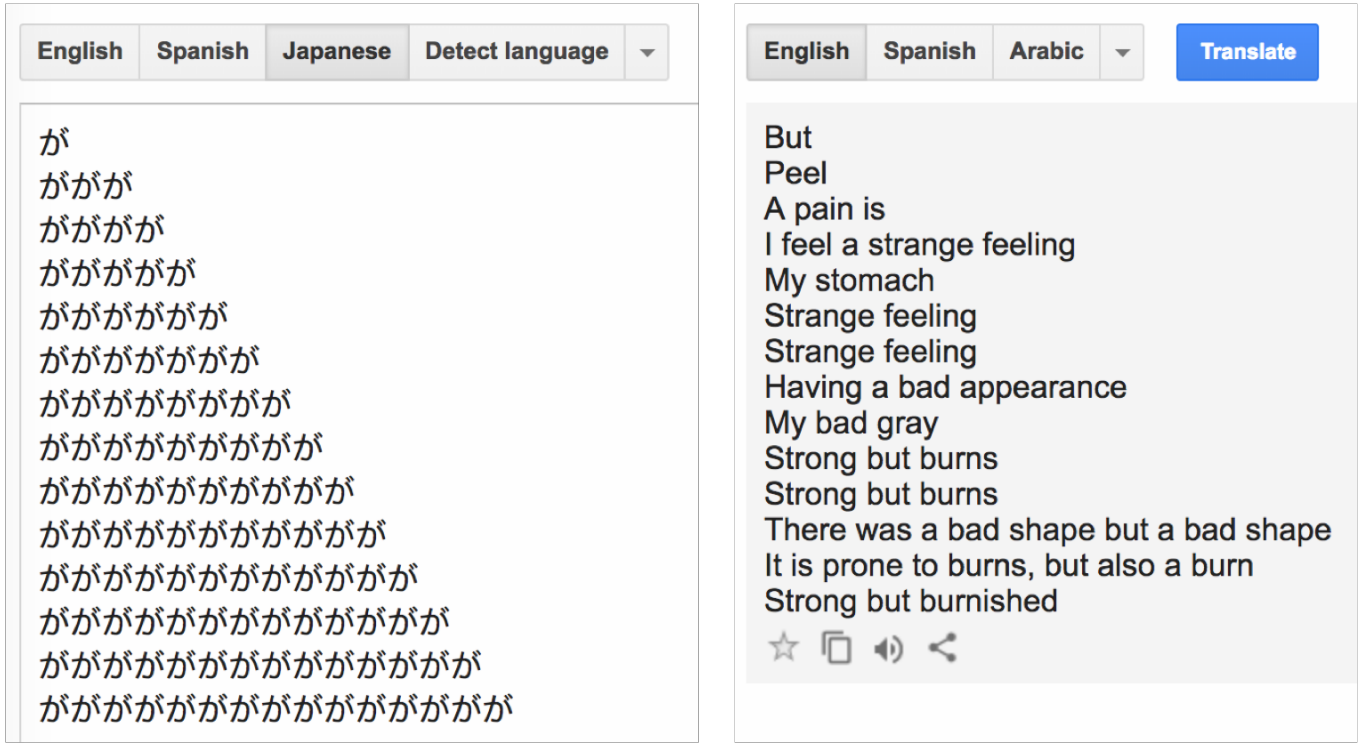


Didn't specify gender

Source: https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

So is Machine Translation Solved?

- Nope!
- Uninterpretable systems do strange things



Source: http://languagelog.ldc.upenn.edu/nll/?p=35120#more-35120

NMT Research Continues

NMT is the **flagship task** for NLP Deep Learning

- NMT research has pioneered many of the recent innovations of NLP Deep Learning

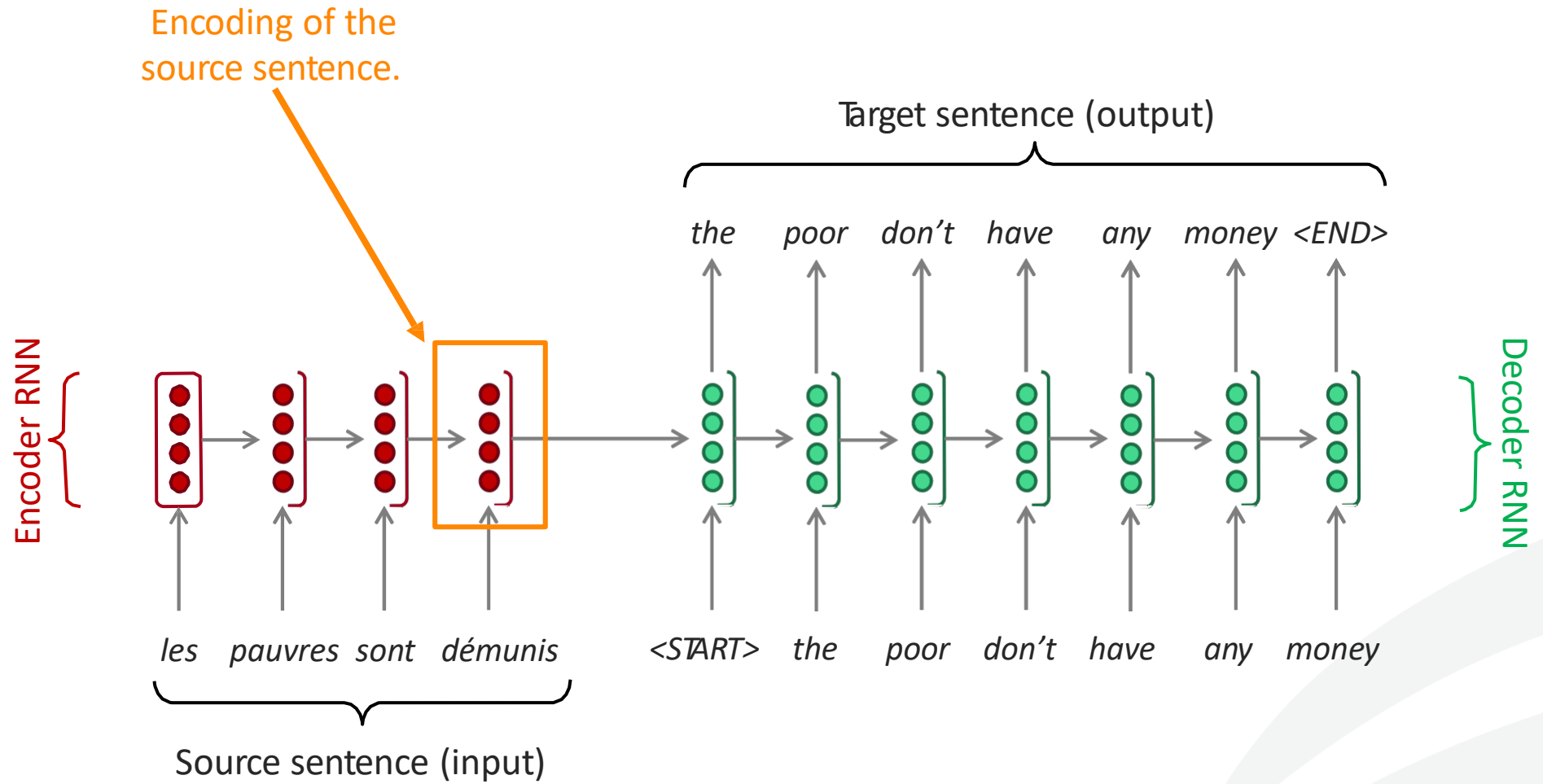
- In **2018**: NMT research continues to thrive
  - Researchers have found *many, many* improvements to the "vanilla" seq2seq NMT system we've presented today
  - But one improvement is so integral that it is the new vanilla…

ATTENTION

Agenda

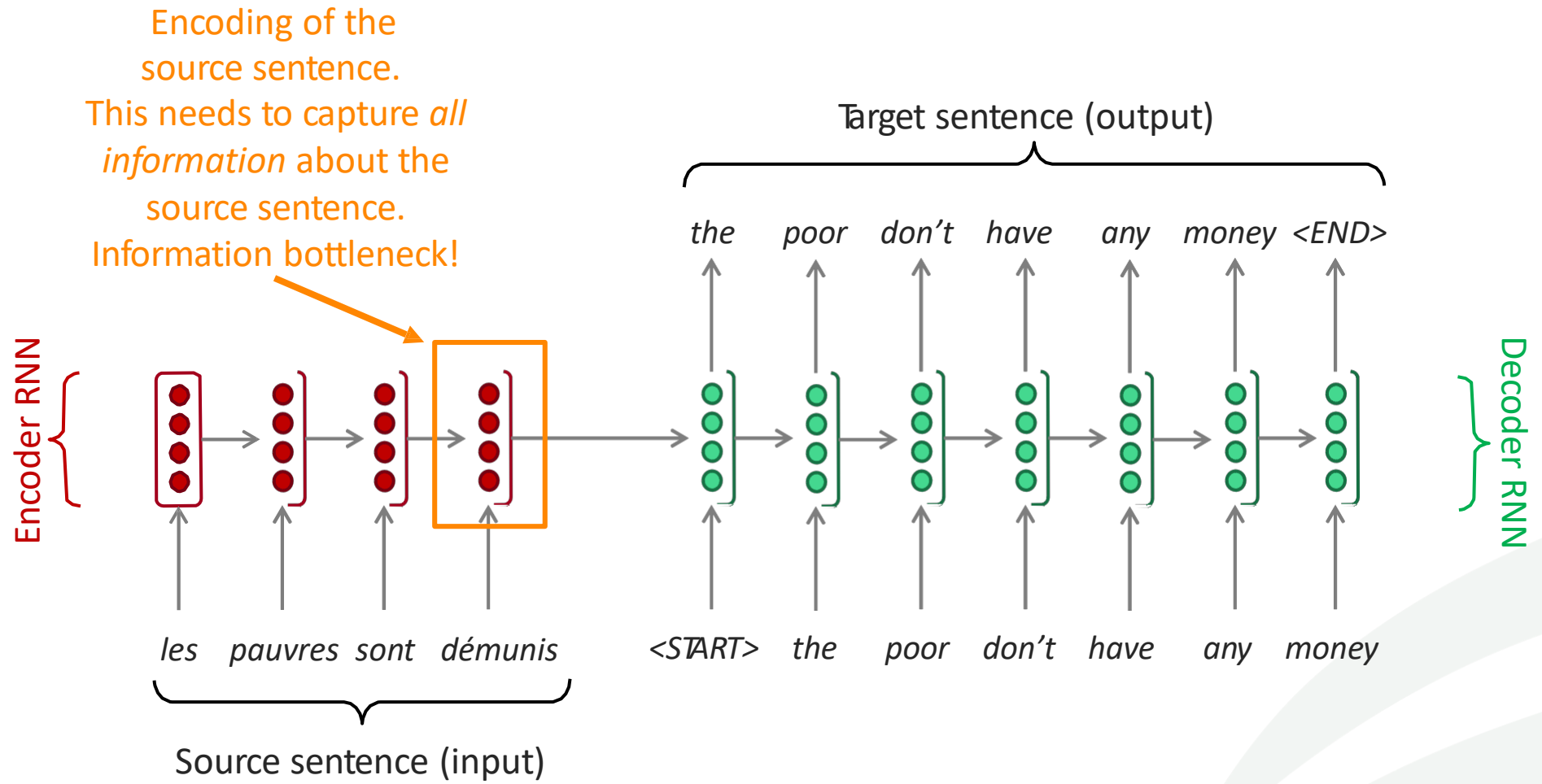
- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

Sequence-to-Sequence: Bottleneck



Problems with this architecture?

