{codemotion}

Online Tech Conference
- Italian edition -

La conferenza tecnica multitrack fatta da sviluppatori per sviluppatori

24-25-26 novembre, 2020

Agenda

Gli orari dell'agenda si riferiscono al fuso orario dell'Europa Centrale (CET).

Conferenza | Workshop

24 Novembre – BACKEND & CLOUD

25 Novembre – SVILUPPO FRONTEND

26 Novembre – AI/MACHINE LEARNING, TEAM & TECH CAREER

	13:45	13:50	13:55	14:00	14:05	14:10	14:15	14:20	14:25	14:30	14:35	14:40	14:45	14:50	14:55	15:00	15:05

Track 1

Track 2

Track 3

Opening

14:10 - Keynote | Inspirational
The Cloud Should Be Fun (and If It's Not You're Probably Doing It Wrong)
Holly Cummins

Q & A

14:50 - Talk | Softwar
Node.js - Consigli su
scalabilitá
Luciano Mammino

14:50 - Talk | Cloud
From YAML to TypeS
Developer's View on
Mikhail Shilkov

14:50 - Talk | Softwar
Big Data codeless p
Vs custom code writ
Fabrizio Marini

Partecipa alle Q&A su

DISCORD

Codice di condotta

Codemotion si impegna a svolgere una conferenza

che rispecchi la diversità della community

e fornisce un'esperienza sicura

per tutti.

OTTIENI MAGGIORI INFORMAZIONI SU DISCORD

Alberto Massidda

Awake

Attention mechanisms in Neural Networks

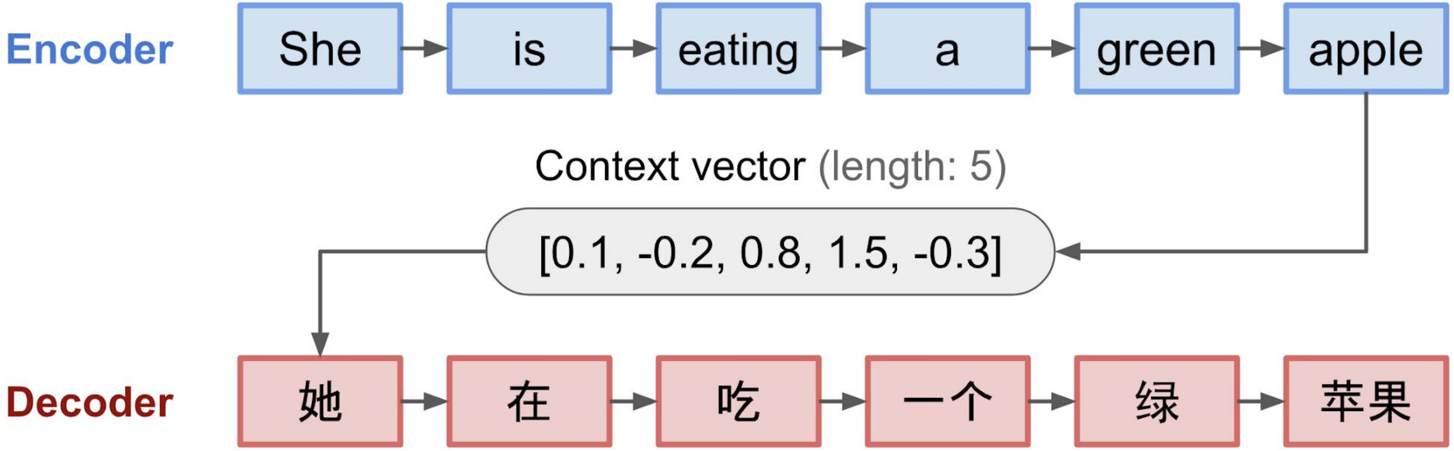
Outline

- The problem with seq2seq: information bottlenecks
- The attention as gradient flow gating
- Different types of attention
- Self attention
- Transformers
- Attention in CV: Show, attend and tell
- Attention in NLP: Neural MT, BERT, GPT-1/2/3
- Attention in GAN: SAGAN

Kudos to Lilian Weng and Jay Alammar

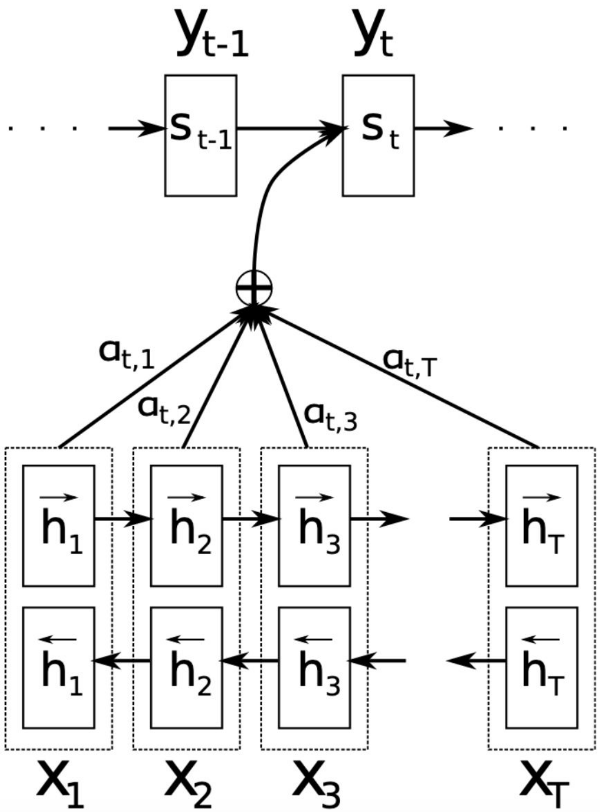
The problem with seq2seq: information bottlenecks

[Sutskever, et al. 2014](#)



The attention

Bahdanau et al., 2015



Decoder computes on current state and the attentive state, which is different for each step.

Attentive state is a weighted sum of all h states by how well current decoder state fits each h state.

Encoder's bidirectional RNN h states

(start here)

The attention

The context matrix is a weighted sum of all *h* states.

$$c_t = \sum_{i=1}^{N} \alpha_{t,i} h_i$$

The weights are computed on a softmax over a scoring function over current input and output.

$$\alpha_{t,i} = align(y_i, x_i)$$

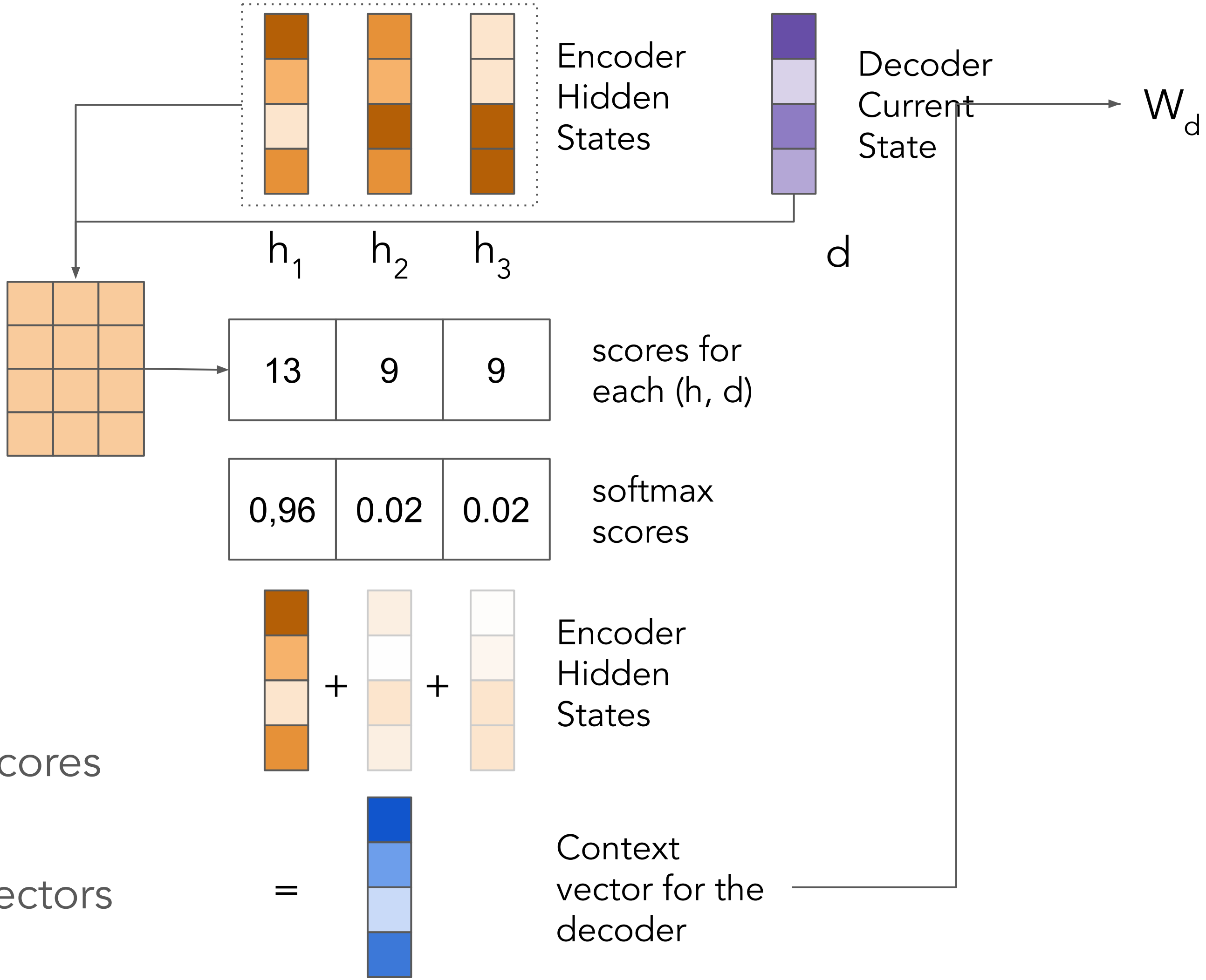
$$= \frac{exp(score(s_{t-1}, h_i))}{\sum_{i=1}^{N} \exp(score(s_{t-1}, h_i))}$$

The original score function proposed by Bahdanau is a Dense layer that takes concatenated input and output.
W_a and v_a are learned matrices.

