# Transformer

Assoc. Prof. Peerapon Vateekul, Ph.D.
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
Peerapon.v@chula.ac.th
(Credit to TA's & This slide is based on 2110572: Natural Language Processing Systems)

# Attention mechanism

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



### ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

### 1 INTRODUCTION

## 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



Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

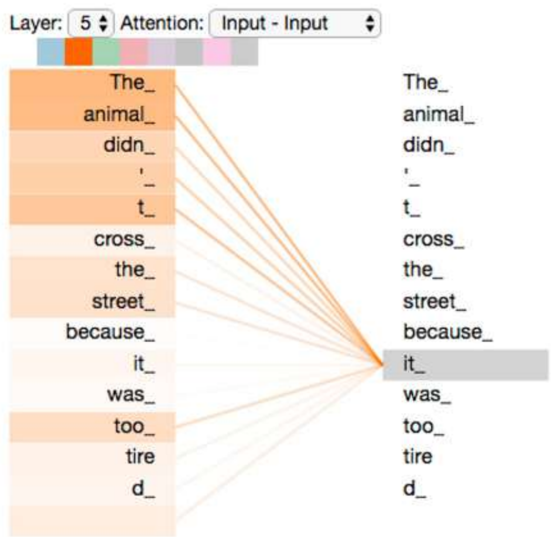
Reference: Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR(2015).

# Scaled Dot-Product Attention (1)

- Introduced in Attention is all you need (Viswani et al., 2017)
- **NO** recurrence nor convolution
- Widely used today in all Transformer-based model
- "Relating different positions of a single sequence in order to compute a representation of the sequence"

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

where the query, keys, values, and output are all vectors and $d_k$ is a number.




Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

# + Scaled Dot-Product Attention (2)

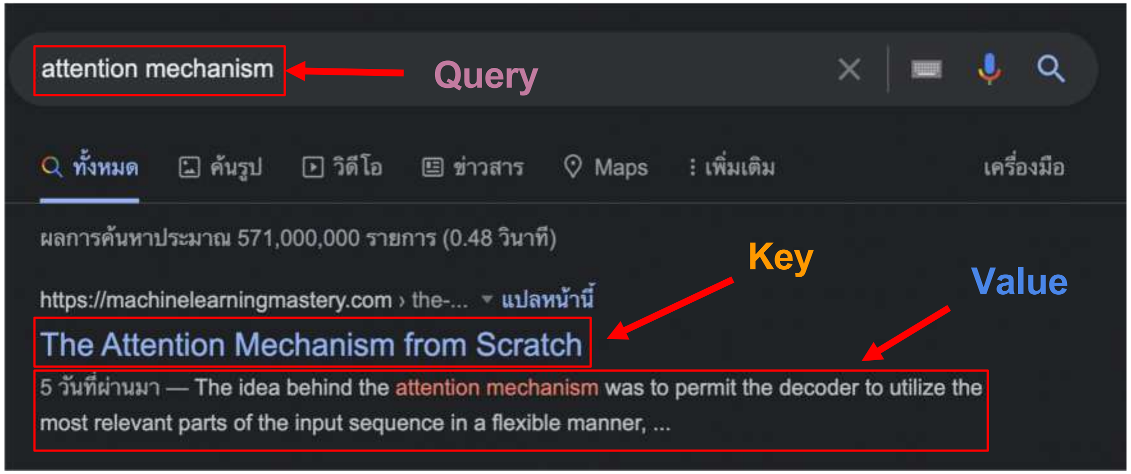
## What are the “query”, “key”, and “value” vectors?

As an analogy, think of Google Search.

