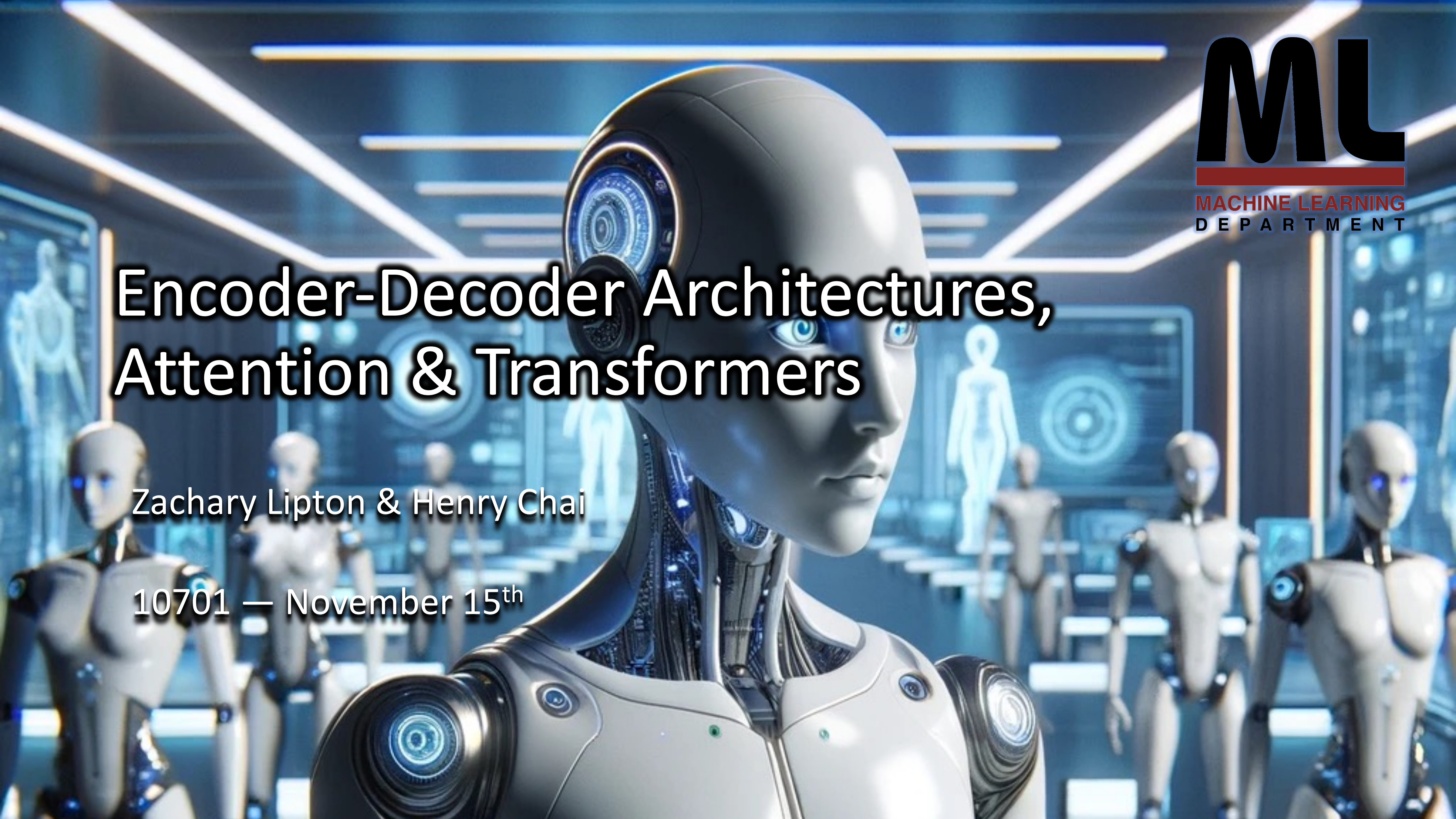


Encoder-Decoder Architectures, Attention & Transformers

Zachary Lipton & Henry Chai

10701 — November 15th

Acknowledgments

- Major thanks to Dr. Bhuwan Dhingra (Duke) & Graham Neubig for fantastic transformer slides!

RNN Tips & Tricks

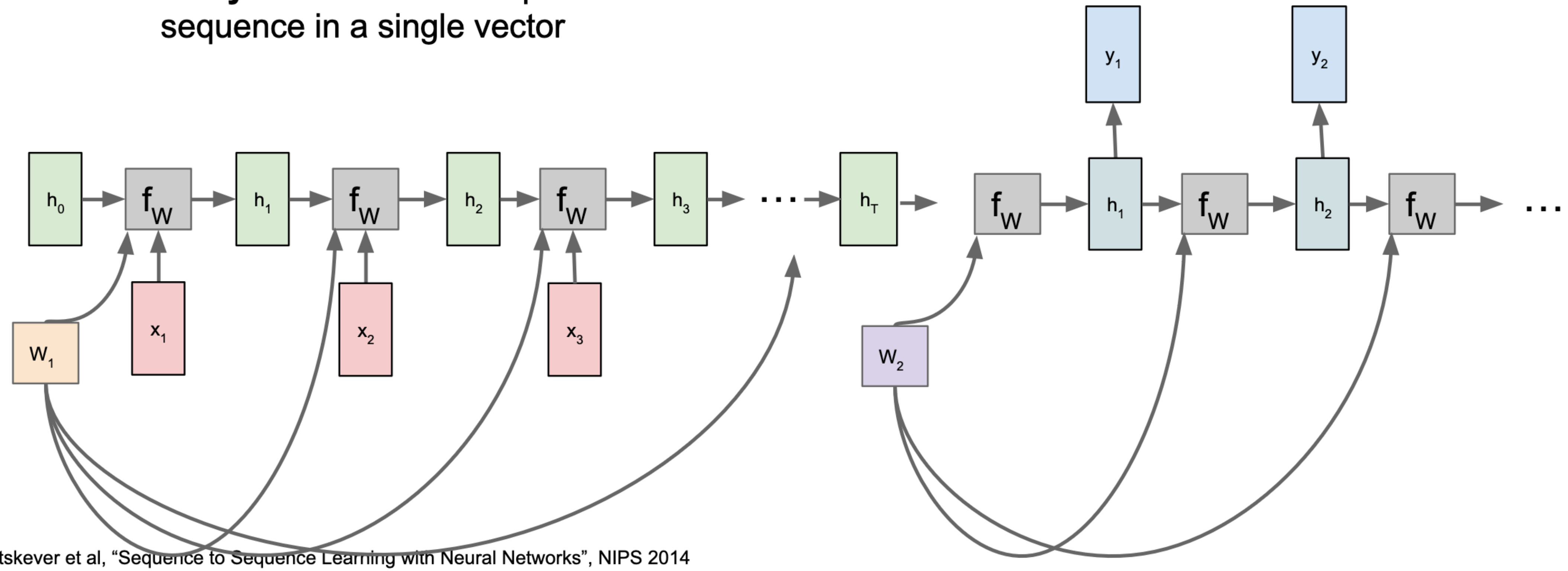
A Brief Review of RNN Tips & Tricks

- Gradient Clipping
- Gated Units
- Bidirectional layers
- Regularization (L2)
- Batch normalization between layers
- Padding (zero padding to match lengths)
- Bucketing (group similar-length sequences)

Encoder–Decoder Architectures

Many to one: Encode input sequence in a single vector

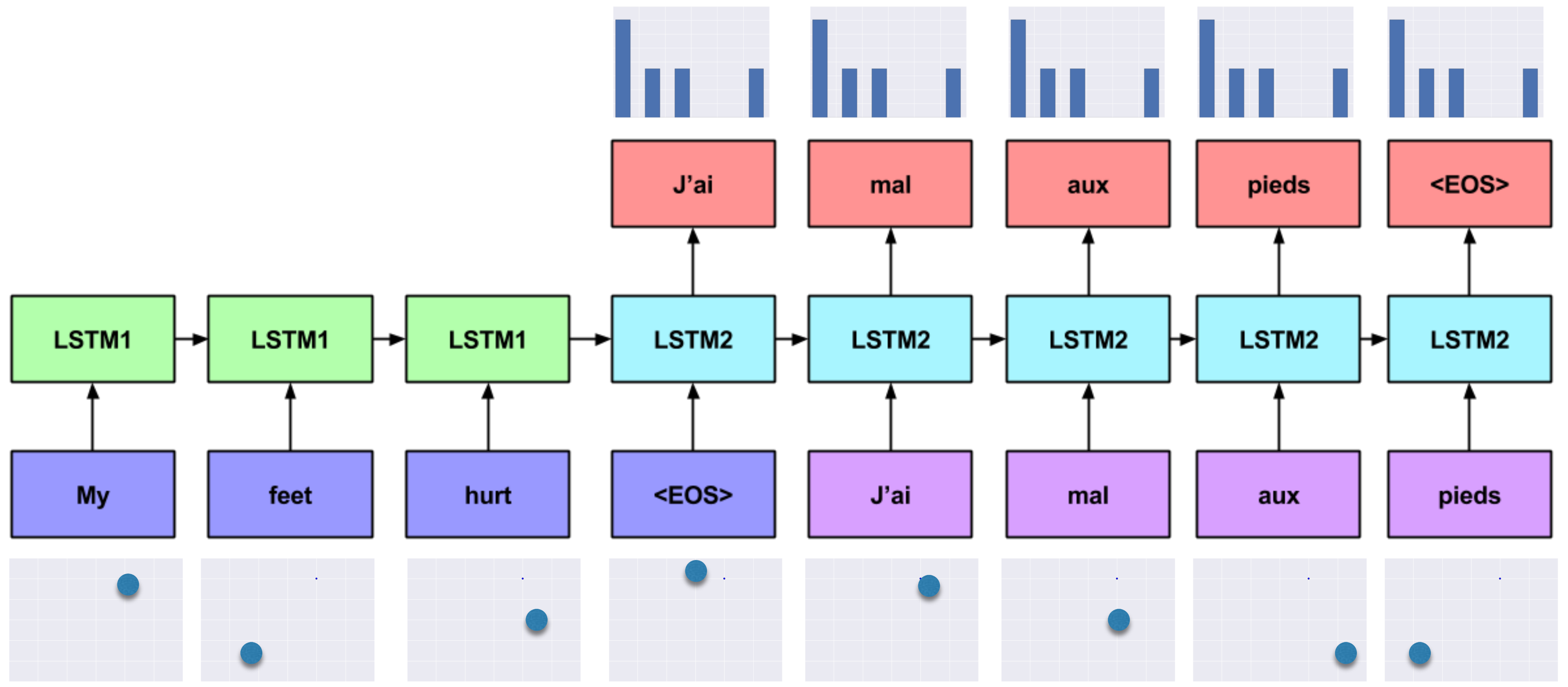
One to many: Produce output sequence from single input vector



Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

slide credit: Andrej Karpathy

Sequence to Sequence (Sutskever et al. 2014)



[Sequence to Sequence Learning with Neural Networks (Sutskever et al. 2014)]

Scale

- Sutskever et al. trained Deep LSTMs with four layers.
- Each layer had 1000 cells
- 384M parameters
- Vocabulary 160,000 words (input) and 80,000 words (output)
- Training Algo: Vanilla SGD, attenuated learning rate
- Other tricks: load the input sentence backwards
- Training time: 10 days with 4 GPUS

Evaluating (Conditional) Language Models

- Perplexity—exponentiated entropy
- BLEU score—average n-gram precision
- ROUGE score—Like BLEU but focused on recall
- METEOR score—incorporates synonyms and paraphrases
- Human evals—Likert scale, preferences among candidates, etc.

