MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CENG 403
Introduction to Deep Learning

Week 14b

Sinan Kalkan

Byte-pair Encoding (BPE)

Example from: https://huggingface.co/learn/llm-course/en/chapter6/5

- Represent frequent byte-pairs as tokens
- E.g., given the corpus:

  Corpus: ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

  our vocabulary would be:

  ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
- “ug” and “un” can be recognized to be very frequent. So, combine them:

  Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]

  Corpus: ("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)

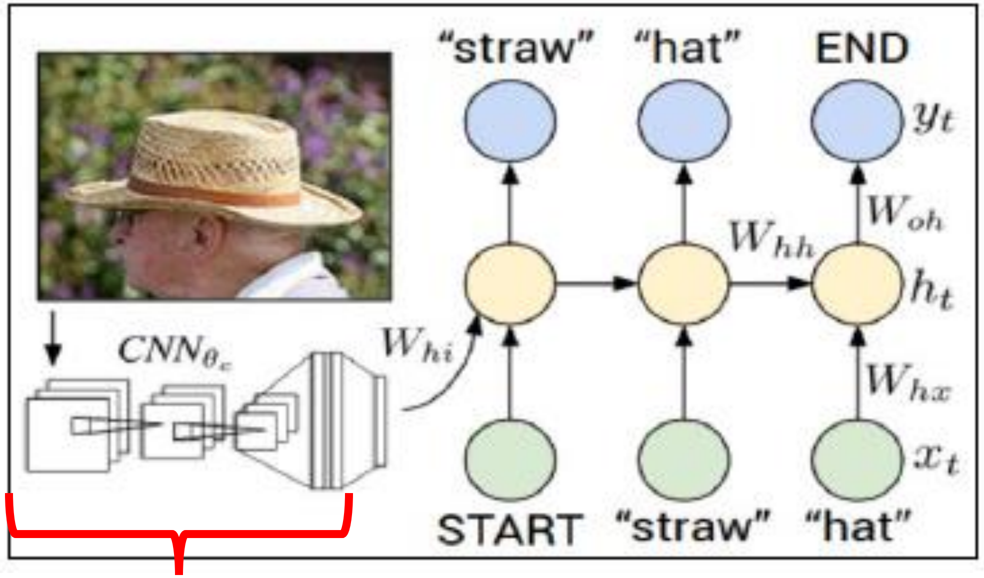
Previously on CENG403

A Comparison among Embeddings

- Out-of-vocabulary (OOV) words
  - Word embeddings struggle with out-of-vocabulary (OOV) words
  - Char embeddings are better with OOV words as they use chars. However, char embeddings fail at capturing semantically meaningful entities larger than chars
- Vocabulary size
  - Word embeddings have large vocabulary size
  - Char embeddings are better in this regard
  - BPE provides a good balance
- Sub-word (prefix, suffix, root/stem) semantics
  - BPE handles sub-words better
- Language specificity
  - Word embeddings are language specific

Previously on CENG403

Image Captioning



Pre-trained word embedding is also used

Pre-trained CNN
(e.g., on imagenet)

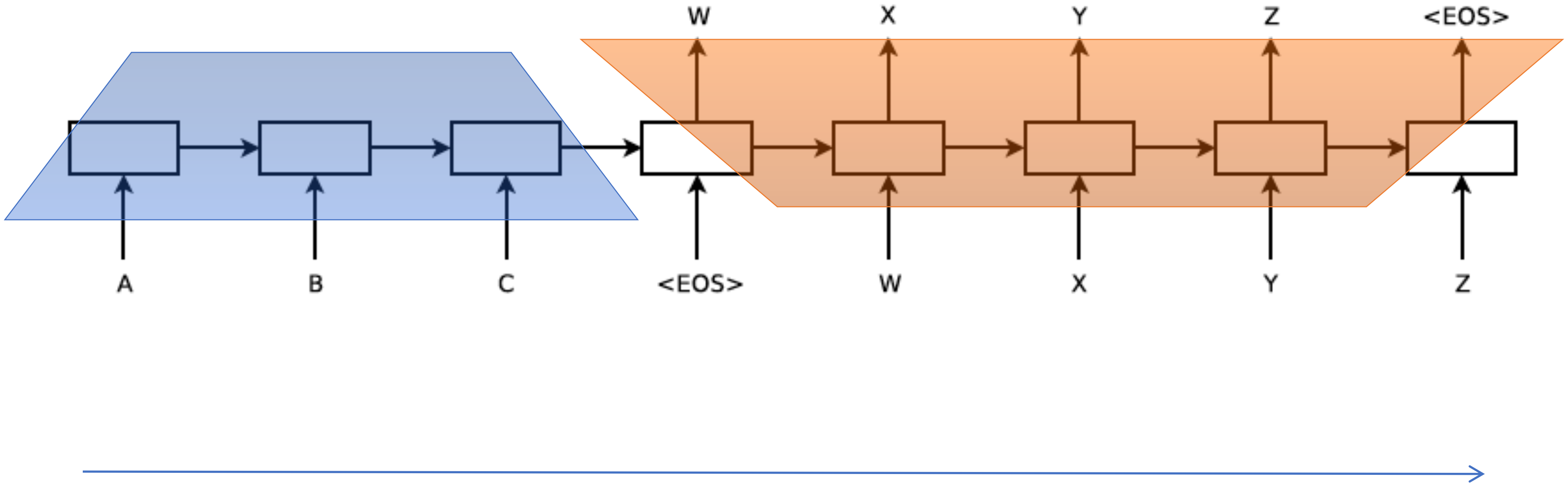
Image: Karpathy

Previously on CENG403

Neural Machine Translation

- Model

Each box is an LSTM or GRU cell.