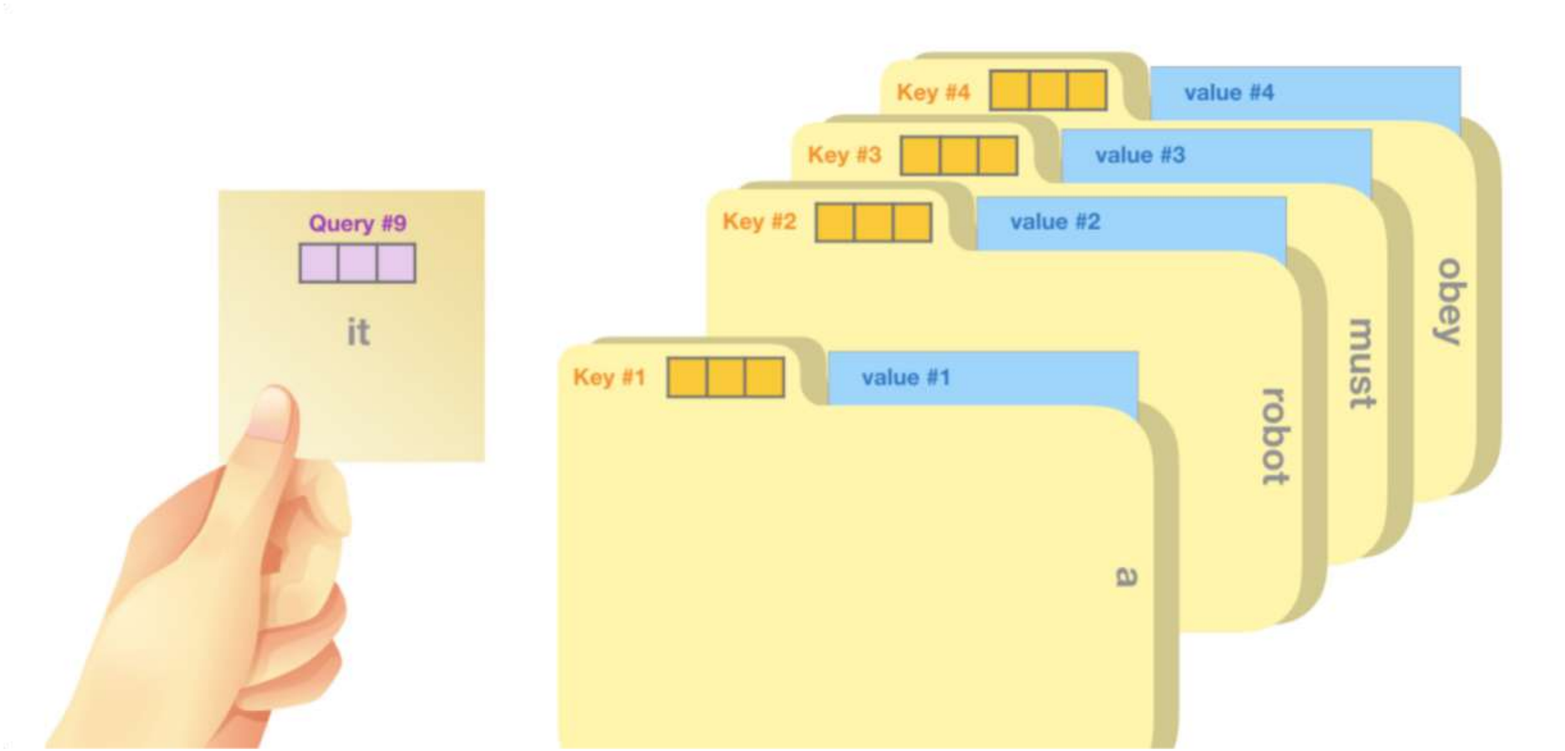
Query - what we want to know

Key - how to index information

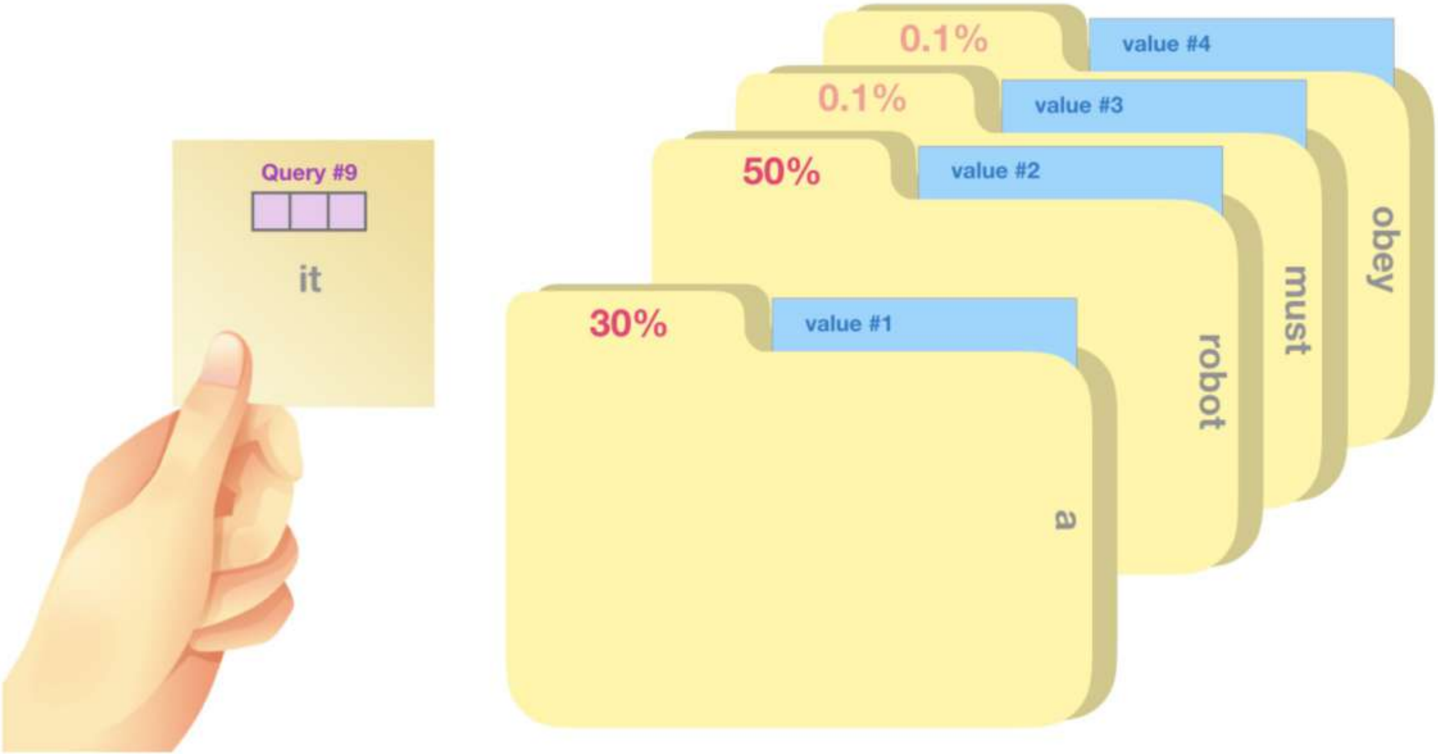
Value - what kind of info is in each website



