# Natural Language Processing with Deep Learning
# CS224N/Ling284

Lecture 13:
Transformer Networks and
Convolutional Neural Networks

**Richard Socher**

# Byte Pair Encoding

- A compression algorithm:
  - Most frequent byte pair ↦ a new byte.

Replace bytes with character ngrams

*Rico Sennrich, Barry Haddow, and Alexandra Birch.* ***Neural Machine Translation of Rare Words with Subword Units****. ACL 2016.*

# Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters.
  - Most frequent ngram pairs ↦ a new ngram.

# Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters.
  - Most frequent ngram pairs ↦ a new ngram.

*Dictionary*

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

*Vocabulary*

l, o, w, e, r, n, w, s, t, i, d

Start with all characters in vocab



*(Example from Sennrich)*

# Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters.
  - Most frequent ngram pairs ↦ a new ngram.

*Dictionary*

5 l o w
2 l o w e r
6 n e w **es** t
3 w i d **es** t

*Vocabulary*

l, o, w, e, r, n, w, s, t, i, d, **es**

Add a pair (e, s) with freq 9



*(Example from Sennrich)*

# Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters.
  - Most frequent ngram pairs ↦ a new ngram.

*Dictionary*

5 l o w
2 l o w e r
6 n e w **est**
3 w i d **est**

*Vocabulary*

l, o, w, e, r, n, w, s, t, i, d, es, **est**

Add a pair (es, t) with freq 9



*(Example from Sennrich)*

# Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters.
  - Most frequent ngram pairs ↦ a new ngram.

*Dictionary*

5 **lo** w
2 **lo** w e r
6 n e w est
3 w i d est

*Vocabulary*

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

Add a pair (l, o) with freq 7



*(Example from Sennrich)*

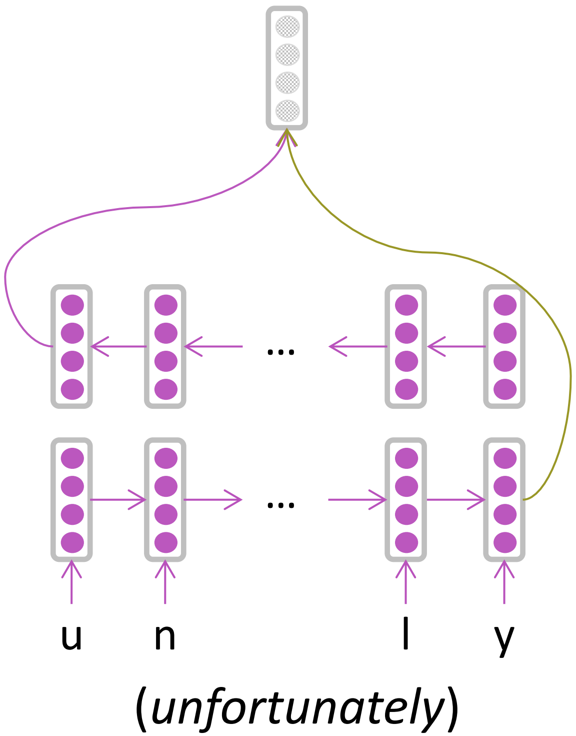
# Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters.
  - Most frequent ngram pairs ↦ a new ngram.
- Automatically decide vocabs for NMT

Top places in WMT 2016!

https://github.com/rsennrich/nematus

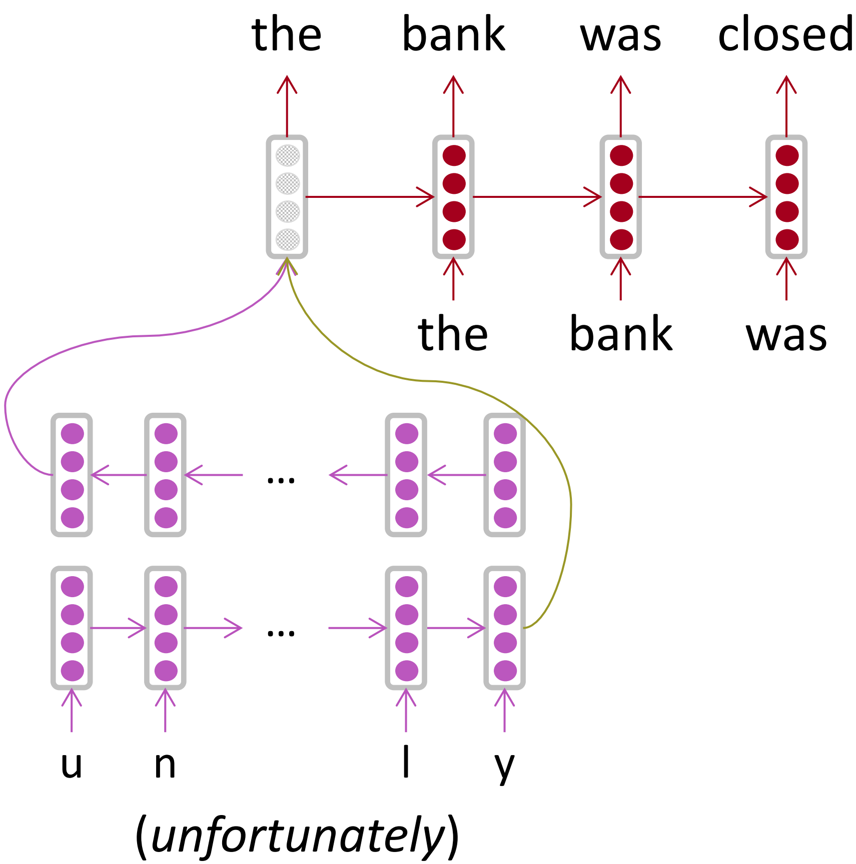
# Character-based LSTM




Bi-LSTM builds word representations

*Ling, Luís, Marujo, Astudillo, Amir, Dyer, Black, Trancoso.* ***Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation***. *EMNLP'15.*