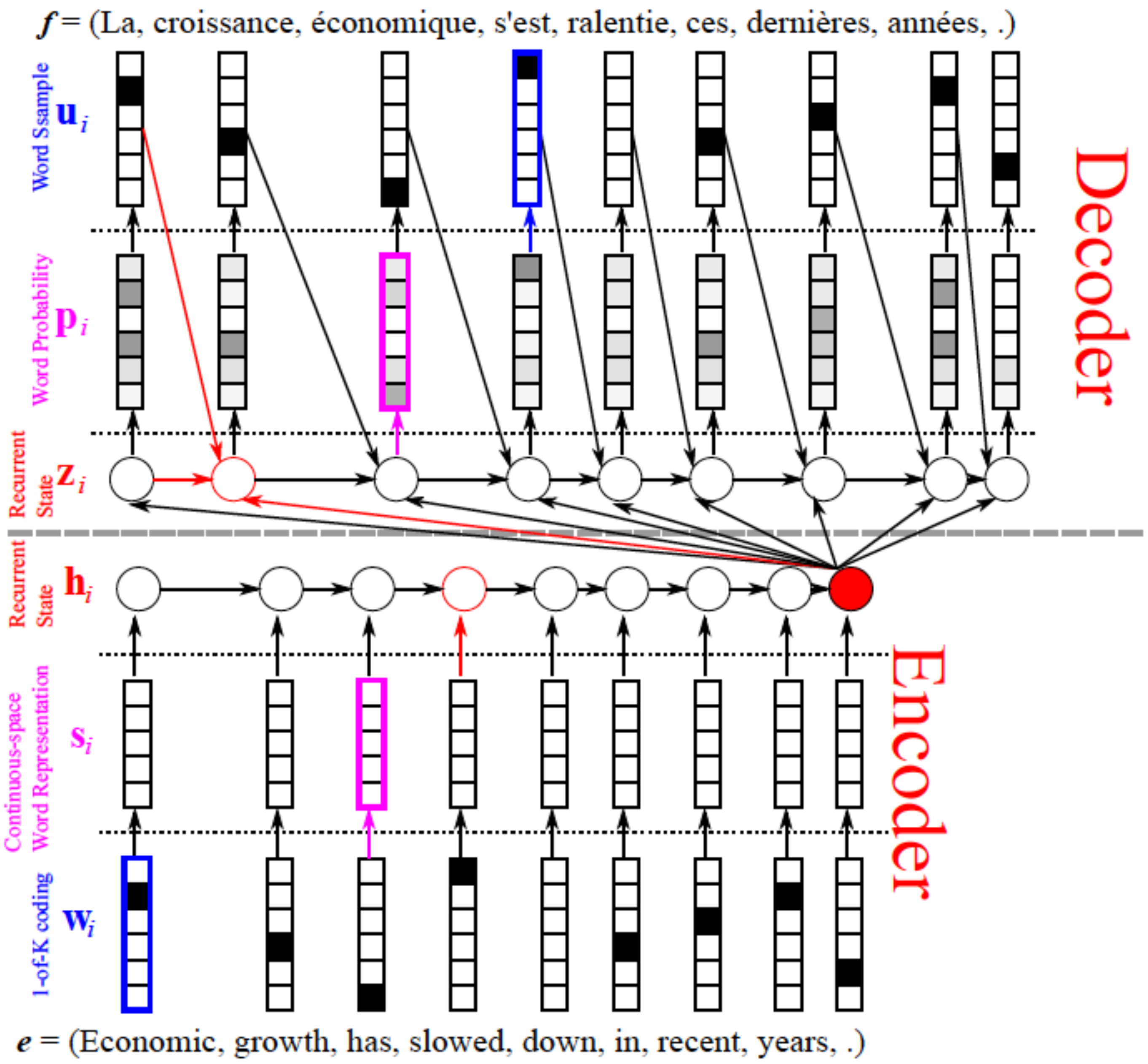


Sutskever et al. 2014

Haitham Elmarakeby

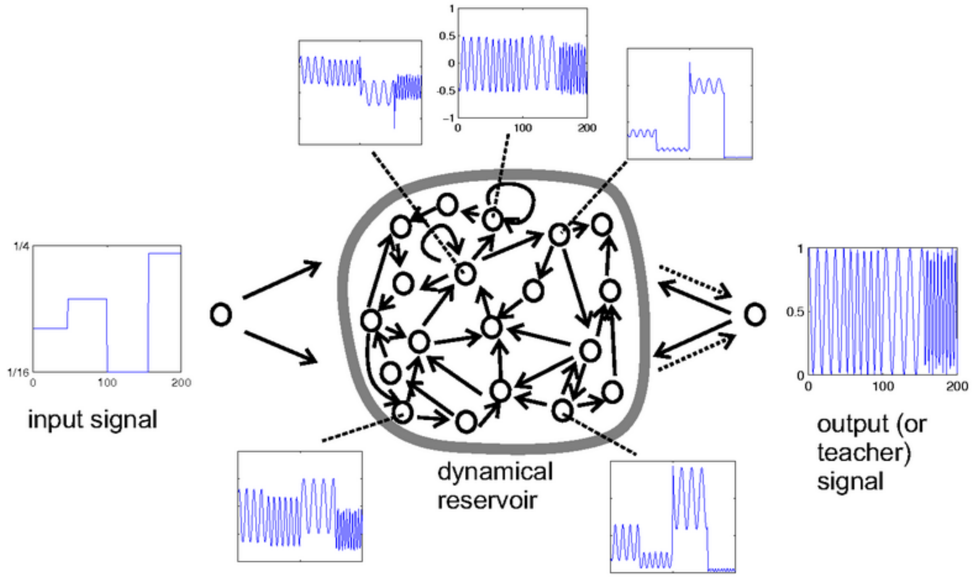
Neural Machine Translation

Previously on CENG403



Cho: From Sequence Modeling to Translation

Previously on CENG403



Echo State Networks

Reservoir Computing

Previously on CENG403

Attention

BLEU: Bilingual Evaluation Understudy

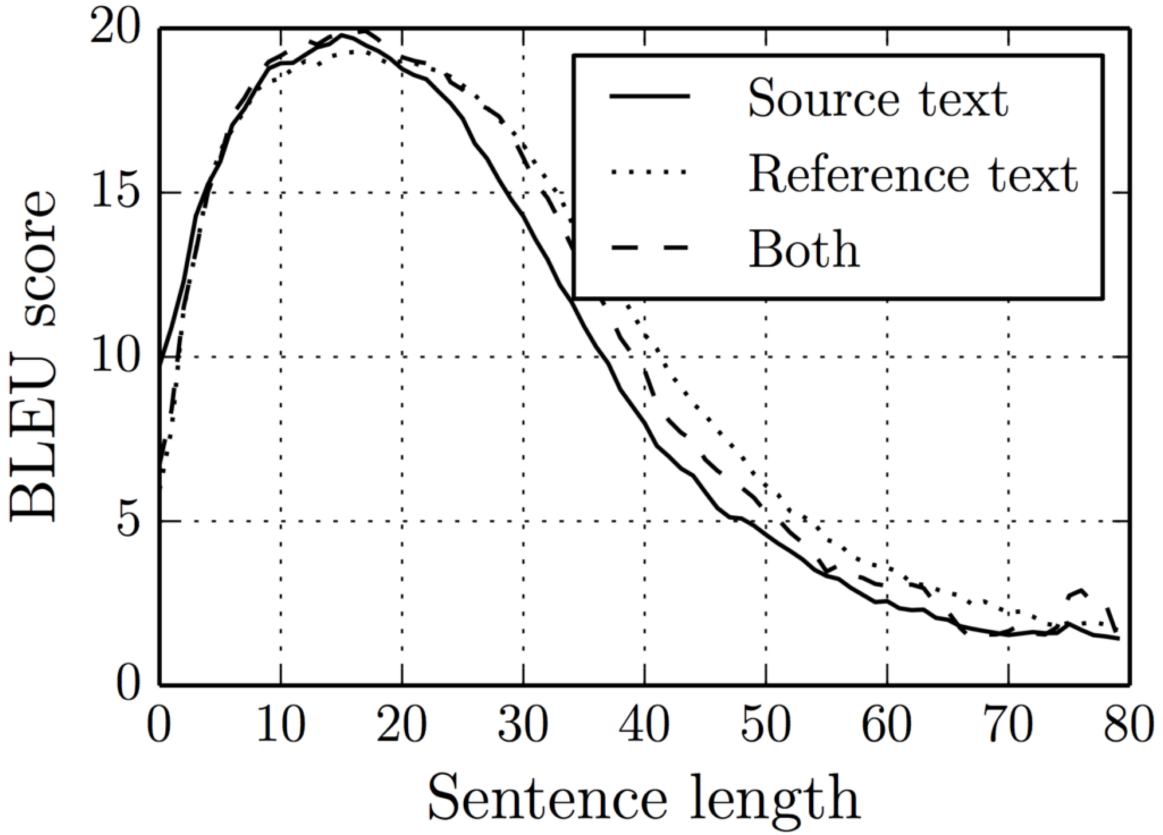
https://cloud.google.com/translate/automl/docs/evaluate#bleu

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal



Attention

Previously on CENG403

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \quad (4)$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

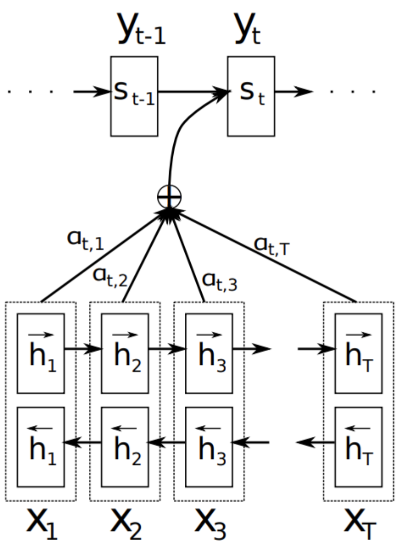


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \quad (6)$$

where

$$e_{ij} = a(s_{i-1}, h_j)$$

is an *alignment model* which scores how well the inputs around position j and the output at position i match. The score is based on the RNN hidden state s_{i-1} (just before emitting y_i, Eq. (4)) and the j-th annotation h_j of the input sentence.

We parametrize the alignment model a as a feedforward neural network which is jointly trained with all the other components of the proposed system. Note that unlike in traditional machine translation,

Previously on CENG403

Attention Types

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

(*) Referred to as "concat" in Luong, et al., 2015 and as "additive attention" in Vaswani, et al., 2017.
(^) It adds a scaling factor $1/\sqrt{n}$, motivated by the concern when the input is large, the softmax function may have an extremely small gradient, hard for efficient learning.