$$score(s_t, h_i) = v_a^T tanh(W_a[s_t; h_i])$$

The attention

1. Prepare inputs
2. Score hidden states
3. Softmax scores
4. Weights inputs by scores
5. Sum up weighted vectors

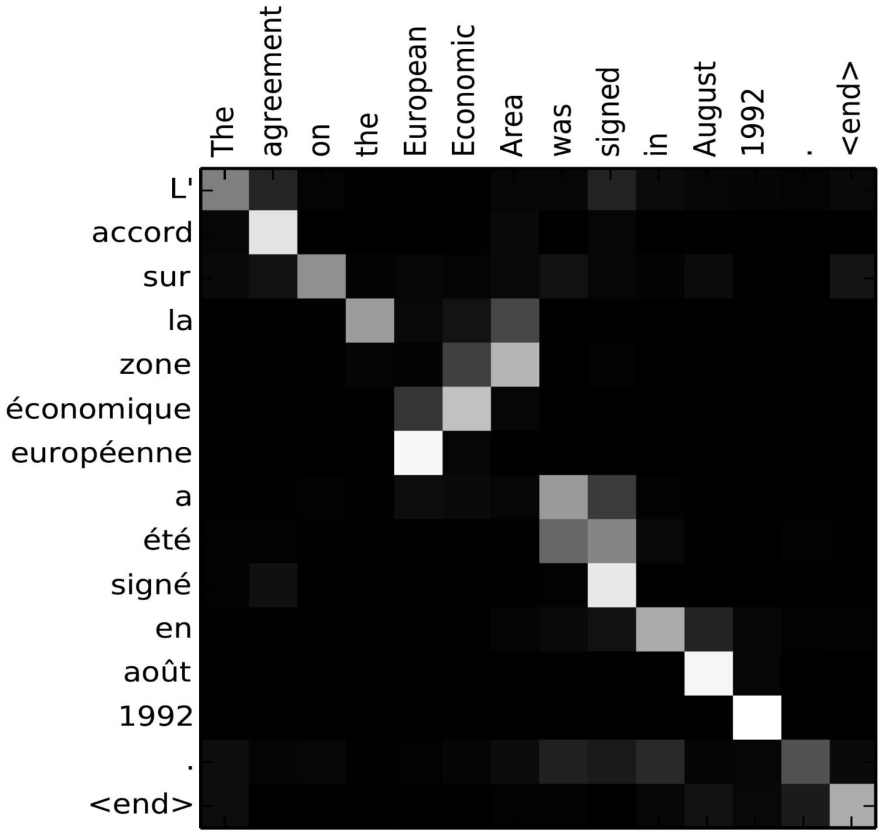


The attention as gradient flow gating

So, if every input to current output is appropriately "faded", the gradient will flow back to relevant states only.

This will create a link between current output and less faded input, saved in the attention matrix.

Some people refer to this as "soft search".



Different types of attention

- Content-base attention (Graves, 2014) $score(s_t, h_i) = \cos[s_t, h_i]$
- Additive (Bahdanau, 2015) $score(s_t, h_i) = v_a^\top tanh(W_a[s_t; h_i])$
- Location-Base (Luong, 2015) $\alpha_{t,i} = softmax(W_a, s_t)$
- General (Luong, 2015) $score(s_t, h_i) = s_t^\top W_a h_i$
- Dot-Product (Luong, 2015) $score(s_t, h_i) = s_t^\top h_i$

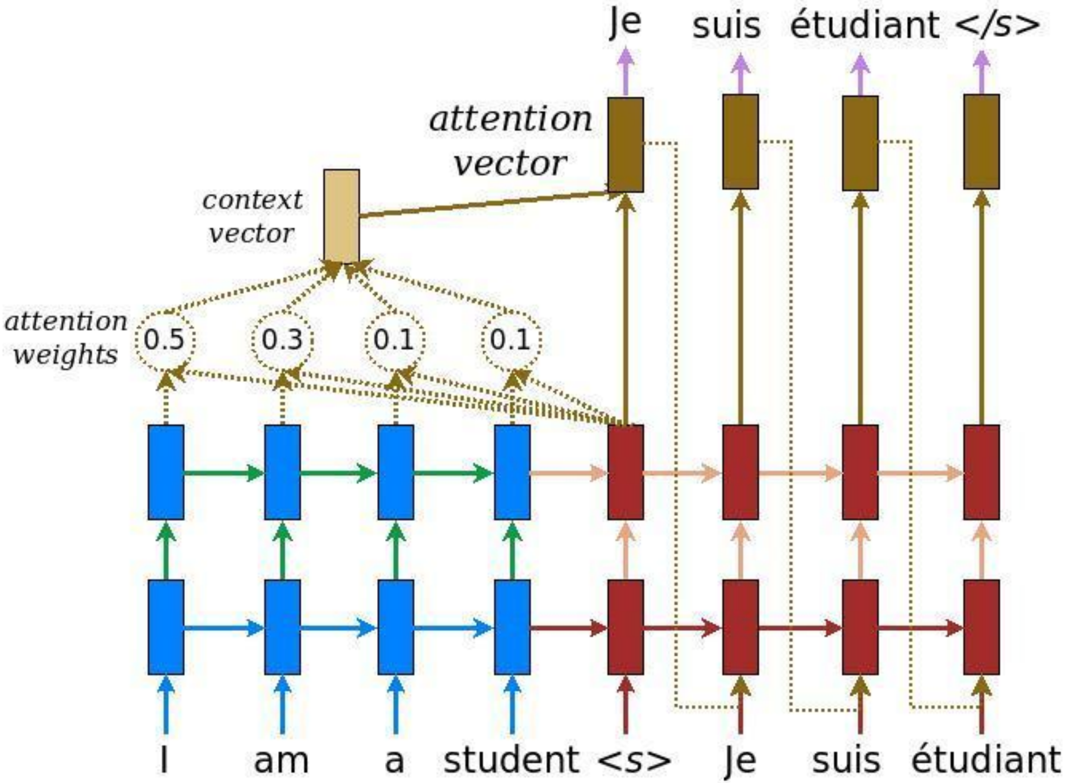
Attention in NLP: Neural MT

Luong et al., 2015

Attention was basically born for this.

- Encoder is a GRU RNN.
- Decoder is a GRU RNN, but hidden state also concatenates attention context.

Colab Notebook



Attention in CV: Show, attend and tell

Xu et al., 2015

The image is first encoded by a CNN to extract features.

Then a LSTM decoder consumes the convolution features to produce descriptive words one by one, where the weights are learned through attention.

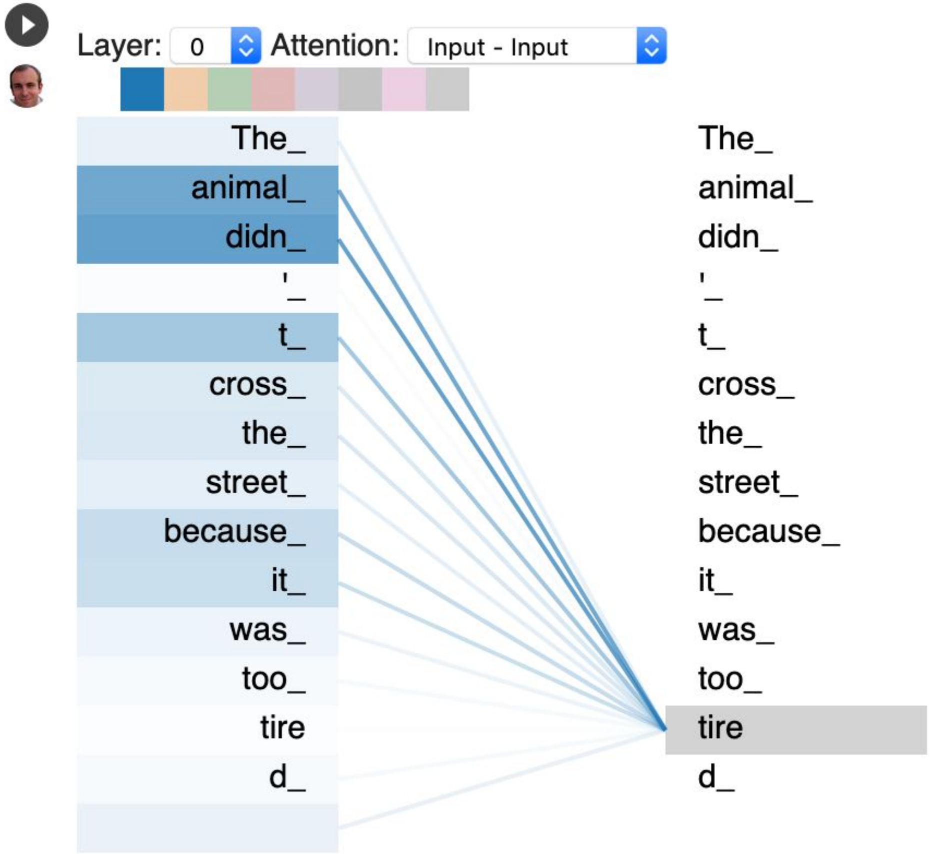
Colab Notebook

Self attention

[Cheng, 2016](#)

Aka, intra-attention.

Relates different positions of a sequence in order to compute a representation of the sequence itself.

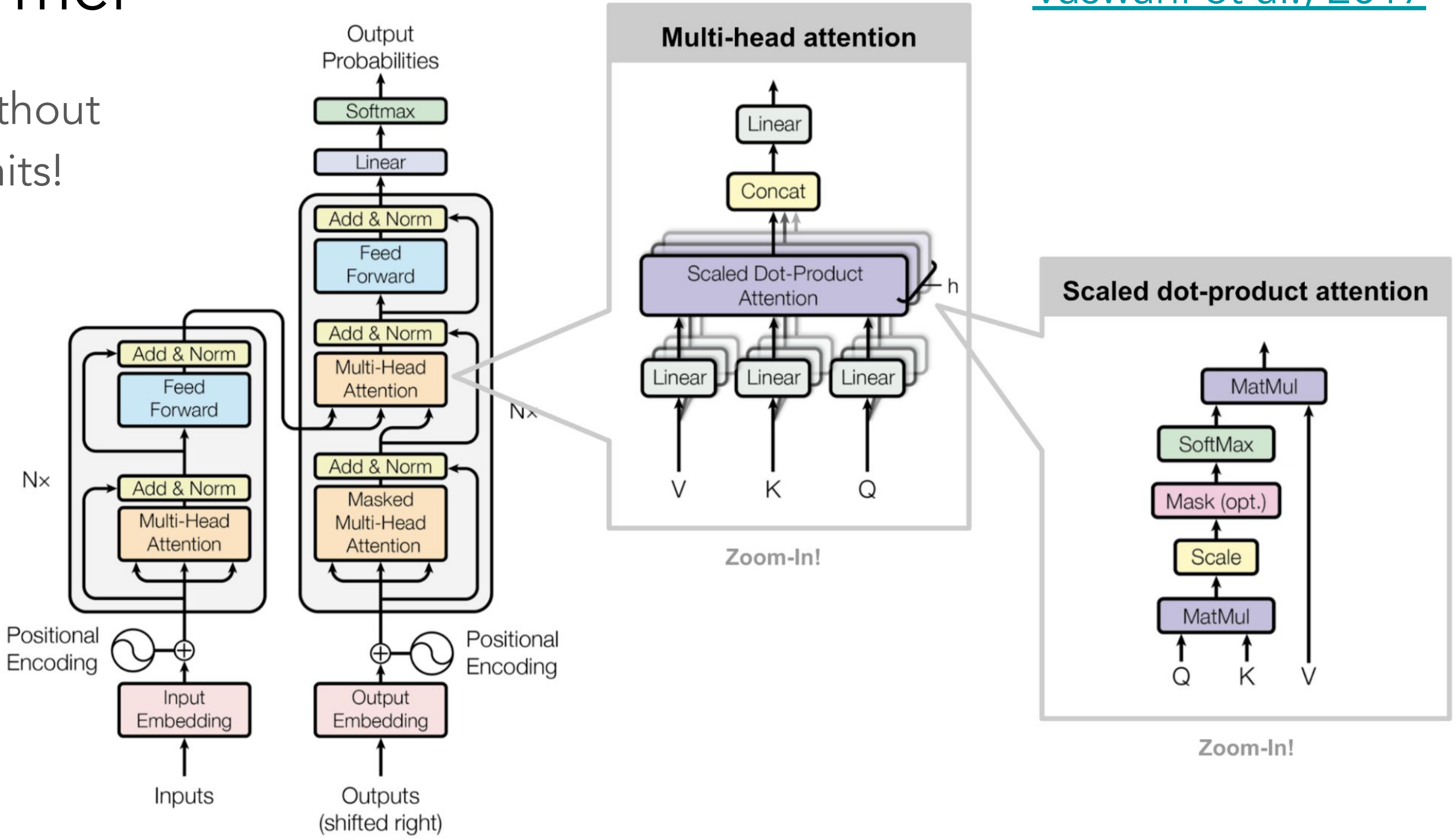


From [Tensor2Tensor Colab notebook](#)