# Character-based LSTM




Recurrent Language Model

Bi-LSTM builds word representations

*Ling, Luís, Marujo, Astudillo, Amir, Dyer, Black, Trancoso.* ***Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation***. *EMNLP'15.*

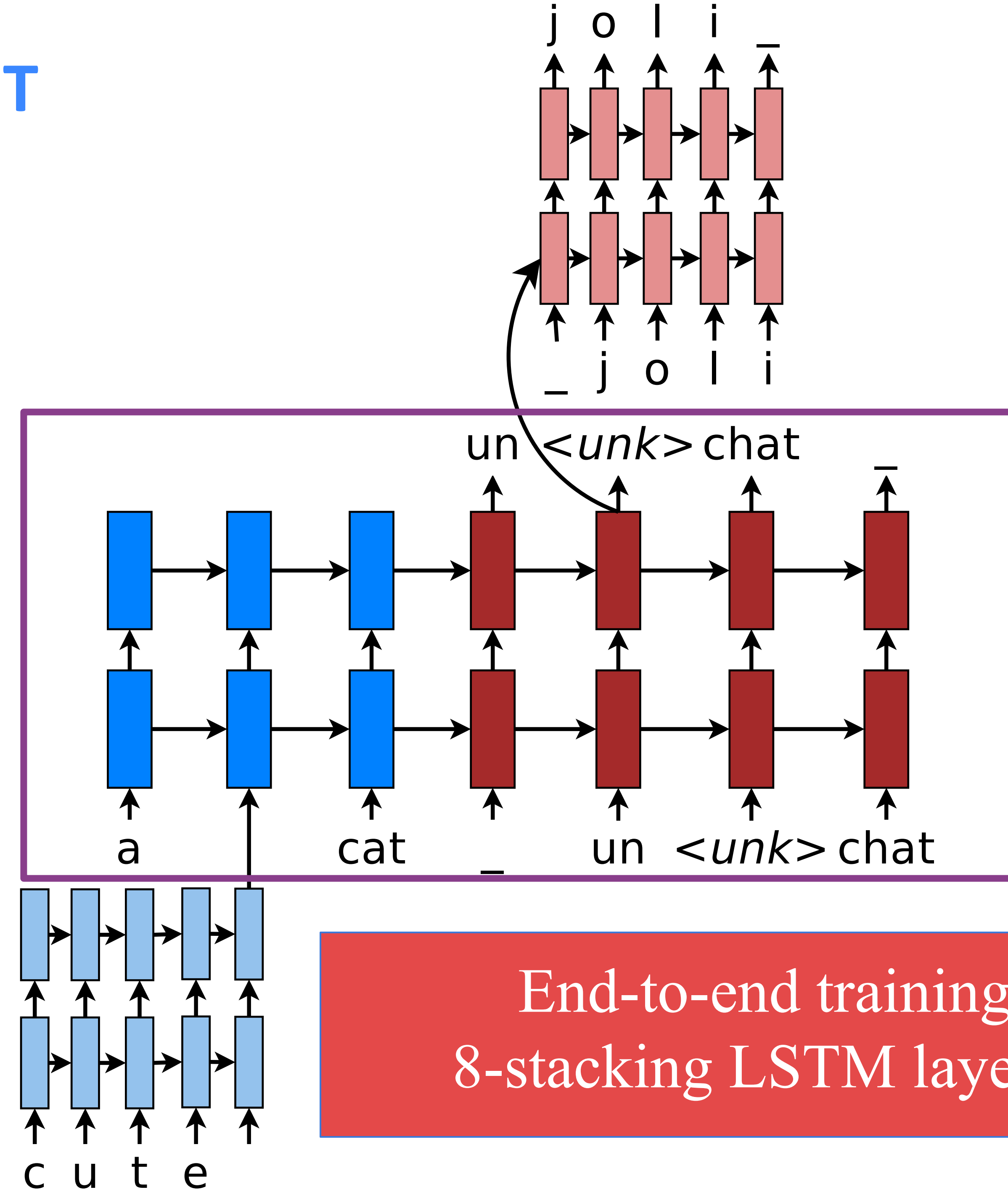
# Hybrid NMT


- A *best-of-both-worlds* architecture:
  - Translate mostly at the word level
  - Only go to the character level when needed.
- More than 2 BLEU improvement over a copy mechanism.

*Thang Luong and Chris Manning.* ***Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models****. ACL 2016.*