Sequence 2 Sequence Results

- Achieved BLEU (measure of translation quality) comparable to best state of the art systems
- Hybrid approaches and ensembling LSTMS led to scores even better than state of the art systems
- No information about language was explicitly modeled or hardwired

Image Captioning: CNN Encoders + RNN Decoders

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

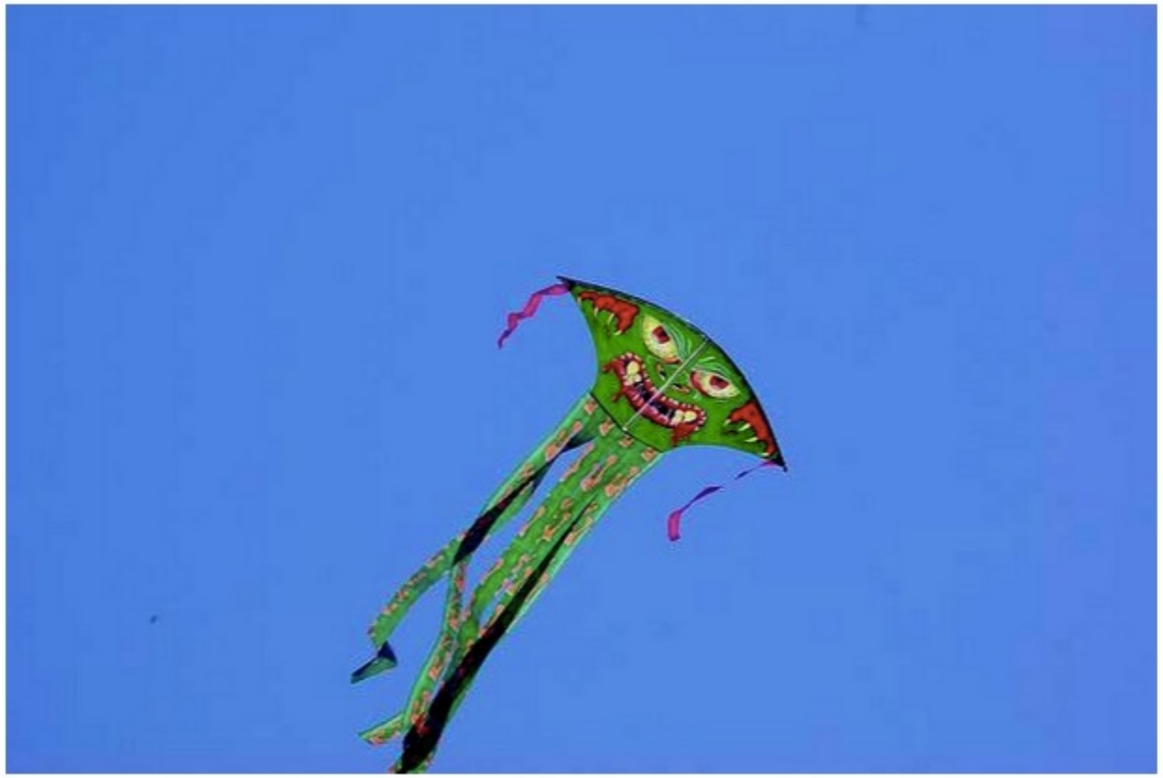
"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

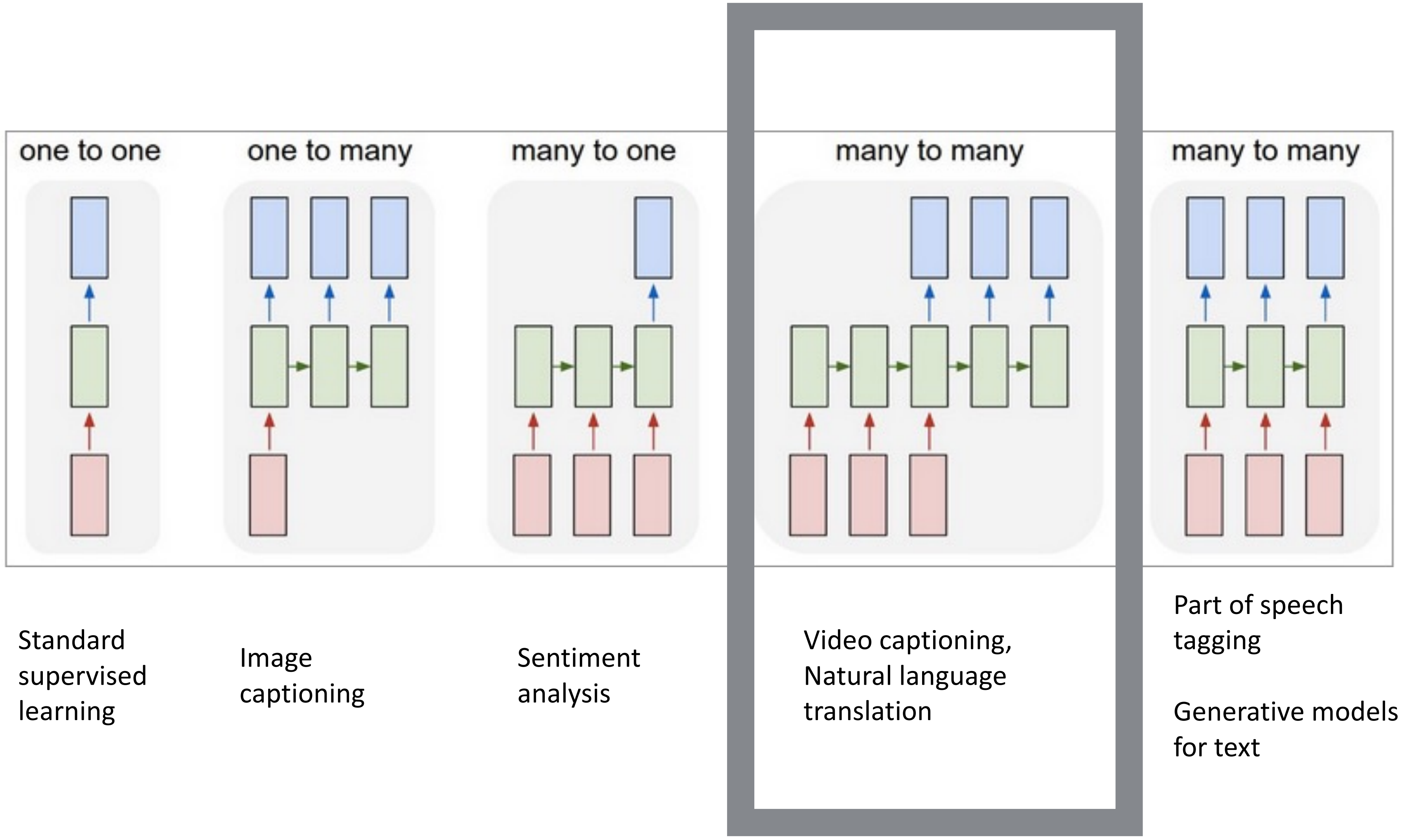
Karpathy et al. (CVPR 2015)

Still a Ways to Go



Human: "A green monster kite soaring in a sunny sky."
Computer model: "A man flying through the air while riding a snowboard."

Simulating Code Execution



Standard supervised learning

Image captioning

Sentiment analysis

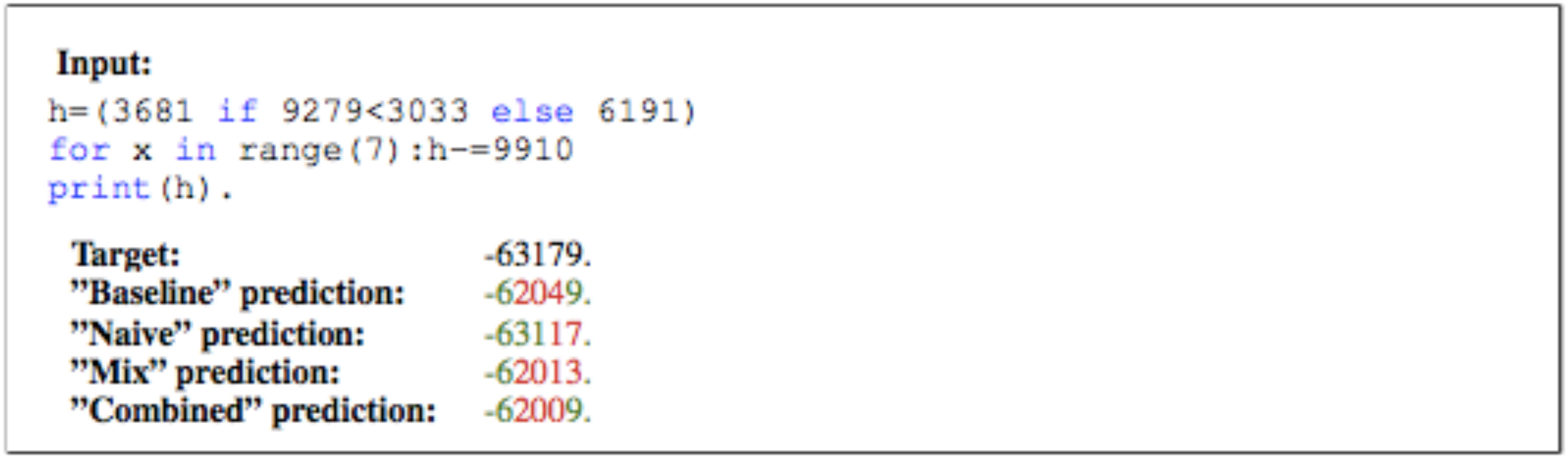
Video captioning, Natural language translation

Part of speech tagging

Generative models for text

Learning to Execute

- RNN reads programs at character level, predicts program output
- Curriculum learning: introduce operators one at a time
- Results: learned to add 9 digits integers with 99% accuracy



Input:

```
h=(3681 if 9279<3033 else 6191)
for x in range(7):h-=9910
print(h).
```

```
Target:                        -63179.
"Baseline" prediction:         -62049.
"Naive" prediction:            -63117.
"Mix" prediction:              -62013.
"Combined" prediction:         -62009.
```

Zaremba & Sutskever 2015

Problems with RNNs / LSTMs

- The entire word history is represented by one (or two) vectors
  - Though increasing the size of the vector(s) helps

"You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!"

Ray Mooney
NLP Prof @ UT Austin

Problems with RNNs / LSTMs

- The entire word history is represented by one (or two) vectors (Though increasing the size of the vector(s) helps)
- We cannot parallelize computation across different words

"You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!"

Ray Mooney
NLP Prof @ UT Austin

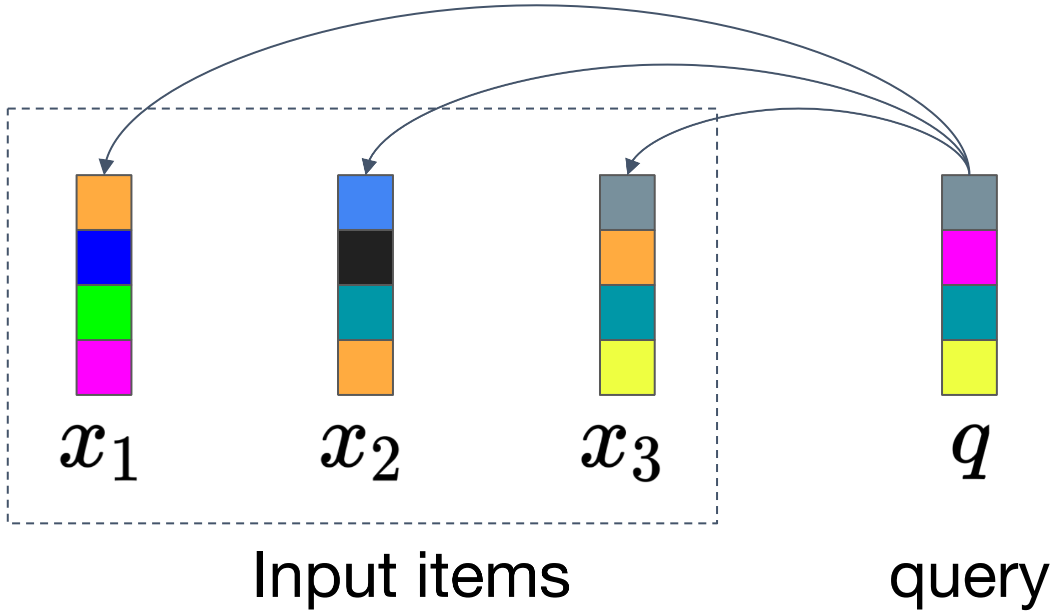
⇒ Transformer Networks (*Attention is all you need, Vaswani et al, 2017*)

Neural attention

- Compare a set of items based on a query

$$\text{score}(x_i, q) = x_i^T q$$

Score can be computed in other ways, e.g. using a feedforward network



Neural attention

- Compare a set of items based on a query

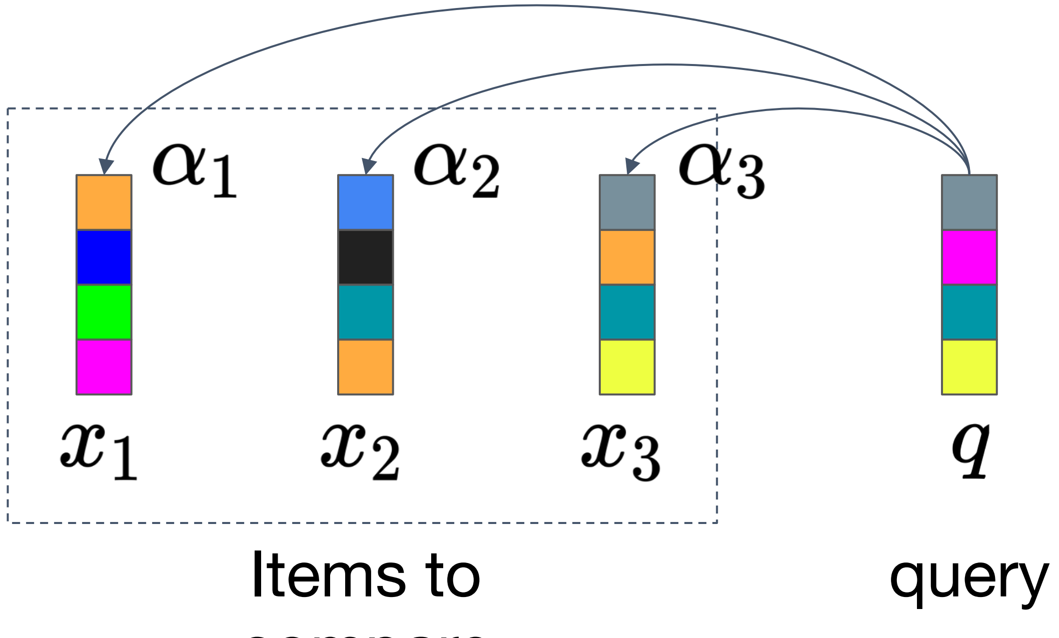
$$\text{score}(x_i, q) = x_i^T q$$

Score can be computed in other ways, e.g. using a feedforward network

$$\alpha_i = \frac{\exp(\text{score}(x_i, q))}{\sum_{i'} \exp(\text{score}(x_{i'}, q))}$$