Sequence-to-Sequence: Bottleneck



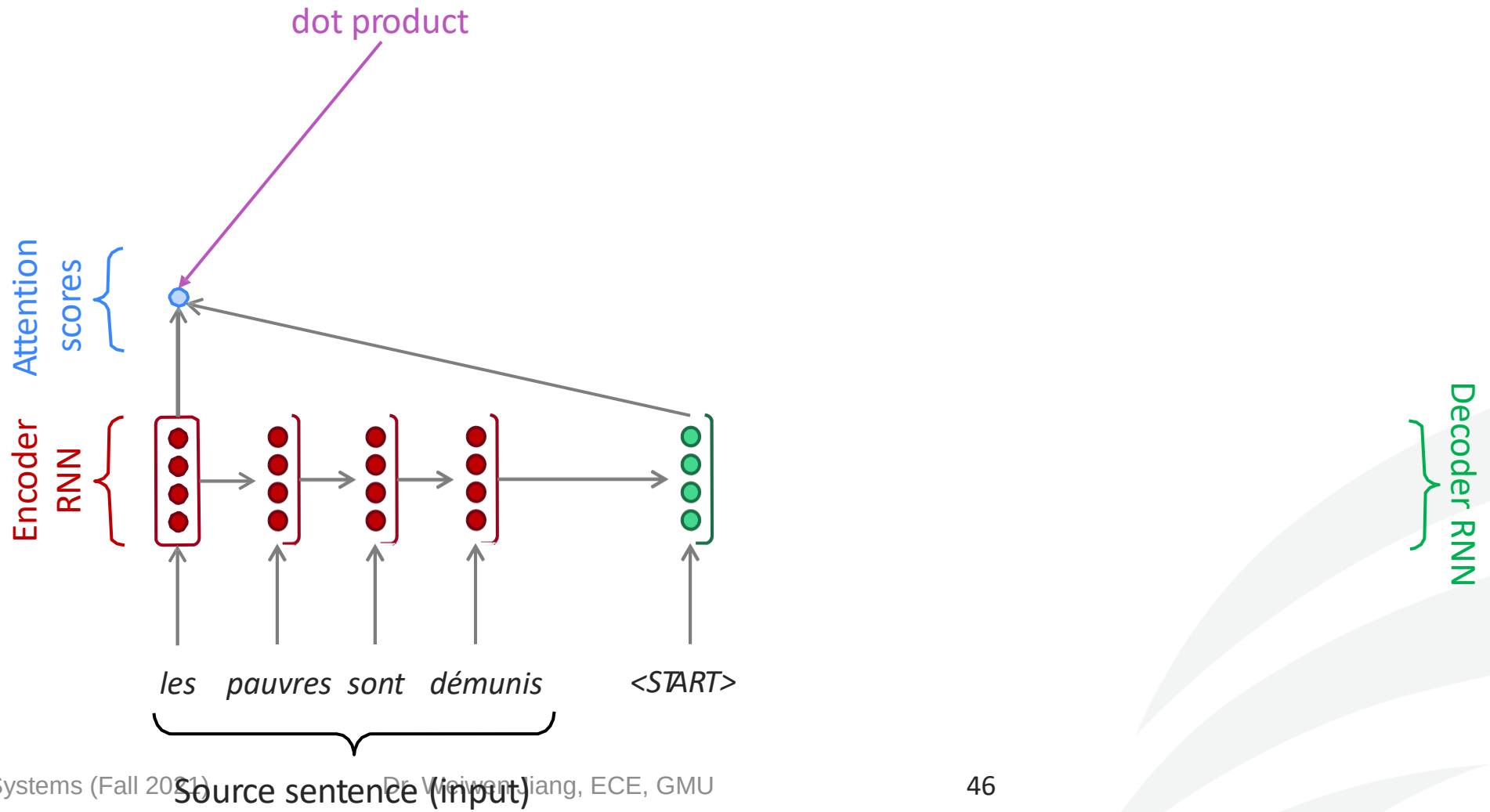
Attention

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

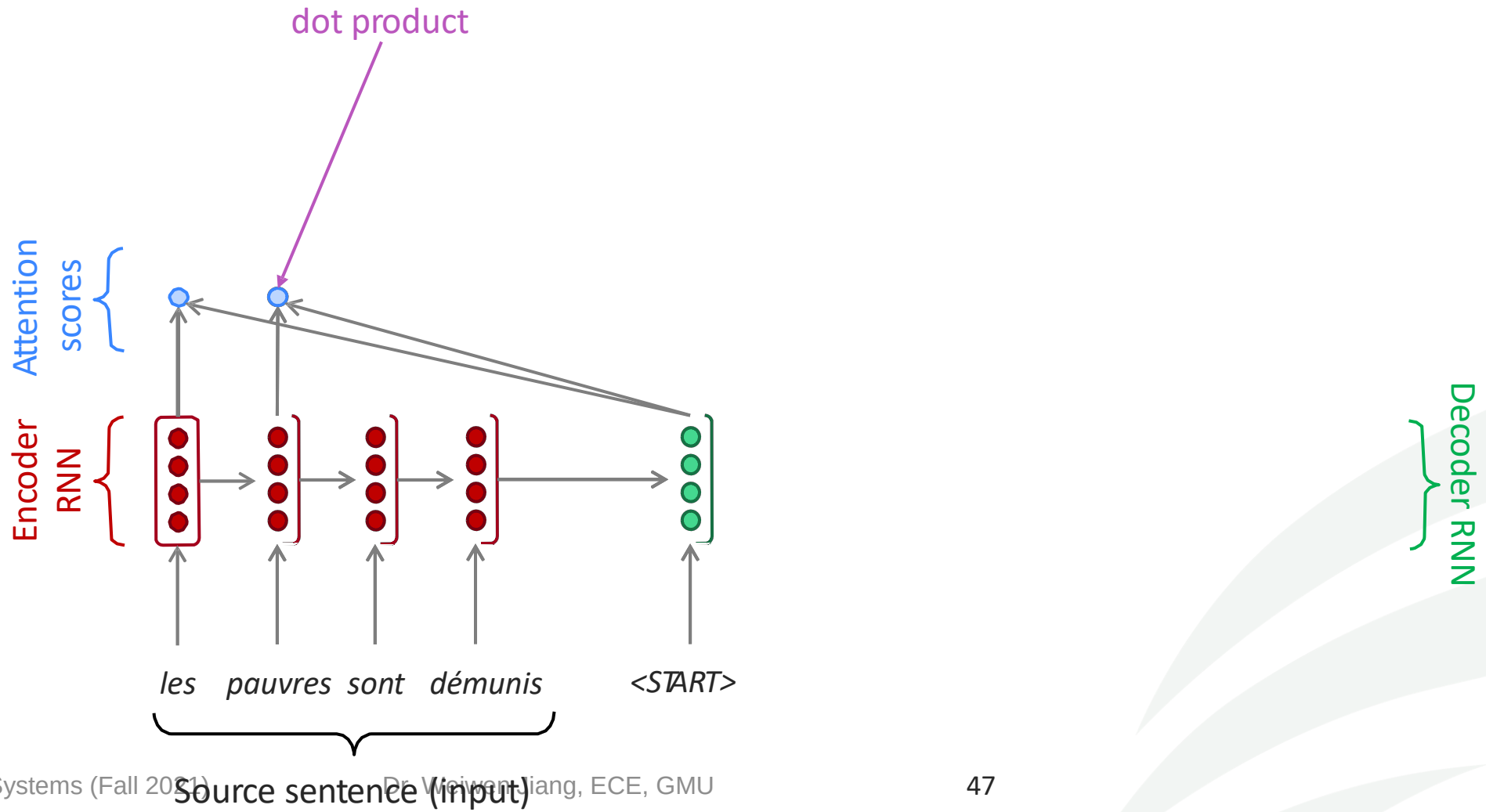
- Core idea: on each step of the decoder, *focus on a particular part* of the source sequence

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

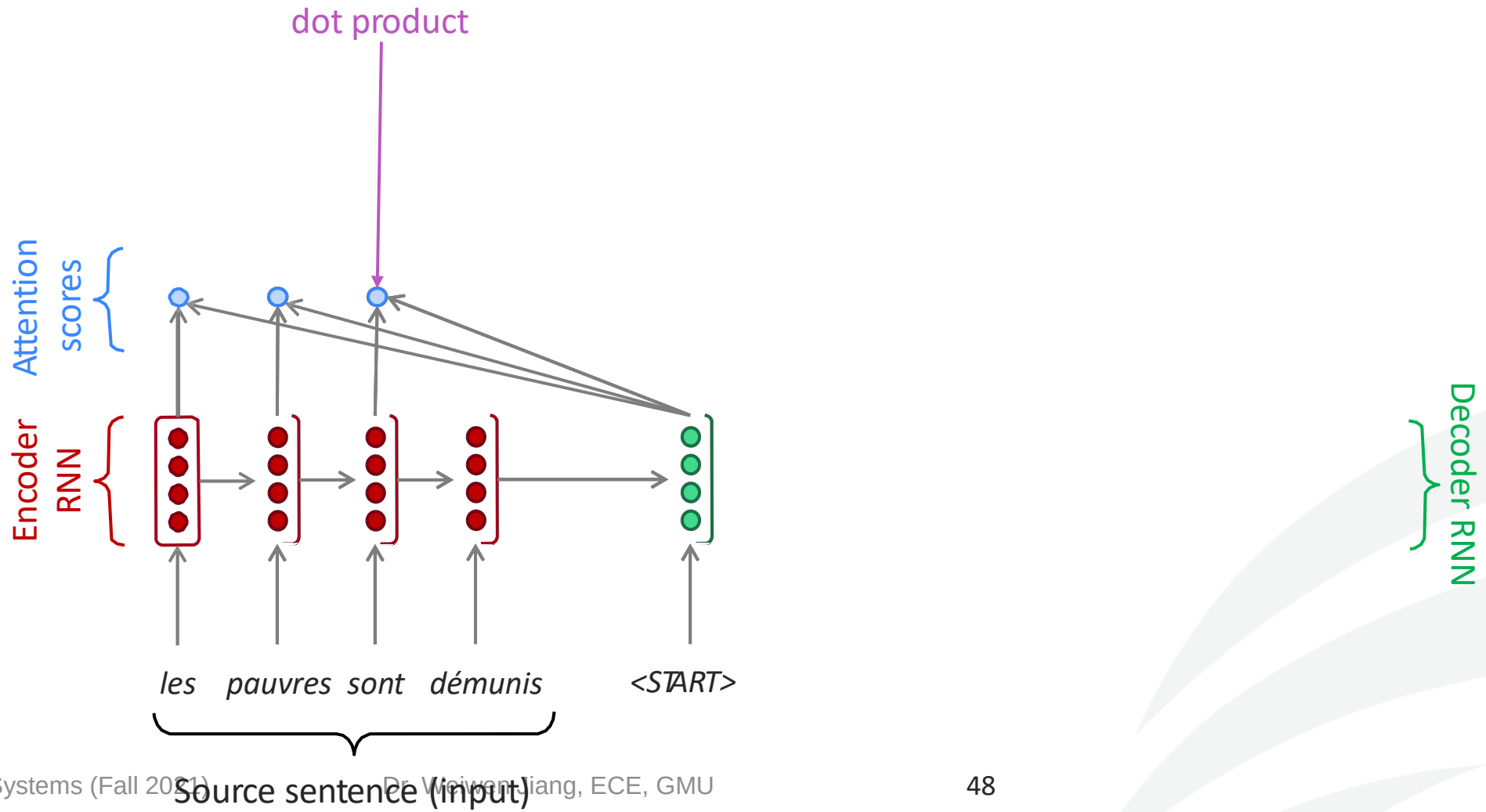
Sequence-to-Sequence with Attention



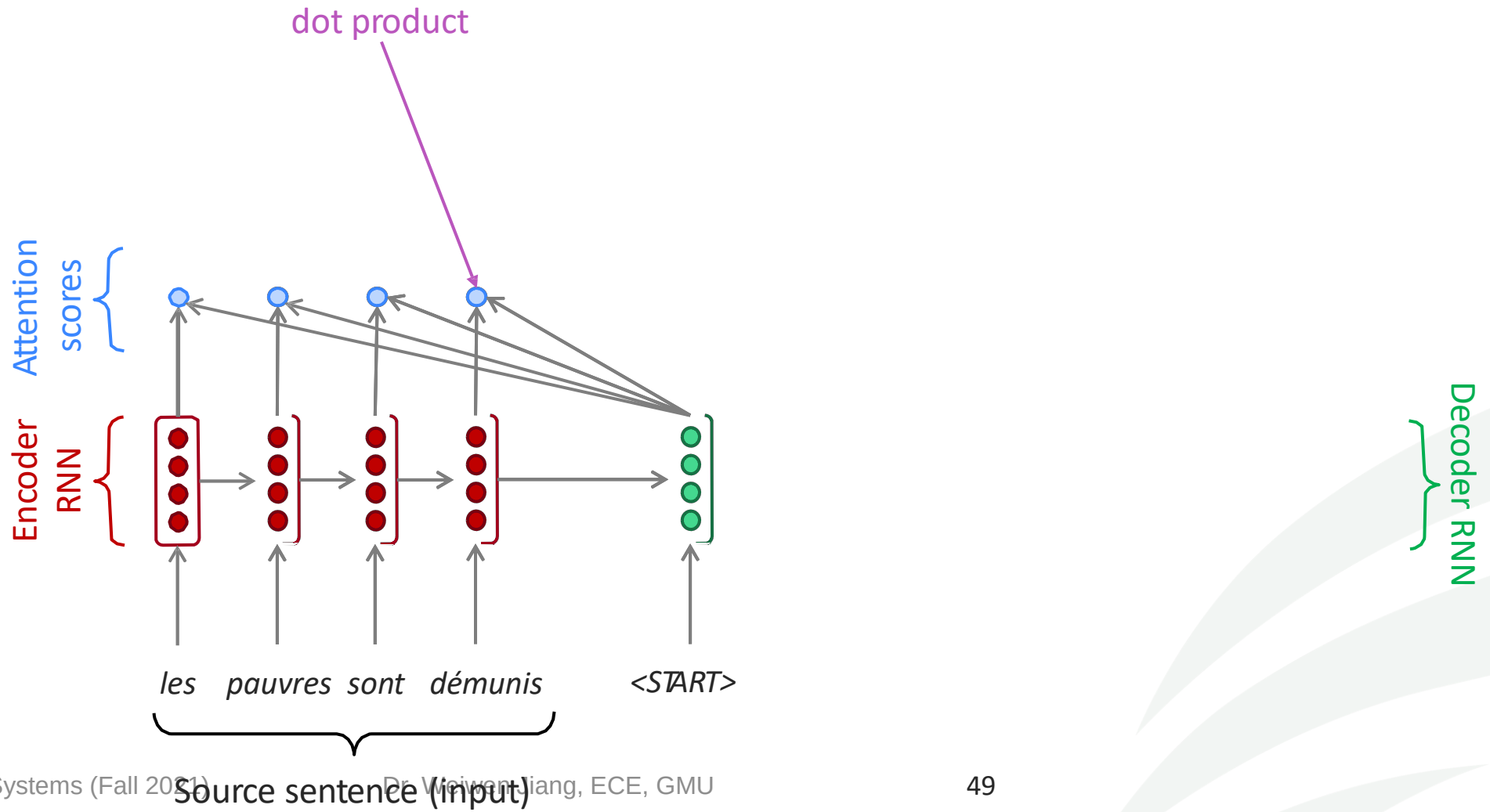
Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention

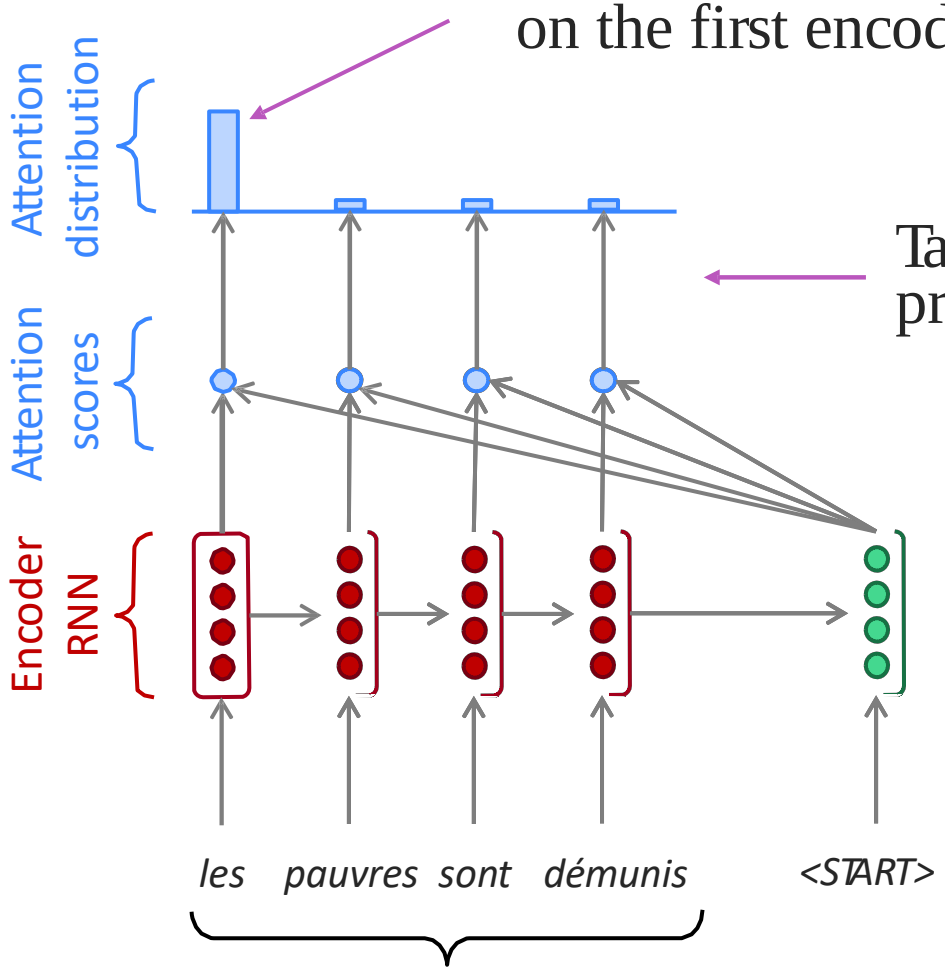


Sequence-to-Sequence with Attention



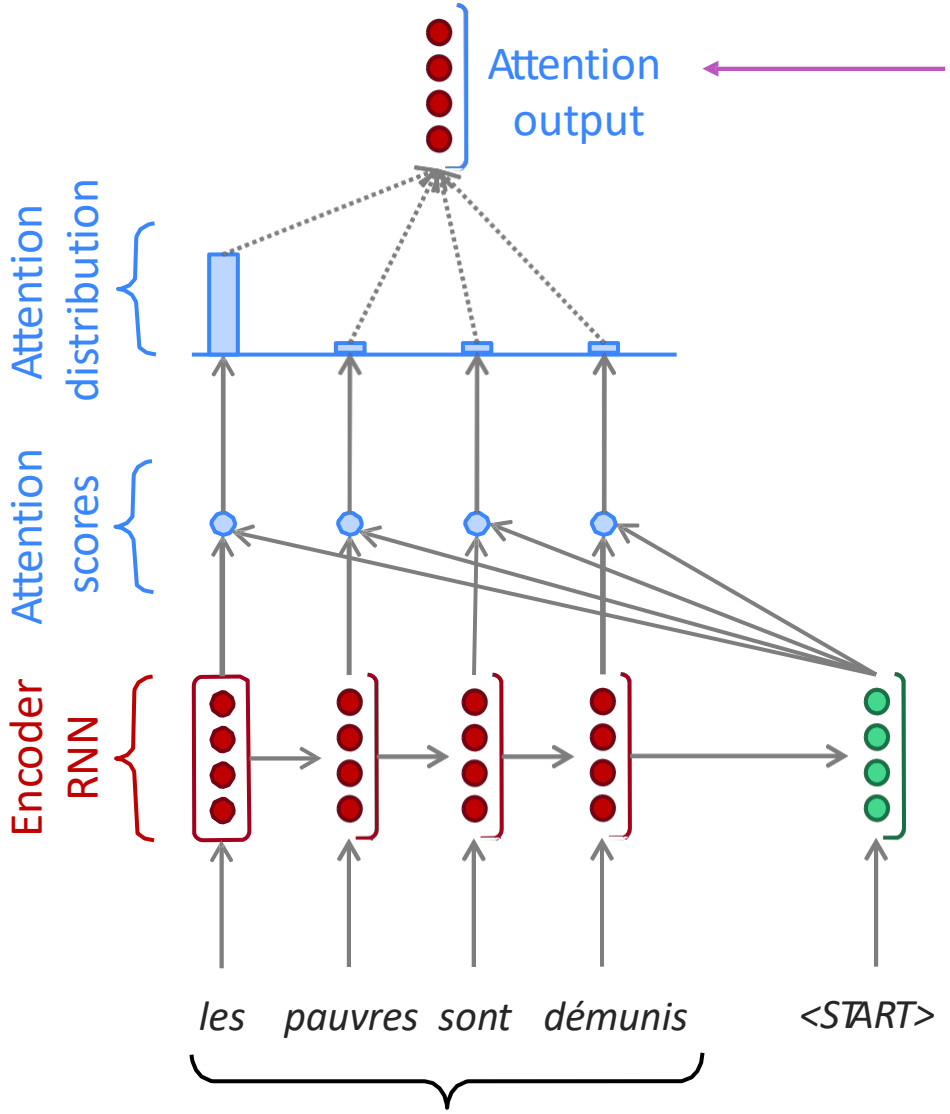
Sequence-to-Sequence with Attention

On this decoder timestep, we're mostly focusing on the first encoder hidden state (*"les"*)



Take softmax to turn the scores into a probability distribution

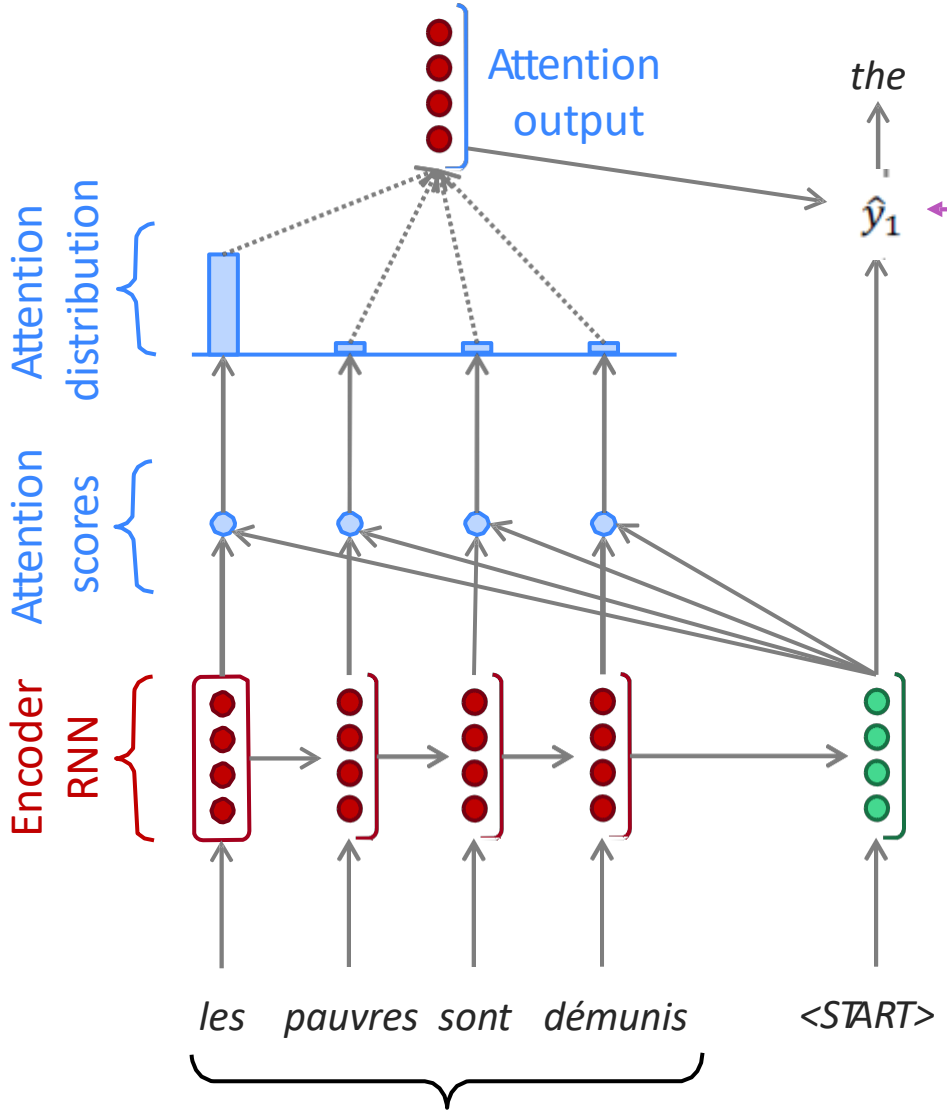
Sequence-to-Sequence with Attention



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

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

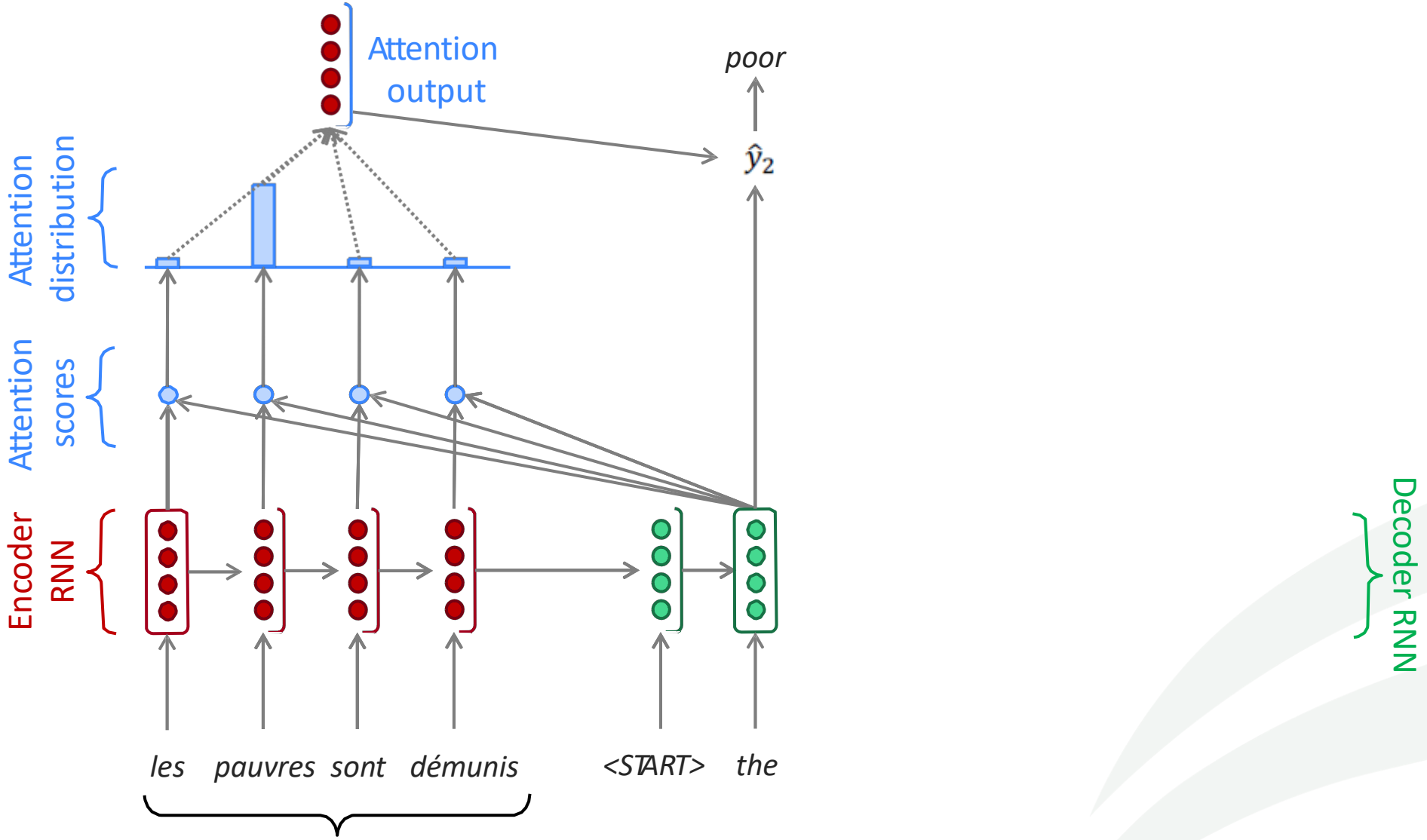
Sequence-to-Sequence with Attention



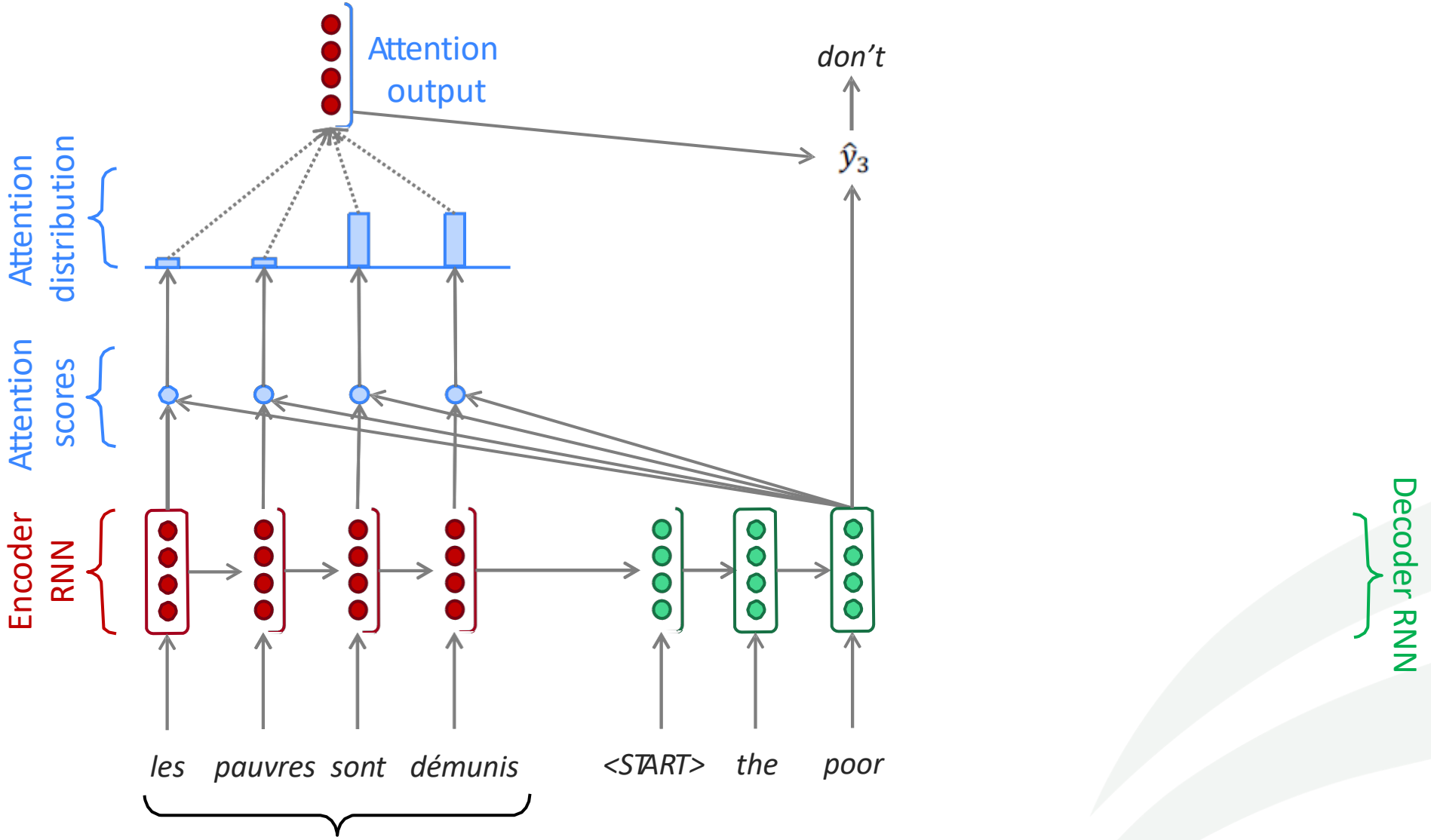
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Decoder RNN