Transformer

Vaswani et al., 2017

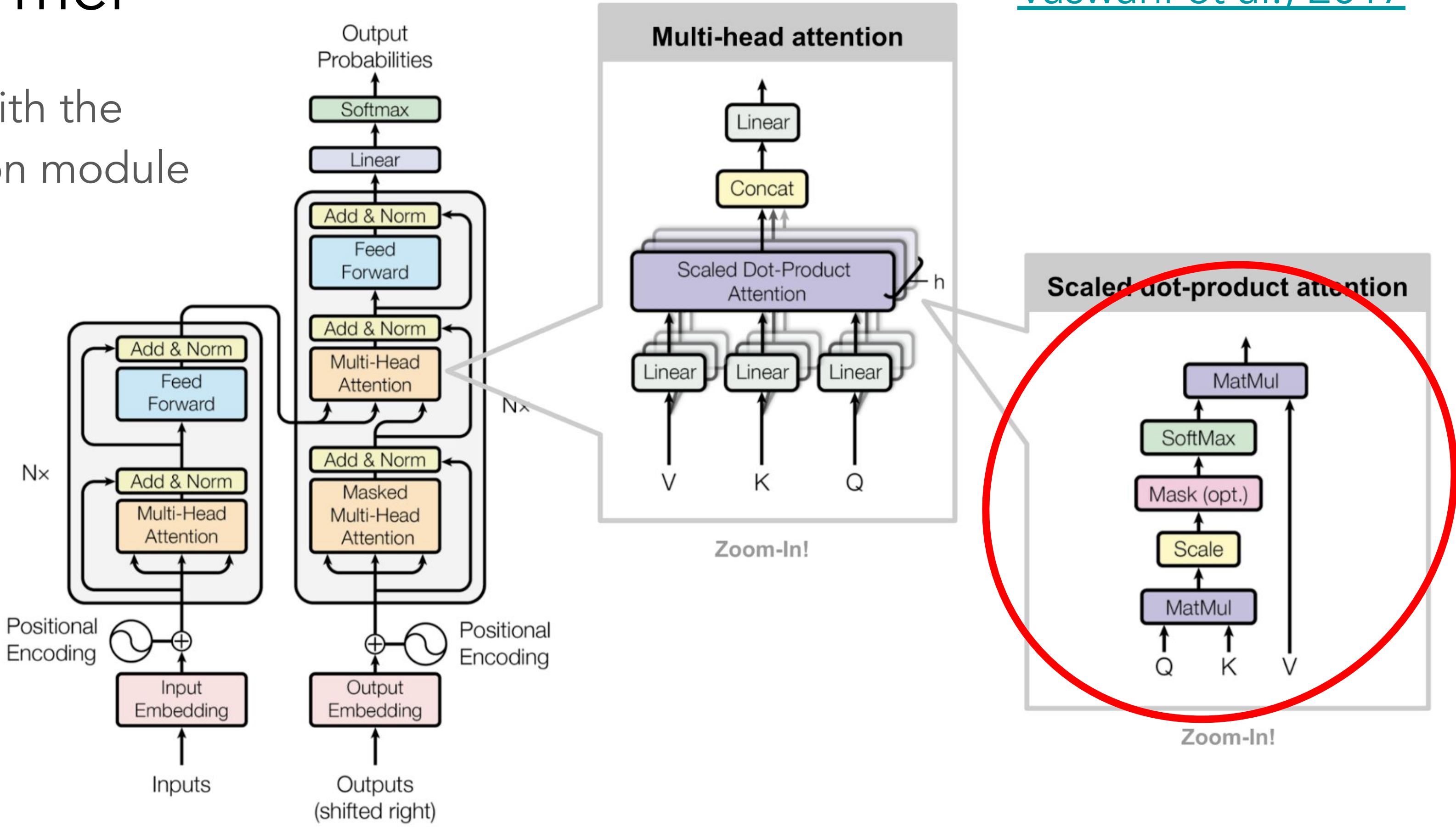
Seq2seq without recurrent units!



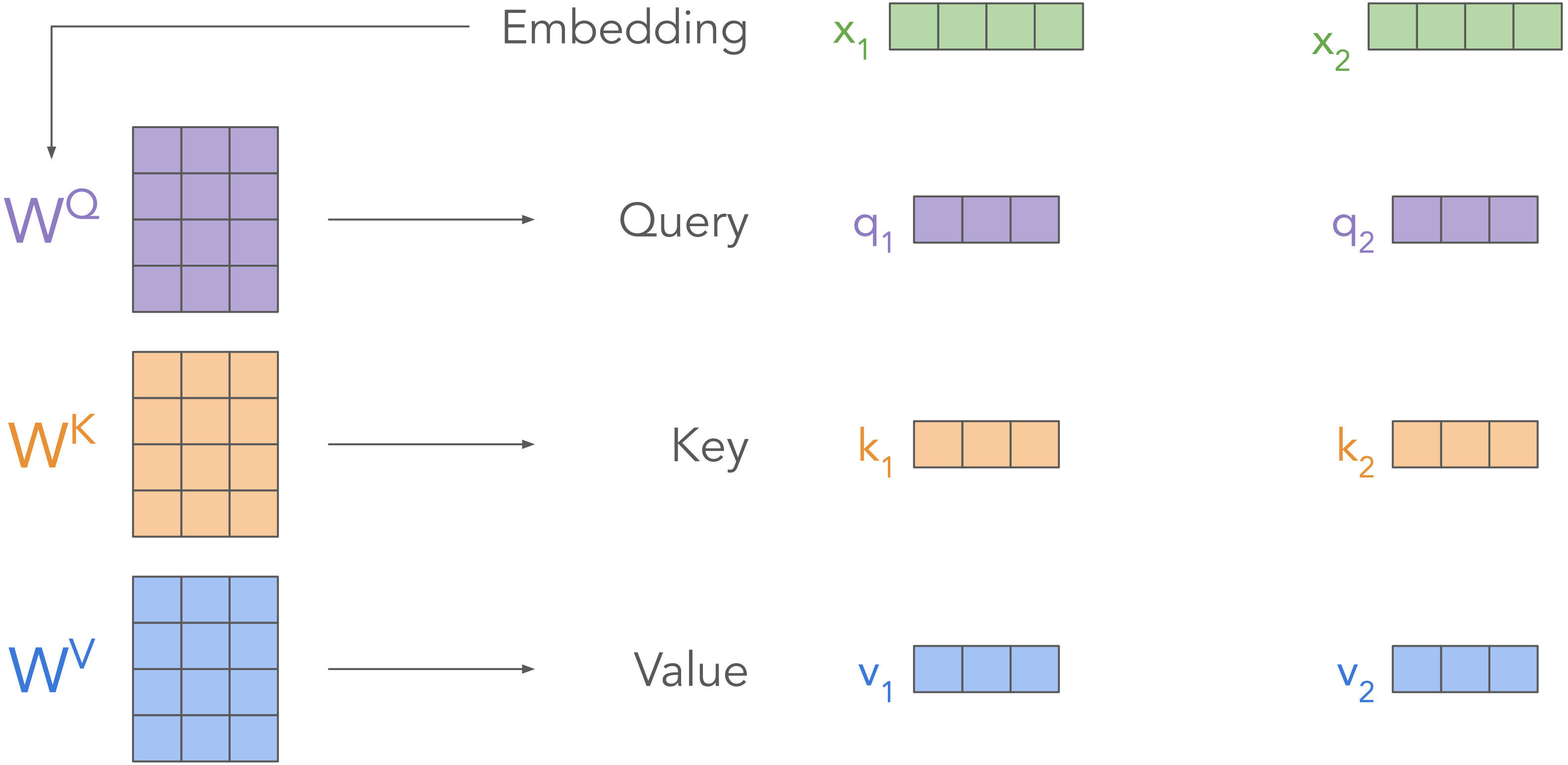
Transformer

[Vaswani et al., 2017]

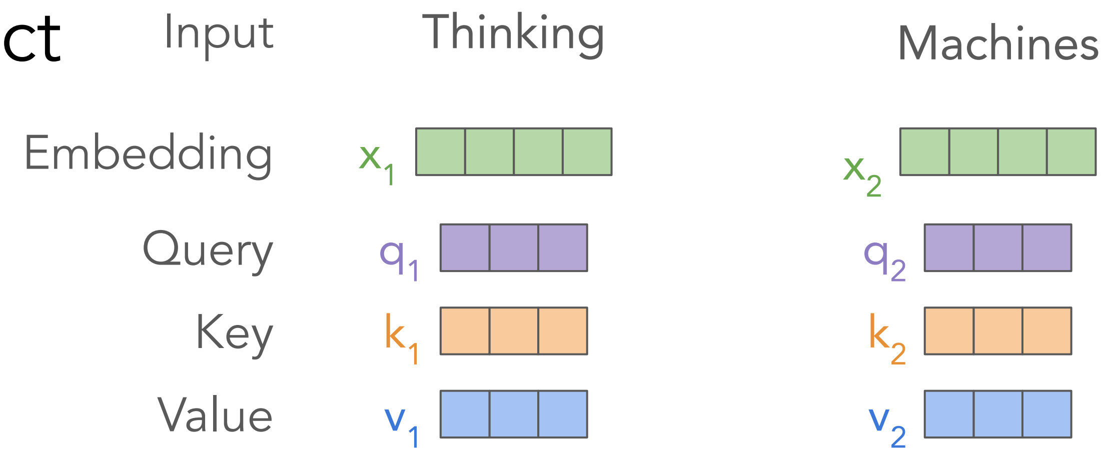
Let's start with the self-attention module