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Today

- Recurrent Neural Networks (RNNs)
  - Image captioning
  - Machine translation
  - Echo State Networks
  - Attention in RNNs

Vanilla Self-attention

$$e_i{}' = \sum_j \frac{\exp\left(e_j^T e_i\right)}{\sum_m \exp(e_m^T e_i)} e_j$$

Attention: Transformer

- Vanilla self attention:

$$e_i{}' = \sum_j \frac{\exp\left(e_j^T e_i\right)}{\sum_m \exp\left(e_m^T e_i\right)} e_j$$

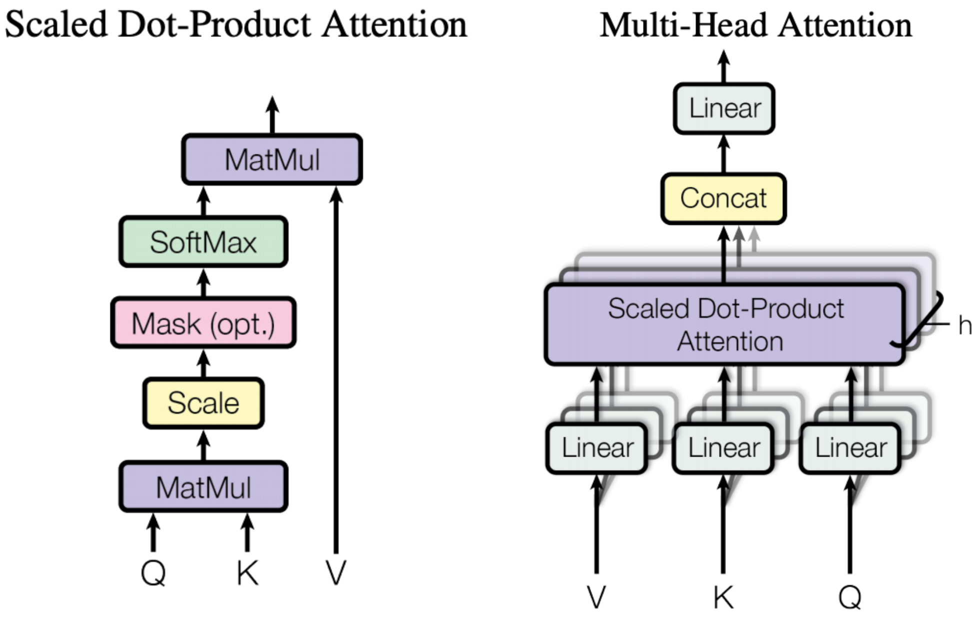
- Scaled-dot product attention:

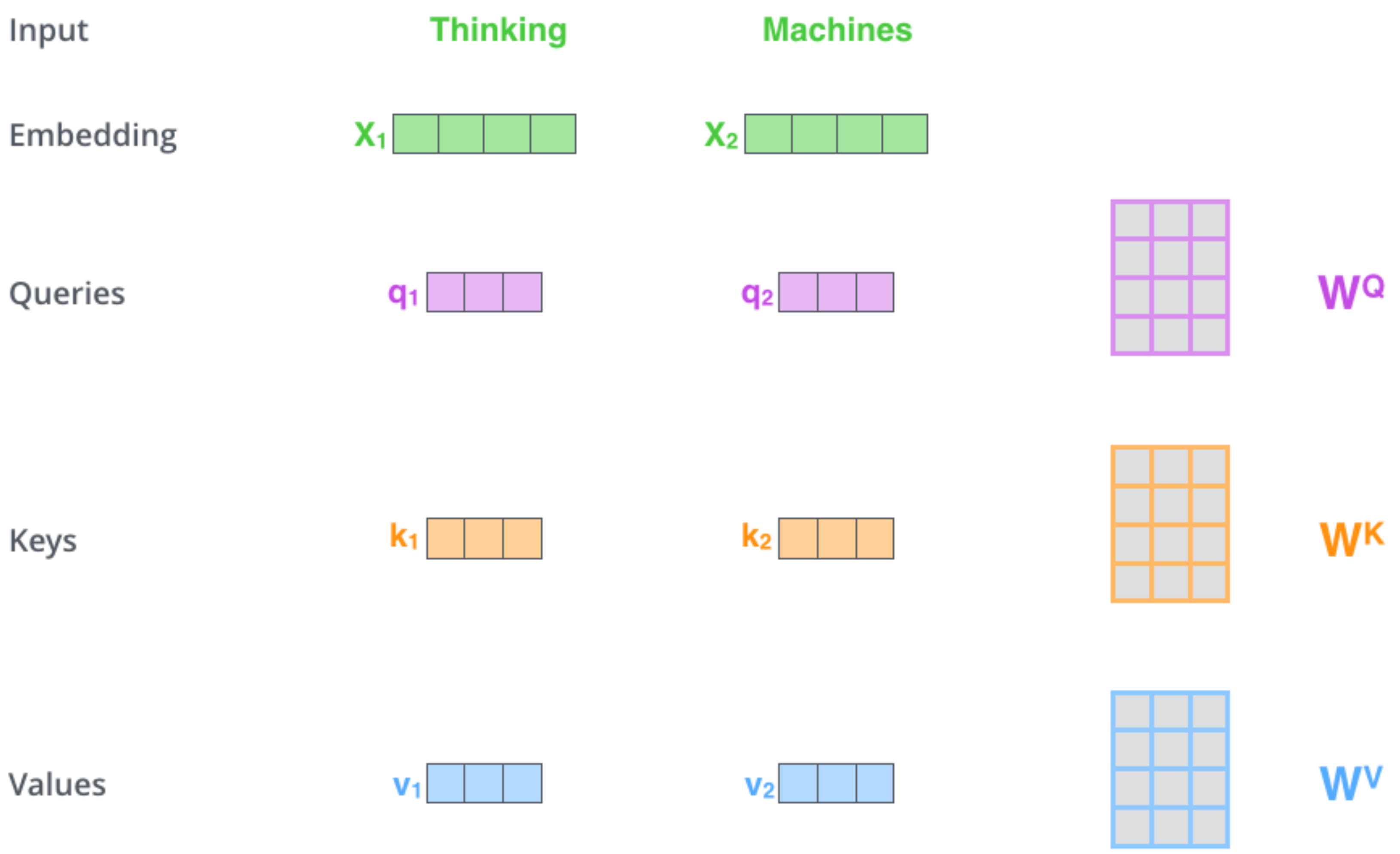
$$e_i{}' = \sum_j \frac{\exp\left(k\left(e_j^T\right) q(e_i)\right)}{\sum_m \exp\left(k\left(e_m^T\right) q(e_i)\right)} v(e_j)$$