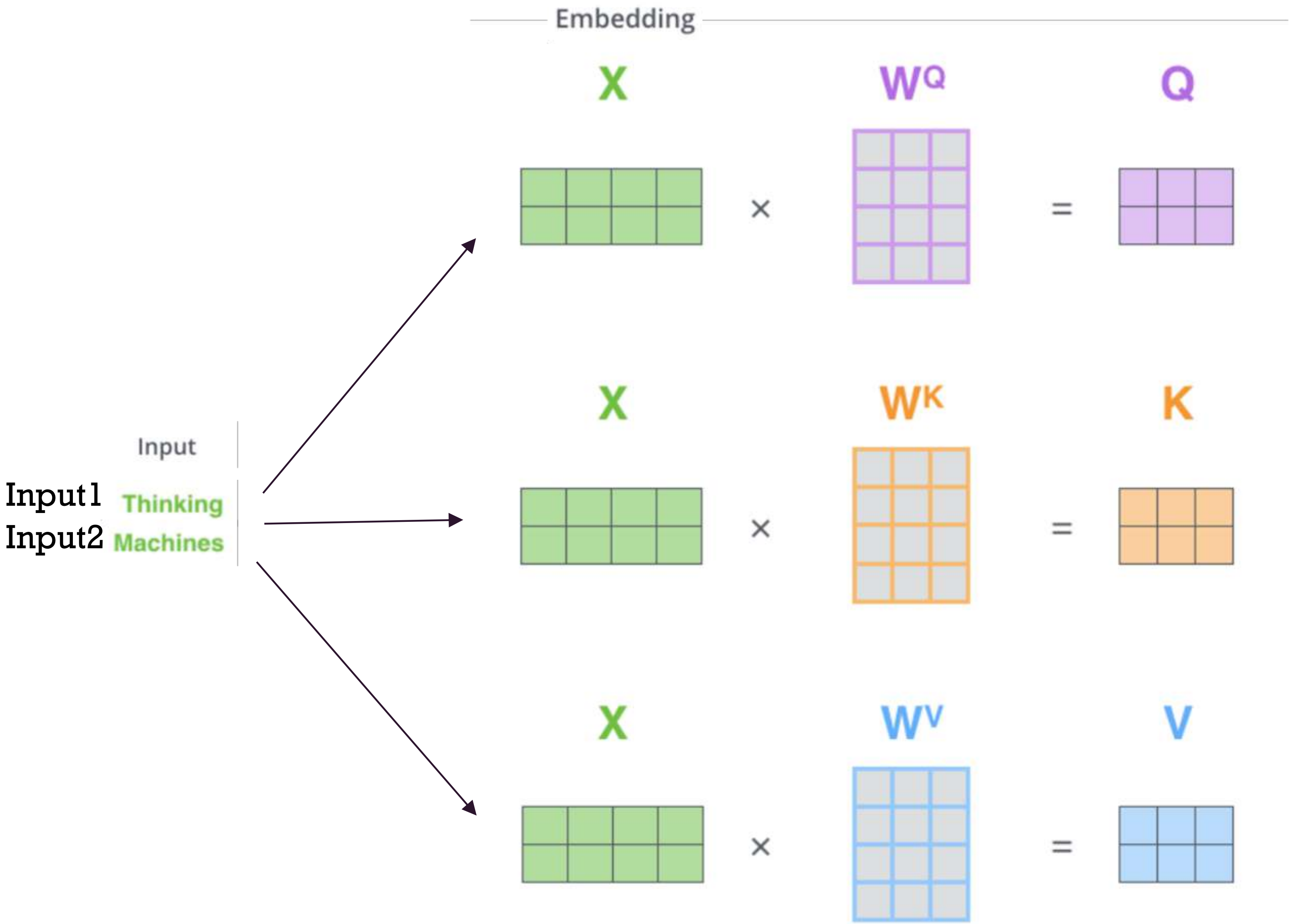

Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

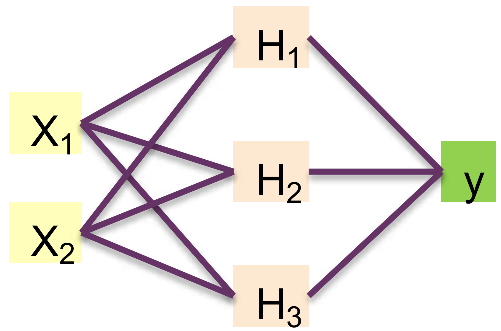



http://jalammar.github.io/illustrated-gpt2/



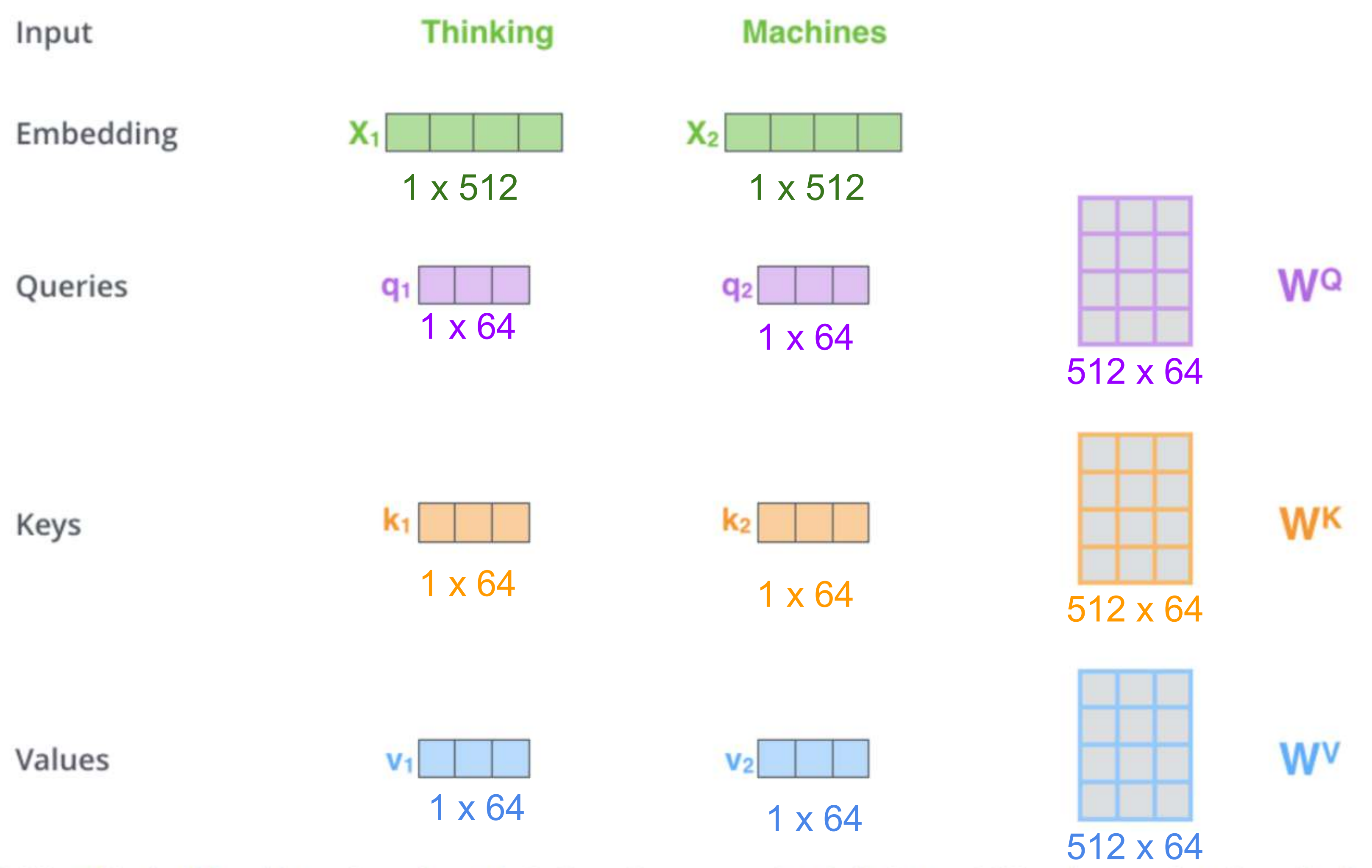