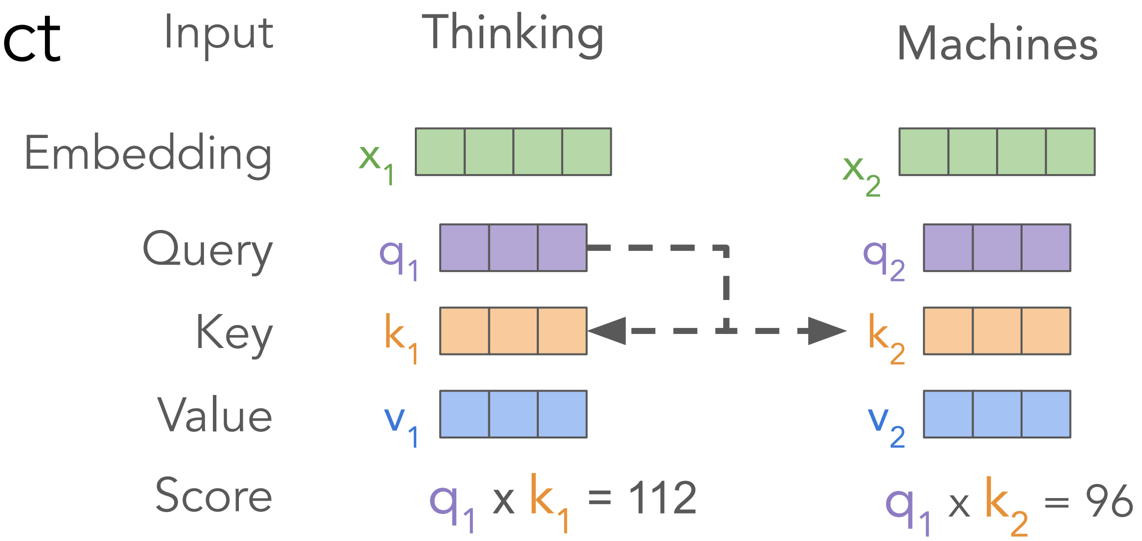
Scaled dot product attention



Scaled dot product attention



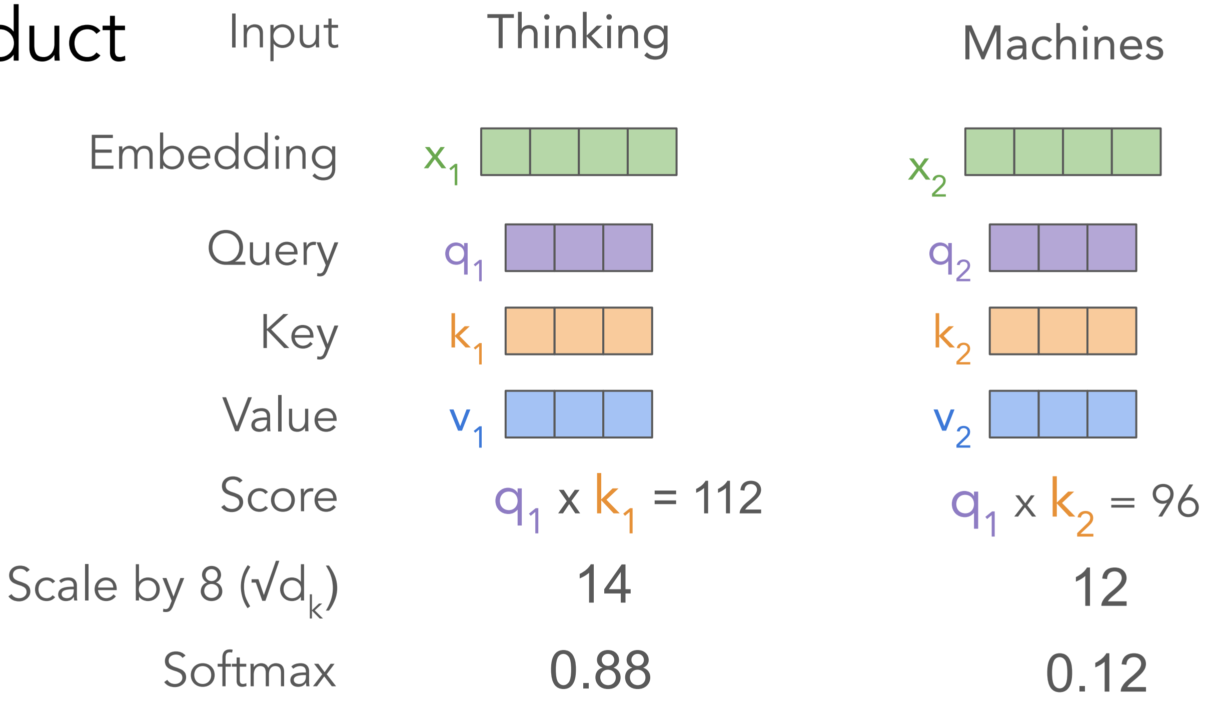
Scaled dot product attention



Scaled dot product attention

Input	Thinking	Machines
Embedding	x_1	x_2
Query	q_1	q_2
Key	k_1	k_2
Value	v_1	v_2
Score	$q_1 \times k_1 = 112$	$q_1 \times k_2 = 96$
Scale by 8 ($\sqrt{d_k}$)	14	12

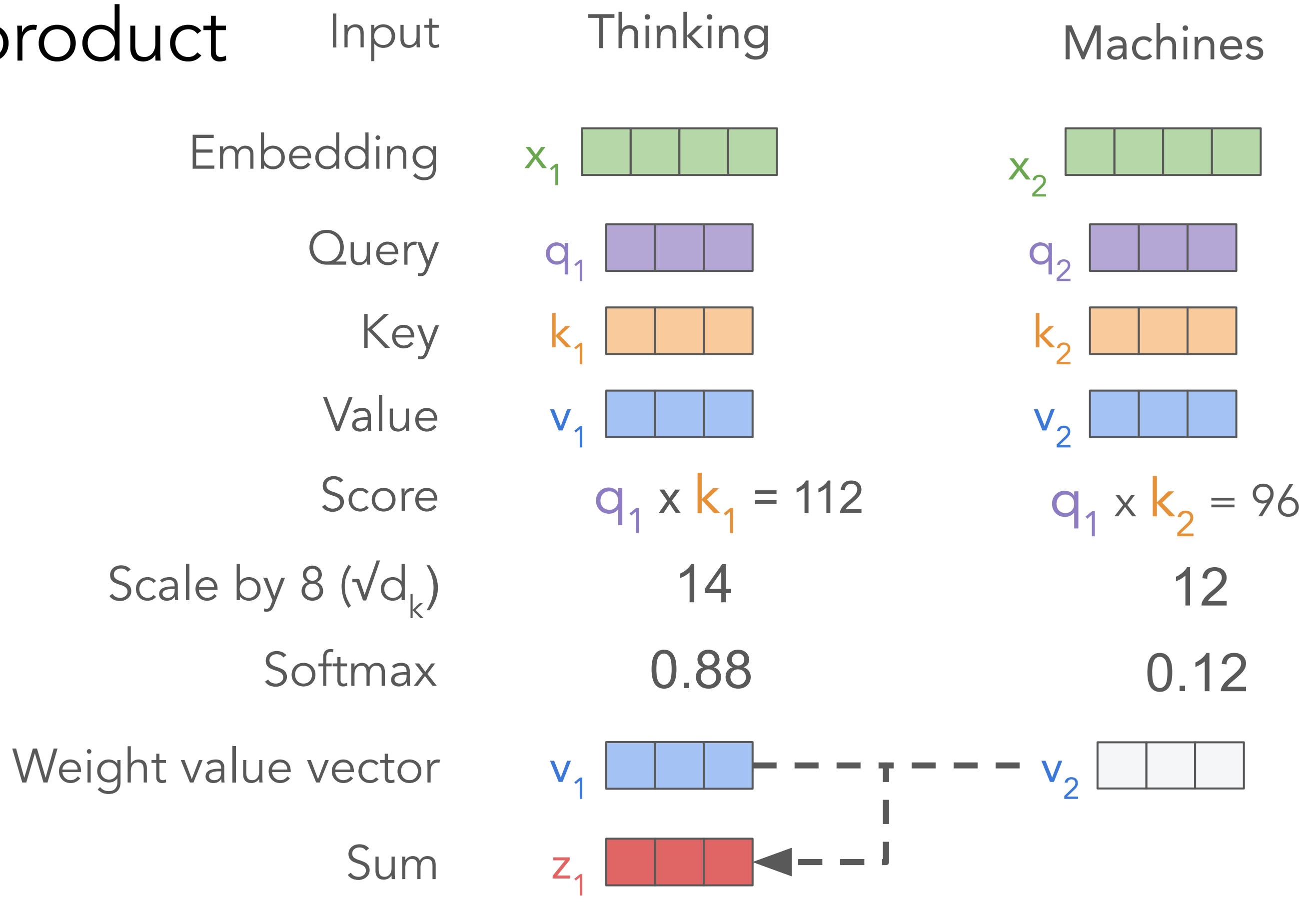
Scaled dot product attention



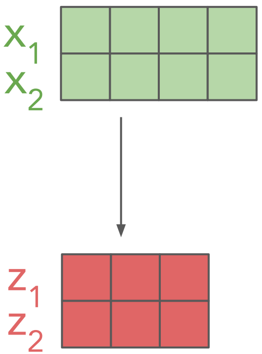
Scaled dot product attention

Input	Thinking	Machines
Embedding	x_1	x_2
Query	q_1	q_2
Key	k_1	k_2
Value	v_1	v_2
Score	$q_1 \times k_1 = 112$	$q_1 \times k_2 = 96$
Scale by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Weight value vector	v_1	v_2