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

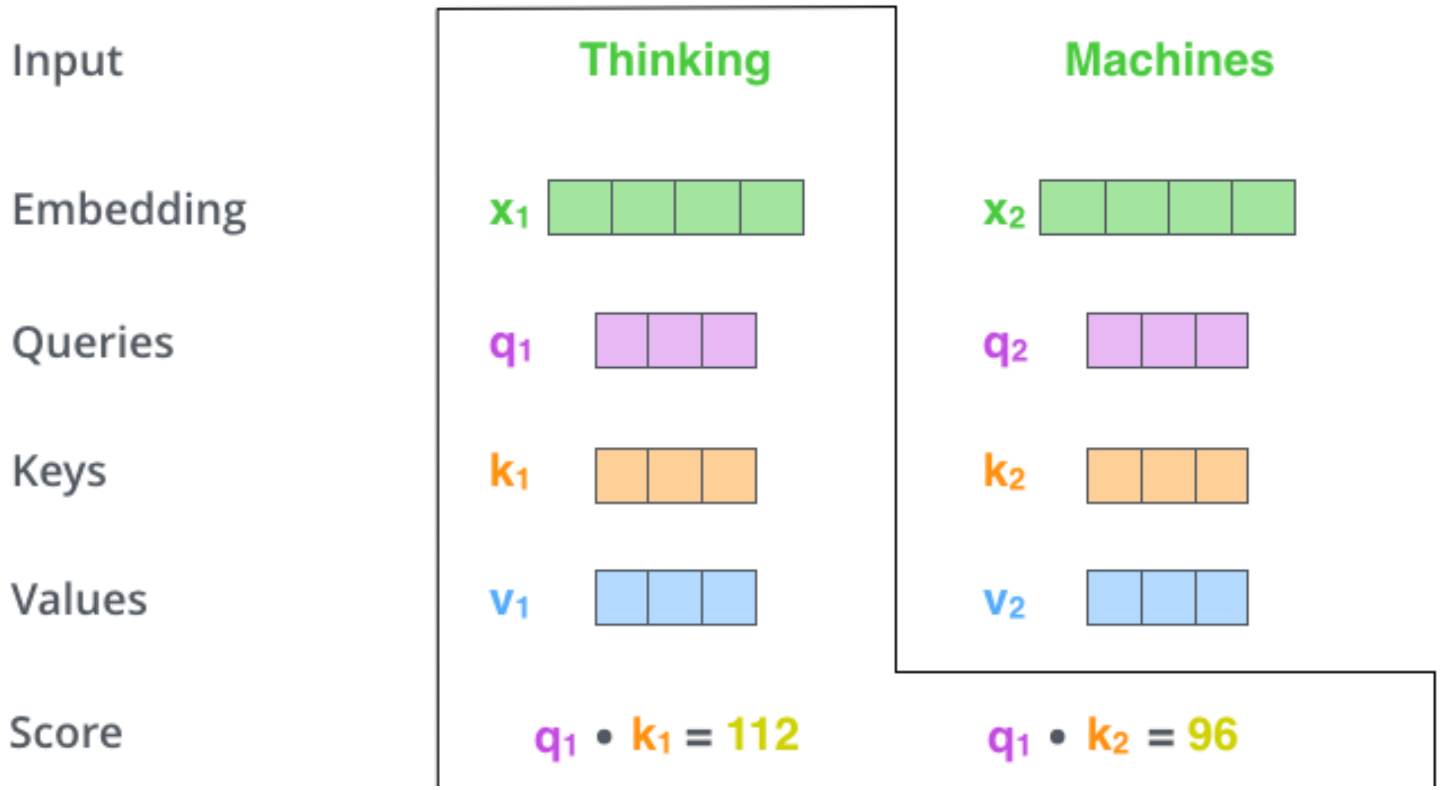
Attention Is All You Need

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

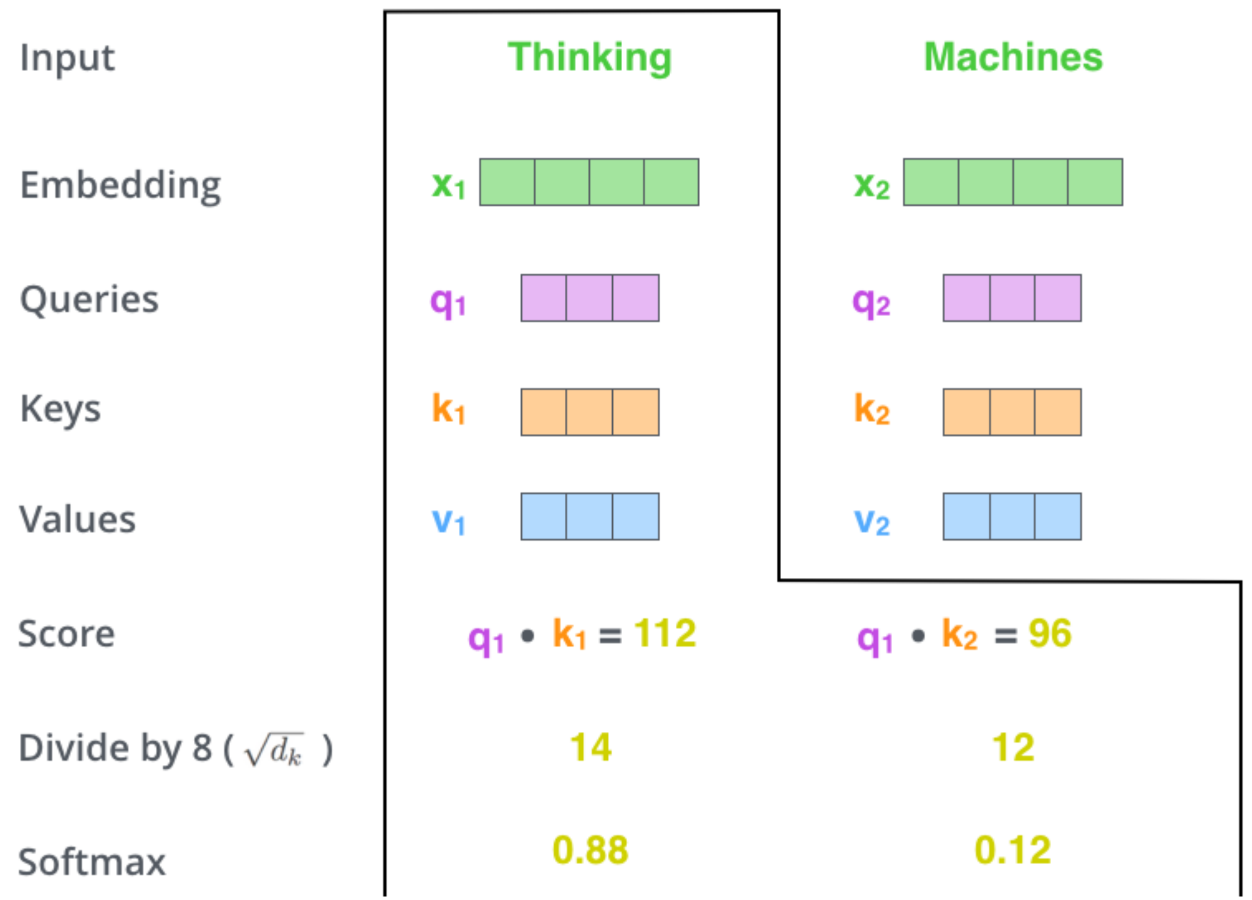




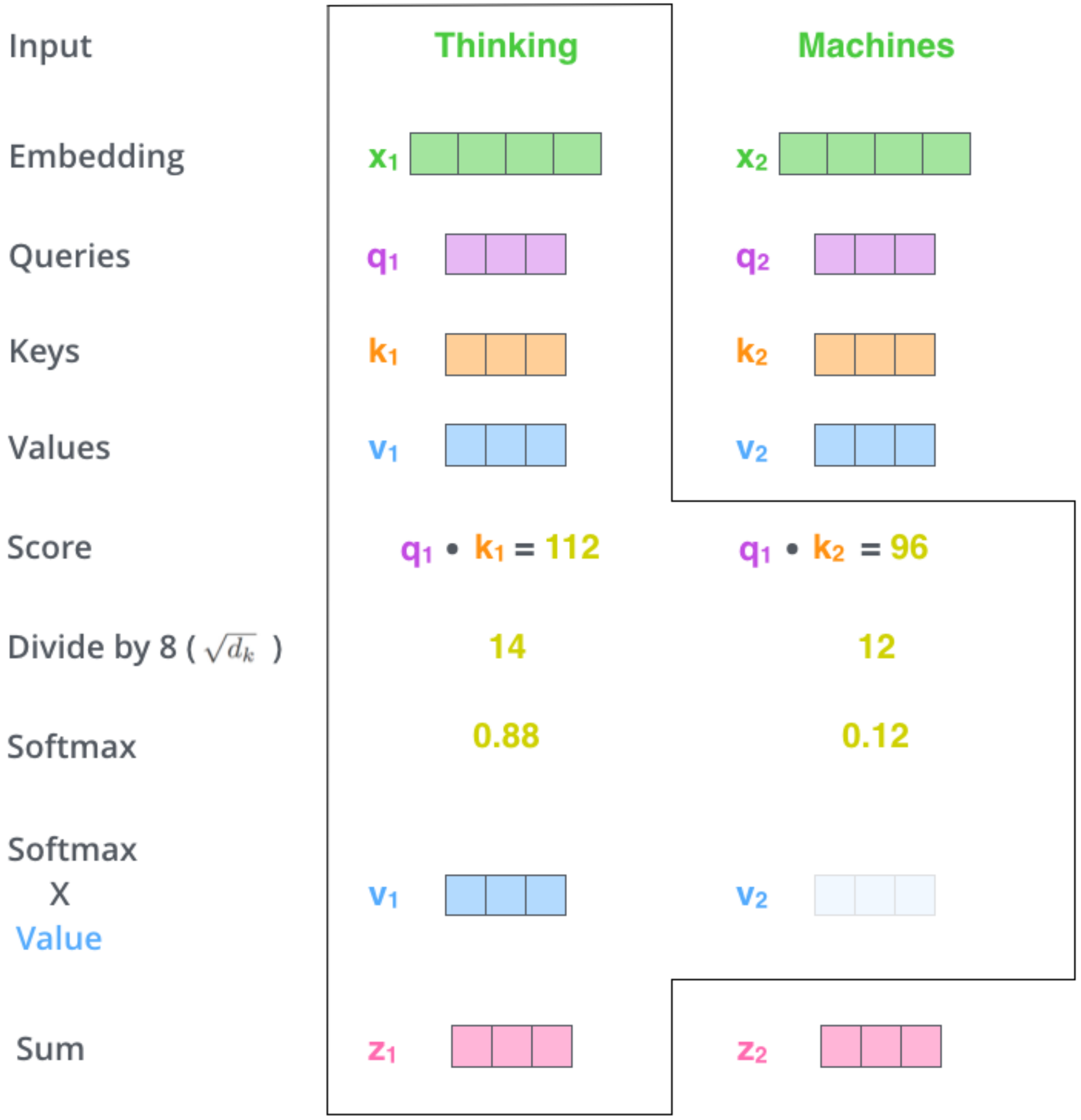
https://jalammar.github.io/illustrated-transformer/