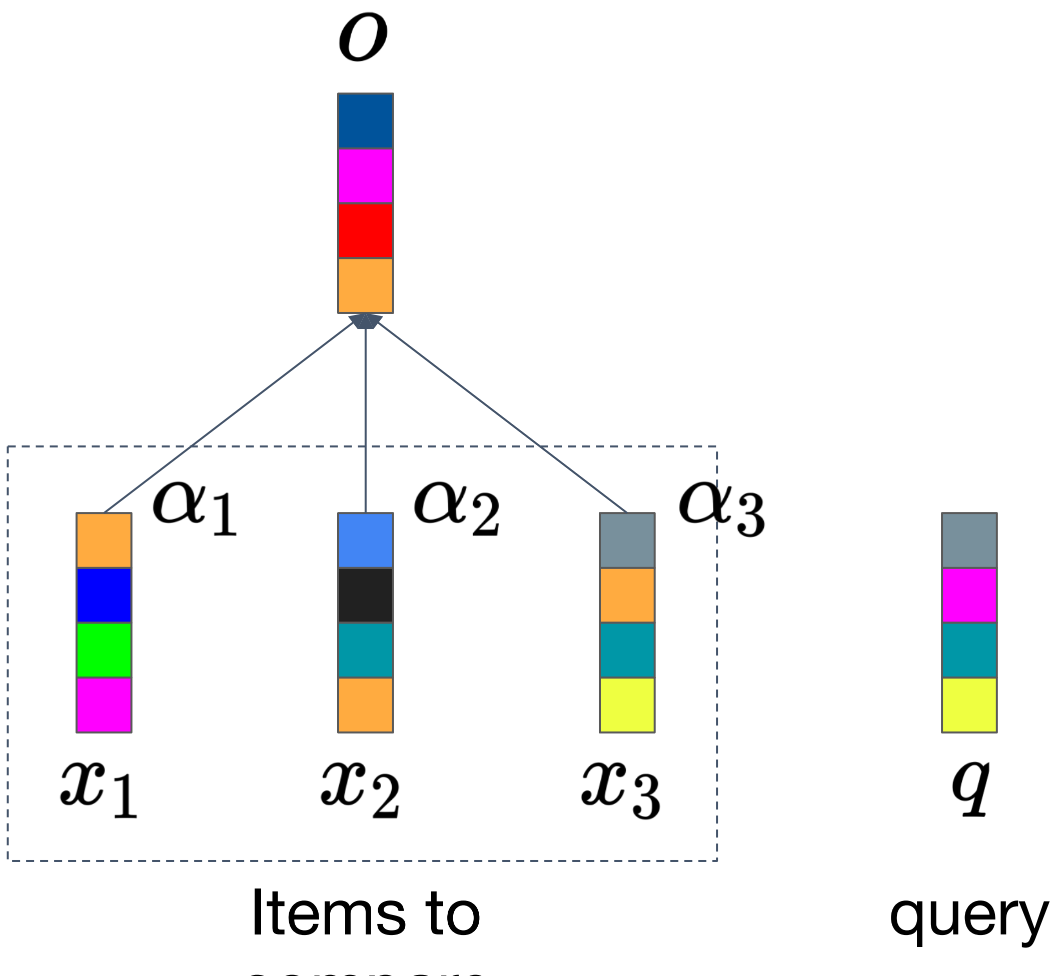
Softmax

Attention from query to item i



Neural attention

- Compare a set of items based on a query
- Aggregate items based on attention weights



$$\text{score}(x_i, q) = x_i^T q$$

Score can be computed in other ways, e.g. using a feedforward network

$$\alpha_i = \frac{\exp(\text{score}(x_i, q))}{\sum_{i'} \exp(\text{score}(x_{i'}, q))}$$

Softmax

$$o = \sum_i \alpha_i x_i$$

Aggregated representation based on query

Attention Score Functions (1)

- ***q*** is the query and ***k*** is the key
- **Multi-layer Perceptron** (Bahdanau et al. 2015)

$$a\ \boldsymbol{q}\quad) = \boldsymbol{w}_2 \tanh\ W\ [\boldsymbol{q};\quad])$$

  - Flexible, often very good with large data

- **Bilinear** (Luong et al. 2015)

$$a\ \boldsymbol{q}\quad) = \boldsymbol{q}\ W$$

Attention Score Functions (2)

- **Dot Product** (Luong et al. 2015)

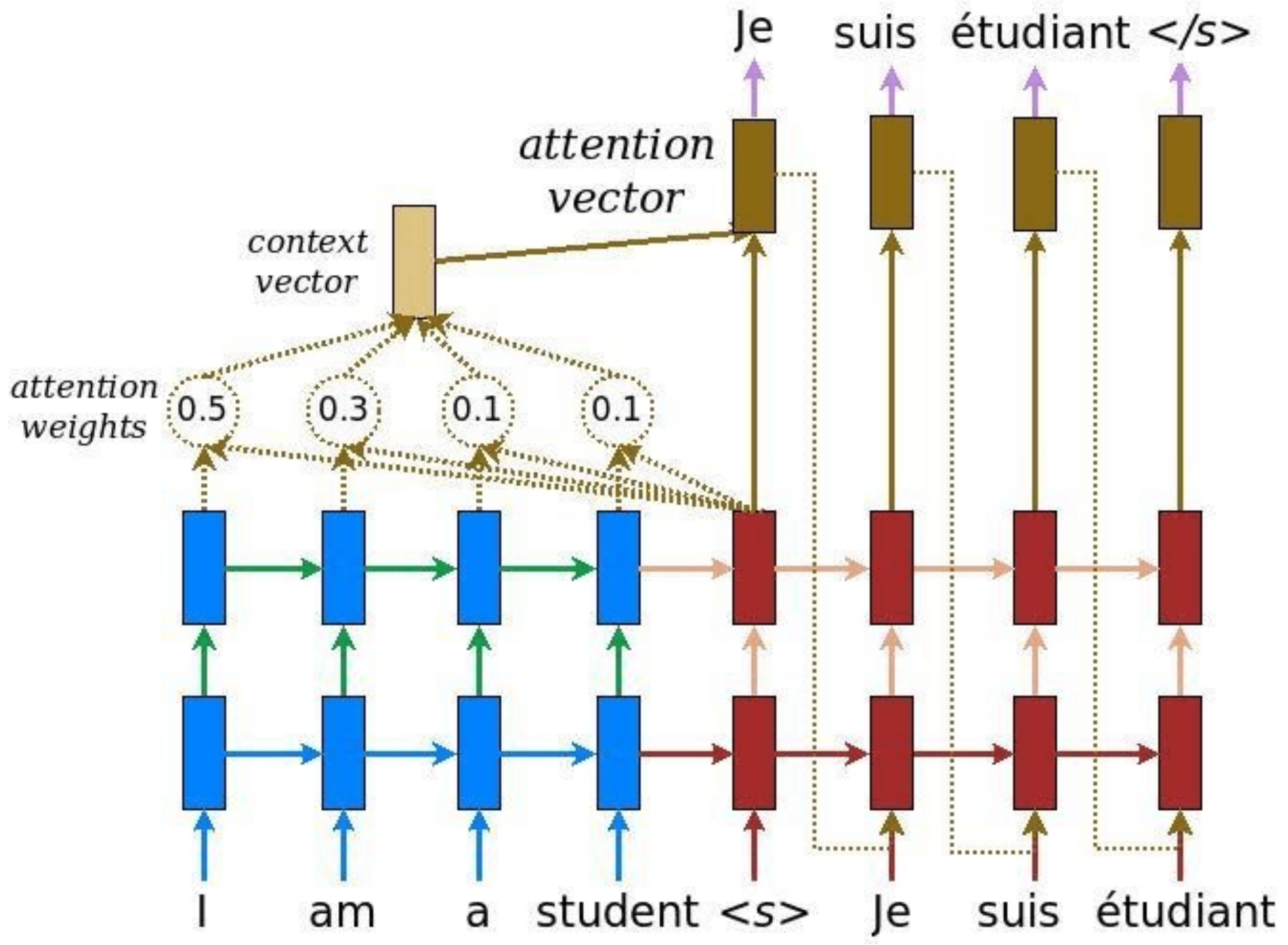
$$a\ \boldsymbol{q}\quad) = \boldsymbol{q}$$

  - No parameters! But requires sizes to be the same.

- **Scaled Dot Product** (Vaswani et al. 2017)
  - *Problem:* scale of dot product increases as dimensions get larger
  - *Fix:* scale by size of the vector

$$a\ \boldsymbol{q}\quad) = \frac{\boldsymbol{q}}{\quad\overline{}}$$

Attention Applied to Encoder-Decoder



Implicitly Learning Alignments

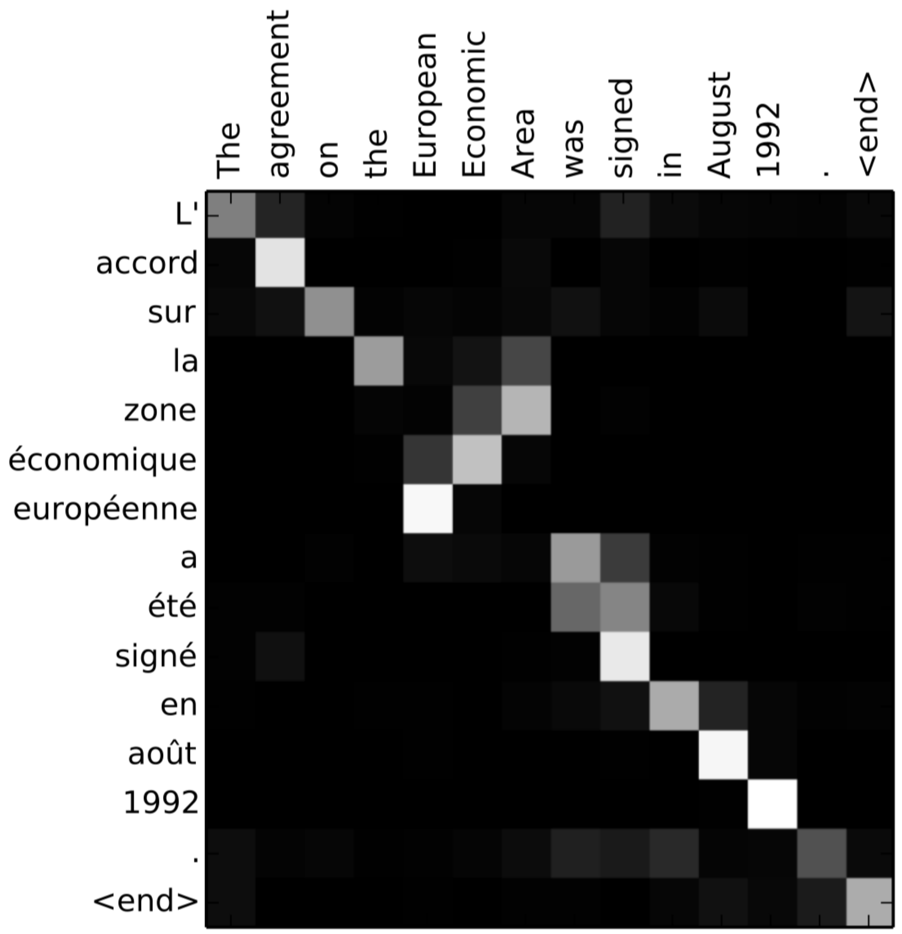


Image from Bahdanau et al. (2015)