Scaled dot product attention



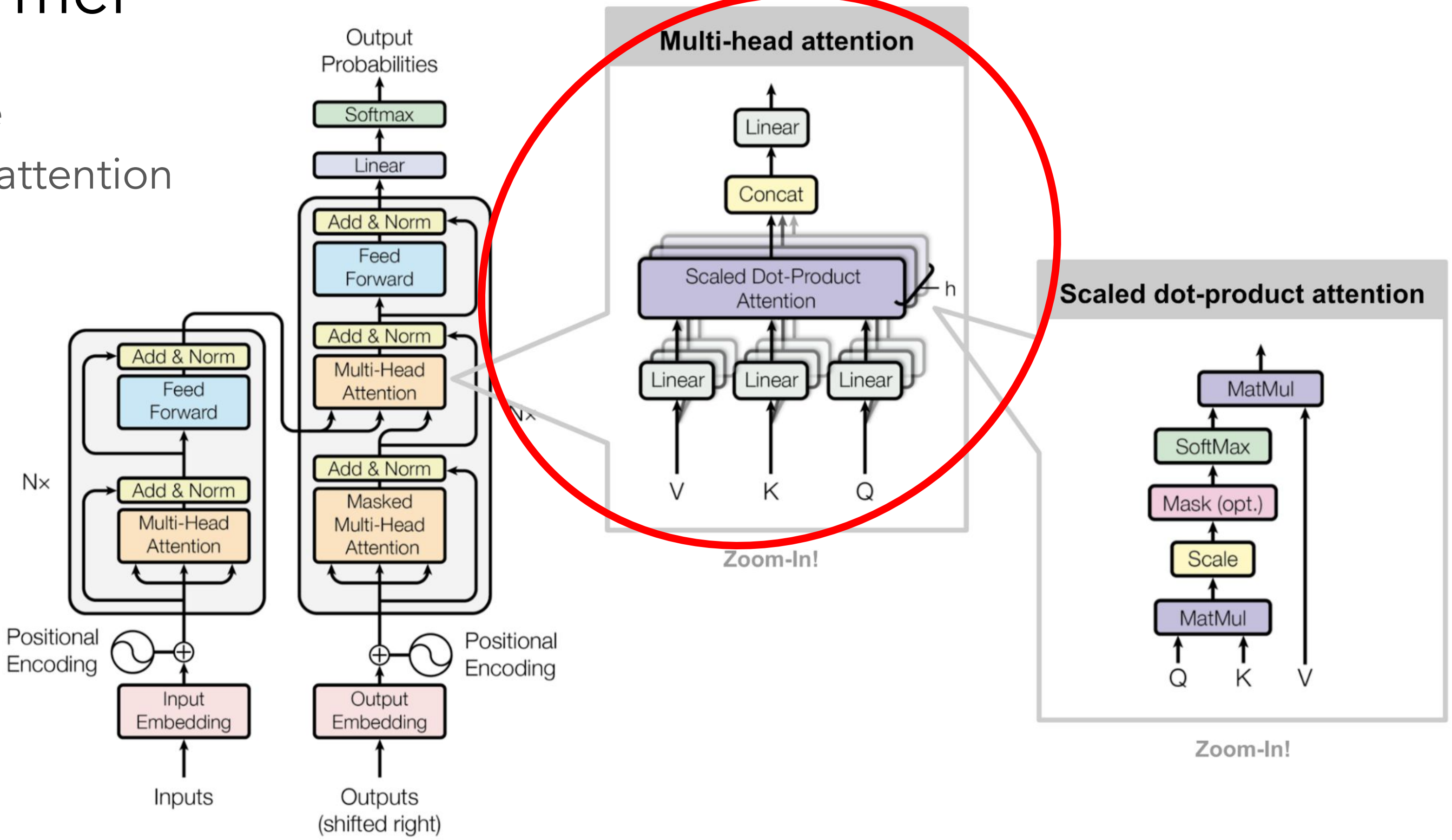
Scaled dot product attention



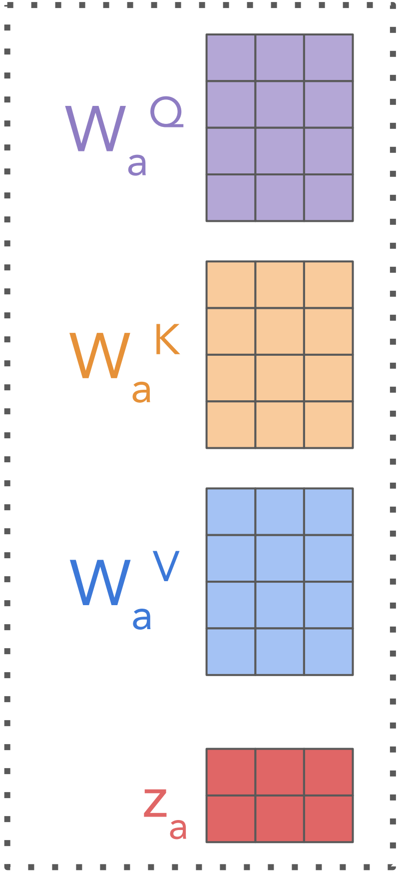
Input	Thinking	Machines
Embedding	x_1	x_2
Query	q_1	q_2
Key	k_1	k_2
Value	v_1	v_2
Score	$q_2 \times k_1 = 56$	$q_2 \times k_2 = 64$
Scale by 8 ($\sqrt{d_k}$)	7	8
Softmax	0.27	0.73
Weight value vector	v_1	v_2
Sum	z_1	z_2

Transformer

Now for the
Multi-head attention

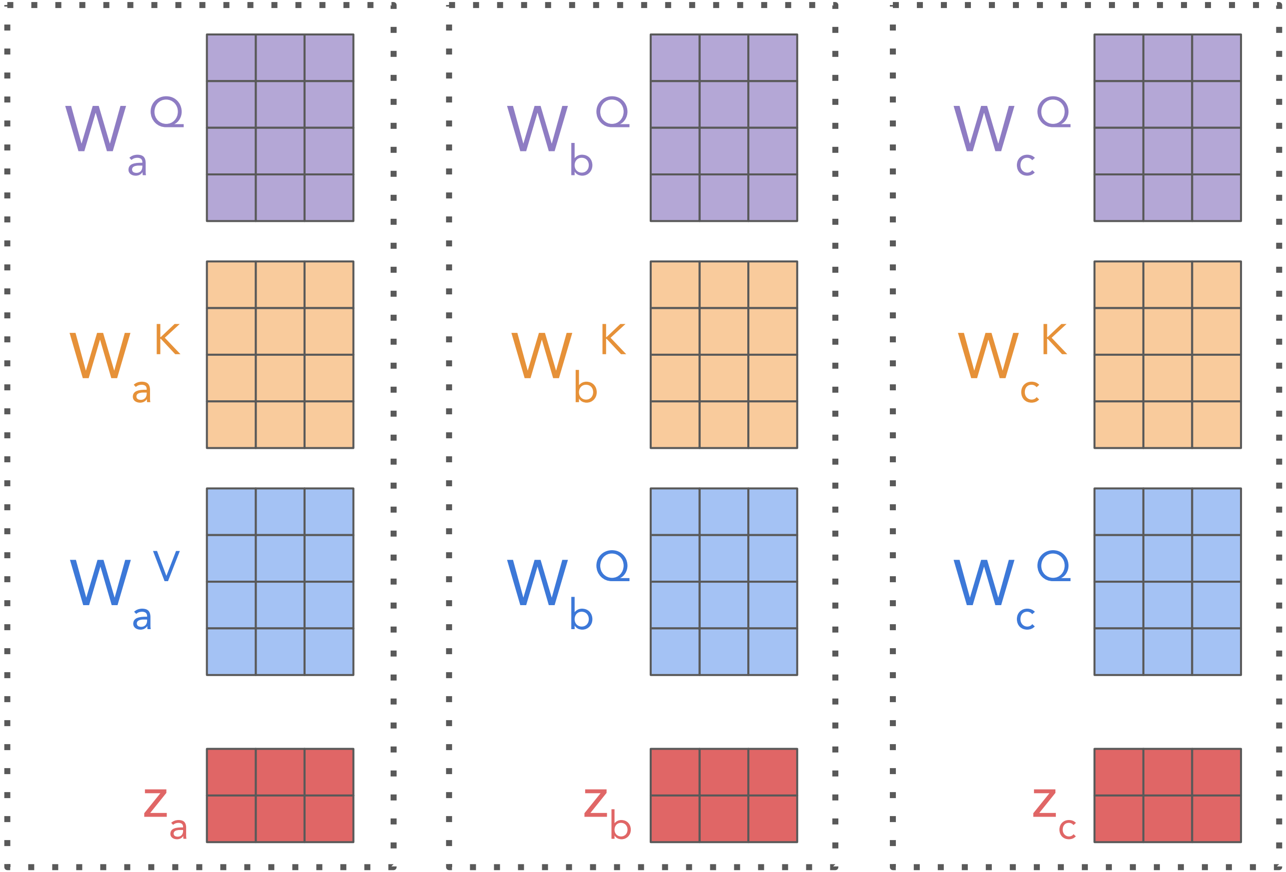


Multi head attention



The struct of Q-K-V matrices is an "attention head".

Multi head attention



The struct of Q-K-V matrices is an “attention head”.

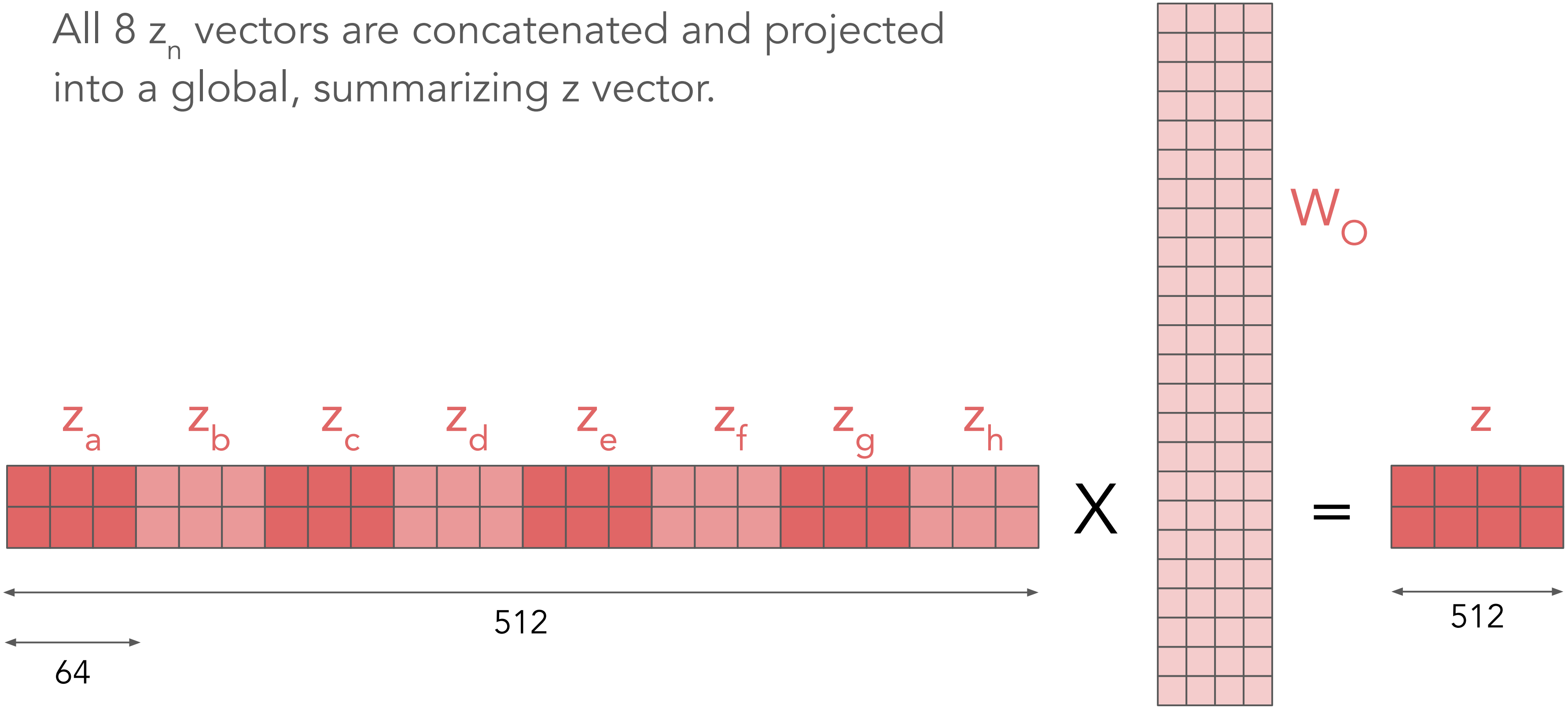
If 1 attention head is nice, 8 heads are better! (here 3 shown)

Each head projects the inputs in different subspaces.