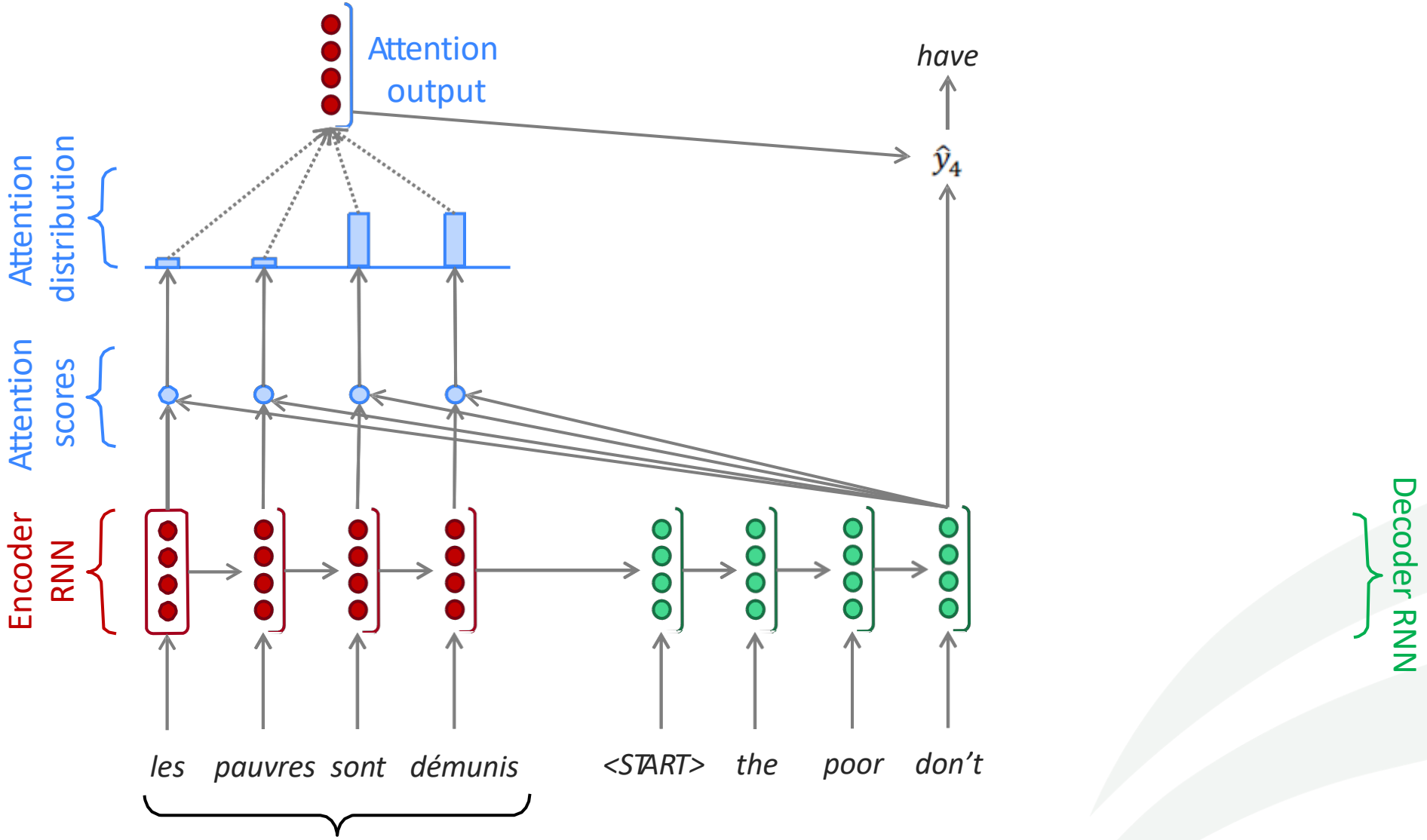
Sequence-to-sequence with attention



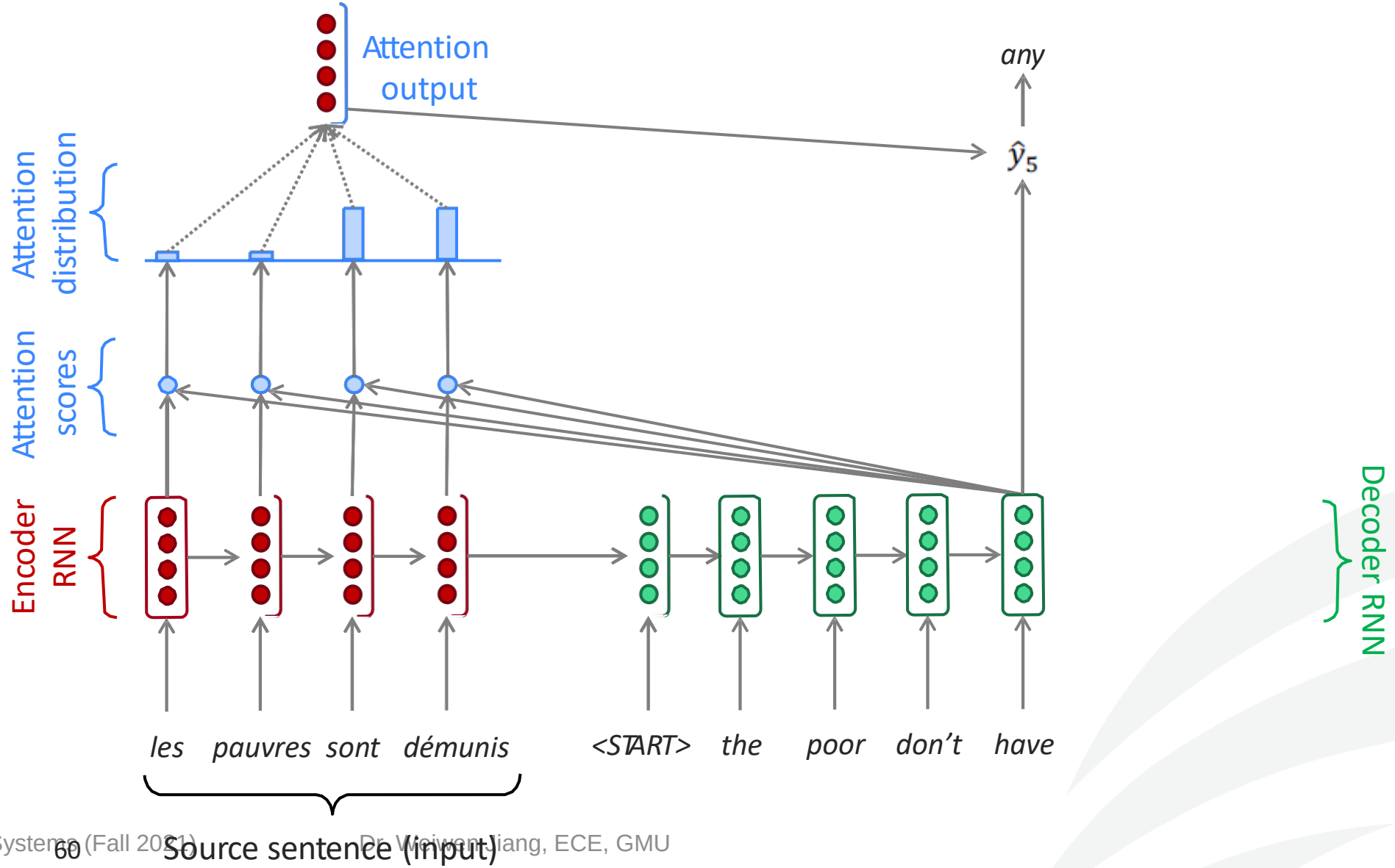
Sequence-to-Sequence with Attention



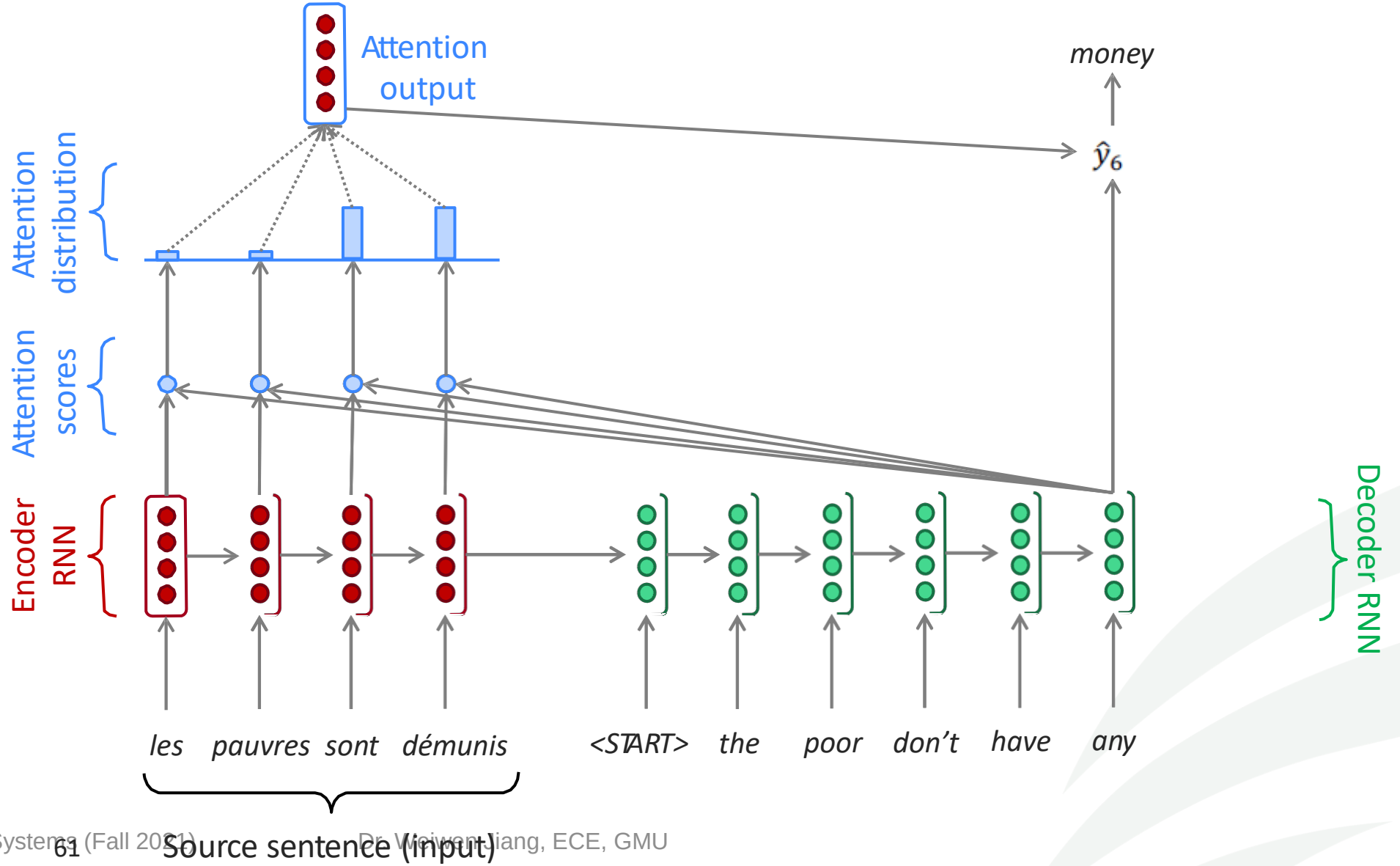
Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



Attention: in Equations

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

$$e^t = [s_t^T h_1, \ldots, s_t^T h_N] \in \mathbb{R}^N$$

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

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

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output a_t

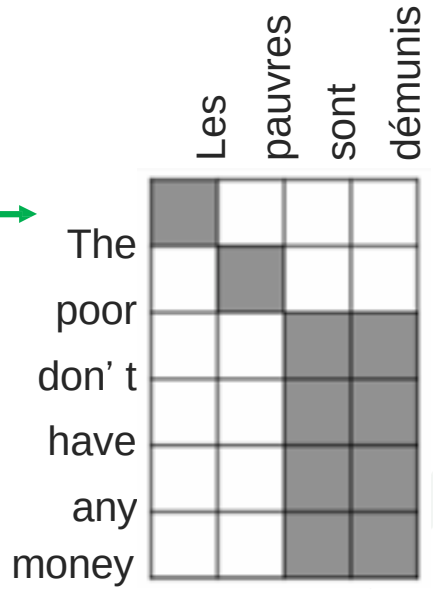
$$a_t = \sum_{i=1}^{N} \alpha_i^t h_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output a_t with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[a_t; s_t] \in \mathbb{R}^{2h}$$