http://jalammar.github.io/illustrated-gpt2/



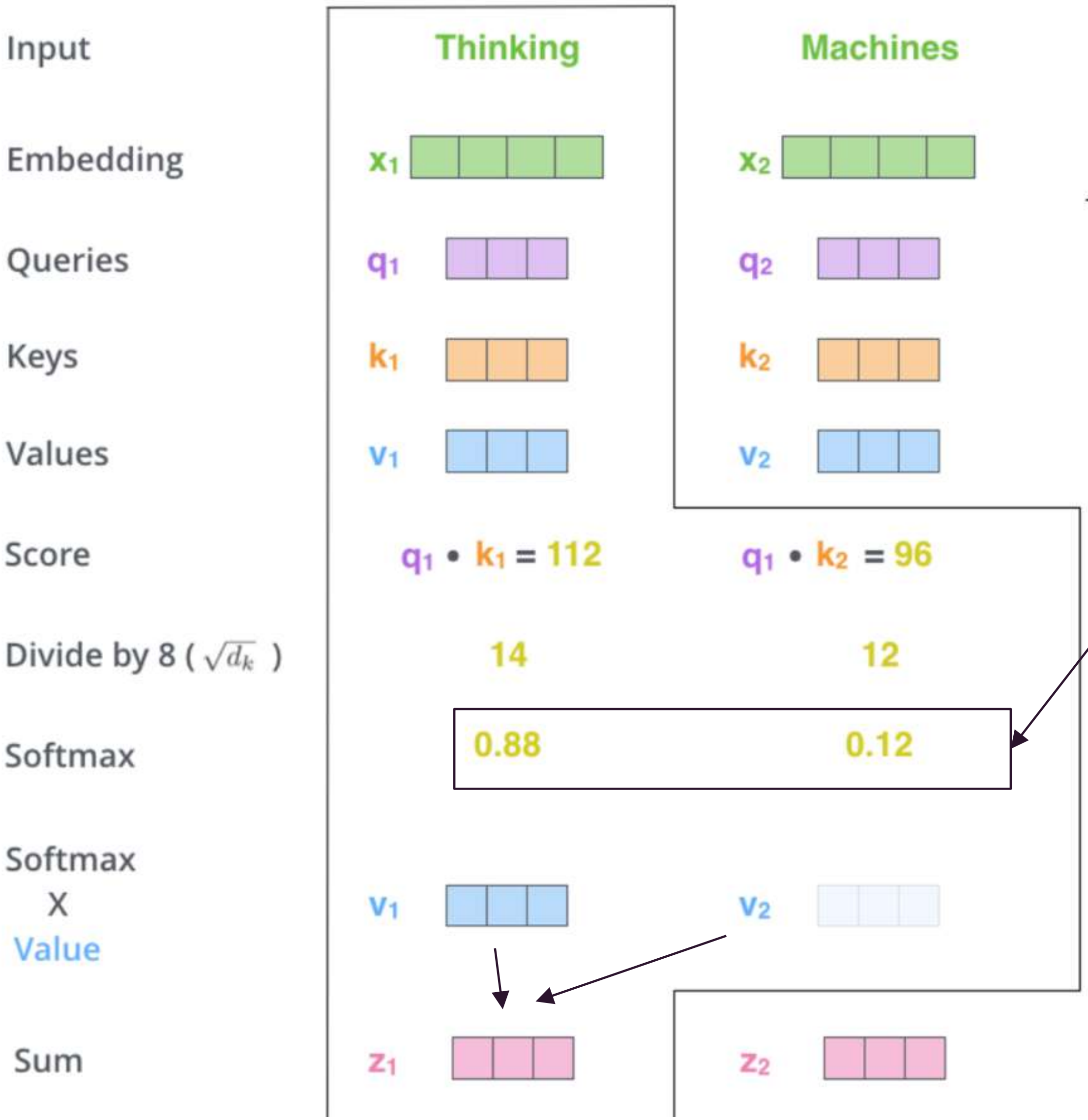

http://jalammar.github.io/illustrated-transformer/

$W^Q$, $W^K$, and $W^V$ are linear layers



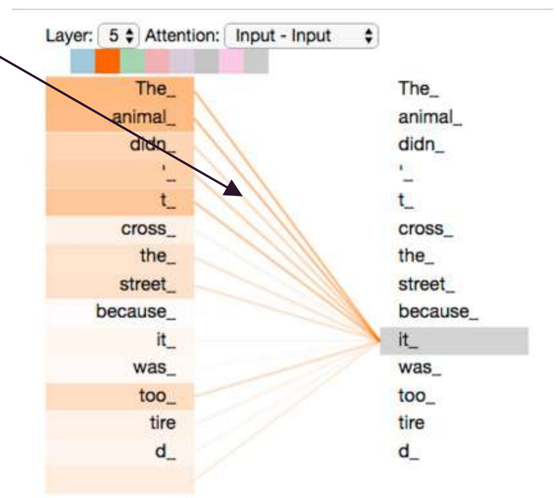

Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

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



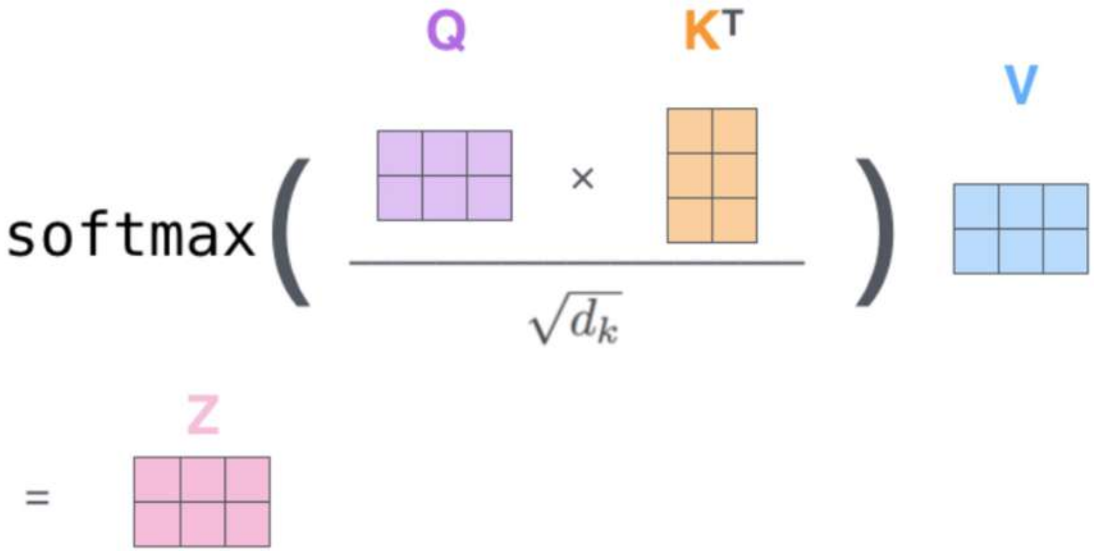

$$\text{Attention}(Q, K, V) = \boxed{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)}V$$