Self-Attention & the Transformer Architecture

arXiv:1706.03762v7 [cs.CL] 2 Aug 2023

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

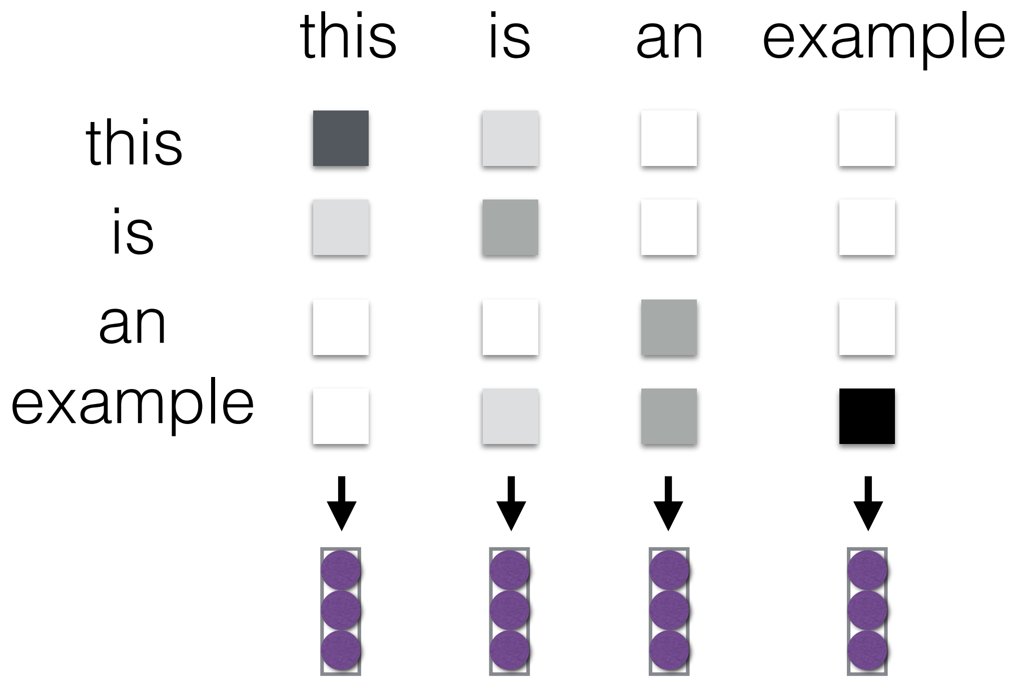
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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

https://arxiv.org/pdf/1706.03762.pdf

Self Attention

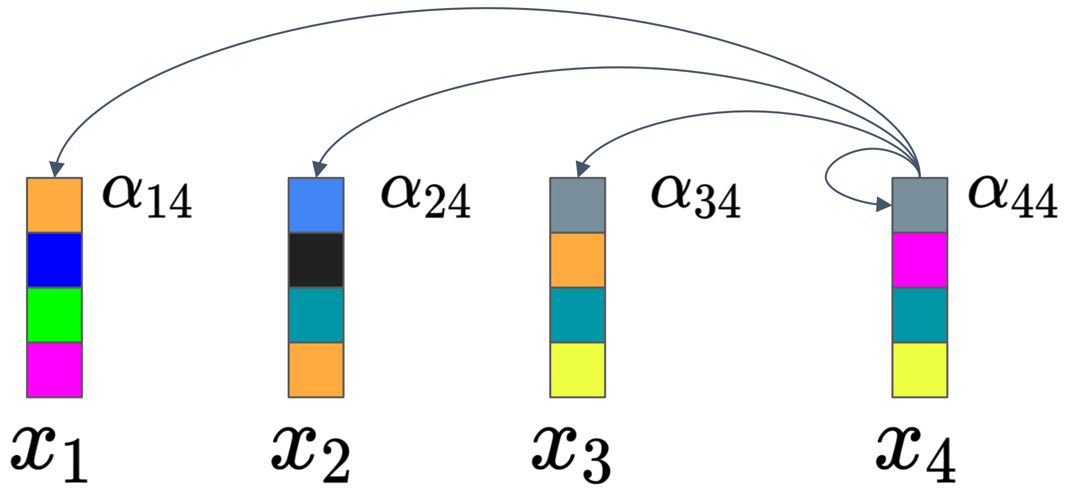
(Cheng et al. 2016, Vaswani et al. 2017)

- Each element in the sentence attends to other elements → context sensitive encodings!



Self-attention layers

- Items and queries come from the same sequence



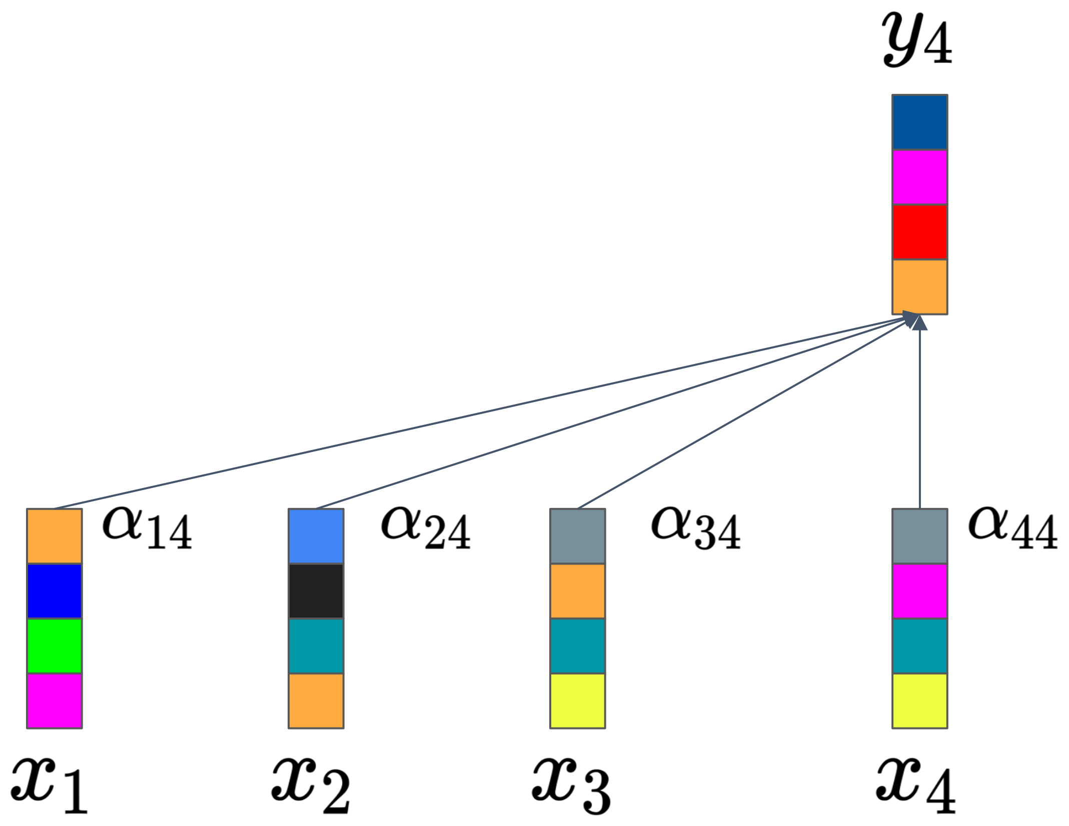
$$\text{score}(x_i, x_j) = x_i^T x_j$$

$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{i'} \exp(\text{score}(x_{i'}, x_j))} \longleftarrow \text{Attention from item } j \text{ to item } i$$

$$y_j = \sum_i \alpha_{ij} x_i \longleftarrow \text{Output for item j}$$

Self-attention layers

- Items and queries come from the same sequence



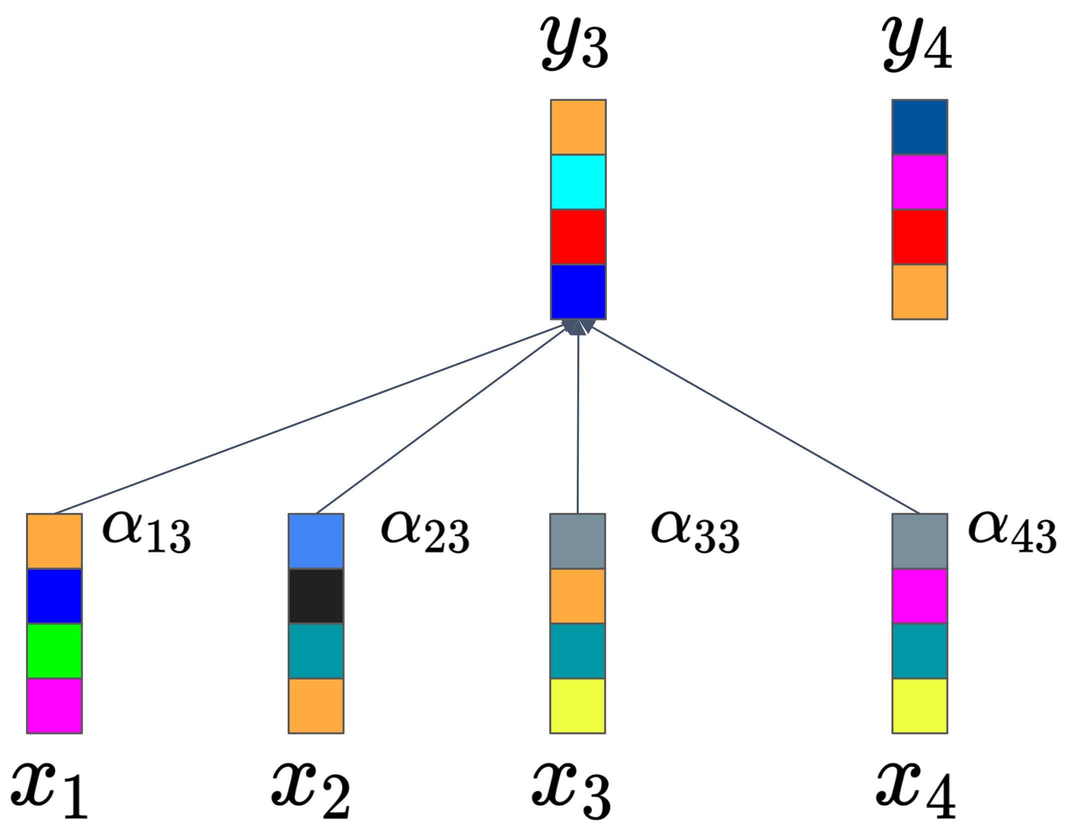
$$\text{score}(x_i, x_j) = x_i^T x_j$$

$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{i'} \exp(\text{score}(x_{i'}, x_j))}$$ ← Attention from item *j* to item *i*

$$y_j = \sum_i \alpha_{ij} x_i$$ ← Output for item j

Self-attention layers

- Items and queries come from the same sequence



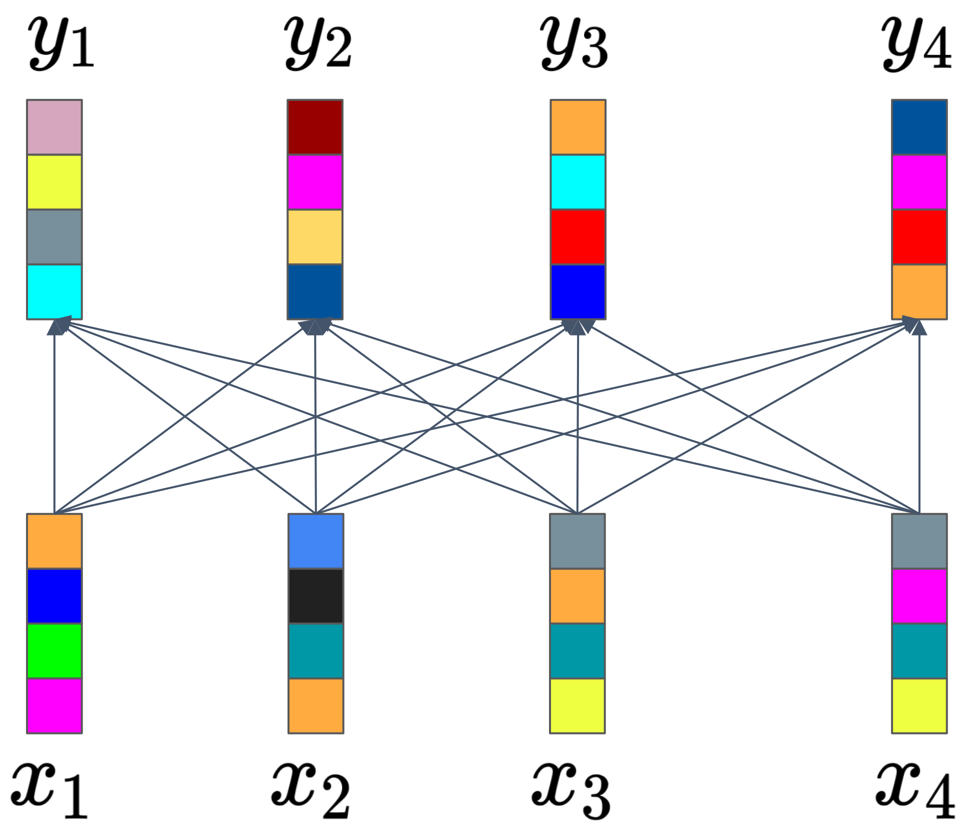
$$\text{score}(x_i, x_j) = x_i^T x_j$$

$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{i'} \exp(\text{score}(x_{i'}, x_j))}$$ ← Attention from item *j* to item *i*

$$y_j = \sum_i \alpha_{ij} x_i$$ ← Output for item j

Self-attention layers

- Items and queries come from the same sequence



$$\mathrm{score}(x_i, x_j) = x_i^T x_j$$

$$\alpha_{ij} = \frac{\exp(\mathrm{score}(x_i, x_j))}{\sum_{i'} \exp(\mathrm{score}(x_{i'}, x_j))}$$ ← Attention from item *j* to item *i*

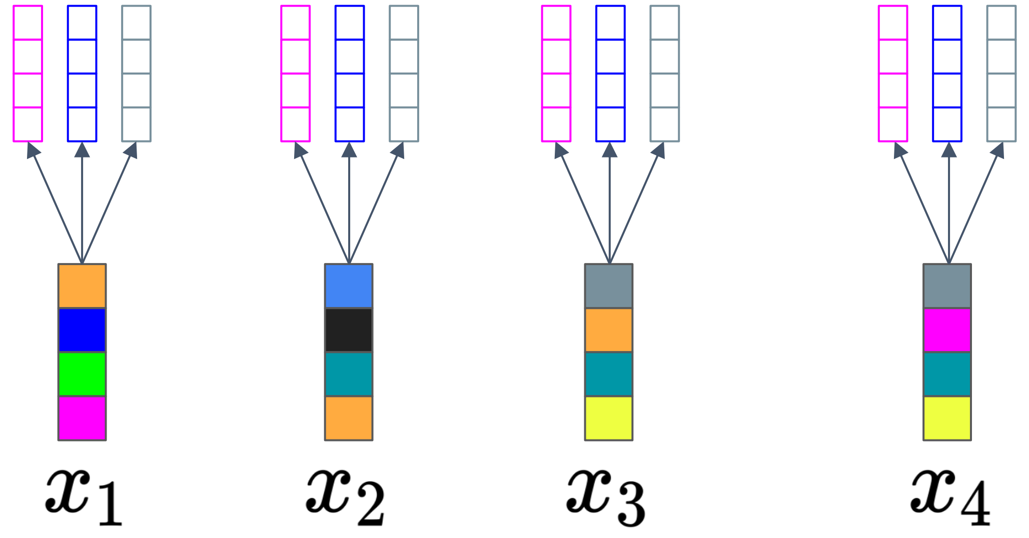
$$y_j = \sum_i \alpha_{ij} x_i$$ ← Output for item j