Attention is Great

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



Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

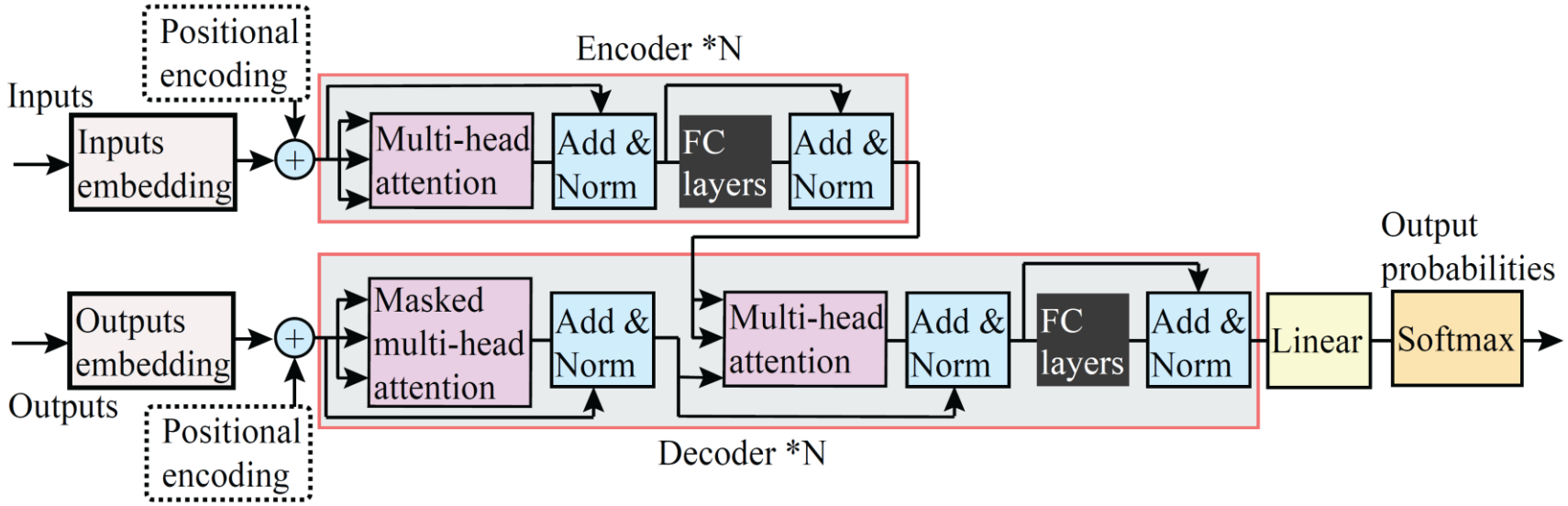
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

The Transformer was proposed in the paper Attention is All You Need

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

Transformer

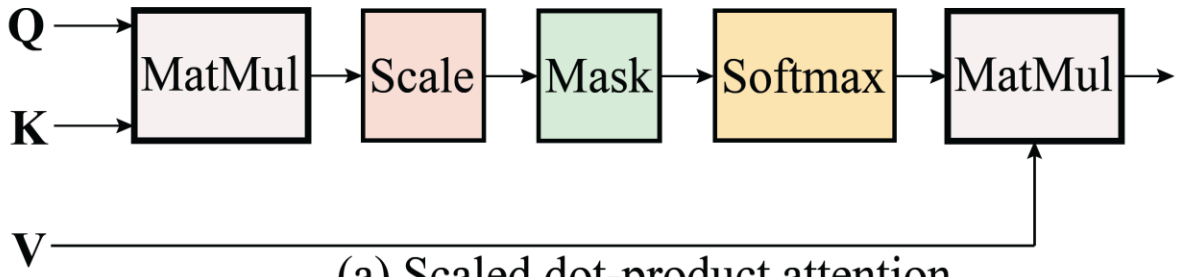


Encoder: a stack of N = 6 identical layers.
Each layer has two sub-layers.

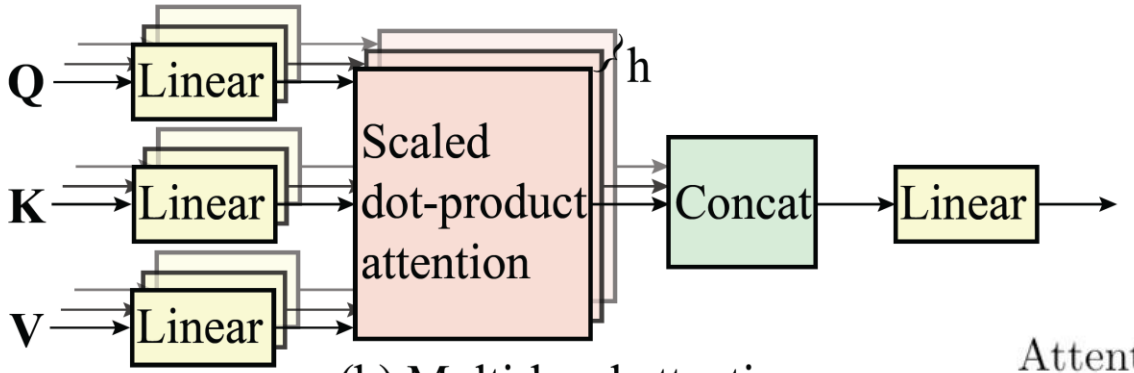
- a **multi-head self-attention** mechanism,
- position wise **fully connected feed-forward** network.
- residual connections.

Decoder: a stack of N = 6 identical layers.
In addition to the two sub-layers in each encoder layer, the decoder inserts a **third sub-layer.**

Transformer



(a) Scaled dot-product attention



(b) Multi-head attention

Attention:

- mapping a query and a set of key-value pairs to an output, where the **query, keys, values**, and output are all vectors.
- The output is computed as a weighted sum of the values.

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

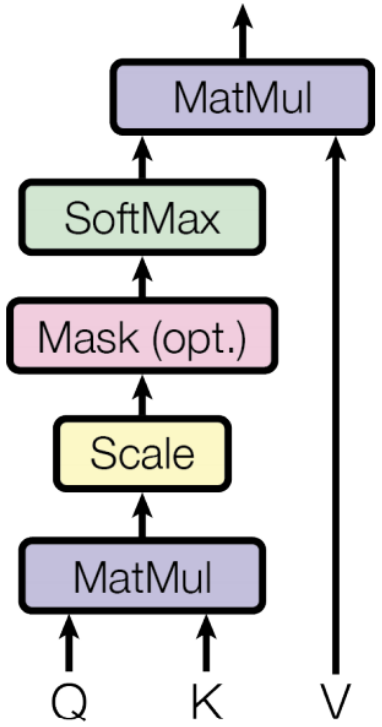
Multi single-head attention are then concatenated as multi-head attention, as shown in Figure 2 (b). MultiHead ($\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$) = Concat ($\text{Head}_1, \cdots, \text{Head}_h$)$\times \mathbf{W}^O$, where the Head is defined as:

$$\text{Head}_i = \textit{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (2)$$

where the projections are parameter matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, and $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$.

Attention in Transformer

Scaled Dot-Product Attention



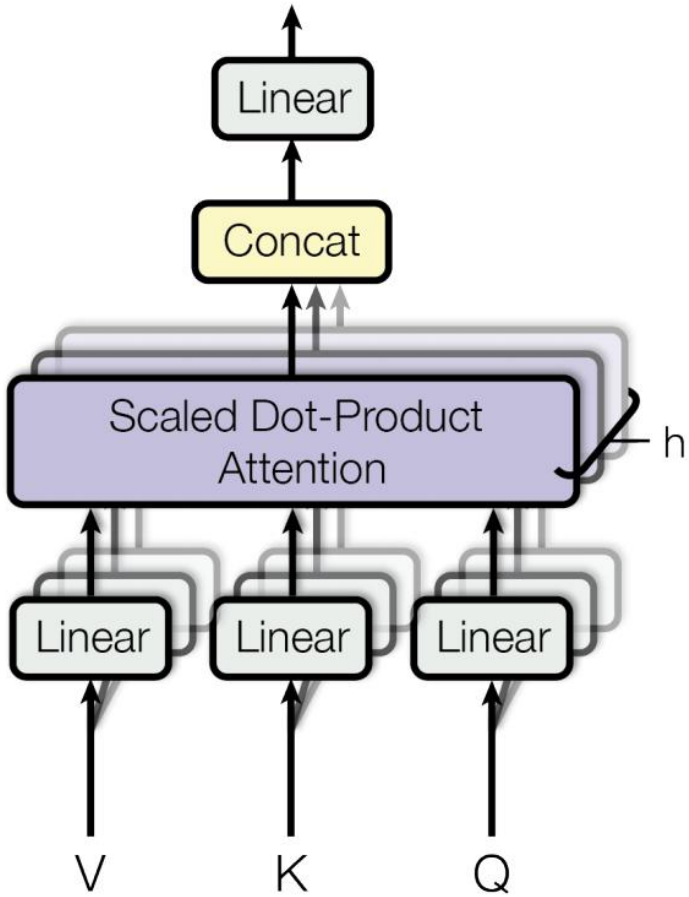
Attention:

- mapping a query and a set of key-value pairs to an output, where the **query, keys, values**, and output are all vectors.
- The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

compute the attention function on a set of queries simultaneously,

- packed together into a matrix Q.
- The keys and values are also packed together into matrices K and V .