Attention scores

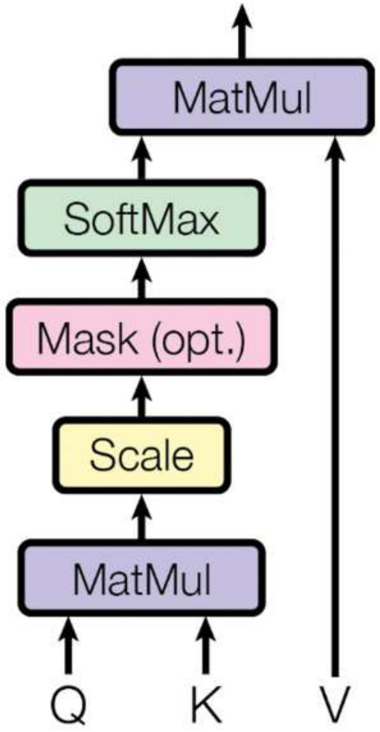



http://jalammar.github.io/illustrated-transformer/

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




Scaled Dot-Product Attention




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

# Attention Is All You Need

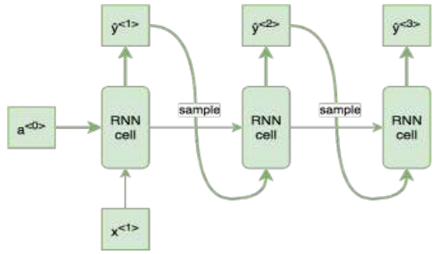
**Ashish Vaswani*** Google Brain avaswani@google.com
**Noam Shazeer*** Google Brain noam@google.com
**Niki Parmar*** Google Research nikip@google.com
**Jakob Uszkoreit*** Google Research usz@google.com
**Llion Jones*** Google Research llion@google.com
**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu
**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com
**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com