# Hybrid NMT

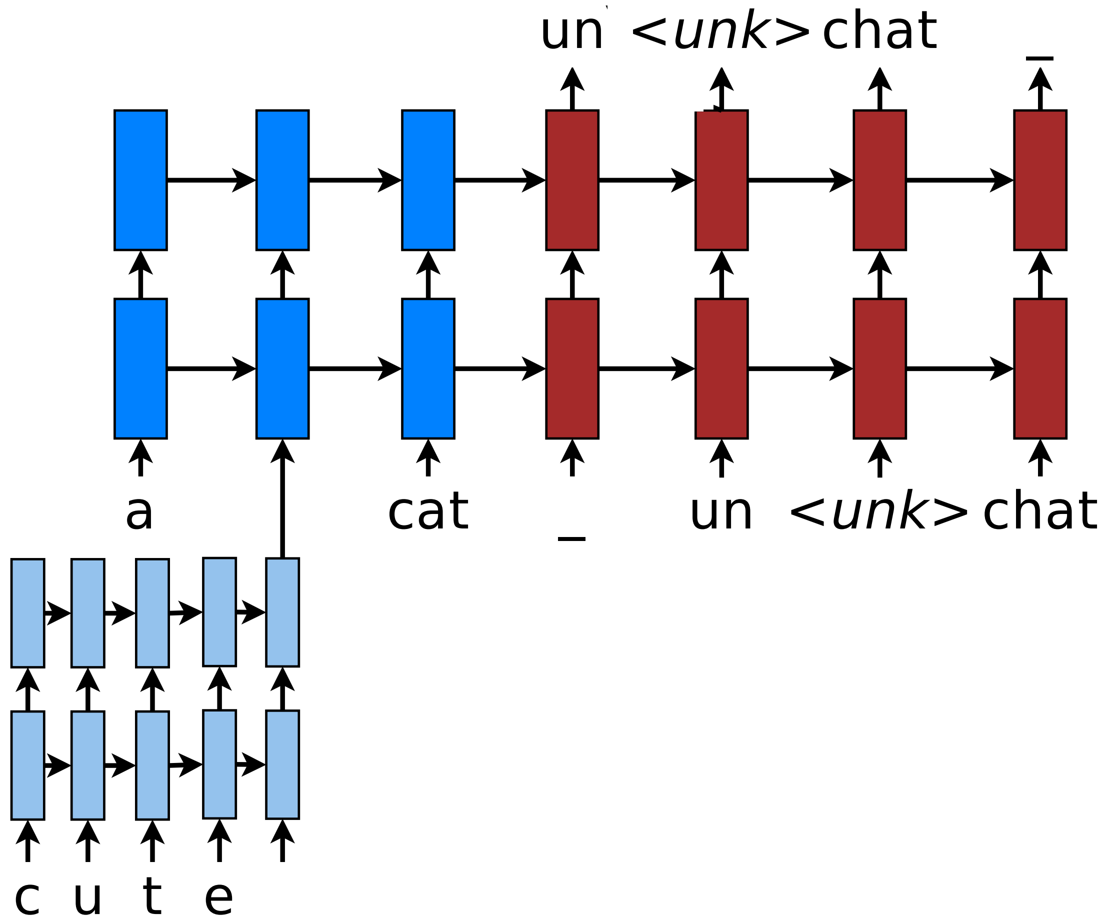
Word-level
(4 layers)




End-to-end training
8-stacking LSTM layers.