Self-attention layers

- Let's add some parameters to the layer to model how the items interact with each other

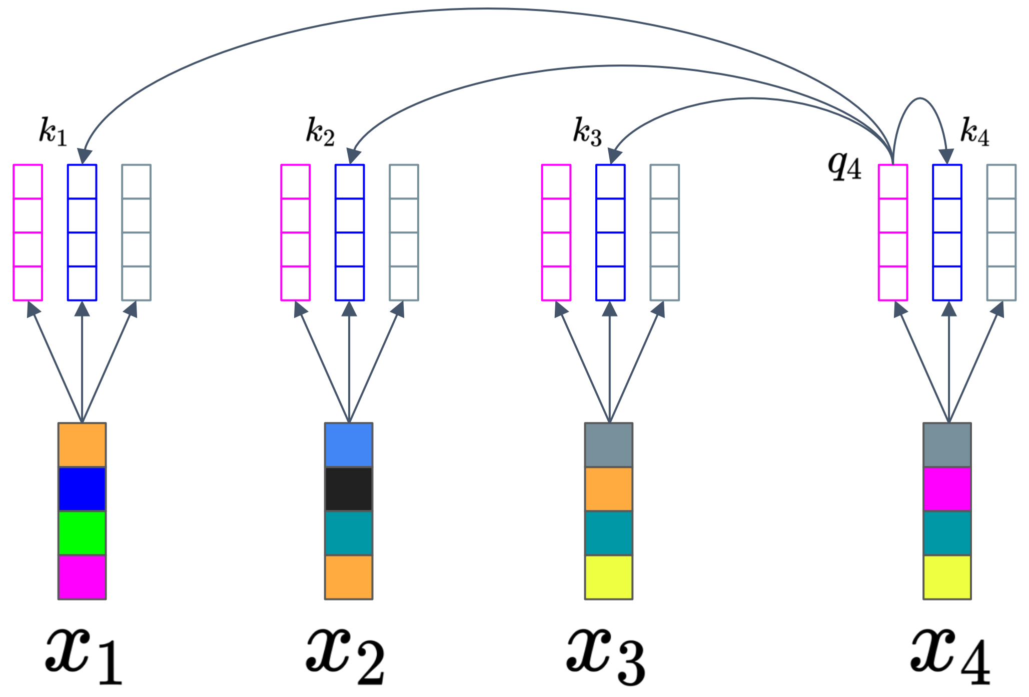
$\boxed{q_i = W_Q x_i}$ $\boxed{k_i = W_K x_i}$ $\boxed{v_i = W_V x_i}$

Project each item to three different roles:
queries, keys and values



Self-attention layers

- Let’s add some parameters to the layer to model how the items interact with each other



$\boxed{q_i = W_Q x_i}$ $\boxed{k_i = W_K x_i}$ $\boxed{v_i = W_V x_i}$

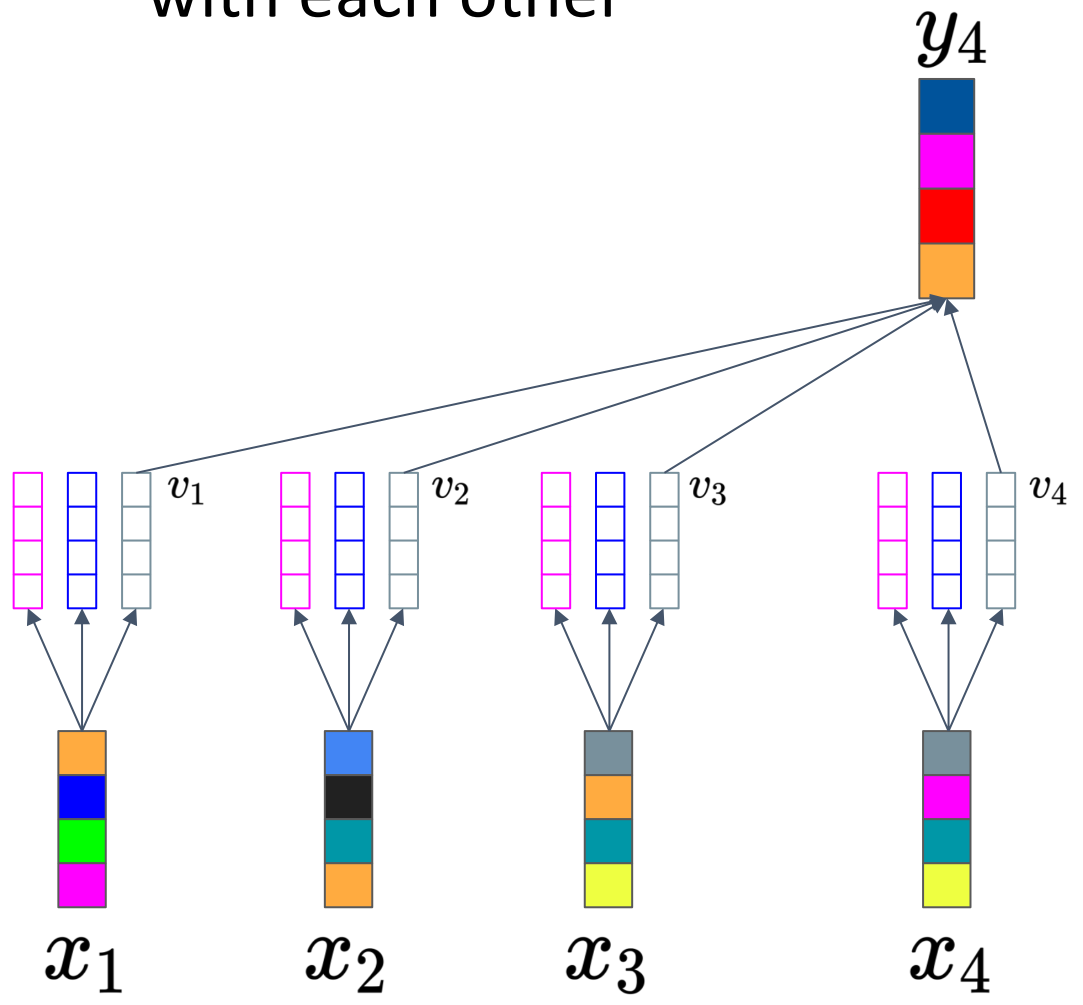
Project each item to three different roles:
queries, **keys** and **values**

$$\text{score}(x_i, x_j) = k_i^T q_j$$

Compare **queries** and **keys**

Self-attention layers

- Let's add some parameters to the layer to model how the items interact with each other



$\boxed{q_i = W_Q x_i}$ $\boxed{k_i = W_K x_i}$ $\boxed{v_i = W_V x_i}$

Project each item to three different roles:
queries, **keys** and **values**

$$\text{score}(x_i, x_j) = k_i^T q_j$$

Compare **queries** and **keys**

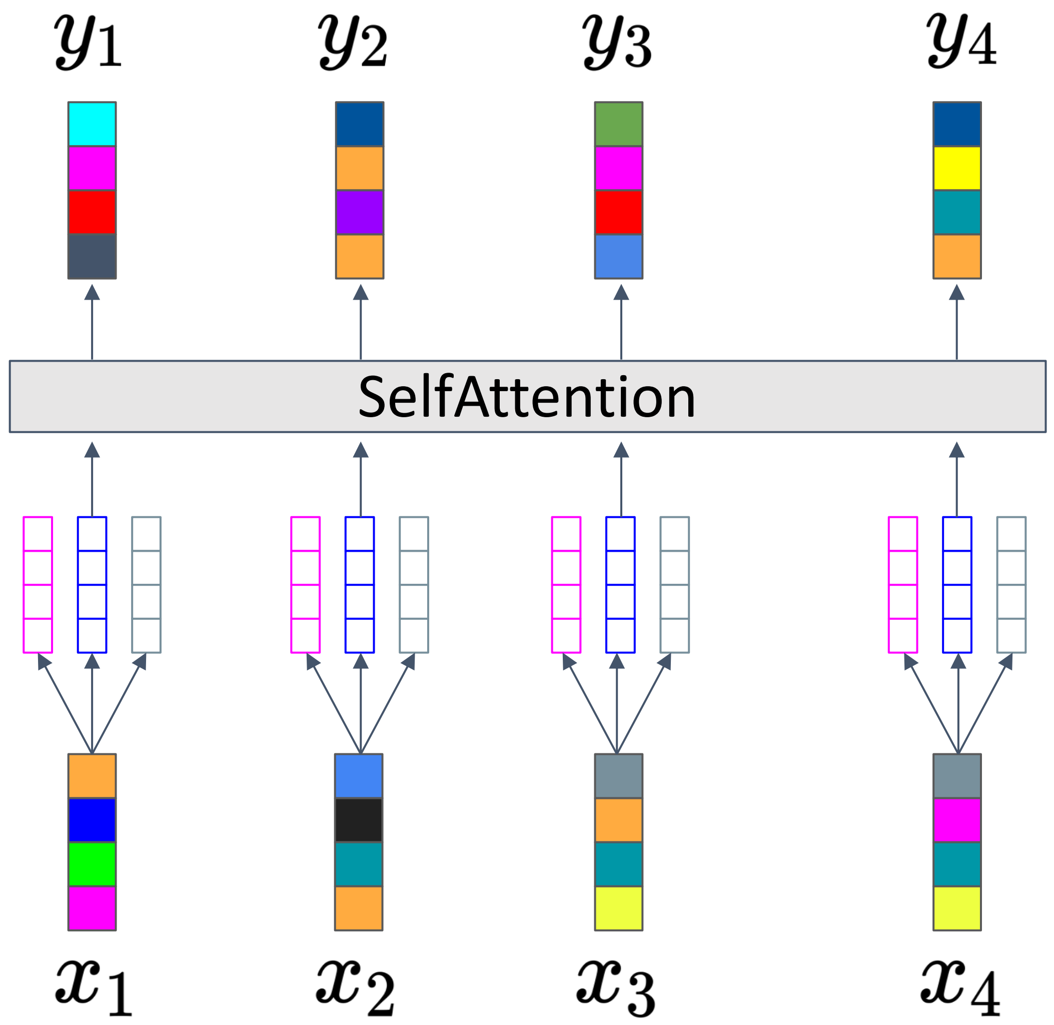
$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{i'} \exp(\text{score}(x_{i'}, x_j))}$$

$$y_j = \sum_i \alpha_{ij} v_i$$

Take attention-weighted sum of the **values**

Self-attention layers

- Same computation performed at each sequence position



$\boxed{q_i = W_Q x_i}$ $\boxed{k_i = W_K x_i}$ $\boxed{v_i = W_V x_i}$

Project each item to three different roles:
queries, **keys** and **values**

$$\text{score}(x_i, x_j) = k_i^T q_j$$

Compare **queries** and **keys**

$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{i'} \exp(\text{score}(x_{i'}, x_j))}$$

$$y_j = \sum_i \alpha_{ij} v_i$$

Take attention-weighted sum of the **values**

Parallelization across the sequence

$$X = [x_1; x_2; \ldots; x_T] \qquad (T \times d)$$

$$W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$$

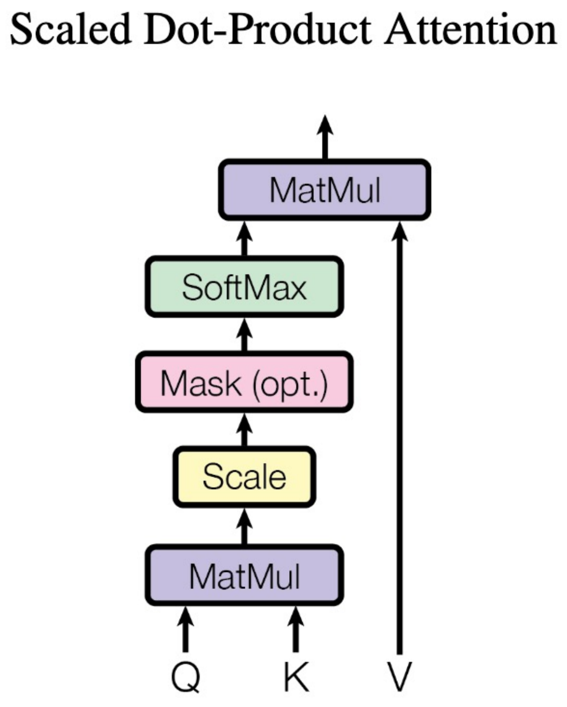
$$Q = XW_Q \qquad K = XW_K \qquad V = XW_V \qquad (T \times d)$$