Better generalization.

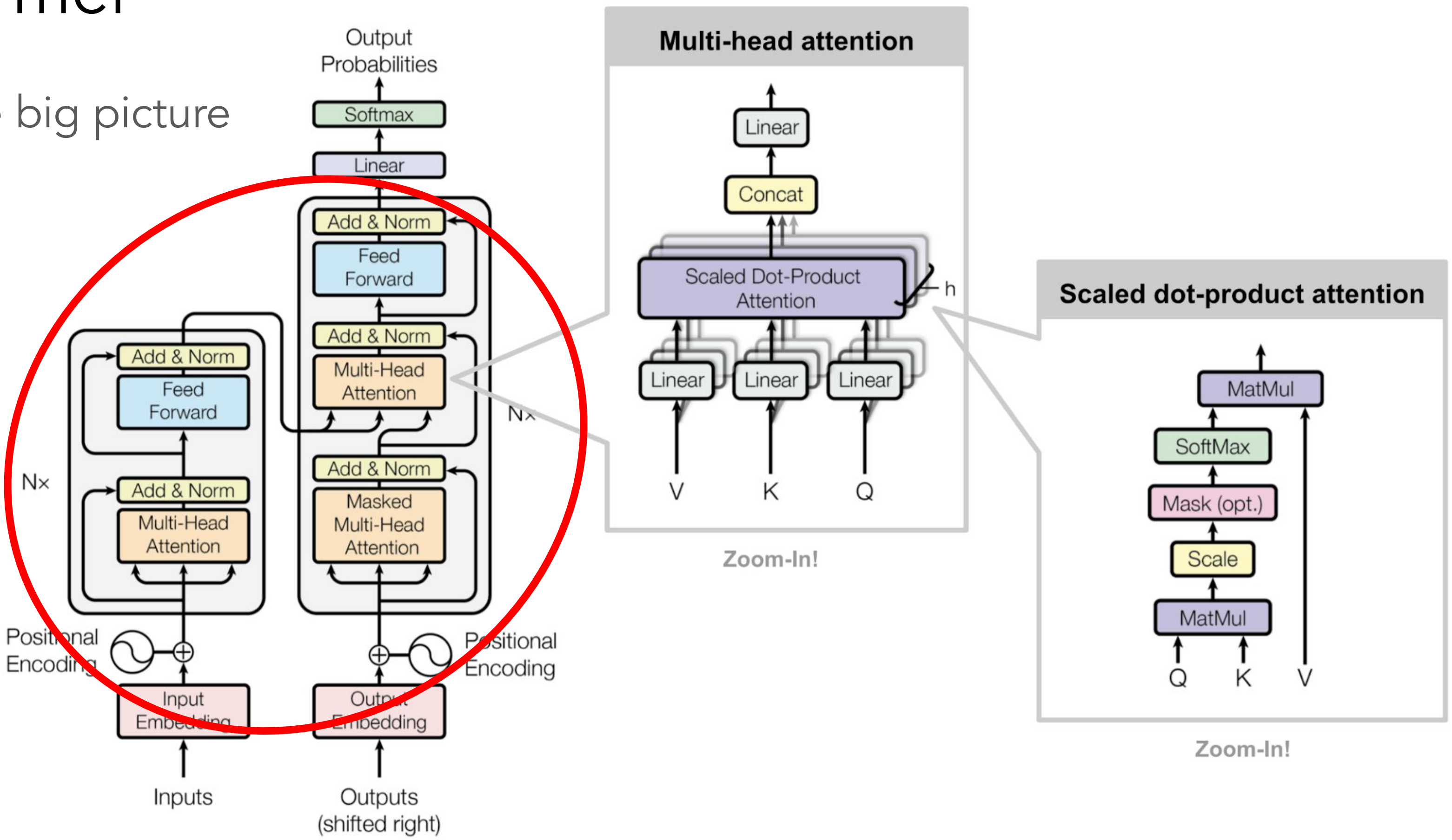
Multi head attention

All 8 z_n vectors are concatenated and projected into a global, summarizing z vector.



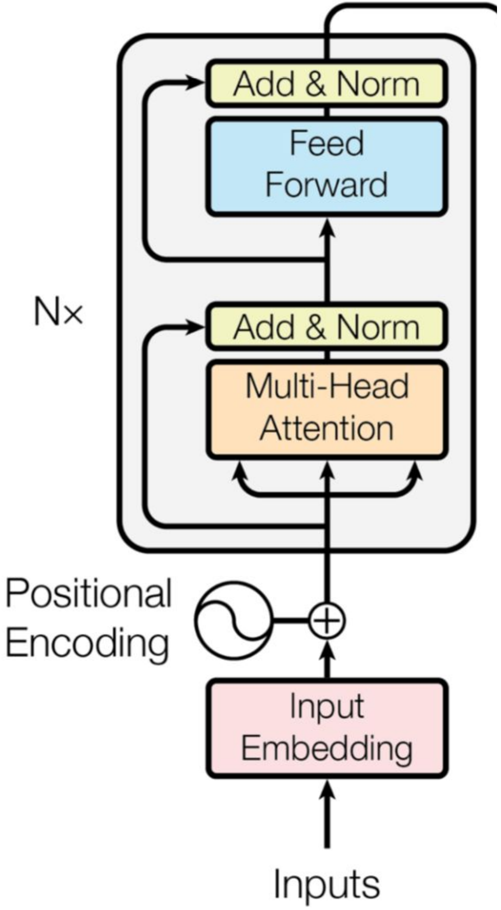
Transformer

Now for the big picture



Transformer

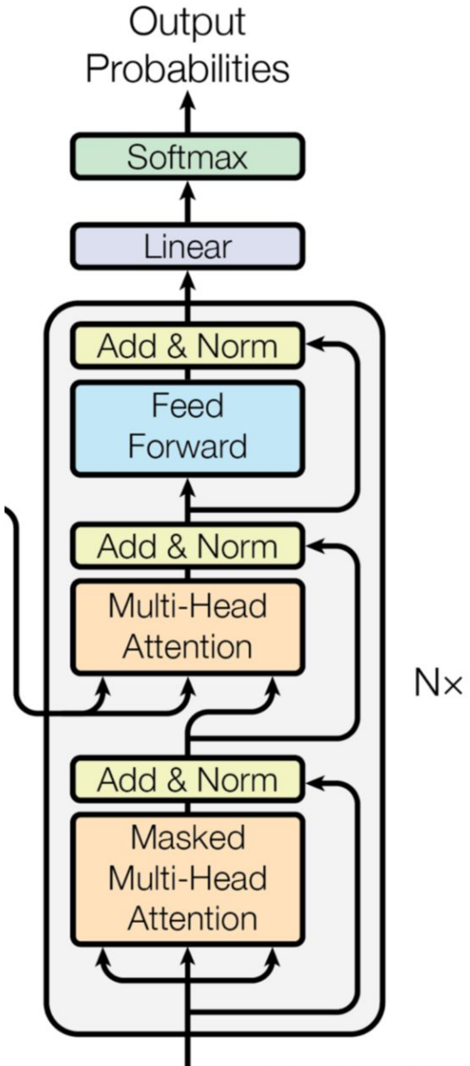
1. Inputs are 512-embedded. A sinusoidal wave is added to give positional context.
2. Embeddings pass through 8 attention heads of dimension 64. Outputs are summed to inputs with a residual connection and layer normalized.
3. Normalized outputs go into a 512-wide feed-forward net. Outputs are summed to inputs with a residual connection and layer normalized.
4. This is repeated in a 6-fold stack.



Transformer

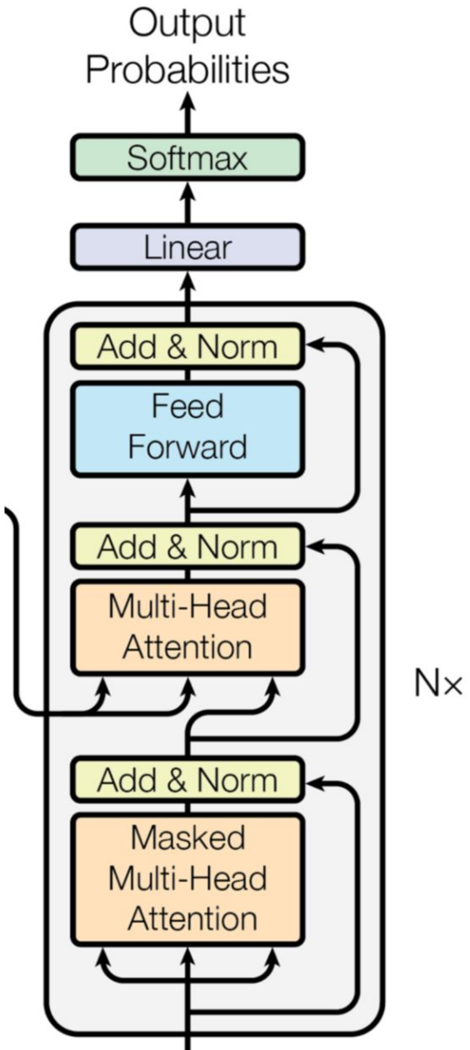
During training, use reference output.
During inference, start with token and then use outputs.

5. Outputs are 512-embedded. A sinusoidal wave is added to give positional context.
6. Embeddings pass through 8 masked attention heads of dimension 64, to avoid looking at future words. Outputs are summed to inputs with a residual connection and layer normalized.
7. Outputs pass through 8 more attention heads as Q, using encoder output as K-V. Outputs are summed to inputs with a residual connection and layer normalized.



Transformer

8. Normalized outputs go into a 512-wide feed-forward net. Outputs are summed to inputs with a residual connection and layer normalized.
9. This is repeated in a 6-fold stack.
10. Final outputs are projected in vocabulary space.
11. Projections are softmaxed to choose most likely output (and to ensure derivability).