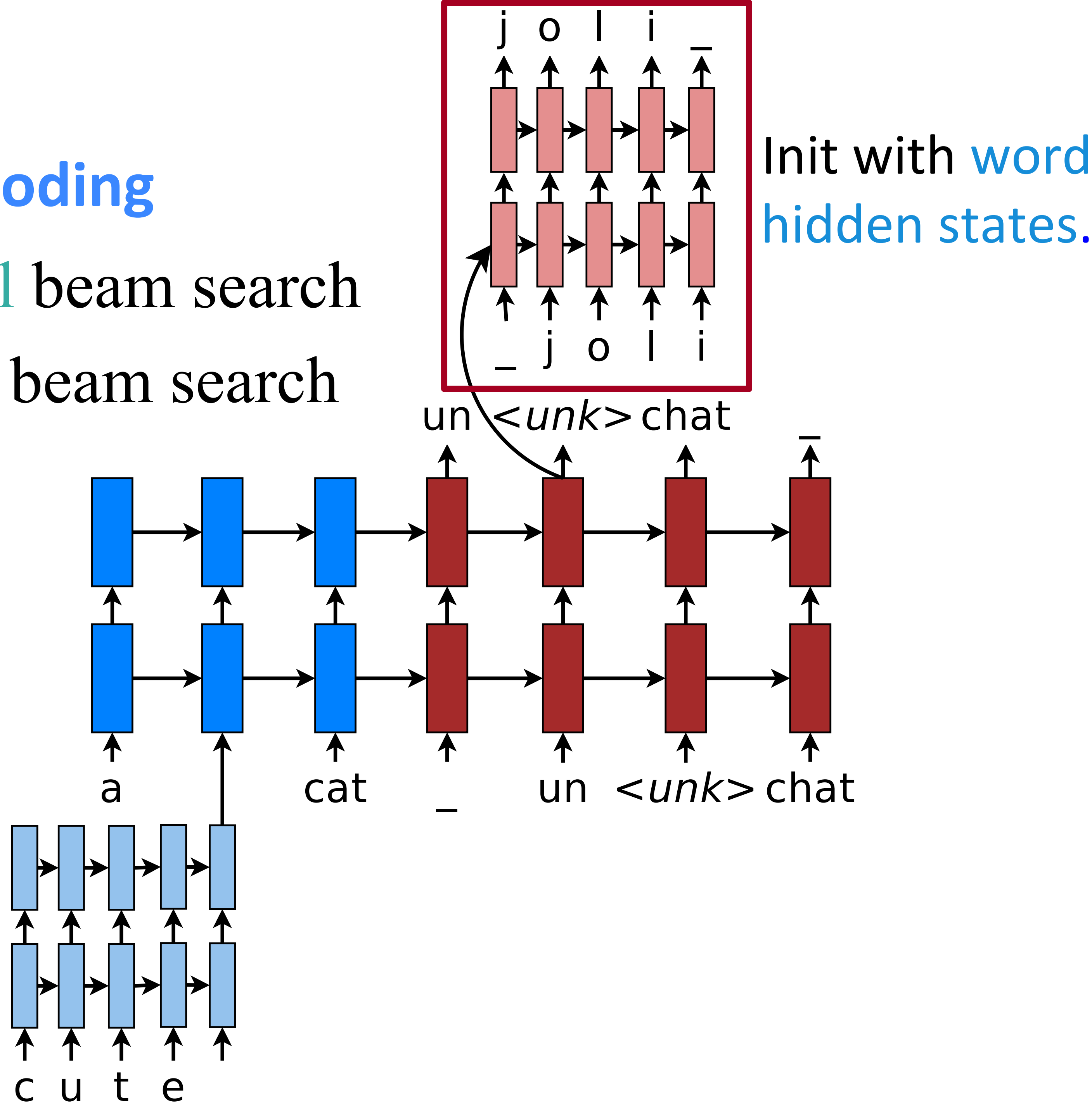
## 2-stage Decoding

- Word-level beam search



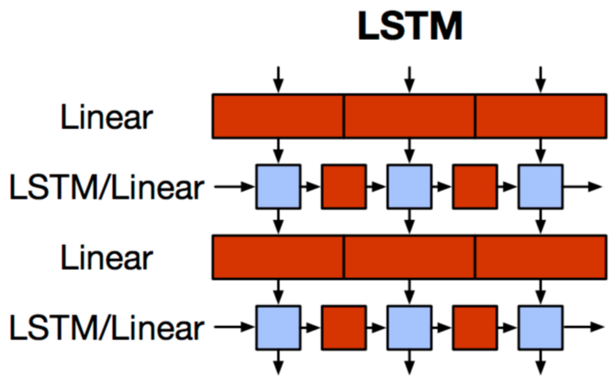
## 2-stage Decoding

- Word-level beam search
- Char-level beam search for <unk>.



# Problems with RNNs = Motivation for Transformers

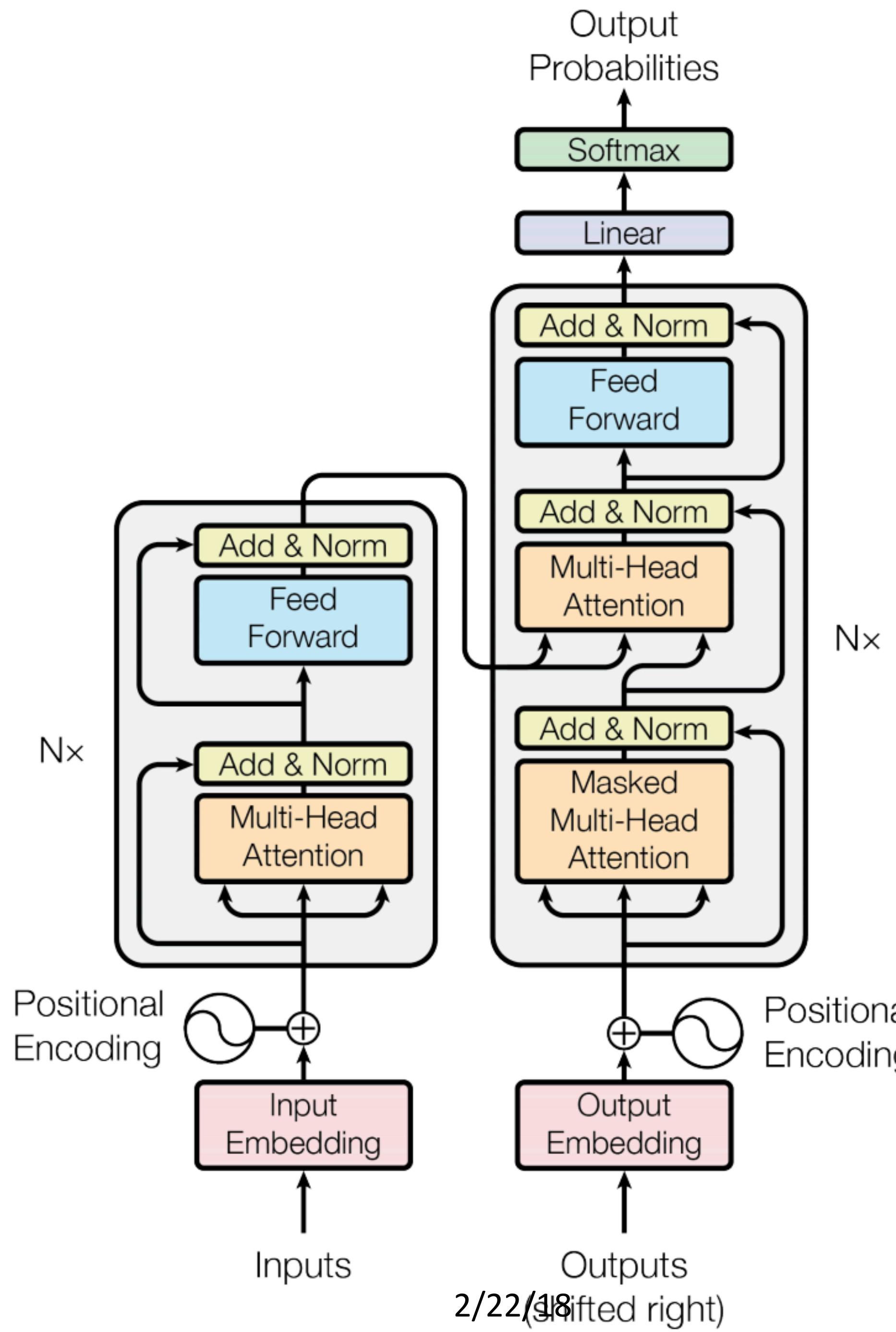
- Sequential computation prevents parallelization




- Despite GRUs and LSTMs, RNNs still need attention mechanism to deal with long range dependencies – path length for co-dependent computation between states grows with sequence
- But if attention gives us access to any state… maybe we don't need the RNN?

# Transformer Overview

- Sequence-to-sequence
- Encoder-Decoder
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier




This and related figures from paper: https://arxiv.org/pdf/1706.03762.pdf

# Transformer Paper

- Attention Is All You Need [2017]
- by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin
- Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

# Transformer Basics

- Let's define the basic building blocks
  of transformer networks first: new attention layers!