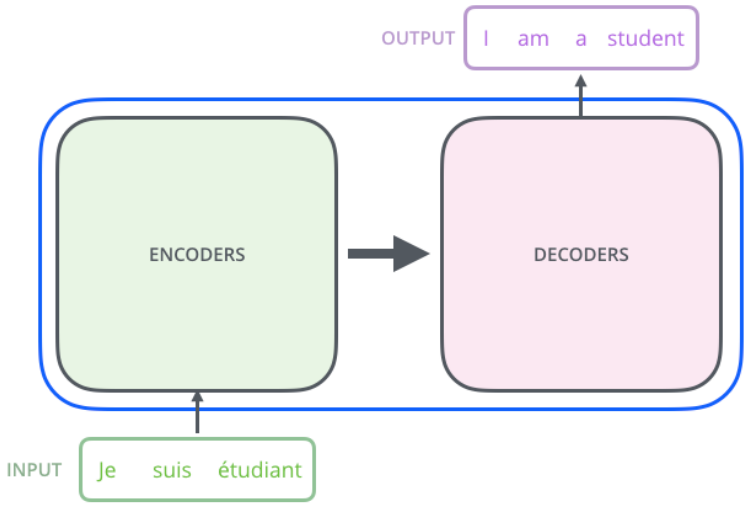
Multi-Head Attention

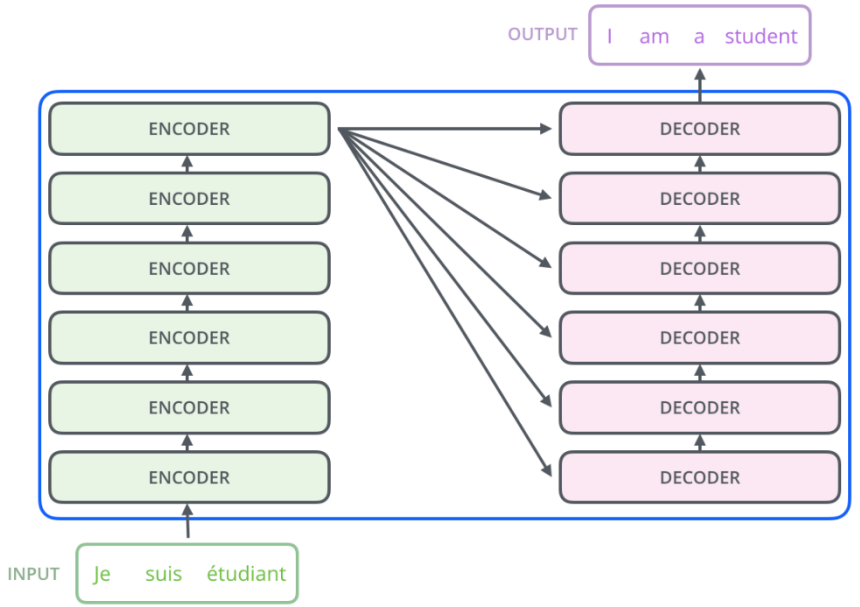


$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}$$

https://jalammar.github.io/illustrated-transformer/

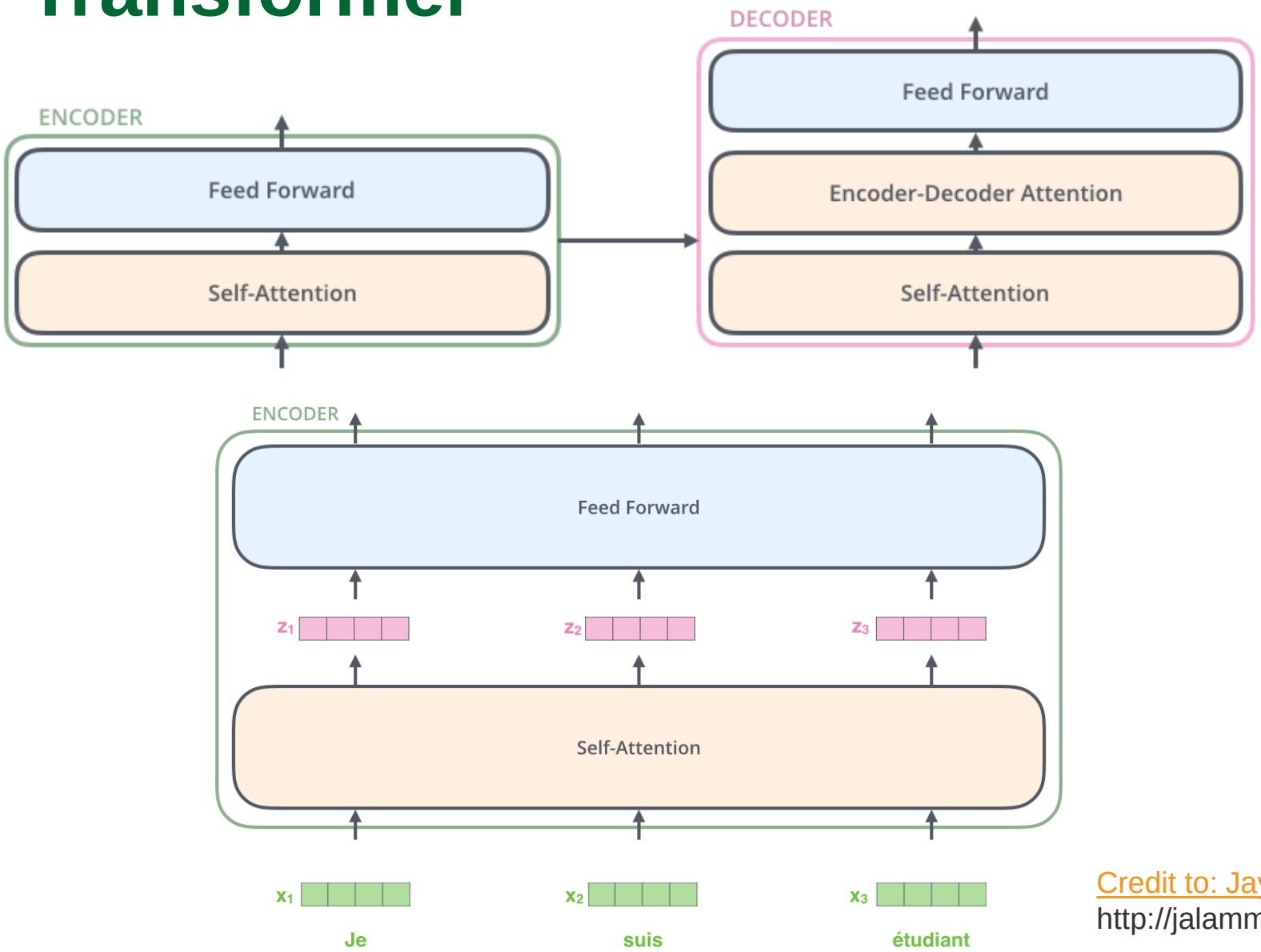
Transformer





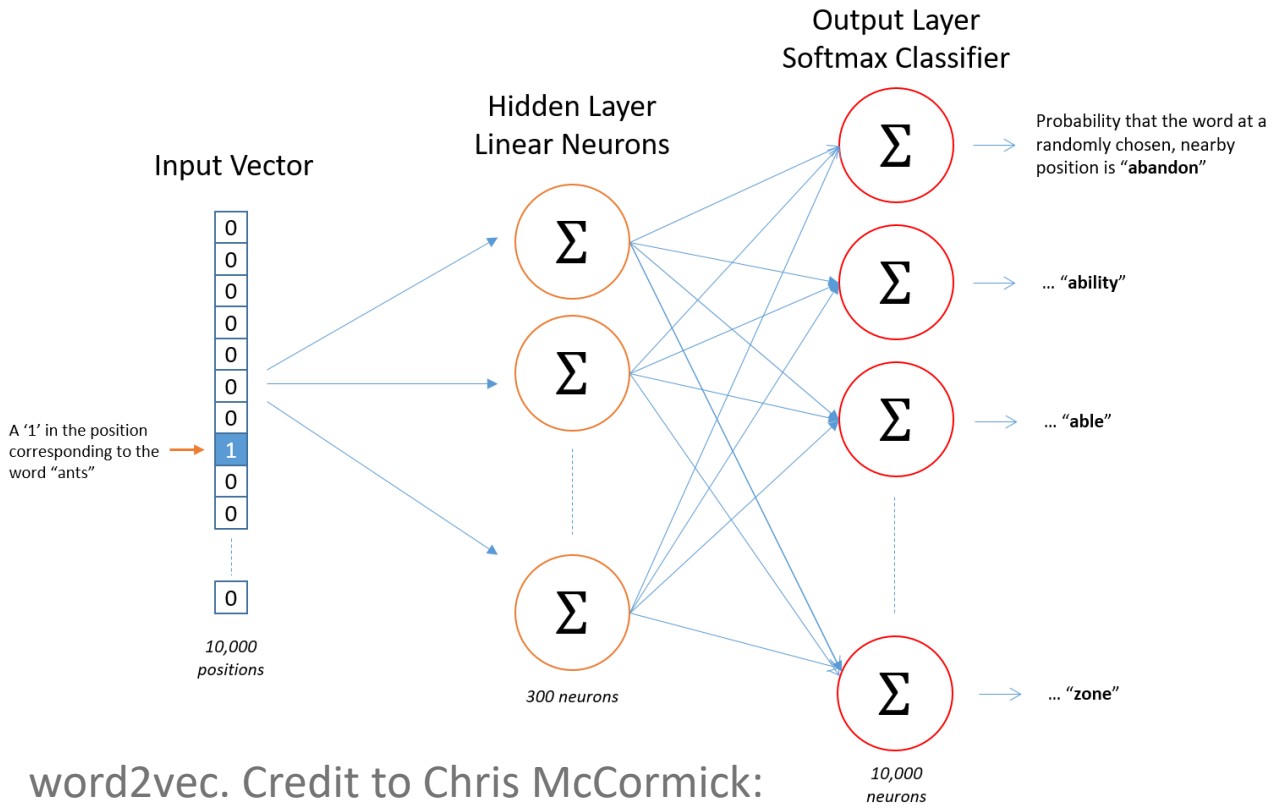
Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

Transformer



Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

Word Embedding



word2vec. Credit to Chris McCormick:

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

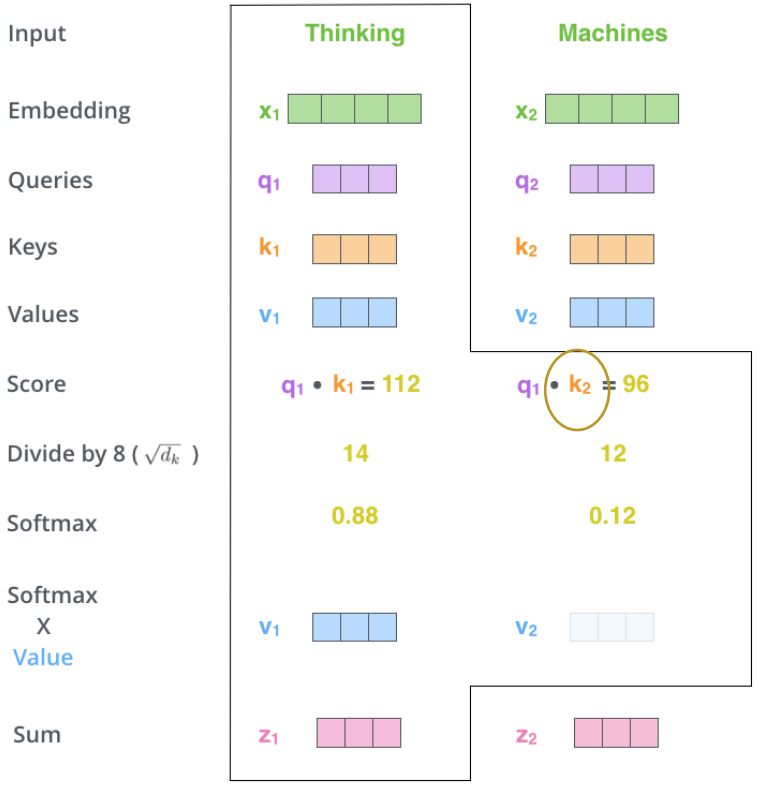
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

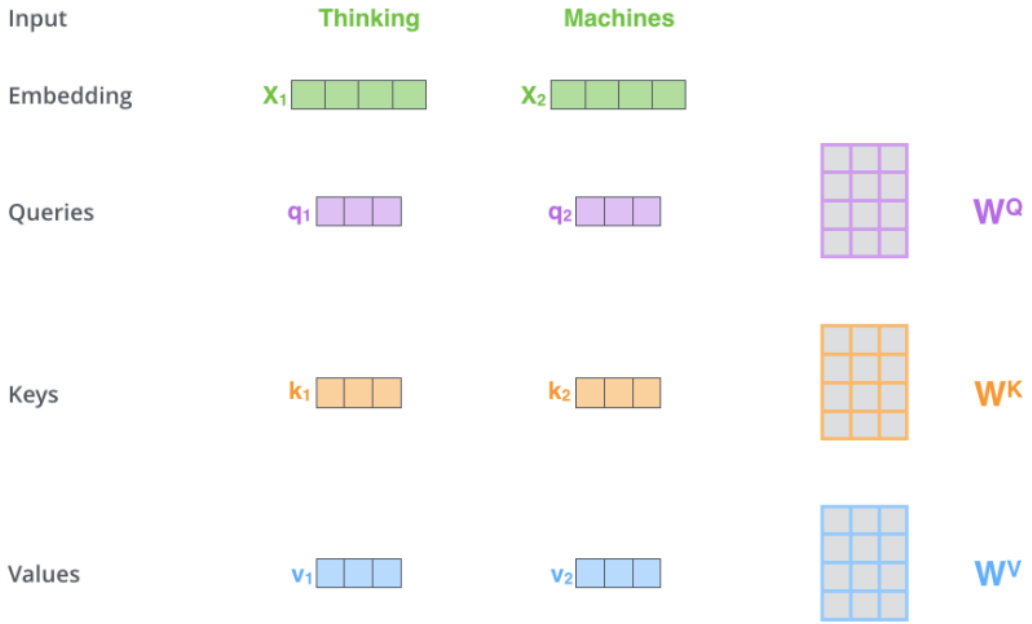
Credit to Chris McCormick:

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

A demo: https://ronxin.github.io/wevi/

Transformer



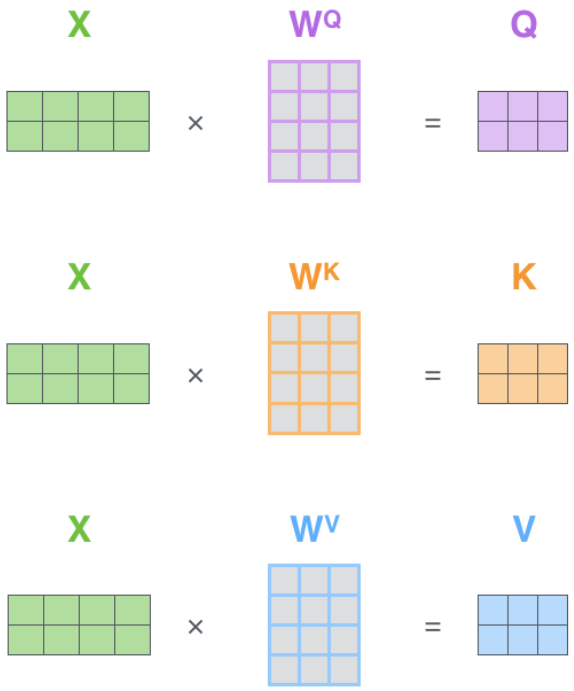


calculating self-attention

1. create three vectors from each of the encoder's input vectors
2. calculate a score
3. divide the scores by $\sqrt{d_k}$
4. Softmax.
5. multiply each value vector by the softmax score
6. sum up the weighted value vectors

Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

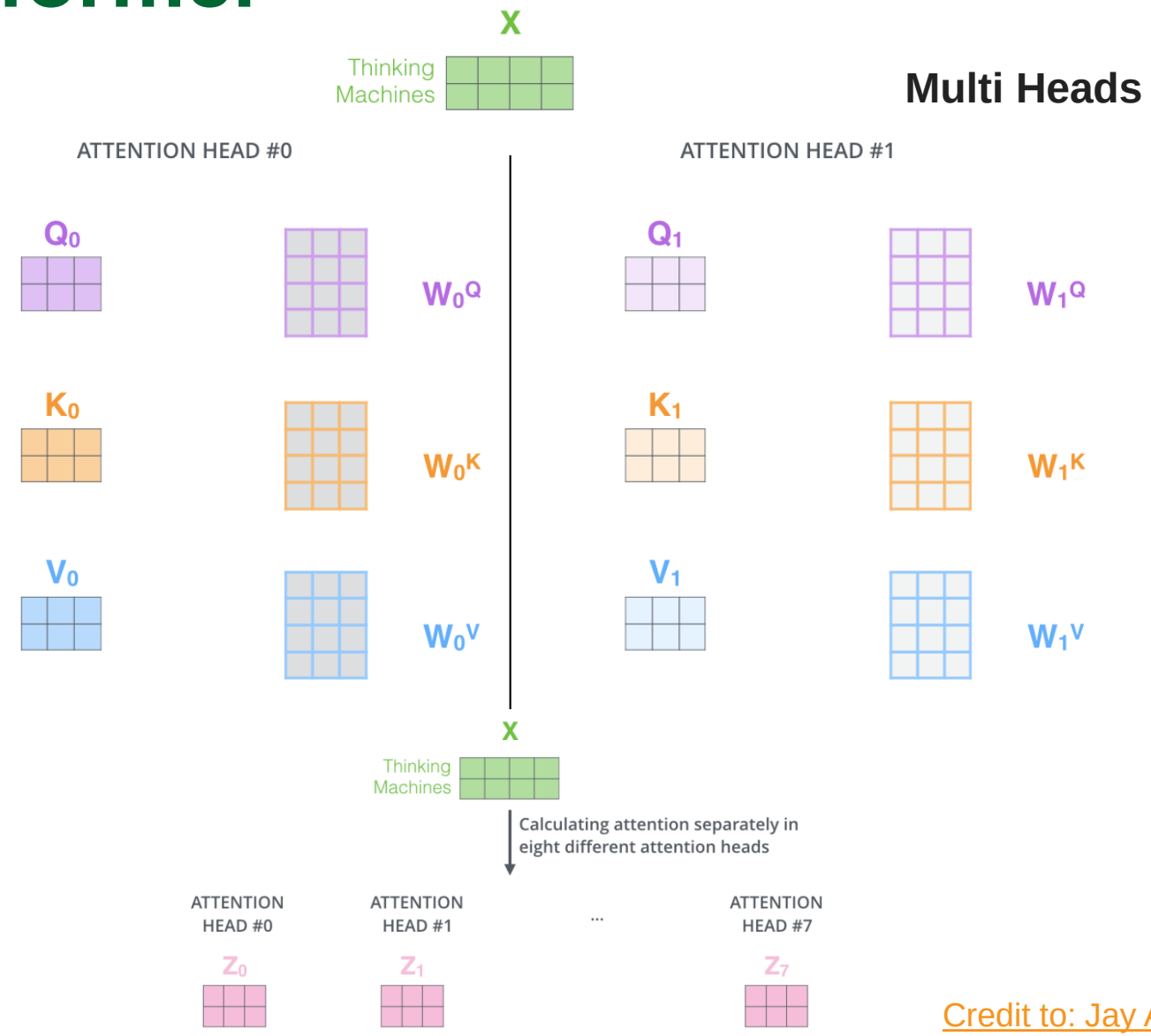
Transformer



$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

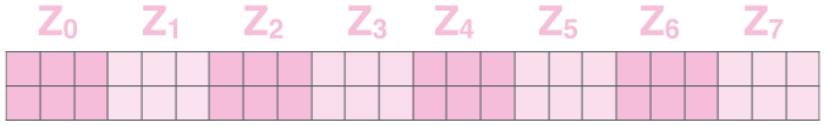
Transformer



Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

Transformer

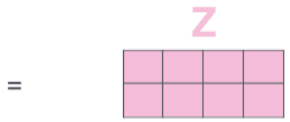
1) Concatenate all the attention heads



2) Multiply with a weight matrix W^O that was trained jointly with the model

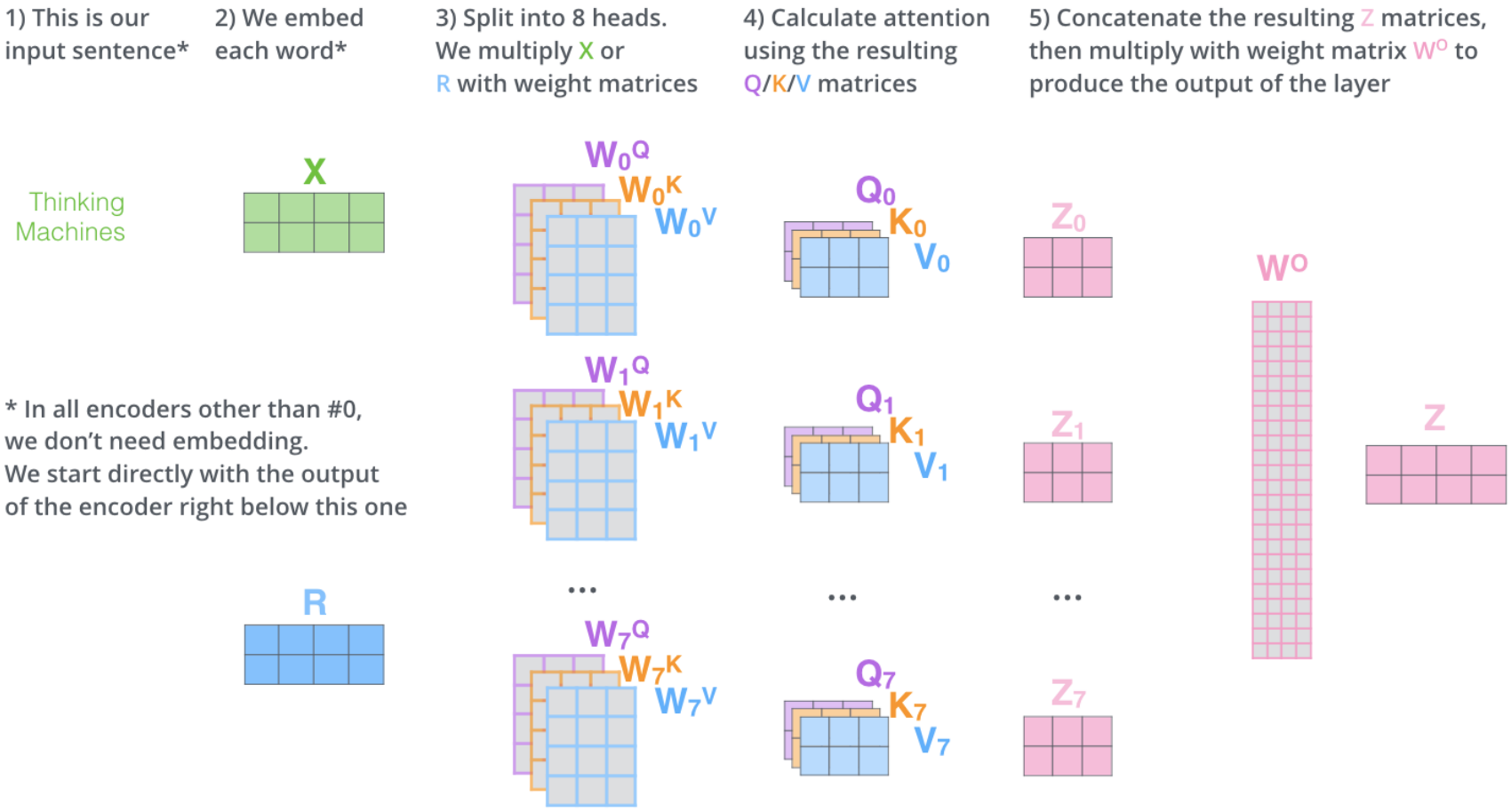
X

3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



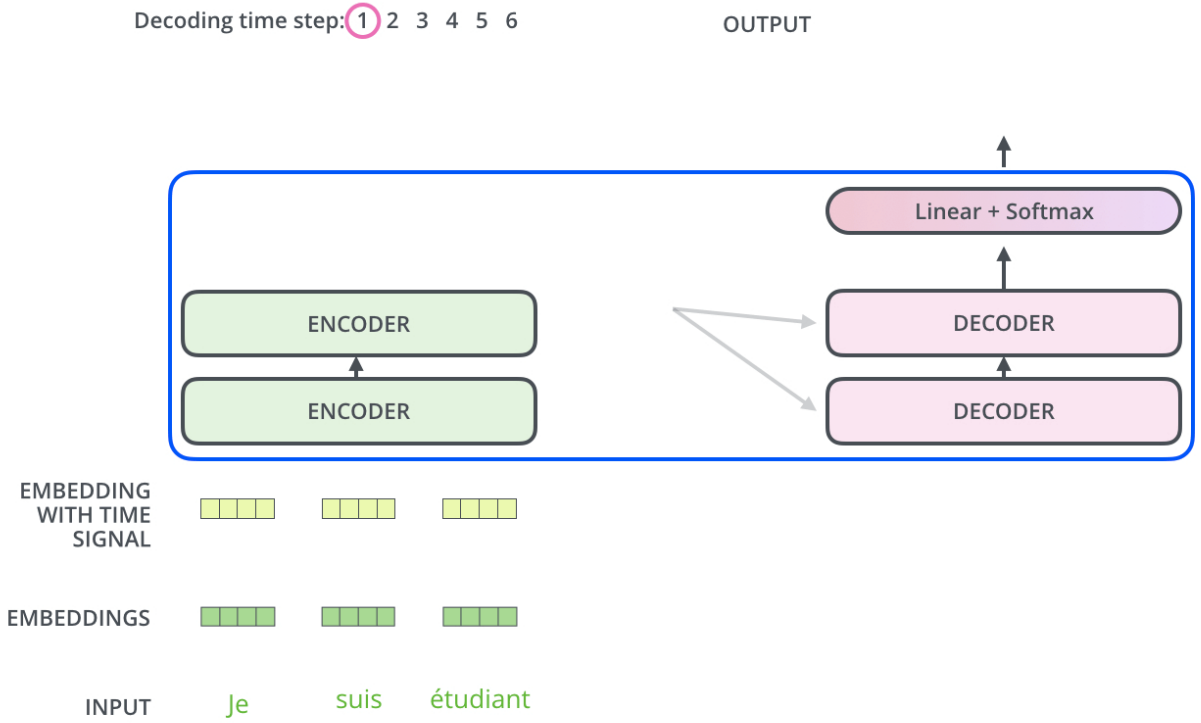
Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

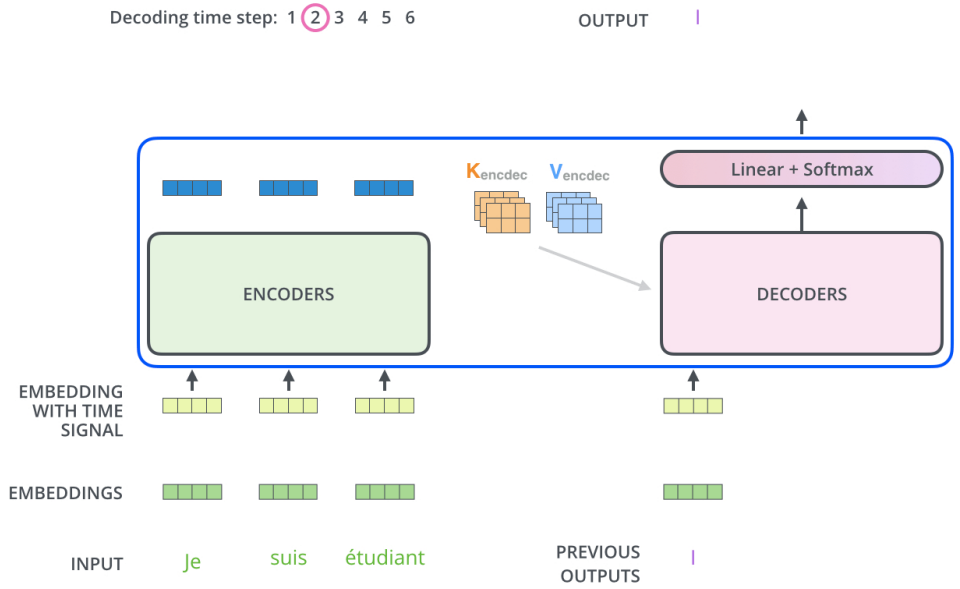
Transformer



Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

Transformer





Credit to: Jay Alammar
http://jalammar.github.io/illustrated-transformer/

Challenges: Model Storage

	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66	**Base**: ~110 **Large**: ~340
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.	**Large:** 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

Comparison of BERT and recent improvements over it

https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8

State-of-the-Art NLP Models from GLUE Ranking Website

BERT-Large Training Times on GPUs

Time	System	Number of Nodes	Number of V100 GPUs
47 min	DGX SuperPOD	92 x DGX-2H	1,472
67 min	DGX SuperPOD	64 x DGX-2H	1,024
236 min	DGX SuperPOD	16 x DGX-2H	256

Using 40 V100, the expected training time would be ~24 hours.

Computing at the network edge puts embedded systems in a new world.

-Intel

It becomes extremely difficult to implement these large-scale language representations into resource constrained devices while achieving desired performance (e.g., real time).

Albert: Compact Version of BERT

Albert: Compact version of BERT
(fewer parameters**, cross-layer parameter sharing**)

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
	xlarge	1270M	24	2048	2048	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	59M	24	2048	128	True
	xxlarge	233M	12	4096	128	True

https://arxiv.org/pdf/1909.11942v1.pdf

Albert-Tiny: (current Chinese GLUE version only)

Parameters: 1.8M, Model size: 16M (1/25 of BERT), training and inference: ten times faster than BERT

NOTE: The following link has many architectures in PyTorch: https://github.com/huggingface/transformers

Agenda

- Recurrent Neural Network (RNN)
- Existing Problem in Vanilla RNN
- Long short-term memory (LSTM)
- Variants of LSTM
- Application: Natural Langue Processing
- Machine Translation
- NMT: Pros and Cons
- Attention
- Transformer
- Conclusion

Conclusions

- Natural language processing (NLP) commonly have sequences of data, which is hard to be handled by CNNs.
- RNN and LSTM was initially proposed to handle the sequence of data
- Attention-based transformer is one of the most popular & state-of-the-art DNN for NLP task
- Models for NLP task mainly have huge model size.

GMU.EDU

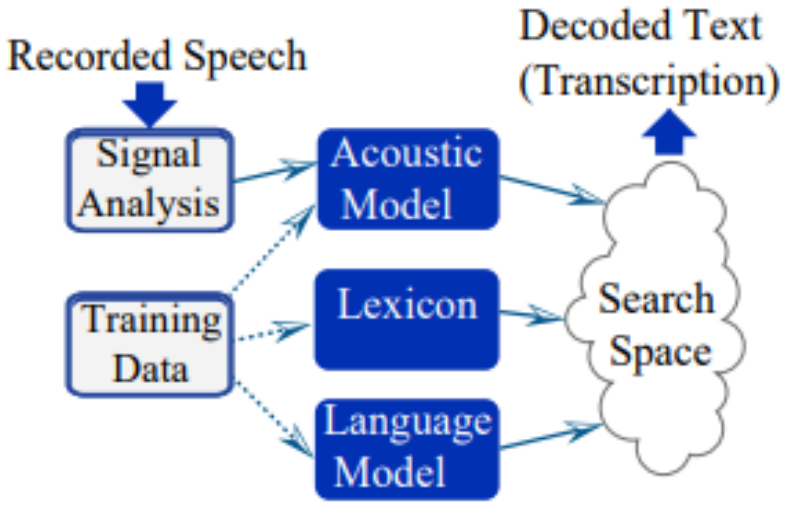
George Mason University

4400 University Drive
Fairfax, Virginia 22030

Tel: (703)993-1000

LSTM and GRU for Automatic Speech Recognitions

- Automatic speech recognition processing flow: acoustic model and language model



$$\mathbb{W}^* = \underset{\mathbb{W}}{\operatorname{argmax}} P(\mathbb{W}|\mathbb{X}) = \underset{\mathbb{W}}{\operatorname{argmax}} \frac{P(\mathbb{X}|\mathbb{W})P(\mathbb{W})}{P(\mathbb{X})}$$
$$= \underset{\mathbb{W}}{\operatorname{argmax}} \underbrace{P(\mathbb{X}|\mathbb{W})}_{Acoustic\,Model} \underbrace{P(\mathbb{W})}_{Language\,Model}$$

- The LSTM or GRU RNN is responsible for the acoustic model, the most computationally intensive part in automatic speech recognition

LSTM and GRU for Automatic Speech Recognitions

- LSTM and GRU RNN cell structures (with peephole connection)