The regicide of Recurrent networks

1. Sequence learning is performed without any Recurrent cell or Convolutions.
2. No vanishing gradient, all inputs are equally attended: stellar performance.
3. While RNNs tend to be extremely efficient in memory terms and to be serial, Transformers require huge memory and sport terrific parallelism because:
    a. multiple heads
    b. K,Q,V transformations
    c. residual + normalizations
    d. feed forward modules

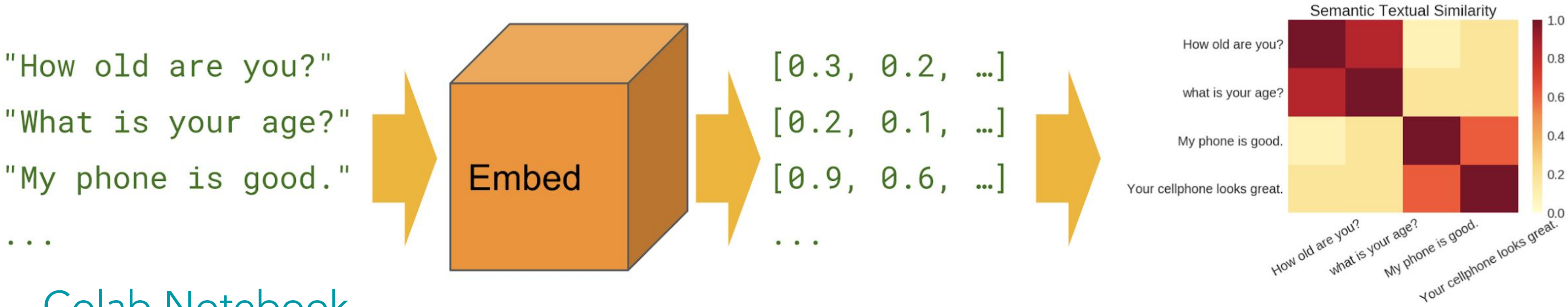
    can all be independently computed across inputs. It's a giant feed forward.

Attention in NLP: Universal Sentence Encoder

Cer et al., 2018

Encoder of text based on a Transformer.

Calculate 512-wide embedding for each word and sum element wise to obtain sentence representation. Rough, but works overall.



Colab Notebook

Hugging Face's Transformer

A very easy library to start with pre-trained models:

ALBERT, BART, BERT, BertGeneration, Blenderbot, CamemBERT, CTRL, DeBERTa, DialoGPT, DistilBERT, DPR, ELECTRA, FlauBERT, FSMT, Funnel Transformer, LayoutLM, Longformer, LXMERT, MarianMT, MBart, MobileBERT, OpenAI GPT/GPT2, Pegasus, ProphetNet, RAG, Reformer, RetriBERT, RoBERTa, SqueezeBERT, T5, Transformer XL, XLM, XLM-ProphetNet, XLM-RoBERTa, XLNet

Basic usage

Various tasks notebook

Multi-label classification notebook

Attention in NLP: GPT

Attention in NLP: GPT

Hold for a second

Attention in NLP: GPT

Although it has been extensively shown that this can generate fluent text because, well, it’s a language model,

THIS IS NOT THE MOST IMPRESSIVE FEATURE OF GPT.

We’ll get to it.

Attention in NLP: GPT

Radford et al., 2018

NLP too dependant on supervised data.

Can we build an as unsupervised as possible model to understand language?

Attention in NLP: GPT

Radford et al., 2018

NLP too dependant on supervised data.

Can we build an as unsupervised as possible model to understand language?

Generative Pre-trained Transformer for Language Understanding.

1. Pre-train a standard Language Model, with unlabeled data, to learn a representation that transfers with little adaptation to a wide range of tasks.
2. Fine-tune on a task with a few labeled data, relying on knowledge transfer with no relevant architectural modification.

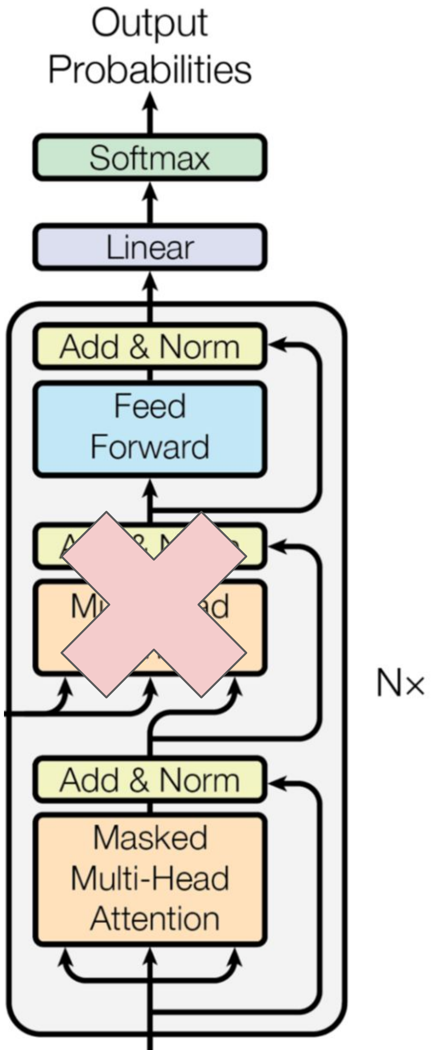
Attention in NLP: GPT

4 goal tasks:

- natural language inference
- question answering
- semantic similarity
- text classification

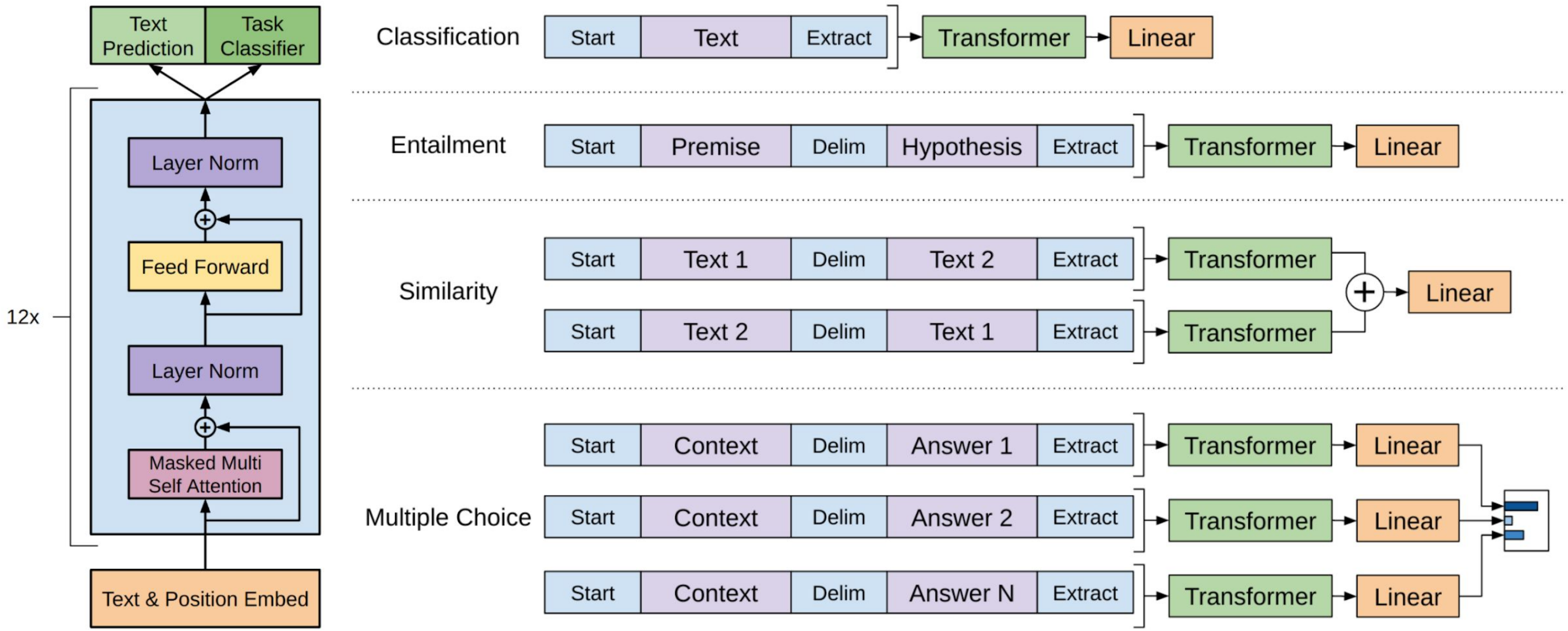
Uses a 12-layer, decoder-only Transformer with 12 Masked Attention heads (768-wide) (110M-340M params):

- good at learning long range dependencies
- doesn’t need Multi-head Attention from Encoder



Attention in NLP: GPT

Tasks inputs are structured as a strings with special separators.



Attention in NLP: GPT

The takeaway is that <u>pre-training a powerful enough</u> (means: enough parameters, enough expressiveness) <u>model builds a sensible foundation of the dependencies between language constituents</u>.

The proof is the knowledge transfer capability with no architecture modifications.

Attention in NLP: BERT

Devlin et al., 2018

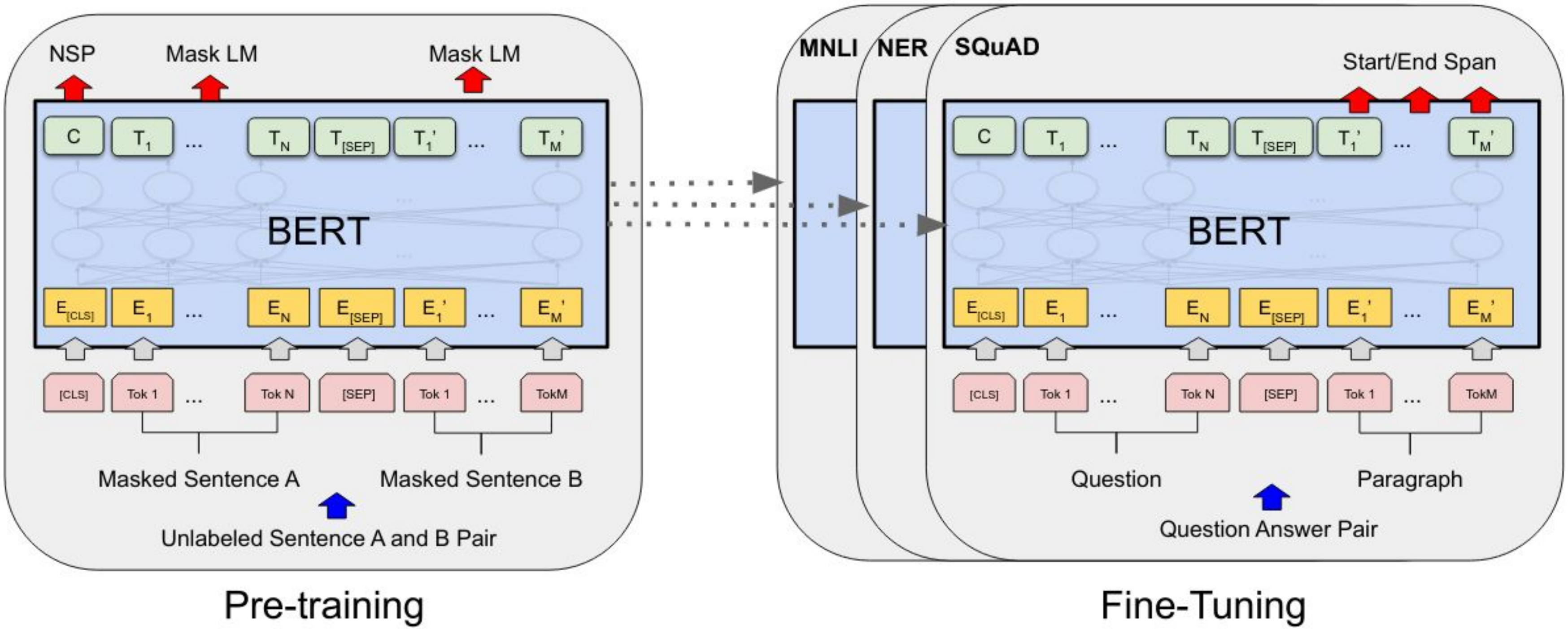
Bidirectional Encoder Representations from Transformers.