# Dot-Product Attention (Extending our previous def.)

- Inputs: a query q and a set of key-value (k-v) pairs to an output
- Query, keys, values, and output are all vectors

- Output is weighted sum of values, where
- Weight of each value is computed by an inner product of query and corresponding key
- Queries and keys have same dimensionality $d_k$ value have $d_v$

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

# Dot-Product Attention – Matrix notation

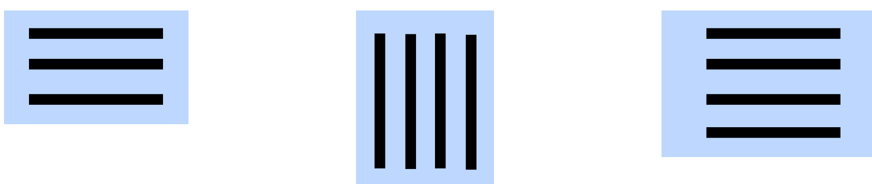
- When we have multiple queries q, we stack them in a matrix Q:

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

- Becomes: $A(Q, K, V) = softmax(QK^T)V$

$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$

softmax
row-wise



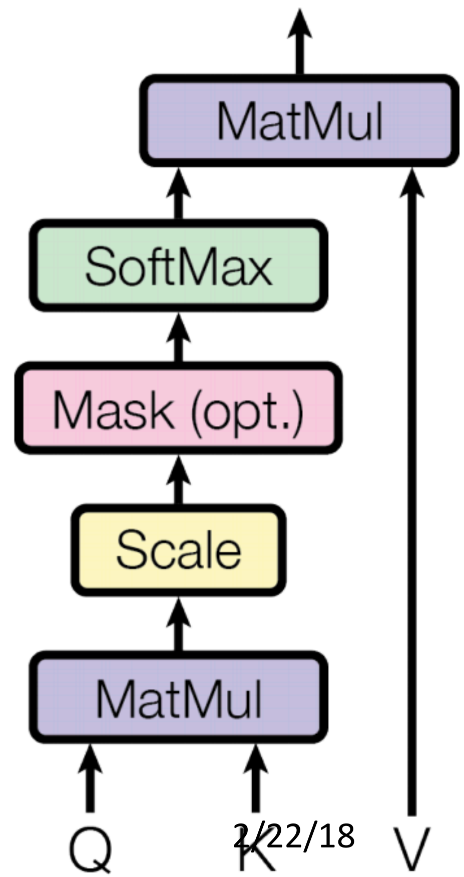
$= [|Q| \times d_v]$

# Scaled Dot-Product Attention

- Problem: As $d_k$ gets large, the variance of $q^Tk$ increases → some values inside the softmax get large → the softmax gets very peaked --> hence its gradient gets smaller.

- Solution: Scale by length of query/key vectors:

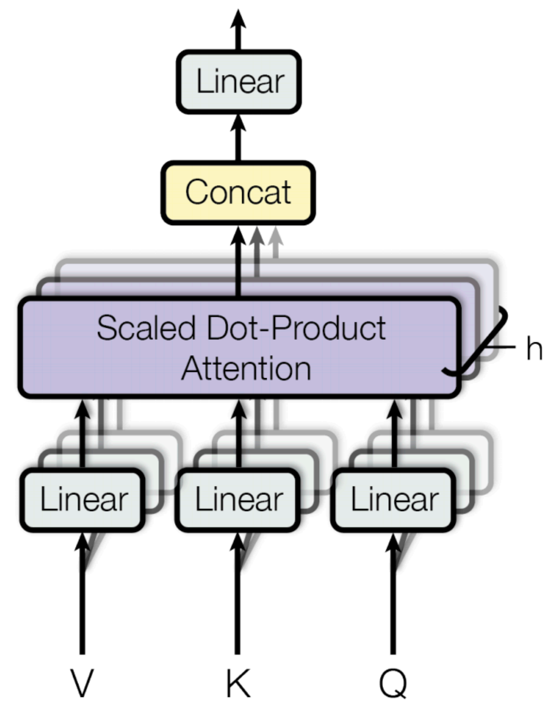
$$A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$



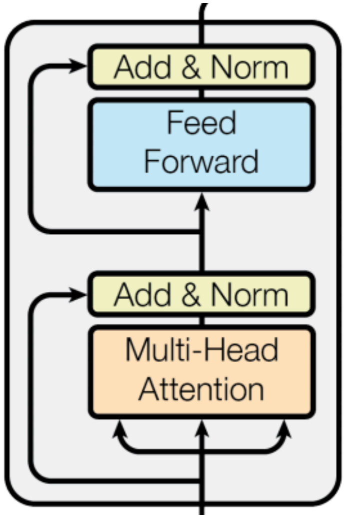
# Self-attention and Multi-head attention

- The input word vectors could be the queries, keys and values
- In other words: the word vectors themselves select each other
- Word vector stack = Q = K = V
- Problem: Only one way for words to interact with one-another
- Solution: Multi-head attention
- First map Q, K, V into h many lower dimensional spaces via W matrices
- Then apply attention, then concatenate outputs and pipe through linear layer

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



# Complete transformer block




- Each block has two “sublayers”

1. Multihead attention
2. 2 layer feed-forward Nnet (with relu)

Each of these two steps also has:

Residual (short-circuit) connection and LayerNorm:

LayerNorm(x + Sublayer(x))

Layernorm changes input to have mean 0 and variance 1,
per layer and per training point (and adds two more parameters)

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad h_i = f\left(\frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i\right)$$