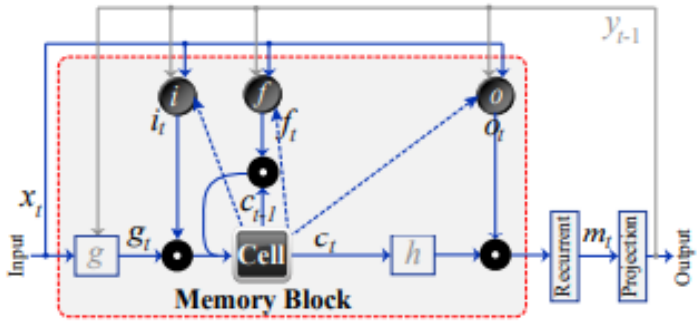


Figure 3: An LSTM based RNN architecture

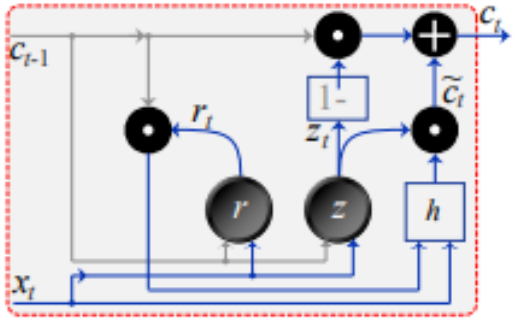


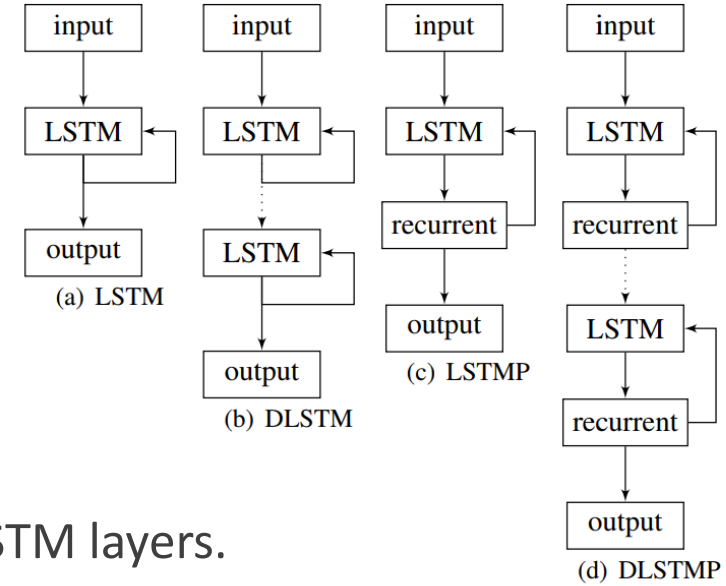
Figure 4: A GRU based RNN architecture

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ir}\mathbf{y}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i),\\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fr}\mathbf{y}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f),\\
\mathbf{g}_t &= \sigma(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{cr}\mathbf{y}_{t-1} + \mathbf{b}_c),\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{g}_t \odot \mathbf{i}_t,\\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{or}\mathbf{y}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o),\\
\mathbf{m}_t &= \mathbf{o}_t \odot \mathbf{h}(\mathbf{c}_t),\\
\mathbf{y}_t &= \mathbf{W}_{ym}\mathbf{m}_t,
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_{zx}\mathbf{x}_t + \mathbf{W}_{zc}\mathbf{c}_{t-1} + \mathbf{b}_z),\\
\mathbf{r}_t &= \sigma(\mathbf{W}_{rx}\mathbf{x}_t + \mathbf{W}_{rc}\mathbf{c}_{t-1} + \mathbf{b}_r),\\
\tilde{\mathbf{c}}_t &= \mathbf{h}(\mathbf{W}_{\tilde{c}x}\mathbf{x}_t + \mathbf{W}_{\tilde{c}c}(\mathbf{r}_t \odot \mathbf{c}_{t-1}) + \mathbf{b}_{\tilde{c}}),\\
\mathbf{c}_t &= (1 - \mathbf{z}_t) \odot \mathbf{c}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{c}}_t
\end{aligned}
$$

Zhe li*, Caiwen Ding*, Siyue Wang, Wujie Wen, Youwei Zhuo, Chang Liu, Qinru Qiu et al. "E-RNN: Design optimization for efficient recurrent neural networks in FPGAs." In 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp.

LSTM RNN architectures



Deep LSTM (DLSTP): are built by stacking multiple LSTM layers.

- The depth in deep LSTM RNNs has an additional meaning.
- They can make better use of parameters by distributing them over the space through multiple layers.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Fifteenth annual conference of the international speech communication association.

LSTM Network (2)- LSTMP

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ir}\mathbf{y}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i), \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fr}\mathbf{y}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f), \\
\mathbf{g}_t &= \sigma(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{cr}\mathbf{y}_{t-1} + \mathbf{b}_c), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{g}_t \odot \mathbf{i}_t, \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{or}\mathbf{y}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o), \\
\mathbf{m}_t &= \mathbf{o}_t \odot \mathbf{h}(\mathbf{c}_t), \\
\mathbf{y}_t &= \mathbf{W}_{ym}\mathbf{m}_t,
\end{aligned}
$$

$\mathbf{W_{ix}}$ is the matrix of weights from the input gate to the input);

$\mathbf{W_{ic}}$;$\mathbf{W_{fc}}$;$\mathbf{W_{oc}}$ are diagonal weight matrices for peephole connections,

b denote bias vectors (bi is the input gate bias vector),

σ is the logistic sigmoid function,

i, f, o and **c** are respectively the input gate, forget gate, output gate and cell activation vectors,

m cell output activation vector,

$\odot$ is the element-wise product of the vectors,

g and *h* are the cell input and cell output activation functions, usually *tanh*.

LSTM Network (2)- LSTMP

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ir}\mathbf{y}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i), \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fr}\mathbf{y}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f), \\
\mathbf{g}_t &= \sigma(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{cr}\mathbf{y}_{t-1} + \mathbf{b}_c), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{g}_t \odot \mathbf{i}_t, \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{or}\mathbf{y}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o), \\
\mathbf{m}_t &= \mathbf{o}_t \odot \mathbf{h}(\mathbf{c}_t), \\
\mathbf{y}_t &= \mathbf{W}_{ym}\mathbf{m}_t,
\end{aligned}
$$

Matrix	Matrix Size
W_{ix}	1024×153
W_{fx}	1024×153
W_{cx}	1024×153
W_{ox}	1024×153
W_{ir}	1024×512
W_{fr}	1024×512
W_{cr}	1024×512
W_{or}	1024×512
W_{ym}	512×1024
Total	**3248128**

LSTM Network (2)- LSTMP

LSTMP - LSTM with Recurrent Projection Layer.

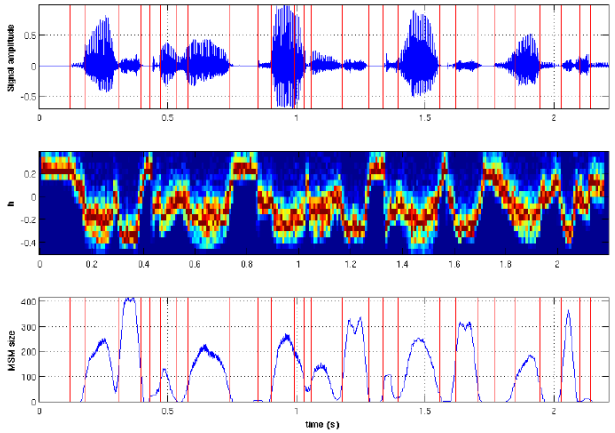
- has a separate linear projection layer after the LSTM layer.
- The recurrent connections now connect from this recurrent projection layer to the input of the LSTM layer.

Peephole connections can learn the fine distinction between sequences of spikes spaced either 50 or 49 time steps apart without the help of any short training exemplars.[1]

[1] Gers, Felix A., Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks." *Journal of machine learning research* 3.Aug (2002): 115-143.

LSTM and GRU for Automatic Speech Recognitions

- Results on phone error rates (PERs) of LSTM and GRU-based RNNs



(TIMIT dataset)

Table 1: Comparison among LSTM based RNNs

ID	Layer Size	Block Size	Peephole	Projection (512)	Phone Error Rate (PER) %	PER degradation (%)
1	256 – 256 – 256	–	×	×	20.83	–
2	256 – 256 – 256	2 – 2 – 2	×	×	20.75	–0.08
3	256 – 256 – 256	4 – 4 – 4	×	×	20.85	0.02
4	512 – 512	–	√	×	20.53	–
5	512 – 512	4 – 4	√	×	20.57	0.04
6	512 – 512	4 – 8	√	×	20.85	0.28
7	512 – 512	8 – 4	√	×	20.98	0.41
8	512 – 512	8 – 8	√	×	21.01	0.48
9	1024 – 1024	–	√	√	20.01	–
10	1024 – 1024	4 – 4	√	√	20.01	0.00
11	1024 – 1024	4 – 8	√	√	20.05	0.04
12	1024 – 1024	8 – 4	√	√	20.10	0.09
13	1024 – 1024	8 – 8	√	√	20.14	0.13
14	1024 – 1024	8 – 16	√	√	20.22	0.21
15	1024 – 1024	16 – 8	√	√	20.29	0.28
16	1024 – 1024	16 – 16	√	√	20.32	0.31

Table 2: Comparison among GRU based RNNs

ID	Layer Size	Block Size	Phone Error Rate (PER) %	PER degradation (%)
1	256 – 256 – 256	–	20.72	–
2	256 – 256 – 256	4 – 4 – 4	20.81	0.09
3	256 – 256 – 256	8 – 8 – 8	20.88	0.16
4	512 – 512	–	20.51	–
5	512 – 512	4 – 4	20.55	0.04
6	512 – 512	4 – 8	20.73	0.22
7	512 – 512	8 – 4	20.89	0.38
8	512 – 512	8 – 8	20.95	0.44
9	1024 – 1024	–	20.02	–
10	1024 – 1024	4 – 4	20.03	0.01
11	1024 – 1024	4 – 8	20.08	0.06
12	1024 – 1024	8 – 4	20.13	0.11
13	1024 – 1024	8 – 8	20.20	0.18
14	1024 – 1024	8 – 16	20.25	0.23
15	1024 – 1024	16 – 8	20.31	0.29
16	1024 – 1024	16 – 16	20.36	0.33

Zhe li*, Caiwen Ding*, Siyue Wang, Wujie Wen, Youwei Zhuo, Chang Liu, Qinru Qiu et al. "E-RNN: Design optimization for efficient recurrent neural networks in FPGAs." In 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp.

LSTM for Automatic Speech Recognitions

Large Scale Acoustic Modeling

They evaluate and compare the performance of LSTM RNN architectures on a large vocabulary speech recognition task – the Google Voice Search task.

C	*P*	Depth	*N*	WER (%)
840	-	5L	37M	10.9
440	-	5L	13M	10.8
600	-	2L	13M	11.3
385	-	7L	13M	11.2
750	-	1L	13M	12.4
6000	800	1L	36M	11.8
2048	512	2L	22M	10.8
1024	512	3L	20M	10.7
1024	512	2L	15M	10.7
800	512	2L	13M	10.7
2048	512	1L	13M	11.3

L indicates the number of layers, for shallow (1L) and deep (2,4,5,7L) networks. *C* indicates the number of memory cells, *P* the number of recurrent projection units, and *N* the total number of parameters. **Word error rate** (**WER**)

Sak, Haşim, Andrew Senior, and Françoise Beaufays. "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition." *arXiv preprint arXiv:1402.1128* (2014).

LSTM for Automatic Speech Recognitions

Large Scale Acoustic Modeling

They evaluate and compare the performance of LSTM RNN architectures on a large vocabulary speech recognition task – the Google Voice Search task.

C	P	Depth	N	WER (%)
840	-	5L	37M	10.9
440	-	5L	13M	10.8
600	-	2L	13M	11.3
385	-	7L	13M	11.2
750	-	1L	13M	12.4
6000	800	1L	36M	11.8
2048	512	2L	22M	10.8
1024	512	3L	20M	10.7
1024	512	2L	15M	10.7
800	512	2L	13M	10.7
2048	512	1L	13M	11.3

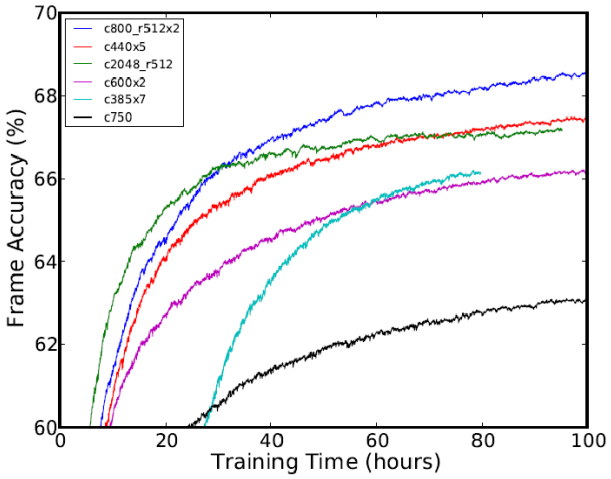
- Conventional LSTM RNNs with a single layer do not perform very well for this large scale acoustic modeling task.
- With two layers of LSTM RNNs, the performance improves but still it is not very good.
- The LSTM RNN with five layers approaches the performance of the best model.

- The LSTMP RNN models give slightly better results than the LSTM RNN model with 5 layers
- increasing the number of parameters in the LSTMP RNN models more than 13M by having more layers or more memory cells does not give performance improvements

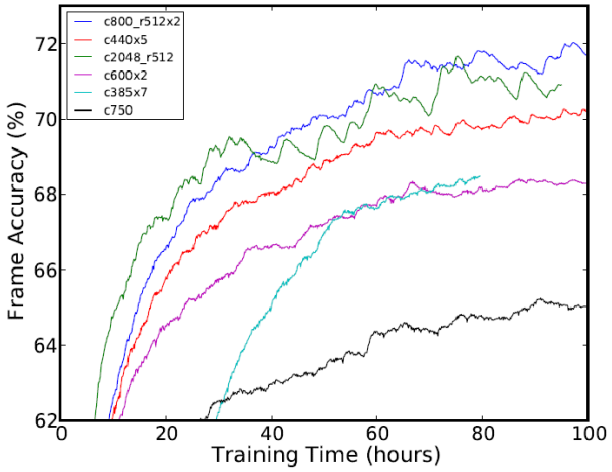
Sak, Haşim, Andrew Senior, and Françoise Beaufays. "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition." *arXiv preprint arXiv:1402.1128* (2014).

LSTM for Automatic Speech Recognitions

Large Scale Acoustic Modeling



(a) Held-out data.



(b) Training data.

- LSTMP RNN architectures converges faster than LSTM RNN architectures
- It is clear that having more layers helps generalization but makes training harder and convergence slower.

Sak, Haşim, Andrew Senior, and Françoise Beaufays. "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition." *arXiv preprint arXiv:1402.1128* (2014).