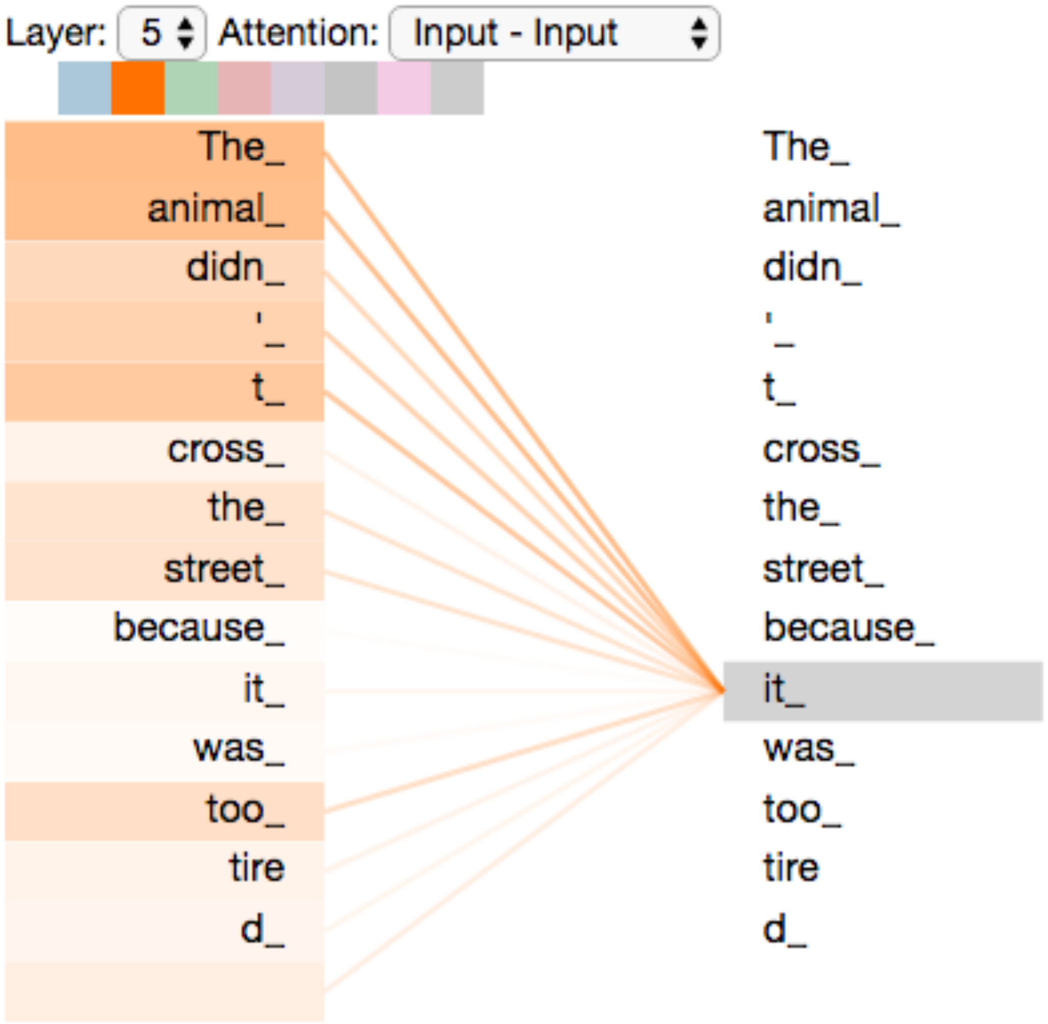
https://jalammar.github.io/illustrated-transformer/



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



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



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

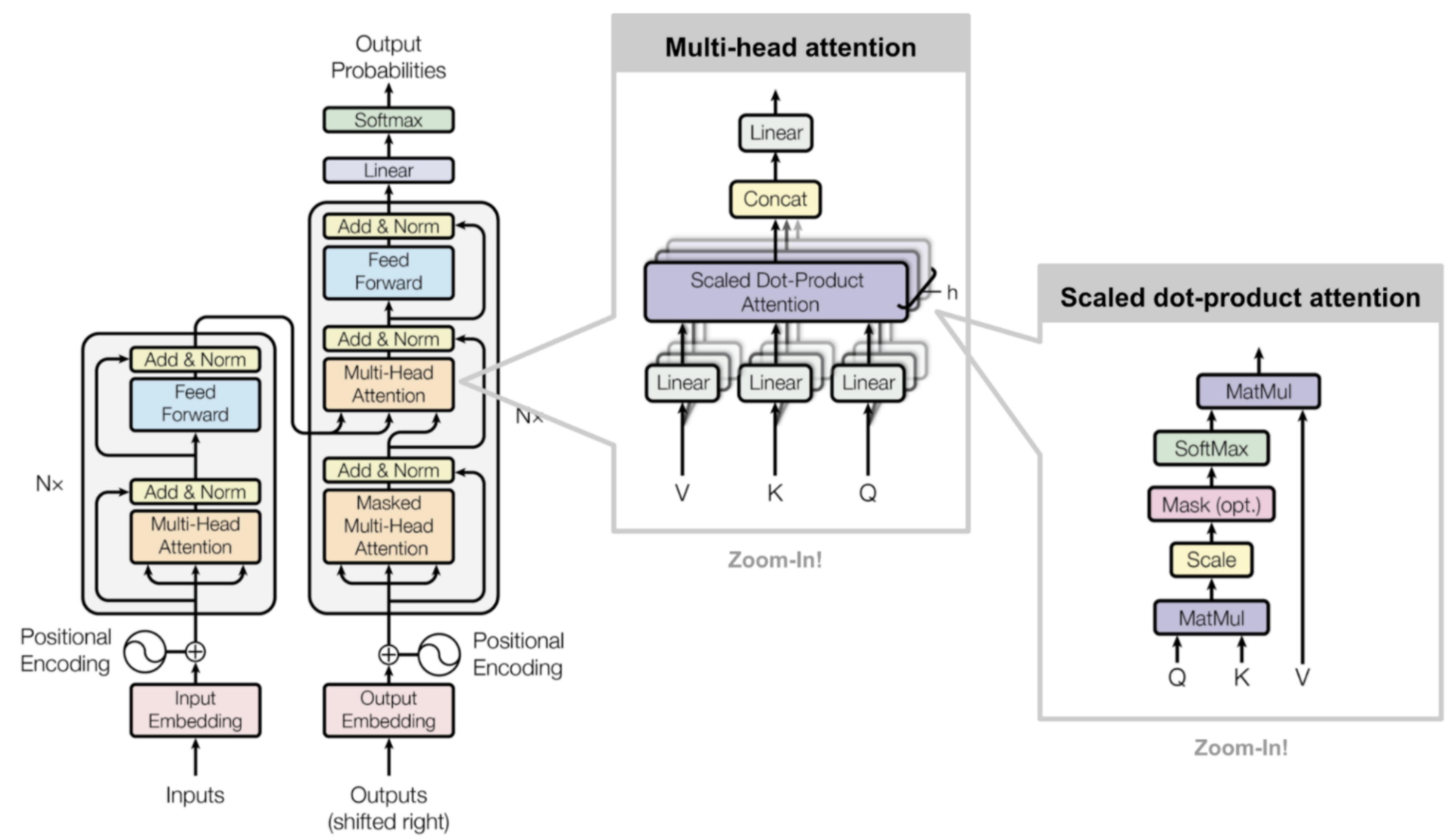


Fig. 17. The full model architecture of the transformer. (Image source: Fig 1 & 2 in Vaswani, et al., 2017.)