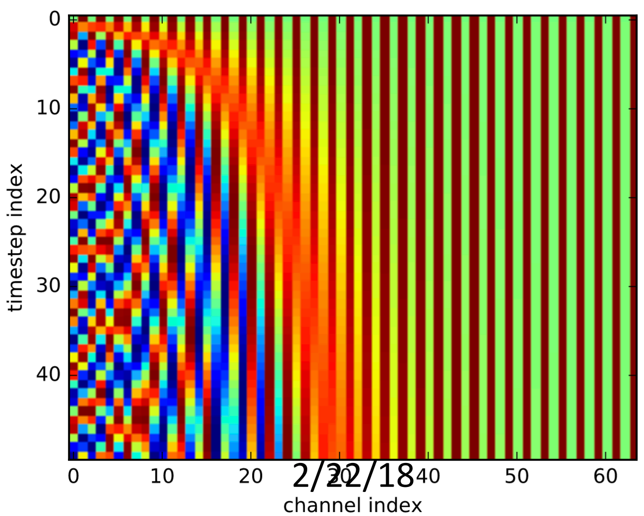
Layer Normalization by Ba, Kiros and Hinton, https://arxiv.org/pdf/1607.06450.pdf

# Encoder Input

- Actual word representations are byte-pair encodings (see last lecture)

- Also added is a positional encoding so same words at different locations have different overall representations:

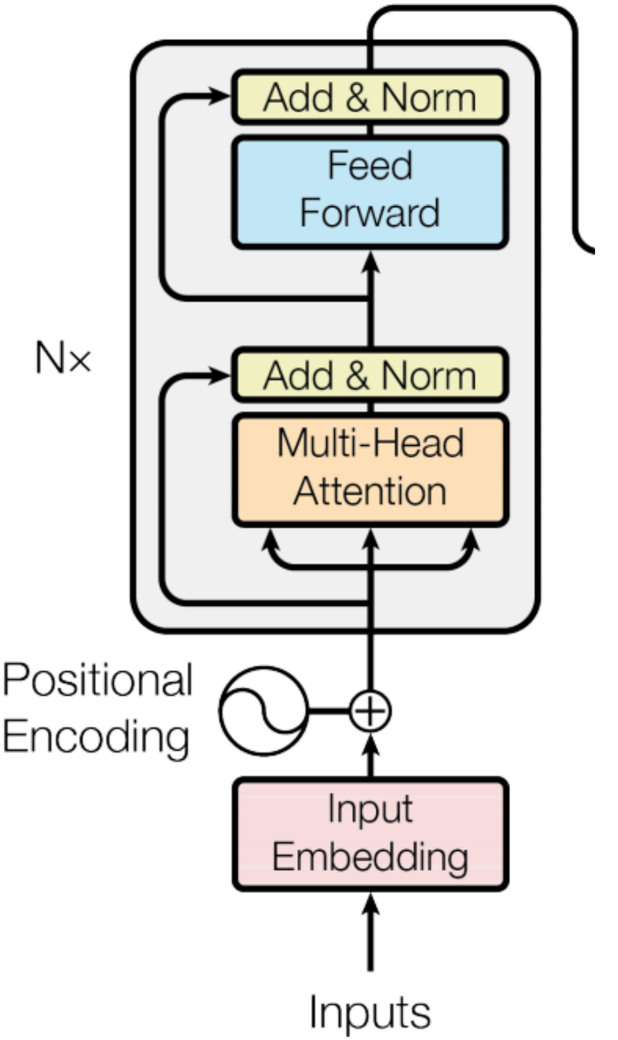
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



# Complete Encoder

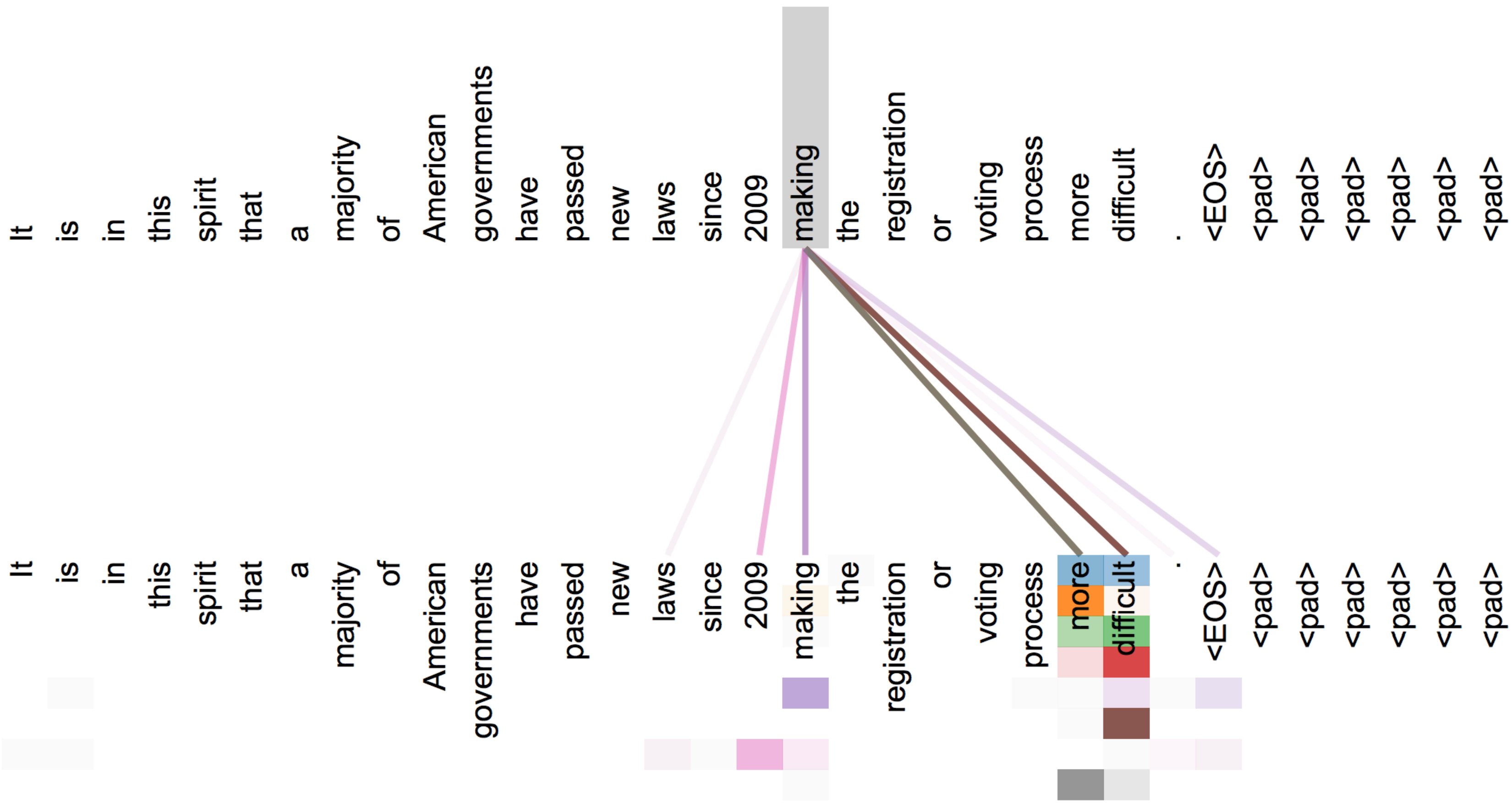
- For encoder, at each block, we use the same Q, K and V from the previous layer