GPT was a decoder, BERT is an encoder; but makes little difference.

BERT pre-trains deep bidirectional representations by jointly conditioning on both left and right context in all layers.

1. Pre-train a "masked language model" (MLM): mask a random word in the input and predict that word using surrounding context.
2. Pre-train a next-sentence prediction binary task: output a binary label
3. Fine-tune on a task with a few labeled data, relying on knowledge transfer with no relevant architectural modification.

Attention in NLP: BERT



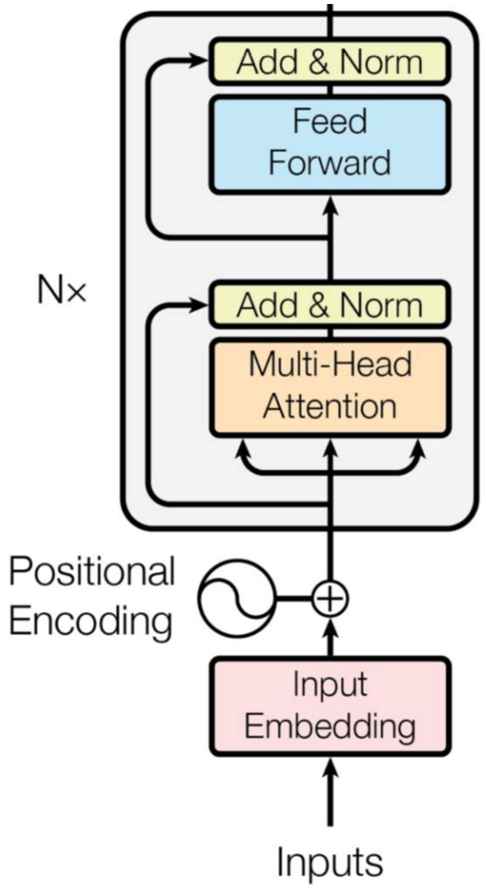
Attention in NLP: BERT

11 goal tasks:

- natural language inference
- question answering
- sentence continuation
- semantic similarity
- text classification

Uses a 12-layer, encoder-only Transformer with 12 Masked Attention heads (768-wide) (110M-340M params):

- yeah, authors wanted same dimensions as OpenAI's GPT to make comparisons



Attention in NLP: GPT-2

NLP models are good at 1 thing and require architecture modifications.

Can we build one model to rule them all, in zero-shot fashion?

Attention in NLP: GPT-2

Radford et al., 2019

NLP models are good at 1 thing and require architecture modifications.

Can we build one model to rule them all, in zero-shot fashion?

From GPT-2 paper:

"The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2,is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting".

NO FINE-TUNING: any task can be modeled at input encoding level.

Attention in NLP: GPT-2

For example,

- a translation training example can be written as the sequence `(translate to french, english text, french text)`.
- a reading comprehension training example can be written as `(answer the question, document, question, answer)`.

The idea is that most training data naturally reflects this encoding scheme:

<<a conversation can be heard between two guys in French: "-Comment on fait pour aller de l'autre cot´e? -Quel autre cot´e?", which means "- How do you get to the other side? - What side?">>

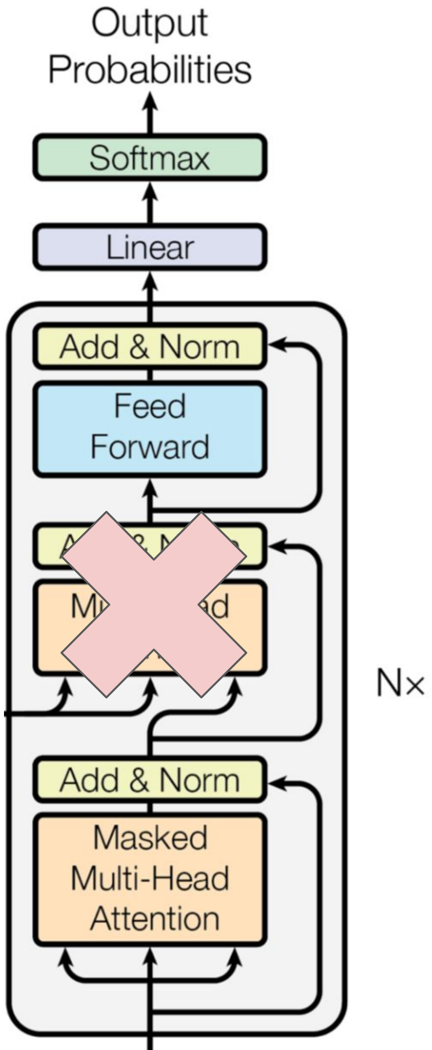
Attention in NLP: GPT-2

8 goal tasks, all zero-shot:

- long range dependency
- masked and normal language modeling
- common sense reasoning
- reading comprehension and summarization
- translation
- question answering

Uses a 48-layer, decoder-only Transformer with 12 Masked Attention heads (768-wide) (117M-1.5B params):

- the main change is the capacity of the model
- smallest one is to compare with BERT



Attention in NLP: GPT-2

Main value of the paper is the study about why this even works:

- generalization vs memorization
- corpus cleanup

The takeaway is that when a large language model is trained on a sufficiently large and diverse dataset, it is able to perform well across many domains and datasets.

Attention in NLP: GPT-3

GPT-2 has shown that a large language model can perform on tasks in zero-shot settings, without fine-tuning at training time.

Attention in NLP: GPT-3

GPT-2 has shown that a large language model can perform on tasks in zero-shot settings, without fine-tuning at training time.

How about if the fine-tuning happens at inference time?

Humans can generally perform a new language task from only a few examples or from simple instructions.

Attention in NLP: GPT-3

[Brown et al., 2020]

GPT-2 has shown that a large language model can perform on tasks in zero-shot settings, without fine-tuning at training time.

How about if the fine-tuning happens at inference time?

Humans can generally perform a new language task from only a few examples or from simple instructions.

GPT-3 is an autoregressive language model with 175B params that demonstrates *few-shot learning*: training samples are passed within the input string, before the actual query input.

A meta-learning system, completely based on input encoding and model capacity.

Attention in GAN

[Zhang et al., 2018]

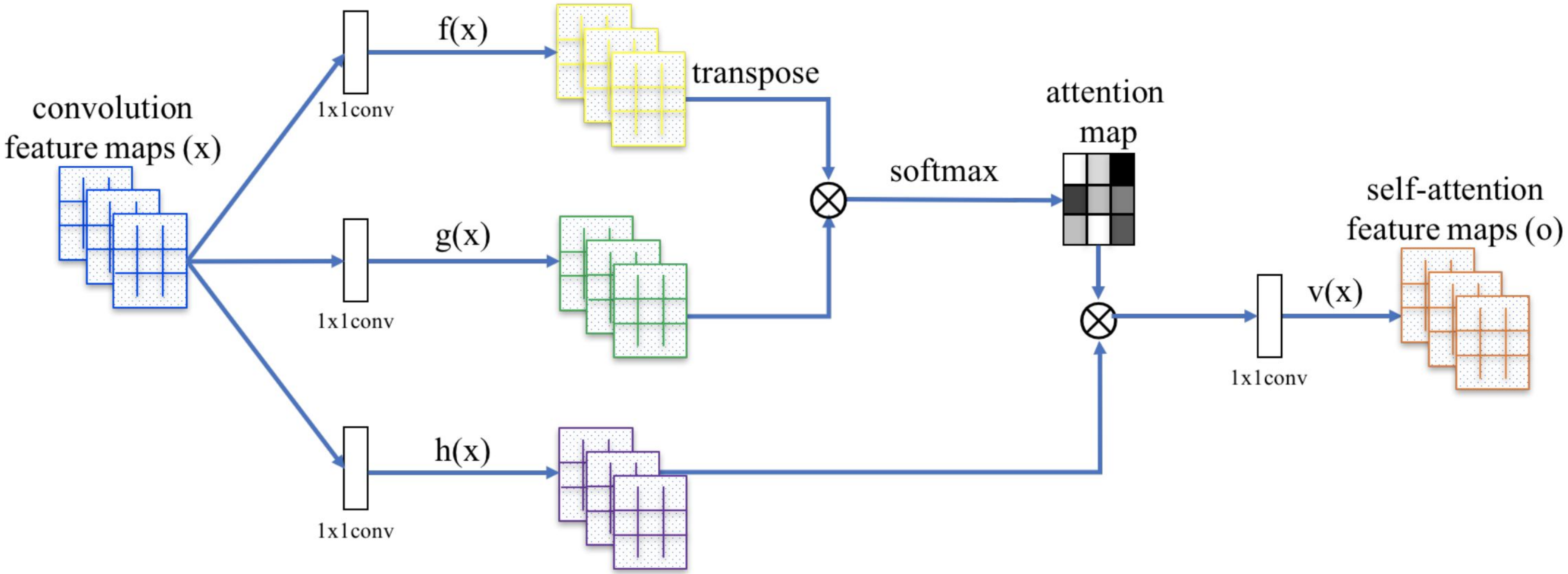
SAGAN is a

- convolutional GAN
- that uses a self-attention layer/block in the generator model,
- does spectral normalization on both the generator and discriminator,
- trains via the two time-scale update rule (TTUR),
- and the hinge version of the adversarial loss.

Self-attention layers enable relationships modeling between spatial regions.

Attention in GAN

The image features from the previous hidden layer x are transformed into two feature spaces (f,g) to calculate the attention and then used to weight x itself.



Links

[Attention? Attention!]

[The Illustrated Transformer]

[Hugging Face – On a mission to solve NLP, one commit at a time.]

#CODEMOTIONCONF_IT