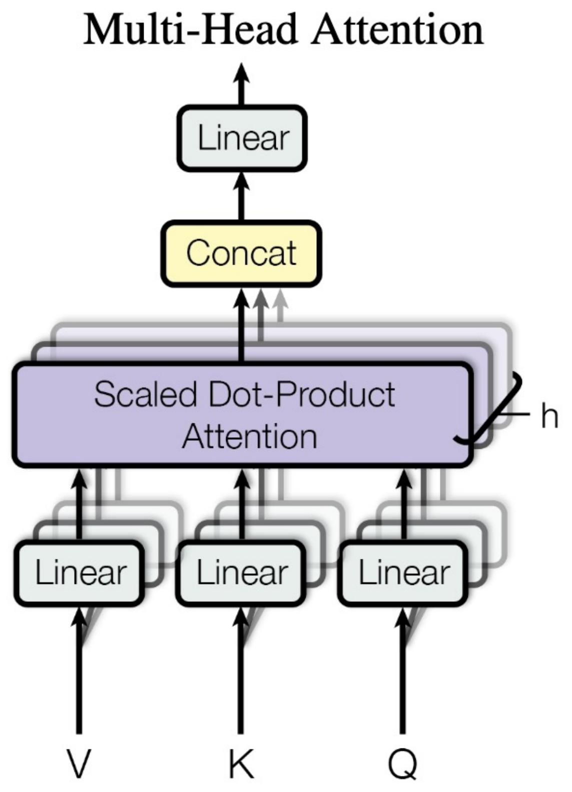
$$\text{SelfAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \qquad (T \times d)$$