- Blocks are repeated 6 times

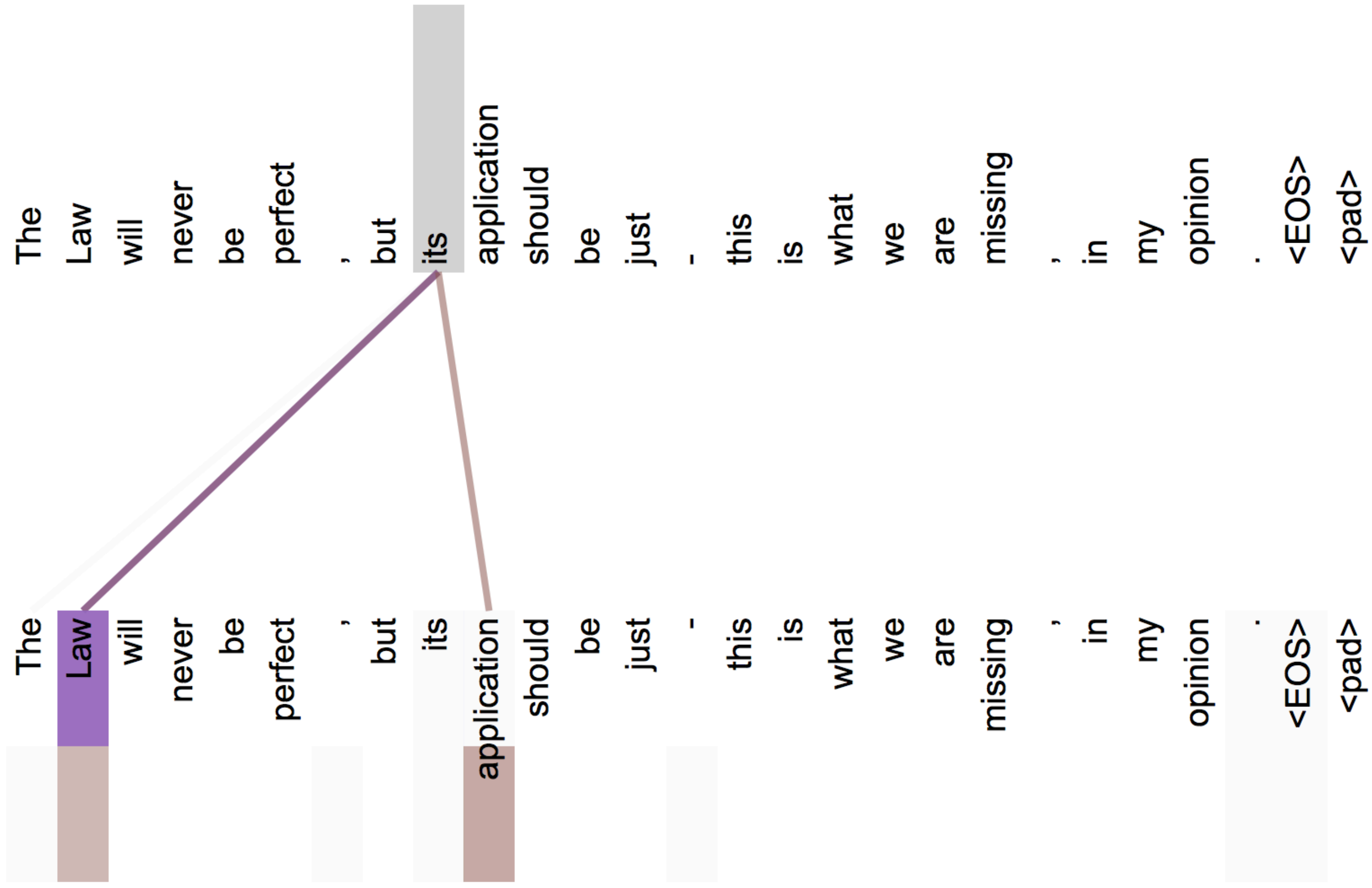


# Attention visualization in layer 5

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



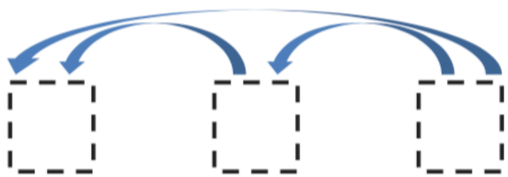
# Attention visualization: Implicit anaphora resolution