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Positional Encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

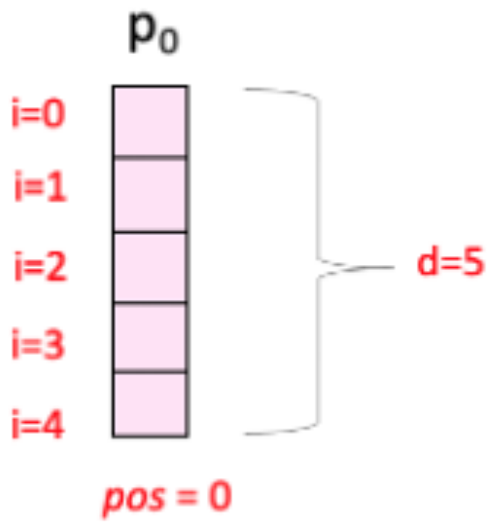


Fig from: https://www.youtube.com/watch?v=dichIcUZfOw

Positional Encoding

$$PE_{(pos,2i)} = \sin\left(\frac{\boxed{pos}}{10000^{\frac{2i}{d}}}\right)$$

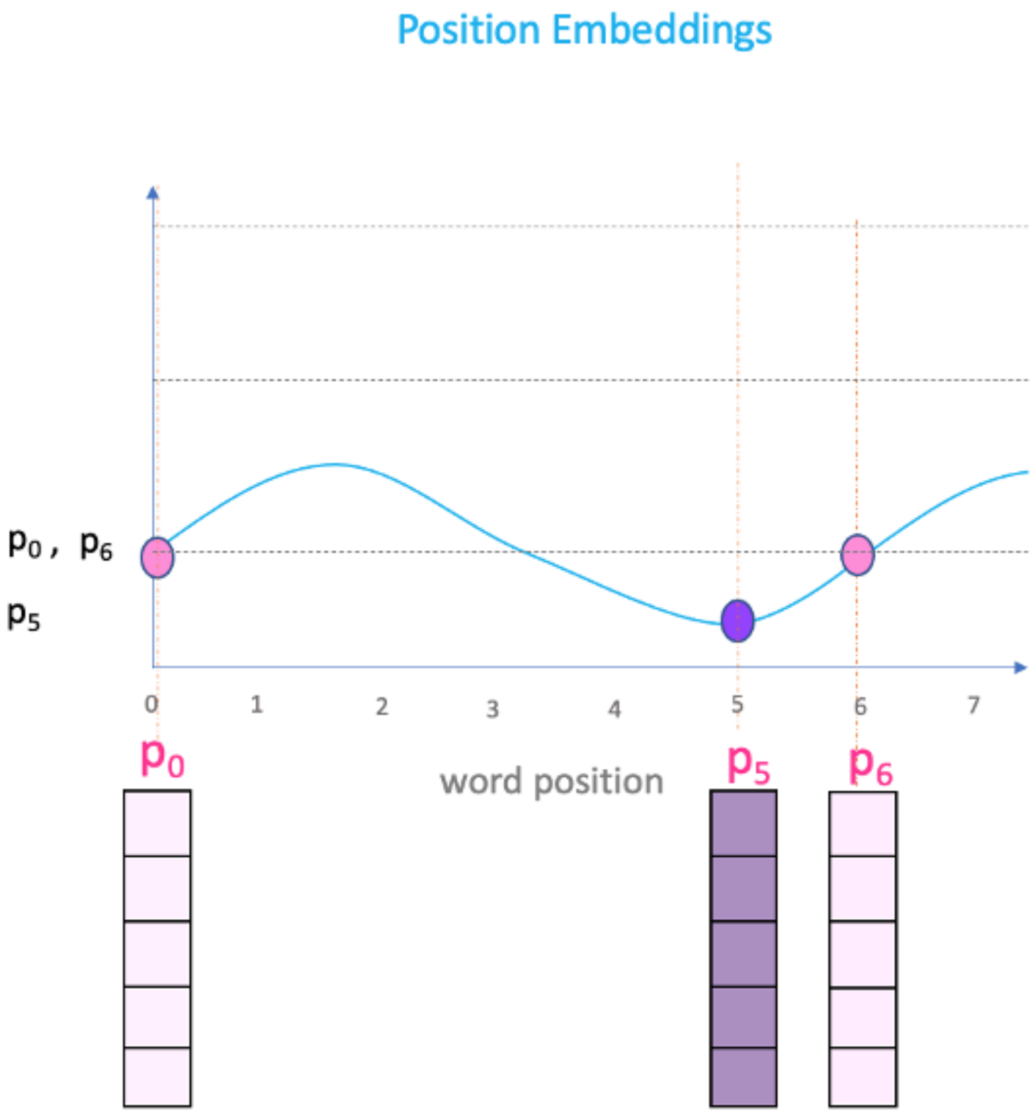


Fig from: https://www.youtube.com/watch?v=dichIcUZfOw

Positional Encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

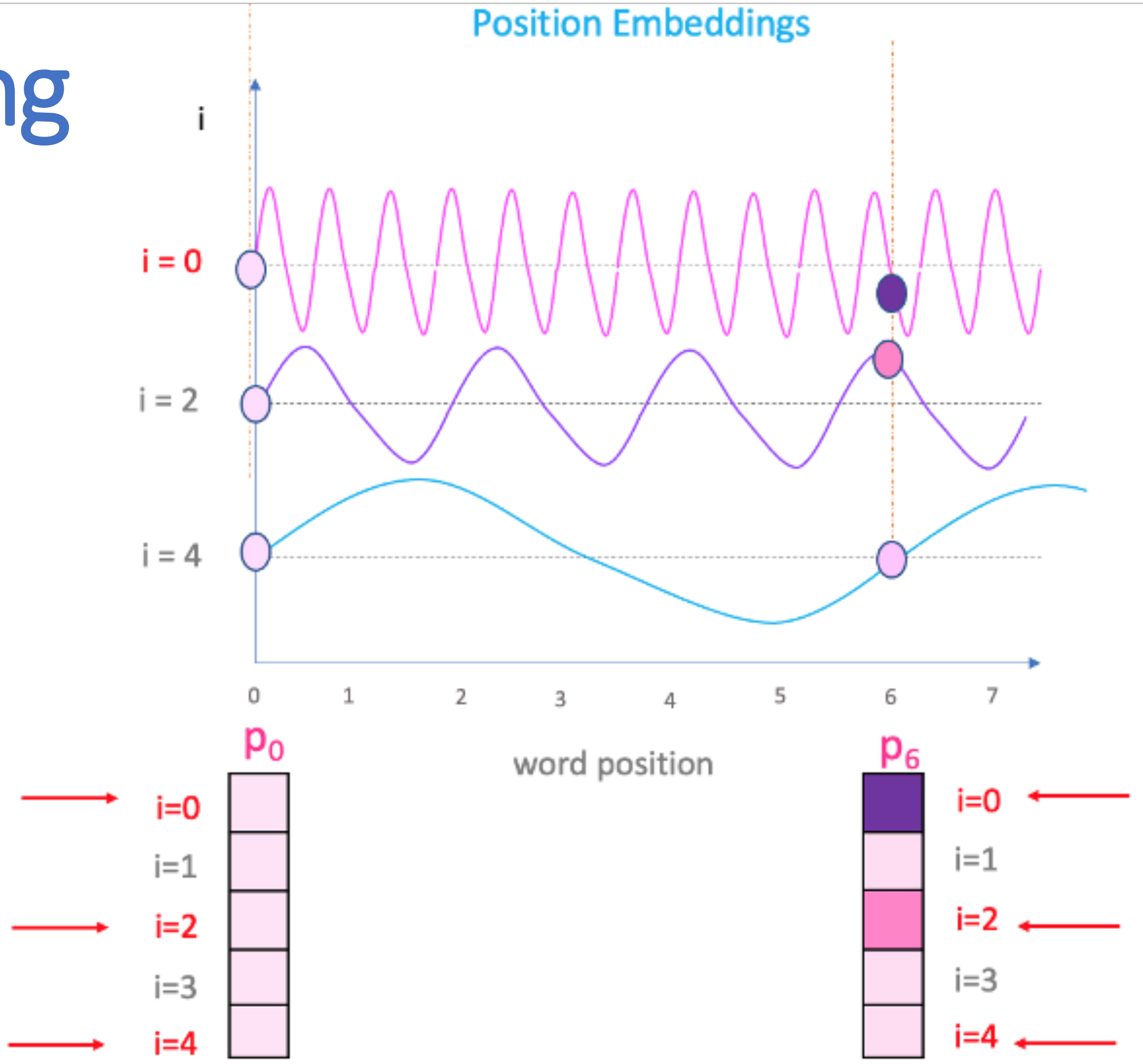
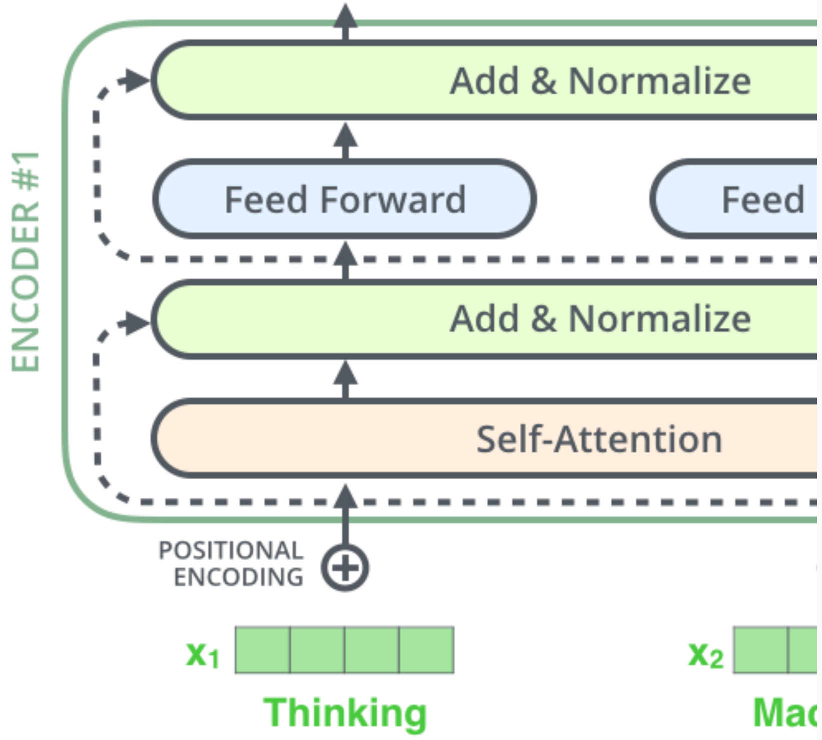
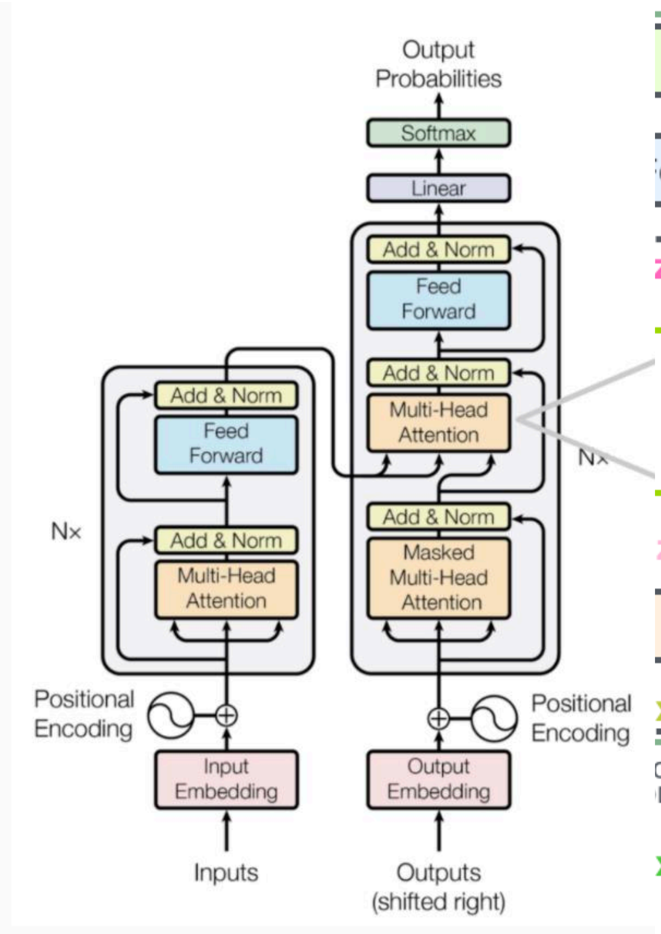
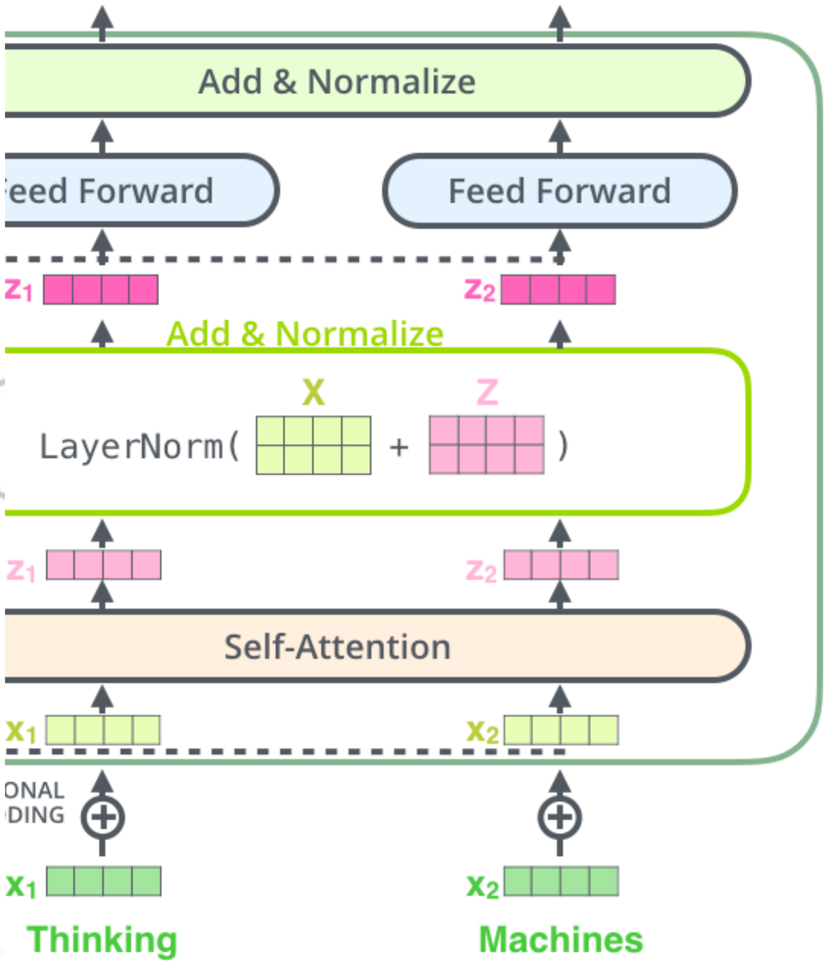