Applied to each row separately

For numerical stability

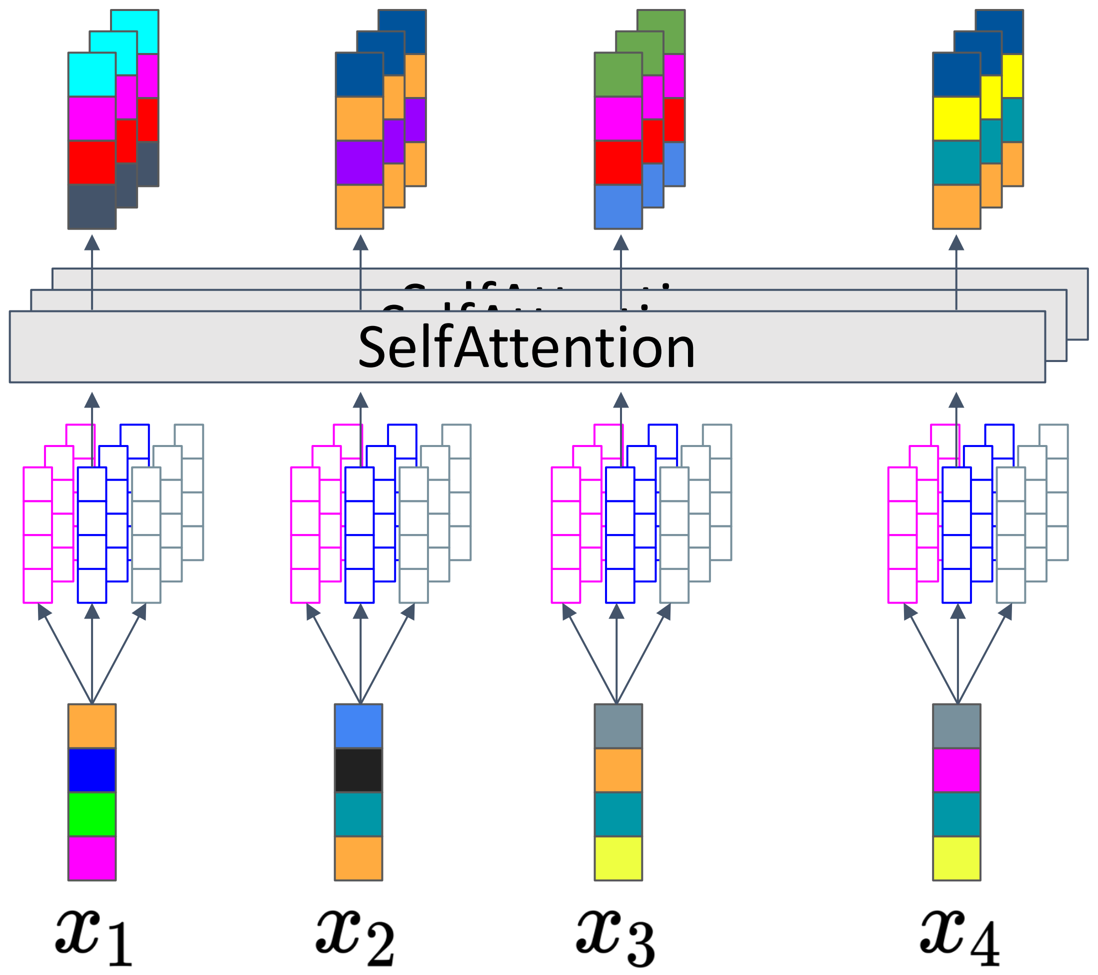
Multihead–Attention





Multi-head attention

- Use multiple self-attention layers, each with its own set of parameters



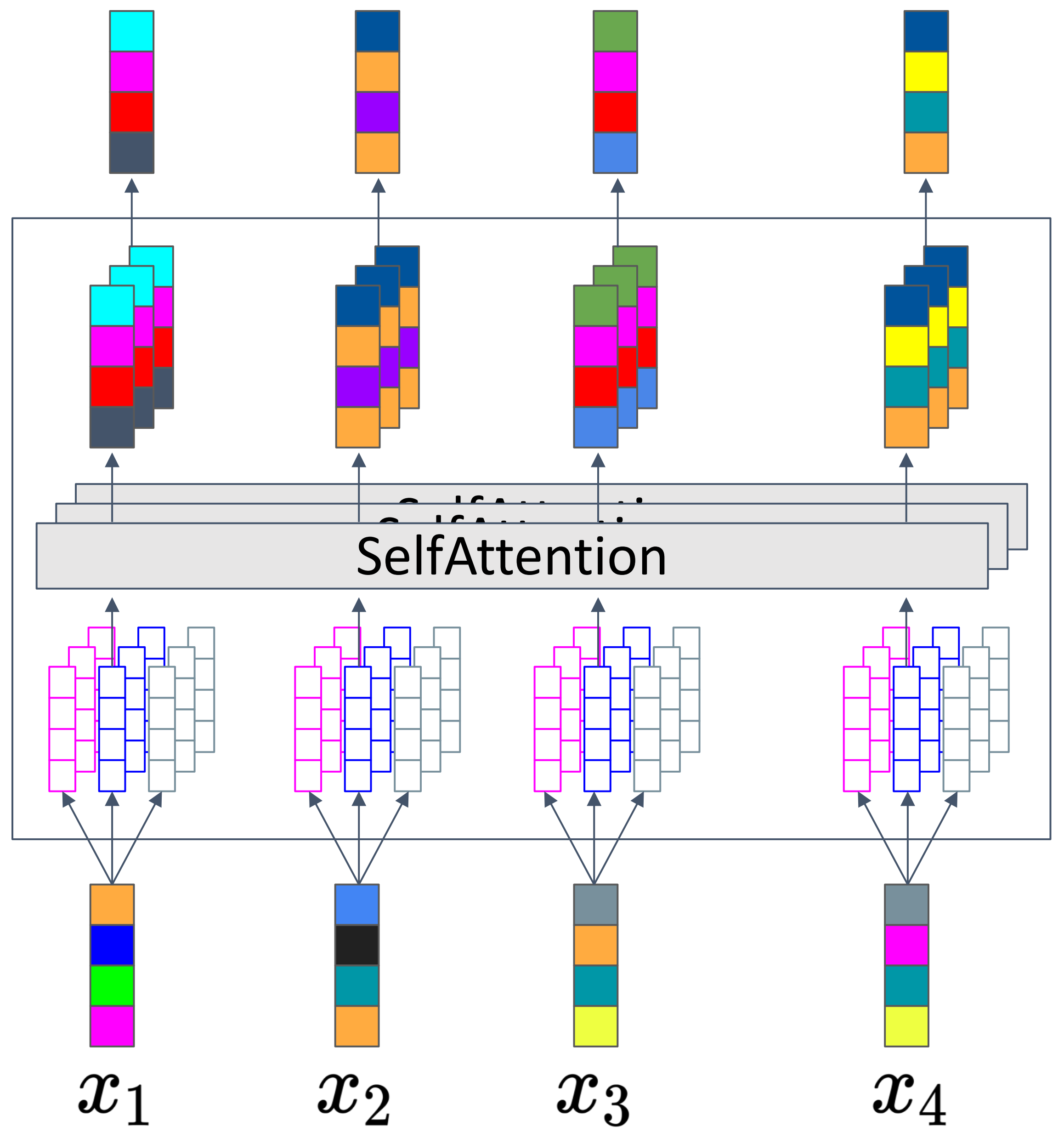
$$Q^h = XW_Q^h$$

$$K^h = XW_K^h$$

$$V^h = XW_V^h$$

$$\text{head}^h = \text{SelfAttention}(Q^h, K^h, V^h)$$

… & keeping dimensionality under control



$$Q^h = XW_Q^h$$

$$K^h = XW_K^h$$

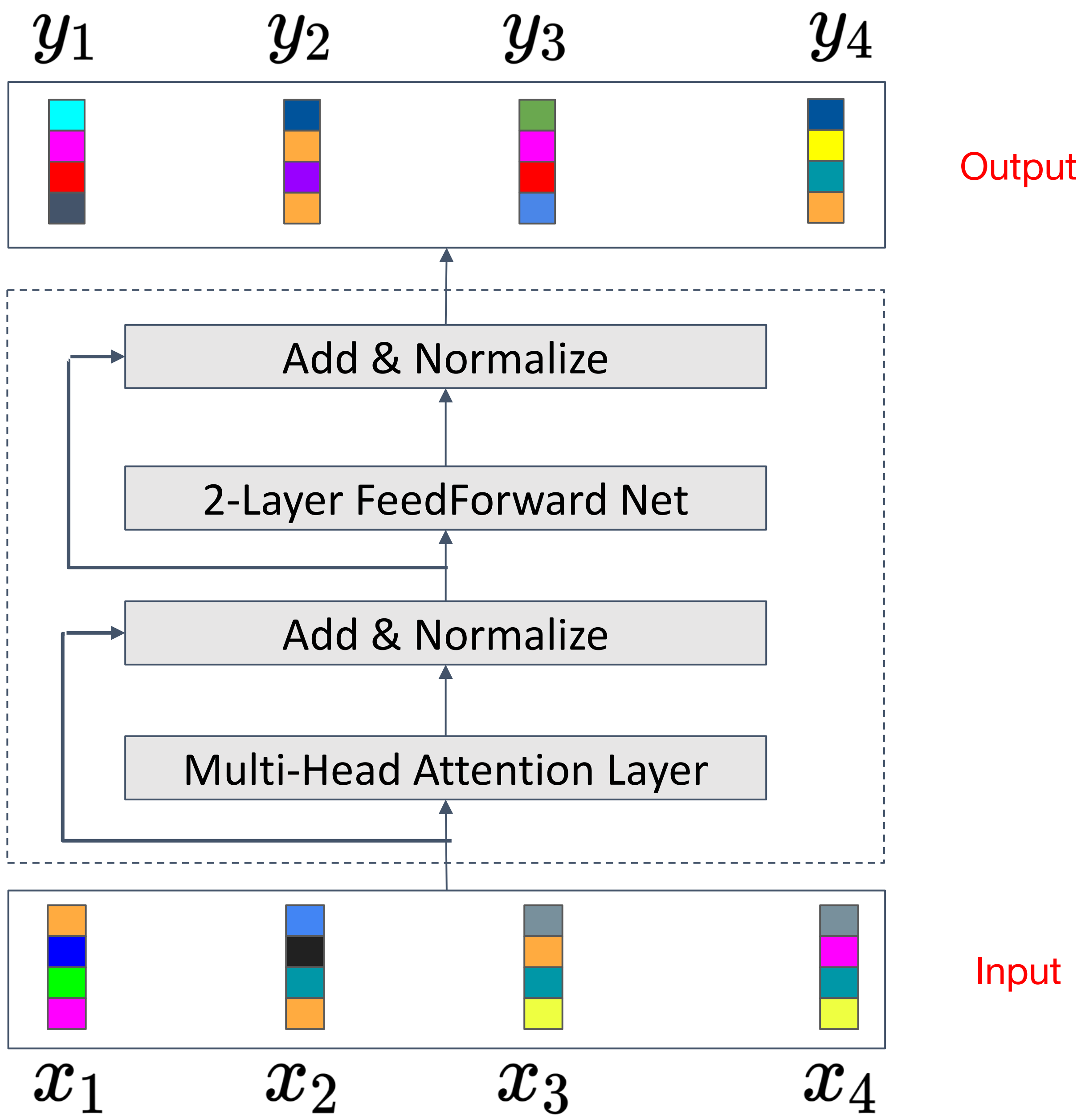
$$V^h = XW_V^h$$

$$\text{head}^h = \text{SelfAttention}(Q^h, K^h, V^h)$$

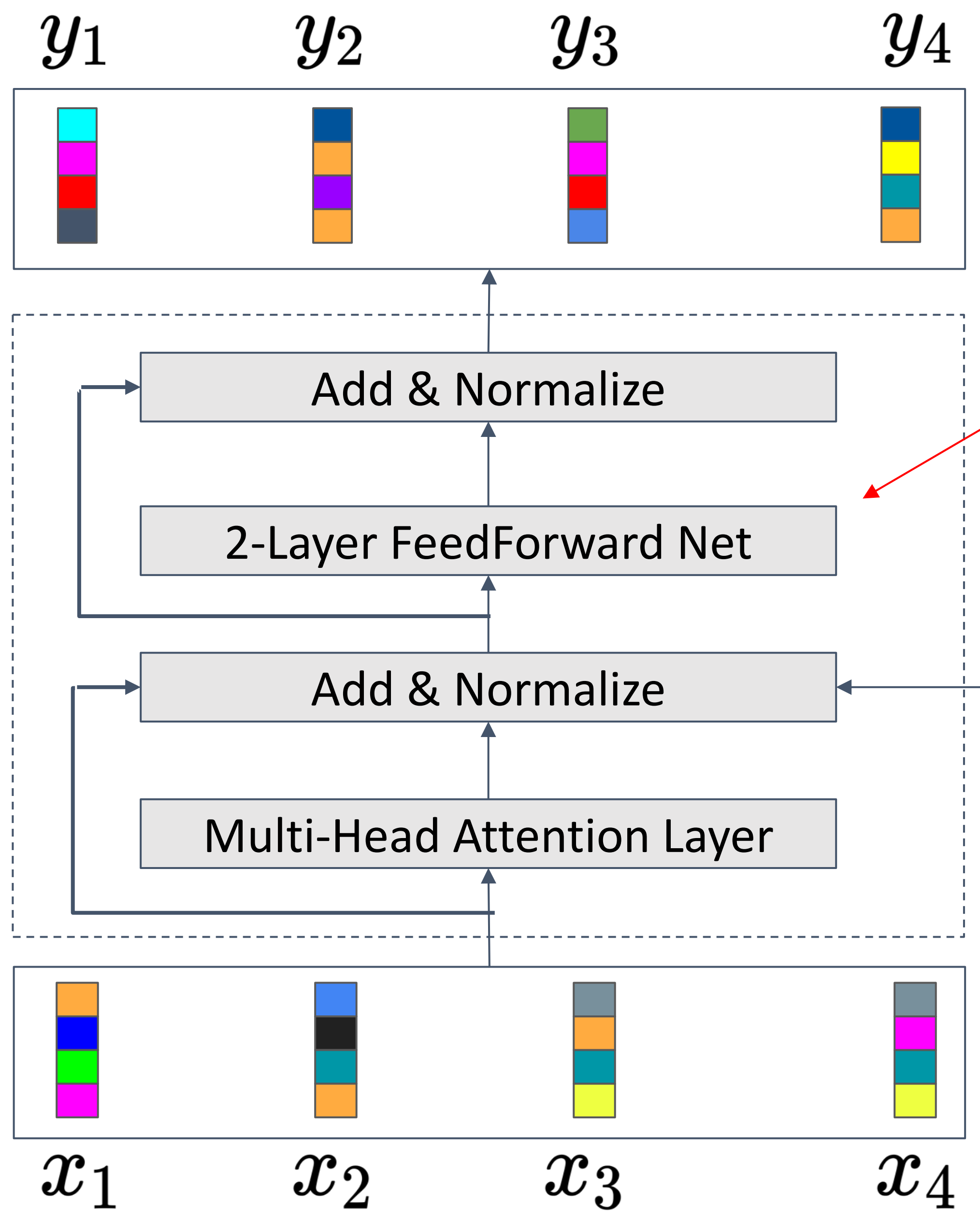
$$\text{MultiHead}(X) = [\text{head}^1, \text{head}^2, \ldots]W_O$$

Concatenate and project back to size *d*

Transformer layers



Transformer layers



Output

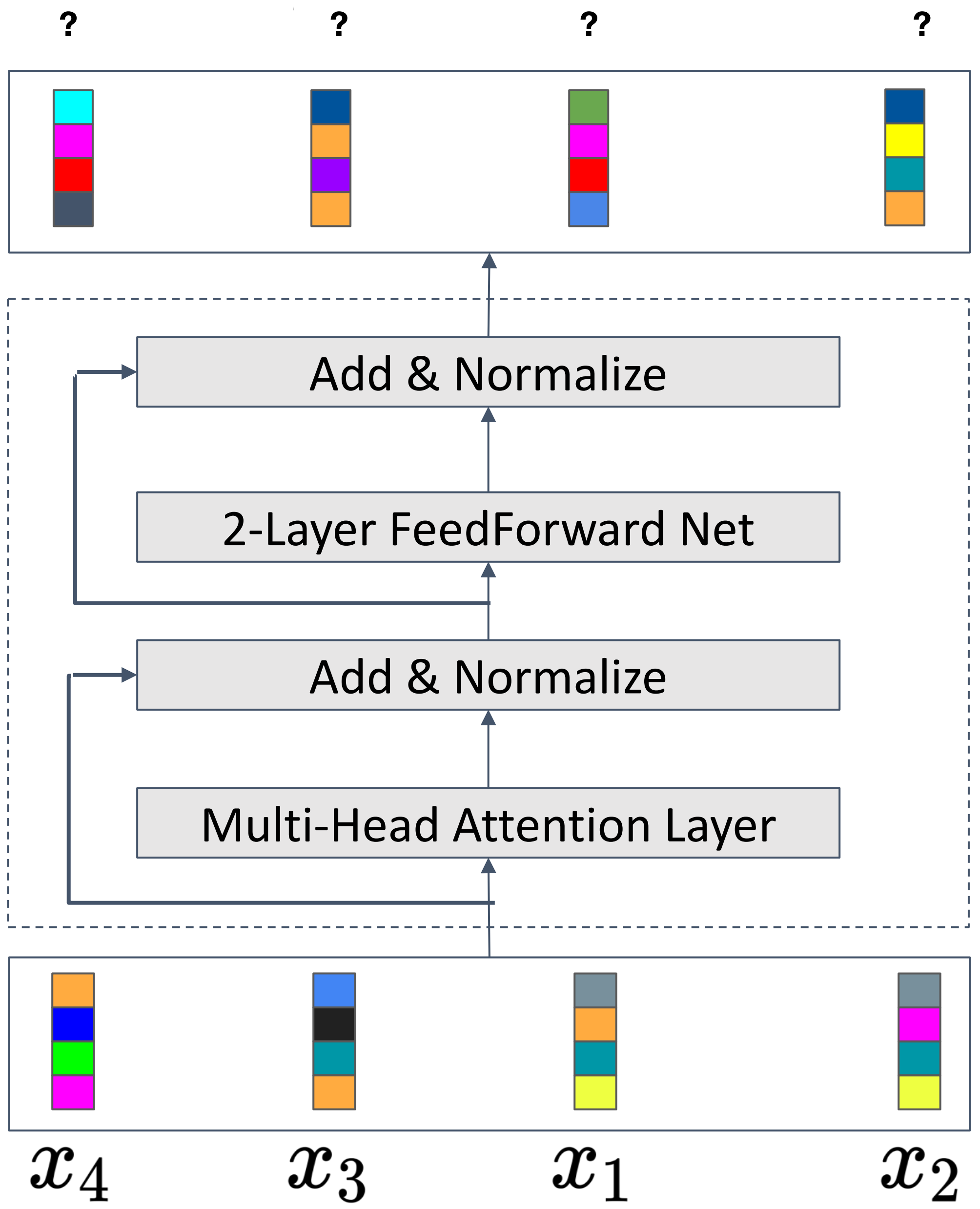
Applied to each vector in parallel

Normalize vectors to zero mean and unit standard dev

$$\mathrm{LayerNorm}(X + \mathrm{MultiHead}(X))$$

Input

What if we switch the order of the inputs?



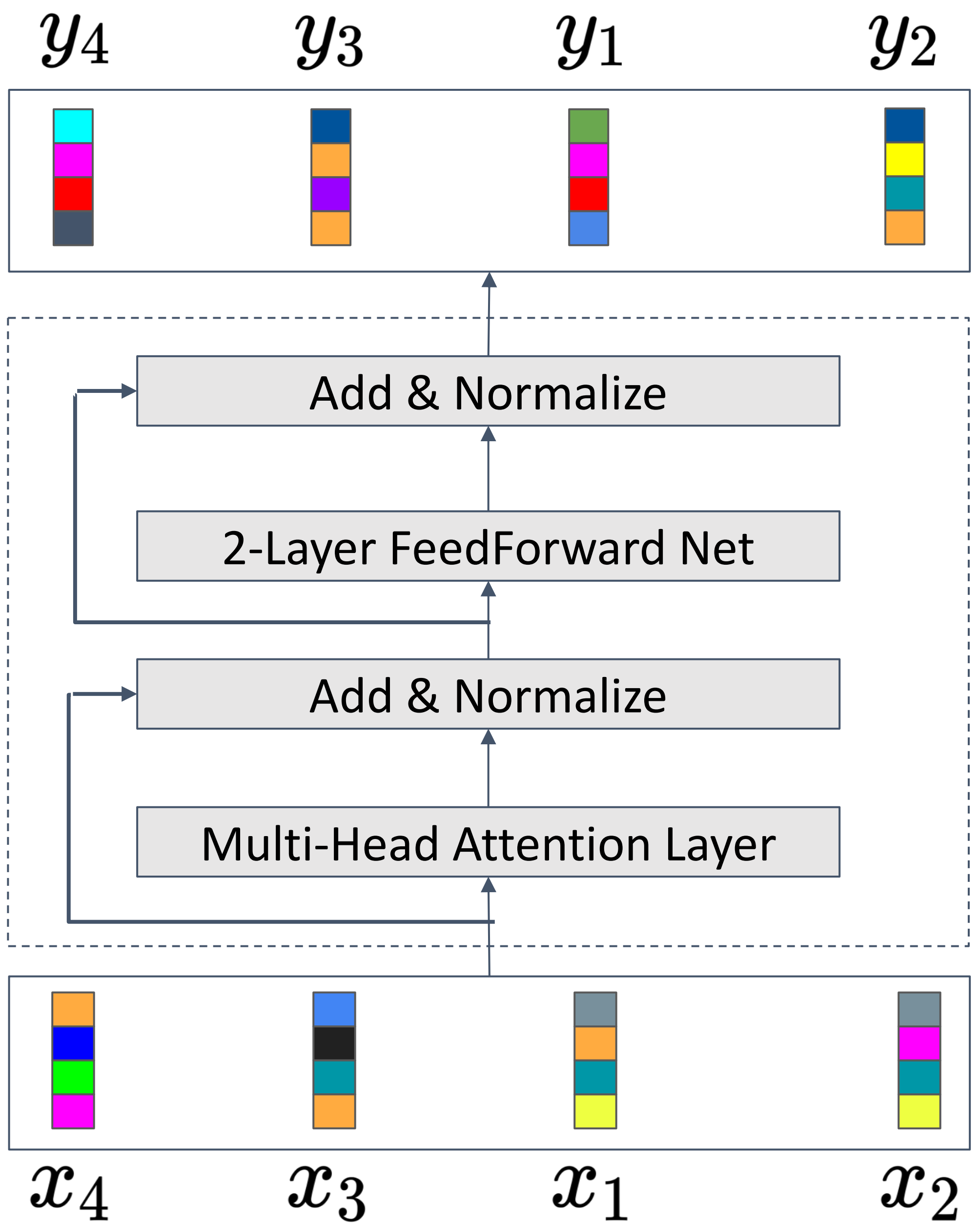
Self-attention:

$$\text{score}(x_i, x_j) = k_i^T q_j$$

$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{i'} \exp(\text{score}(x_{i'}, x_j))}$$

$$y_j = \sum_i \alpha_{ij} v_i$$

What if we switch the order of the inputs?



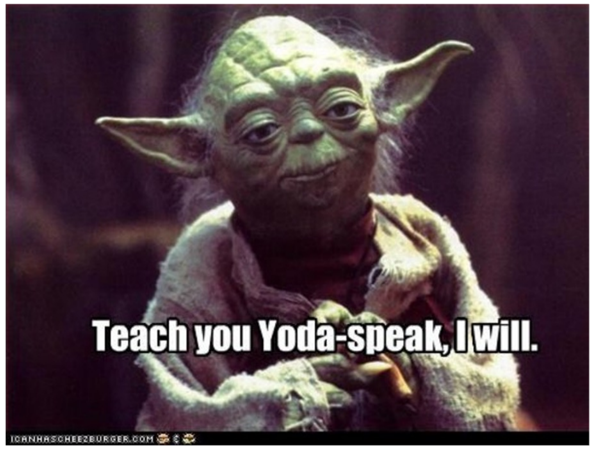
Self-attention:

$$\text{score}(x_i, x_j) = k_i^T q_j$$

$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{i'} \exp(\text{score}(x_{i'}, x_j))}$$

$$y_j = \sum_i \alpha_{ij} v_i$$

Permuting the inputs produces the same representations, with bidirectional self-attention all information about order lost