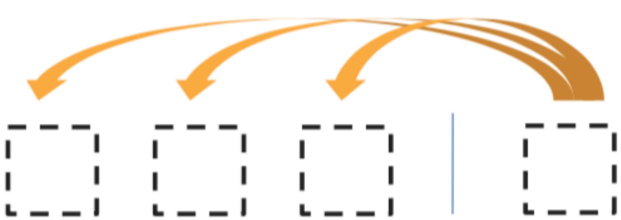

In 5th layer. Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

# Transformer Decoder

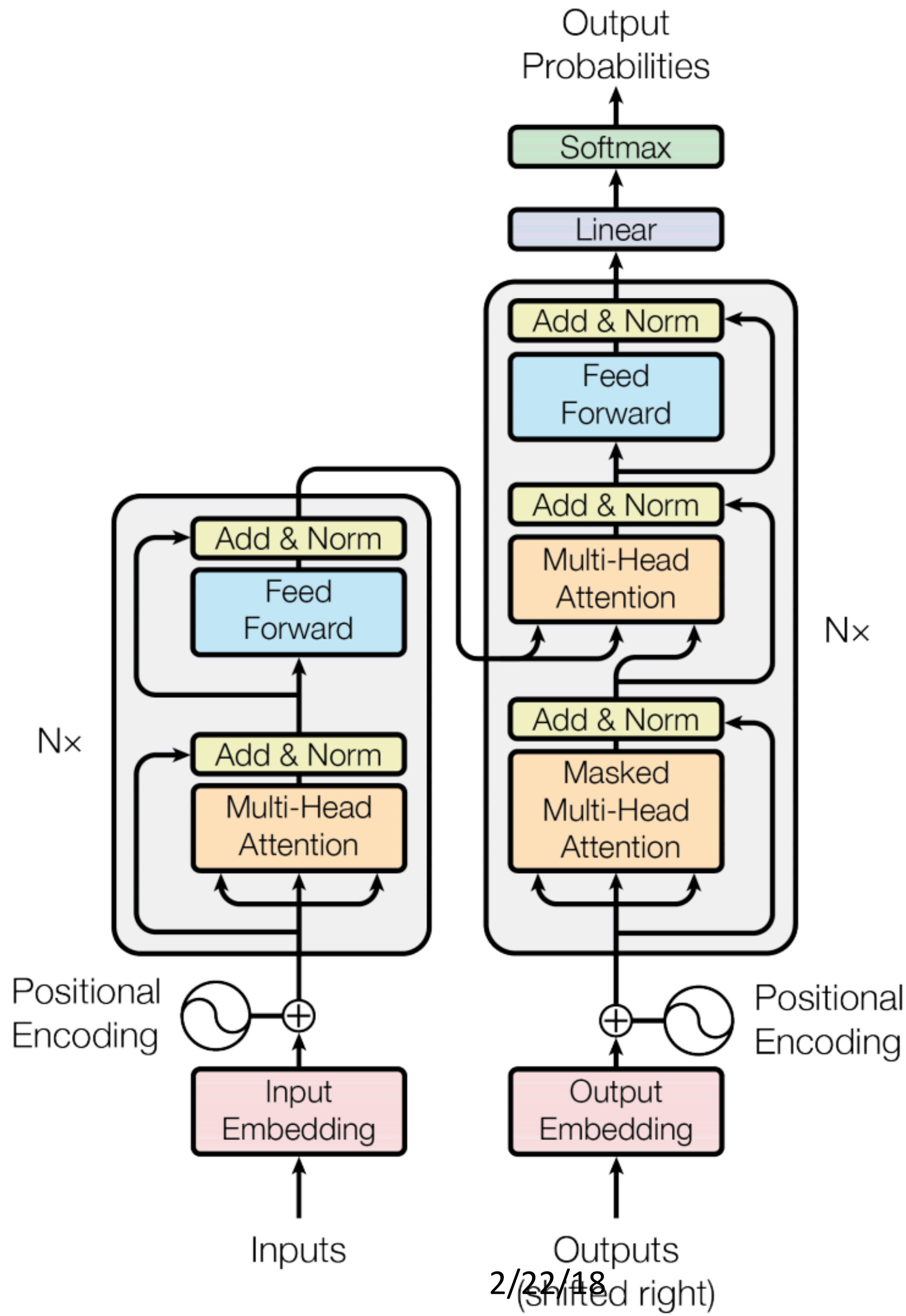
- 2 sublayer changes in decoder
- Masked decoder self-attention on previously generated outputs:



- Encoder-Decoder Attention, where queries come from previous decoder layer and keys and values come from output of encoder



- Blocks repeated 6 times also



# Tips and tricks of the Transformer

Details in paper and later lectures:

- Byte-pair encodings
- Checkpoint averaging
- ADAM optimizer with learning rate changes
- Dropout during training at every layer just before adding residual
- Label smoothing
- Auto-regressive decoding with beam search and length penalties

- → Overall, they are hard to optimize and unlike LSTMs don't usually work out of the box and don't play well yet with other building blocks on tasks.

# Experimental Results for MT

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

# Experimental Results for Parsing

| Parser | Training | WSJ 23 F1 |
|---|---|---|
| Vinyals & Kaiser el al. (2014) [37] | WSJ only, discriminative | 88.3 |
| Petrov et al. (2006) [29] | WSJ only, discriminative | 90.4 |
| Zhu et al. (2013) [40] | WSJ only, discriminative | 90.4 |
| Dyer et al. (2016) [8] | WSJ only, discriminative | 91.7 |
| Transformer (4 layers) | WSJ only, discriminative | 91.3 |
| Zhu et al. (2013) [40] | semi-supervised | 91.3 |
| Huang & Harper (2009) [14] | semi-supervised | 91.3 |
| McClosky et al. (2006) [26] | semi-supervised | 92.1 |
| Vinyals & Kaiser el al. (2014) [37] | semi-supervised | 92.1 |
| Transformer (4 layers) | semi-supervised | 92.7 |
| Luong et al. (2015) [23] | multi-task | 93.0 |
| Dyer et al. (2016) [8] | generative | 93.3 |