Fig from: https://www.youtube.com/watch?v=dichIcUZfOw

Skip Connections & Normalization

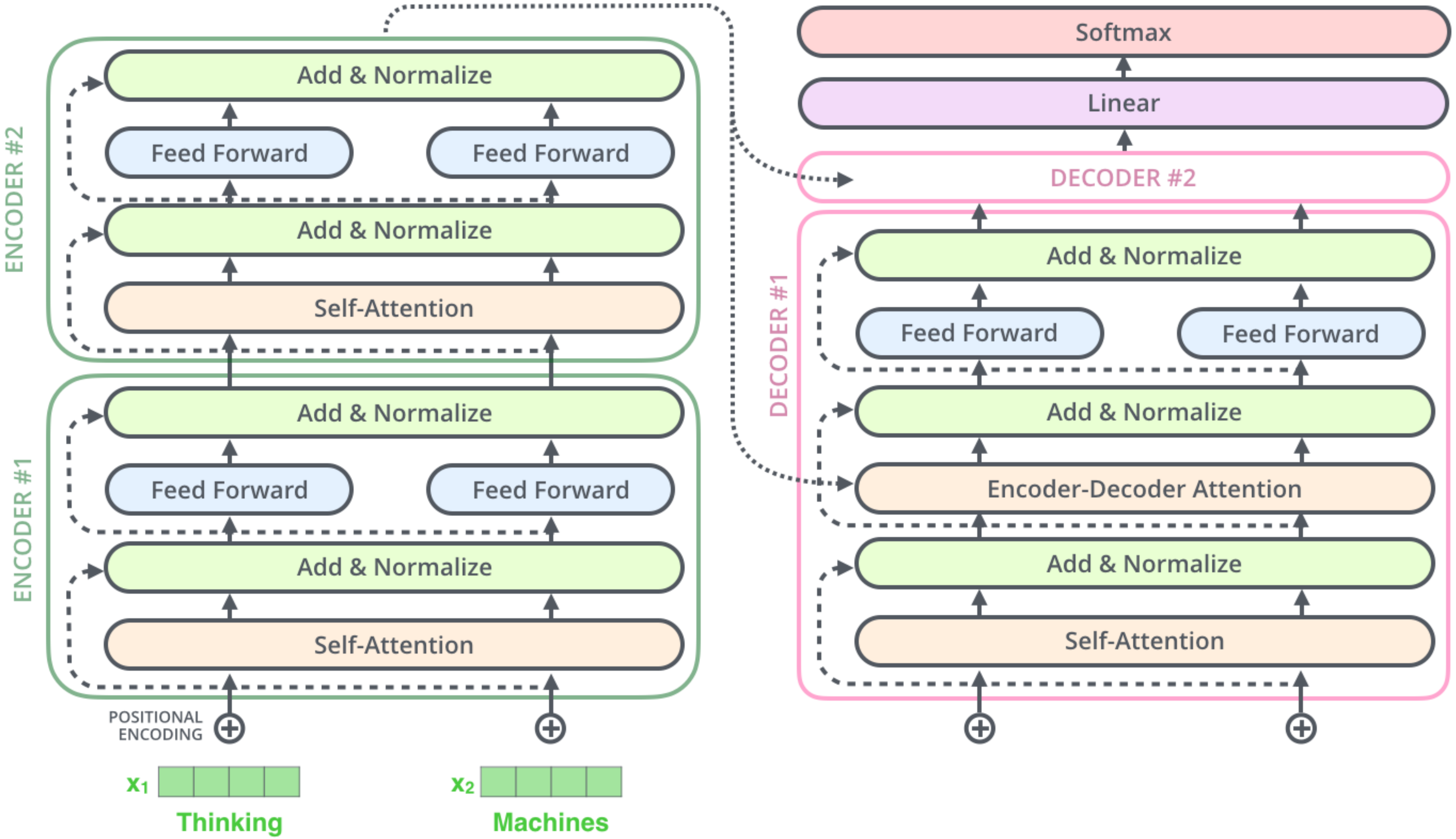






https://jalammar.github.io/illustrated-transformer/

Skip Connections & Normalization



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

Transformers without Normalization

Jiachen Zhu[1,2], **Xinlei Chen**[1], **Kaiming He**[3], **Yann LeCun**[1,2], **Zhuang Liu**[1,4,†]

[1]FAIR, Meta, [2]New York University, [3]MIT, [4]Princeton University

[†]Project lead

Normalization layers are ubiquitous in modern neural networks and have long been considered essential. This work demonstrates that Transformers without normalization can achieve the same or better performance using a remarkably simple technique. We introduce Dynamic Tanh (DyT), an element-wise operation $\mathrm{DyT}(\boldsymbol{x}) = \tanh(\alpha \boldsymbol{x})$, as a drop-in replacement for normalization layers in Transformers. DyT is inspired by the observation that layer normalization in Transformers often produces tanh-like, S-shaped input-output mappings. By incorporating DyT, Transformers without normalization can match or exceed the performance of their normalized counterparts, mostly without hyperparameter tuning. We validate the effectiveness of Transformers with DyT across diverse settings, ranging from recognition to generation, supervised to self-supervised learning, and computer vision to language models. These findings challenge the conventional understanding that normalization layers are indispensable in modern neural networks, and offer new insights into their role in deep networks.

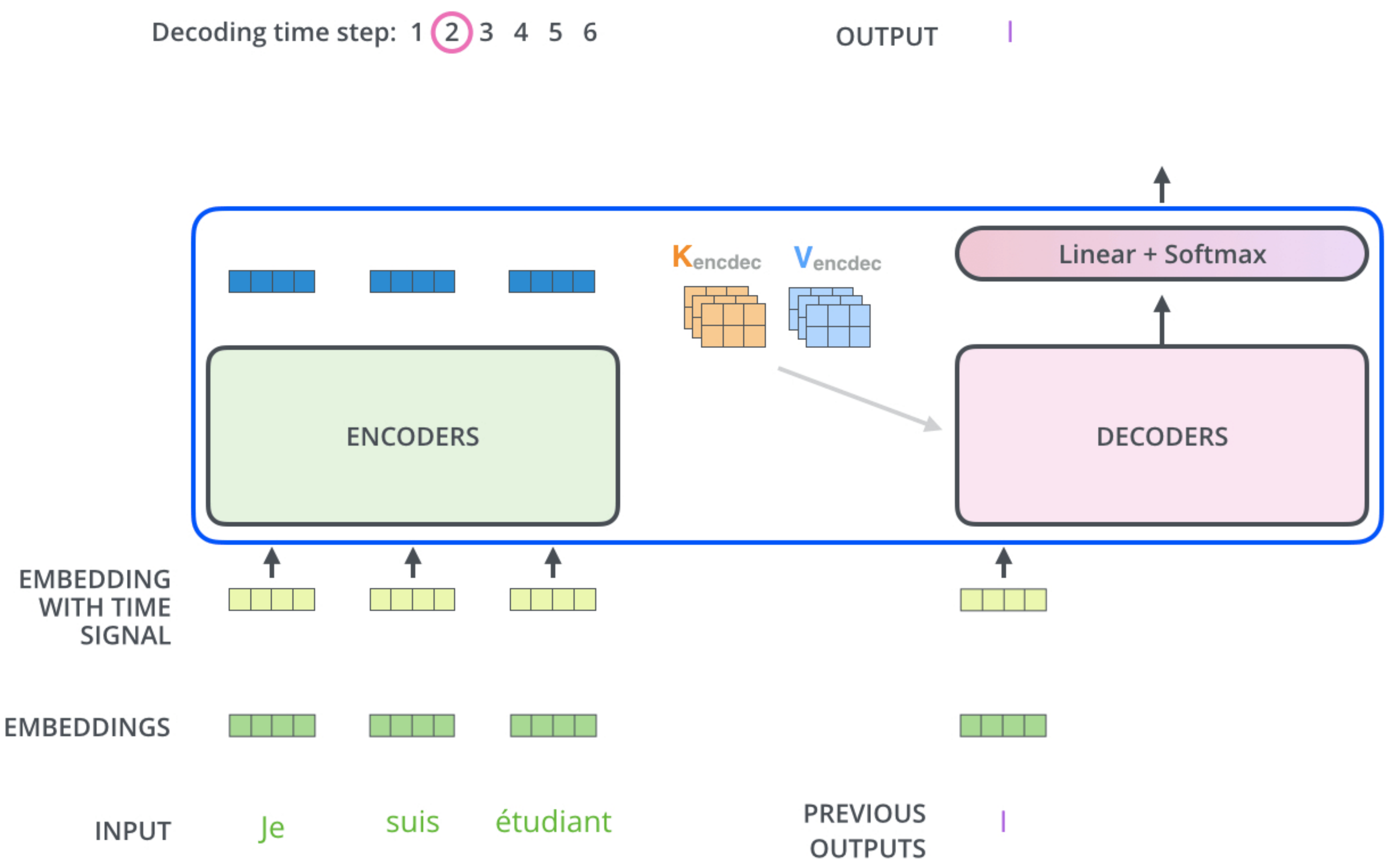
Date: March 14, 2025

Project page and code: jiachenzhu.github.io/DyT

Correspondence: jiachen.zhu@nyu.edu, zhuangl@princeton.edu

Meta

Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
K encdec
V encdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Tutorial on transformers

- https://e2eml.school/transformers.html
- https://jalammar.github.io/illustrated-transformer/

A Significant Issue with Self-Attention: Complexity

$$e_i' = \sum_j \frac{\exp\left(k(e_j^T)q(e_i)\right)}{\sum_m \exp\left(k(e_m^T)q(e_i)\right)} v(e_j)$$

- If there are n tokens/embeddings,
  - Updating a single tokens require $O(n)$ operations.
  - Overall: $O(n^2)$
- What is the complexity of an RNN layer with n time steps?