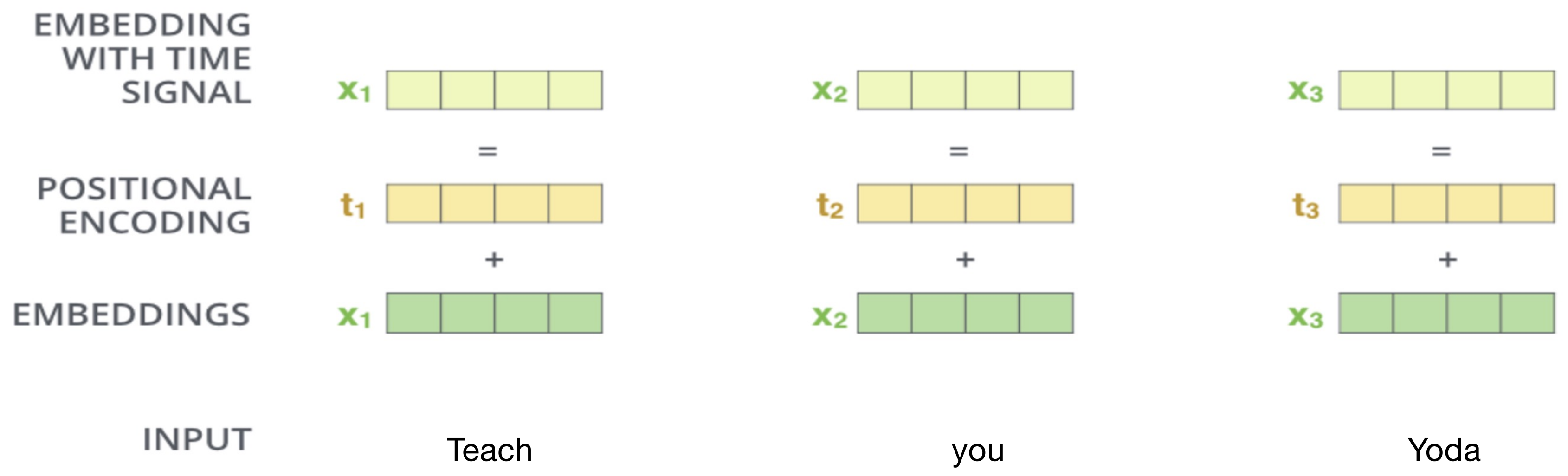


Common Transformer Tricks

- Layer Normalization
- Specialized Training Schedule (usually based on Adam optimizer)
- Label Smoothing

Positional encodings

- Add an embedding to each input which depends on its position



- We can either learn a separate embedding for each position [1, max-length]
- Or we can use a fixed function which maps integers to real vectors and preserves distances
  - The transformer paper used *sin* and *cos* of various frequencies

Figure credit: Jay Alammar's blog

Variants of Positional Embeddings

- Sinusoidal (Vaswani et al 2017)
- Absolute position embeddings (e.g. BERT)
- Relative Position Embeddings (Shaw et al 2018)
- Better relative position embeddings (Transformer XL paper, XLNet0
- Rotary Position Embeddings
- Alibi embeddings (e.g. Train Short, Test Long)
- No Position Embeddings (NoPos)

Sinusoidal Embeddings

- Sinusoidal embeddings:

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d\times 1} \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

- Learned positional embeddings:
  Randomly initialize, look up embedding based on time step t

What about tokenization?

- WordPiece (BERT paper, Fast WordPiece: https://arxiv.org/abs/2012.15524)
- Byte-Pair Encodings (https://arxiv.org/pdf/1508.07909.pdf)
- Unigram LM (https://arxiv.org/abs/1804.10959)
- Adaptive softmax (Baevski and Auli, 2018) and adaptive inputs (Joulin et al., 2017).
- vocabulary-free models like ByT4 (Xue et al., 2022) and CANINE (Clark et al., 2022)
- Better multilingual vocabularies (XLM-V): https://arxiv.org/abs/2301.10472)

Wordpiece Embeddings / Vocabs

Greedy algorithm — basic idea:

1. Initialize the word unit inventory with all characters
2. Build a (n-gram) language model on the training data, using the inventory from 1.
3. ”Generate a new word unit by combining two units out of the current word inventory to increment the word unit inventory by one. Choose the new word unit out of all the possible ones that increases the likelihood on the training data the most when added to the model.” (text src)
4. “Go to 2 until a predefined limit of word units is reached or the likelihood increase falls below a certain threshold.” (text src)

Wu et al. 2016 (Google NMT), used by BERT

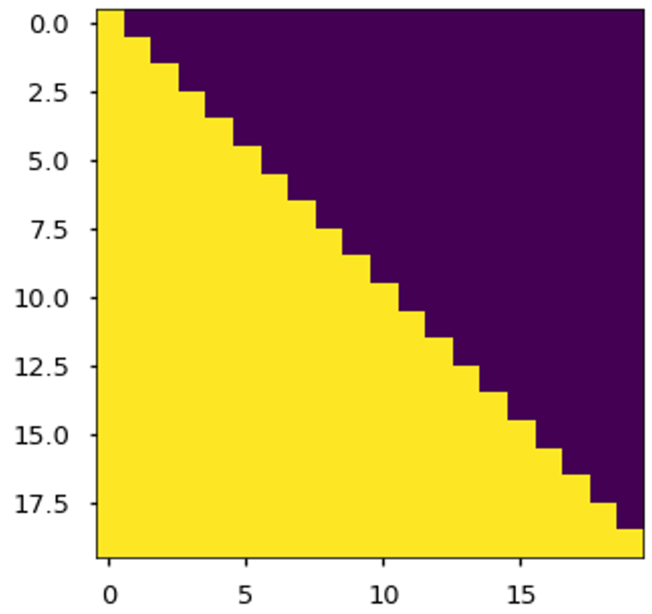
Byte Pair Encoding

1. Initialize the word unit inventory with all characters
2. Build a language model on the training data using the inventory from 1.
3. “Count all symbol pairs and replace each occurrence of the most frequent pair (‘A’, ‘B’) with a new symbol ‘AB’”
4. Iterate until a predefined limit of word units is reached.

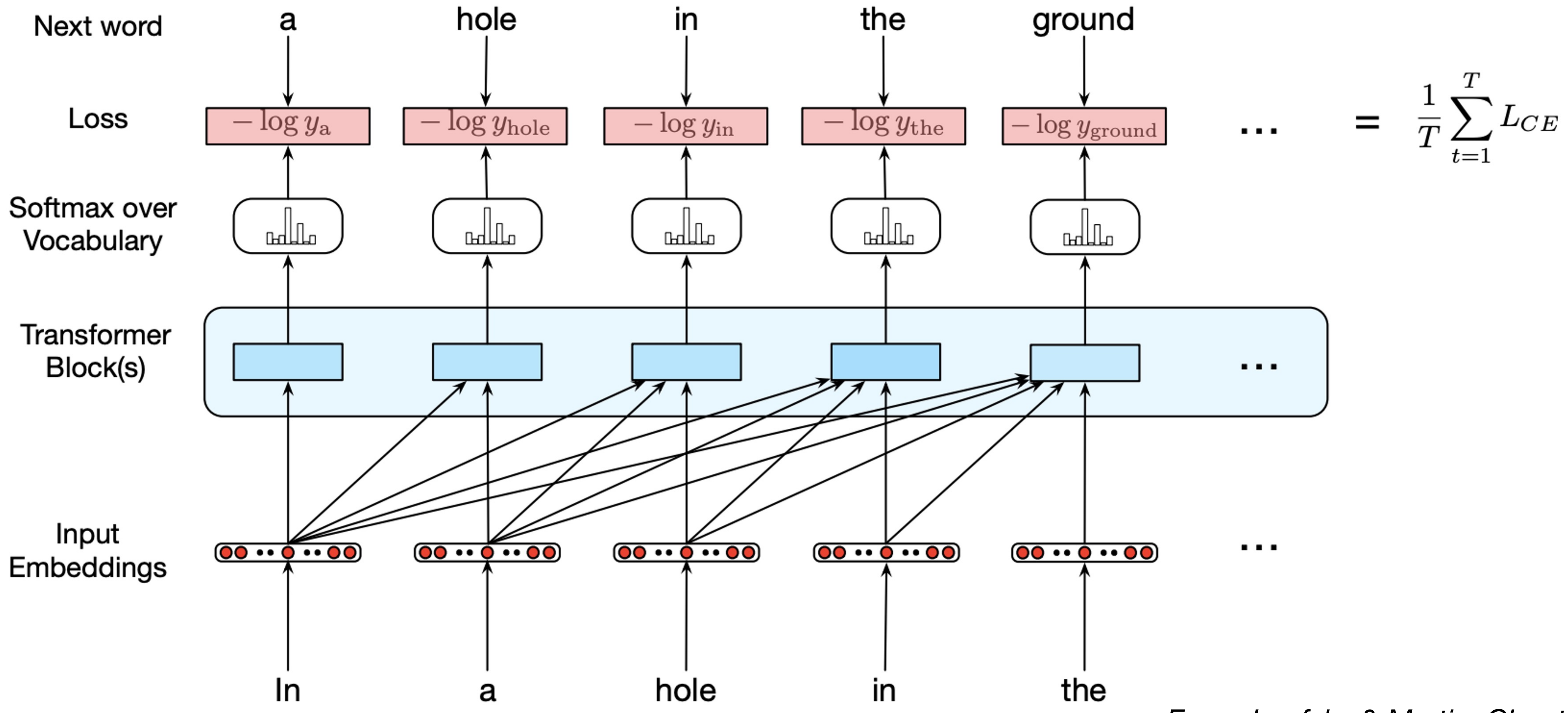
Old compression algorithm, adapted by Sennrich et al. 2016 to NLP.

Transformers for language modeling

- $P(w_t | w_1, \ldots, w_{t-1})$
- We need to prevent attention mechanism from using words after position *t* $\text{score}(x_i, x_j) = -\infty \quad \forall \quad i > j$
- **Attention masking**:



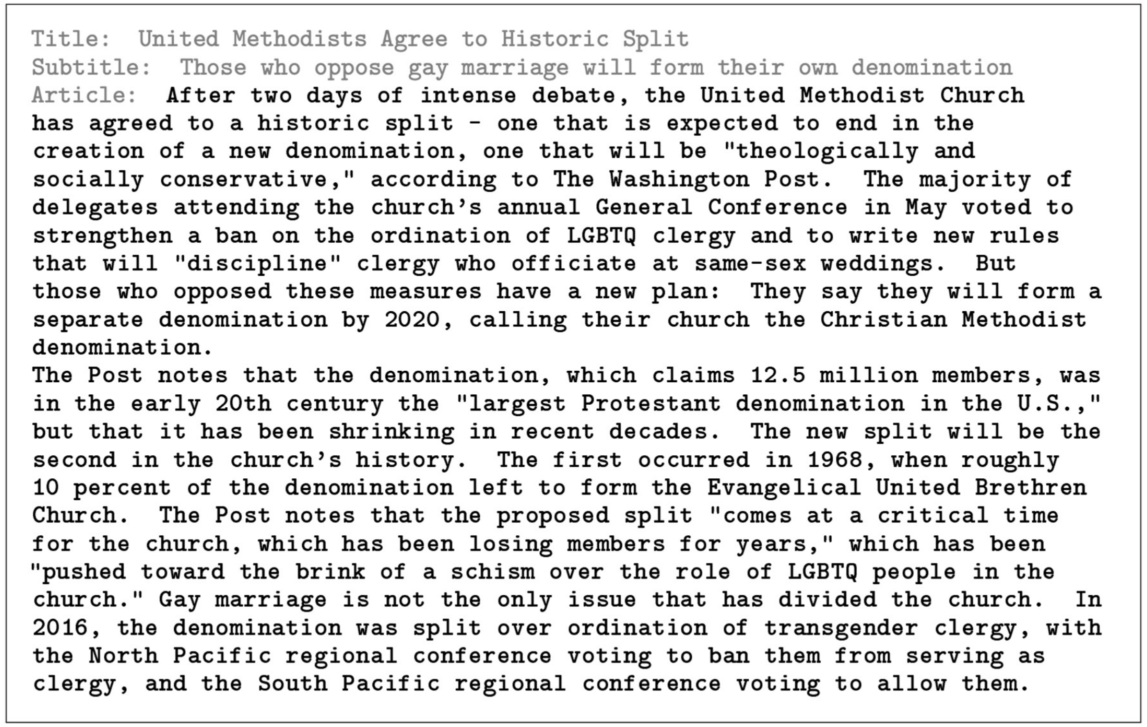
Transformers for language modeling



From Jurafsky & Martin, Chapter 9

GPT3

- 175 billion parameter transformer LM trained on roughly 500 billion tokens of text
- Perplexity on Penn Treebank: 20.5



```
Title:  United Methodists Agree to Historic Split
Subtitle:  Those who oppose gay marriage will form their own denomination
Article:  After two days of intense debate, the United Methodist Church
has agreed to a historic split - one that is expected to end in the
creation of a new denomination, one that will be "theologically and
socially conservative," according to The Washington Post.  The majority of
delegates attending the church's annual General Conference in May voted to
strengthen a ban on the ordination of LGBTQ clergy and to write new rules
that will "discipline" clergy who officiate at same-sex weddings.  But
those who opposed these measures have a new plan:  They say they will form a
separate denomination by 2020, calling their church the Christian Methodist
denomination.
The Post notes that the denomination, which claims 12.5 million members, was
in the early 20th century the "largest Protestant denomination in the U.S.,"
but that it has been shrinking in recent decades.  The new split will be the
second in the church's history.  The first occurred in 1968, when roughly
10 percent of the denomination left to form the Evangelical United Brethren
Church.  The Post notes that the proposed split "comes at a critical time
for the church, which has been losing members for years," which has been
"pushed toward the brink of a schism over the role of LGBTQ people in the
church." Gay marriage is not the only issue that has divided the church.  In
2016, the denomination was split over ordination of transgender clergy, with
the North Pacific regional conference voting to ban them from serving as
clergy, and the South Pacific regional conference voting to allow them.
```

Language Models are Few-Shot Learners, Brown et al, 2020

“Here is the image representing the phrase "The End," designed for use as the closing slide of a PowerPoint presentation.”