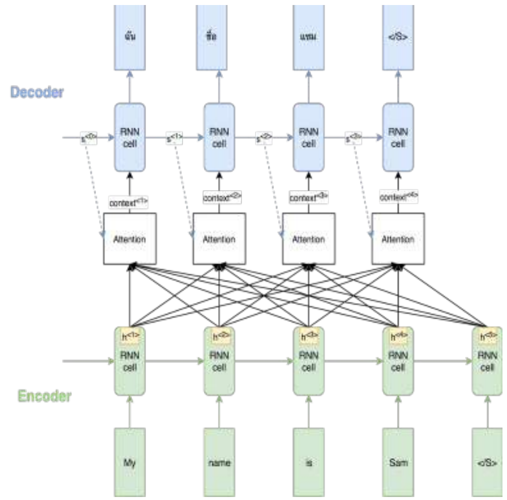




Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

# The Transformer

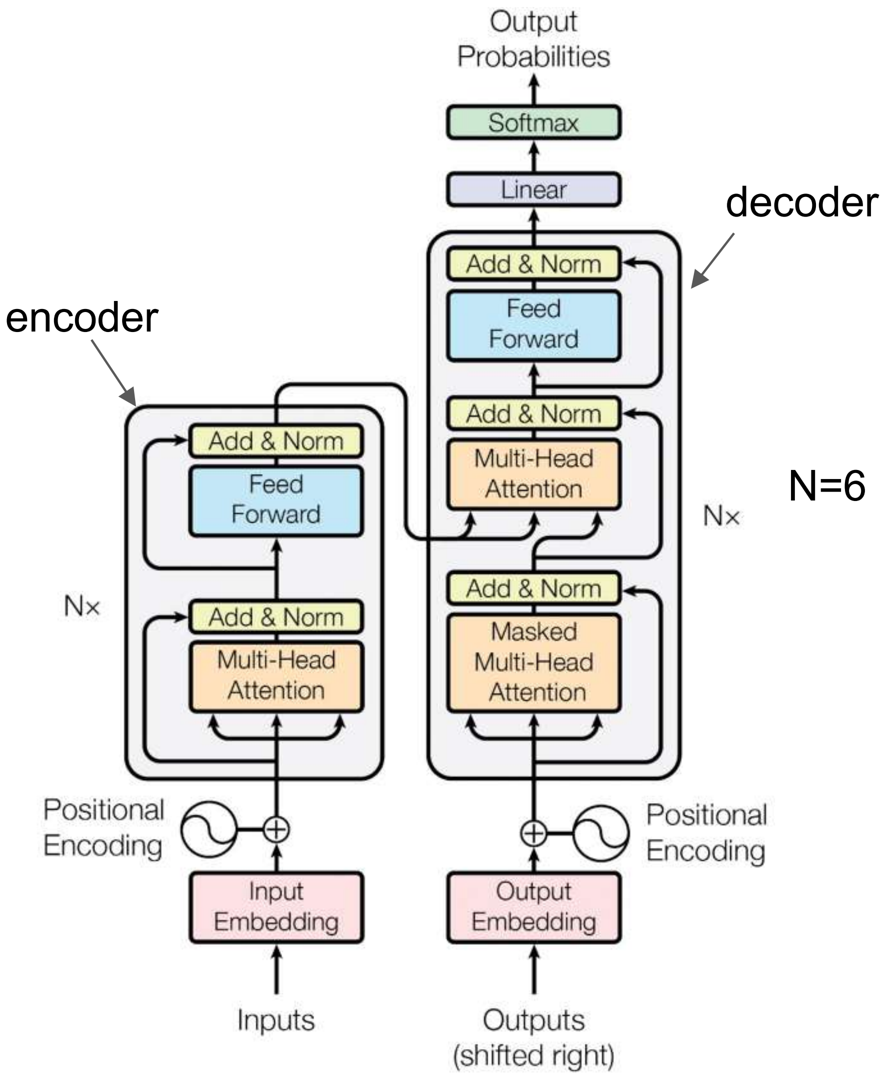


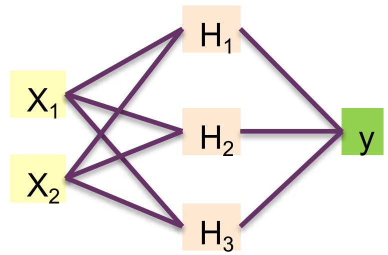

Only rely on the attention mechanism - **No RNN**!

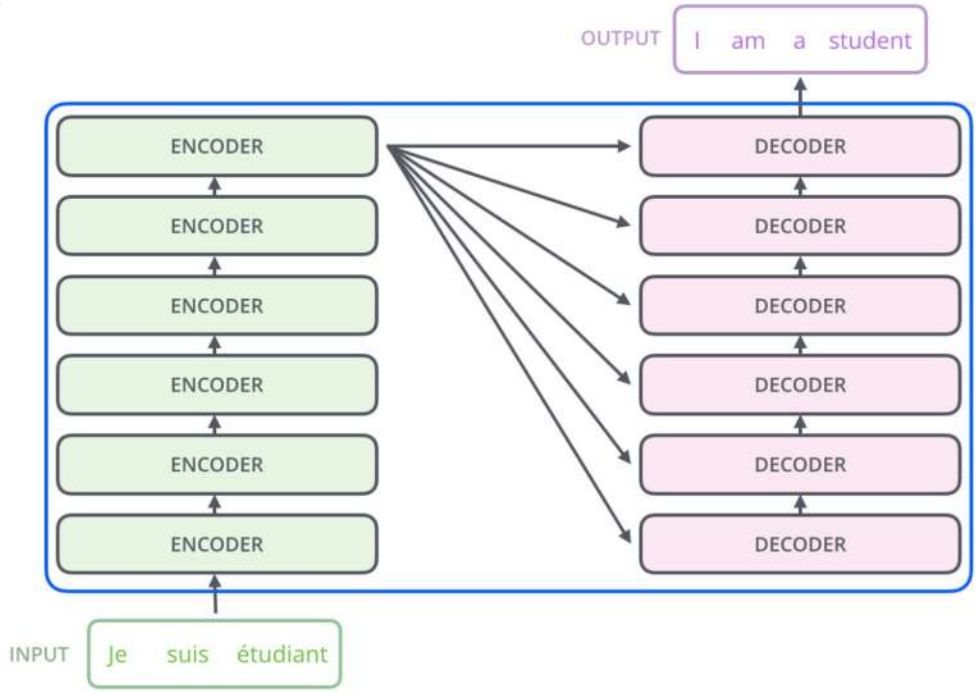
- More parallelizable -> Faster training time
- Better at longer sequence


Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



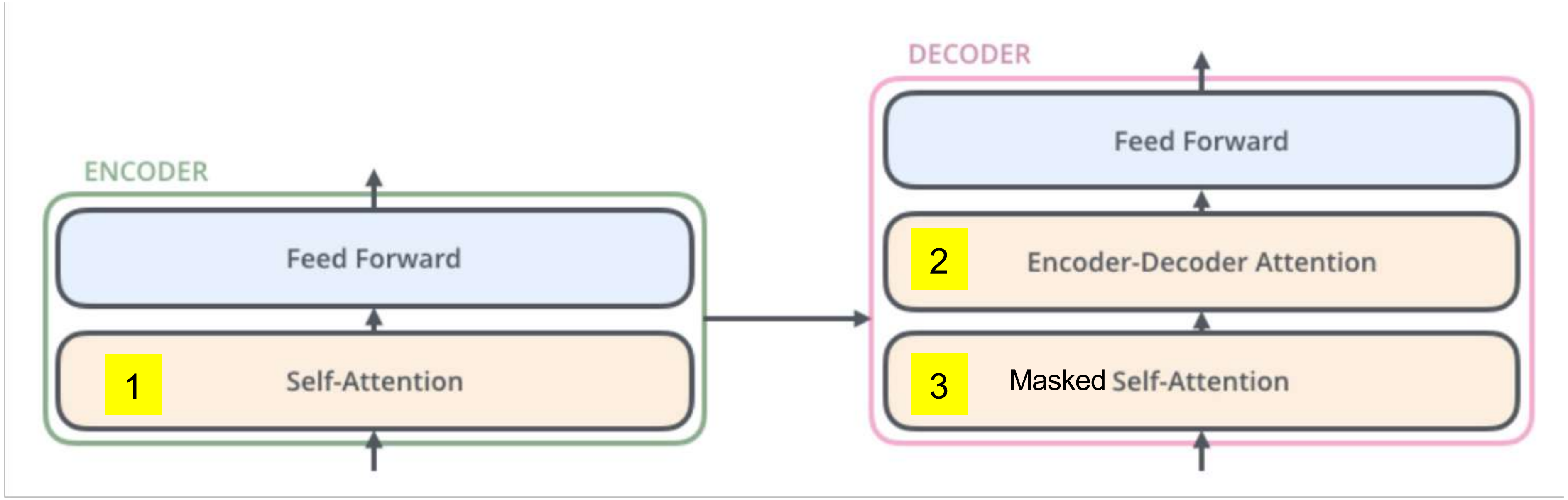

=




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

# In each layer..

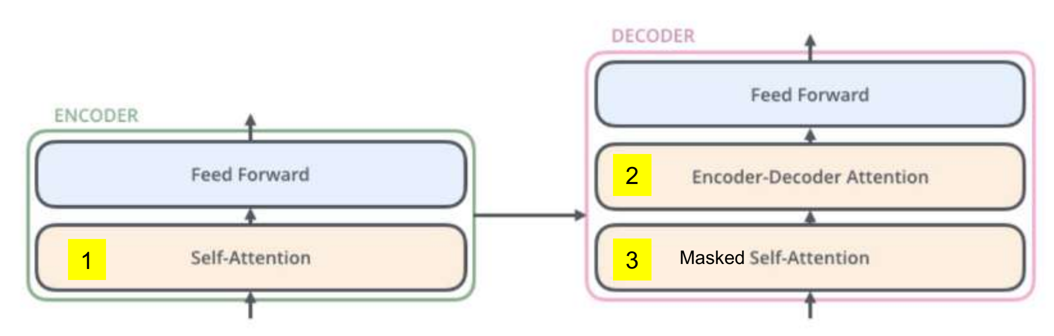


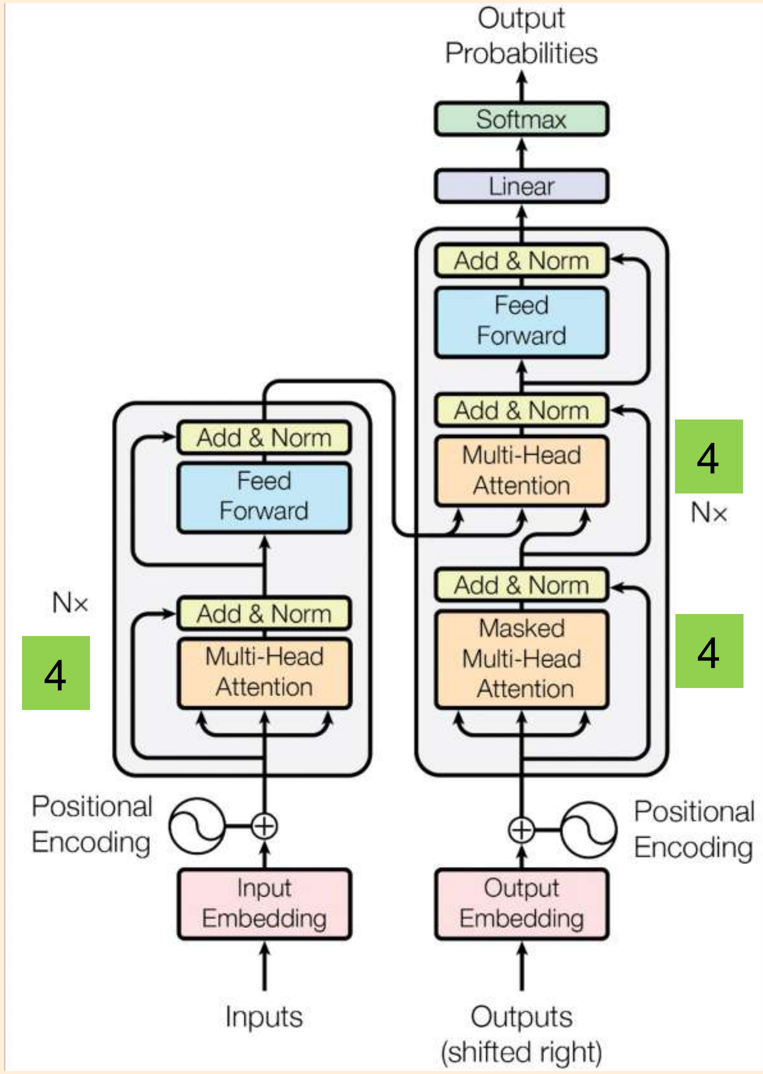

http://jalammar.github.io/illustrated-transformer/

# Scaled Dot-Product Attention

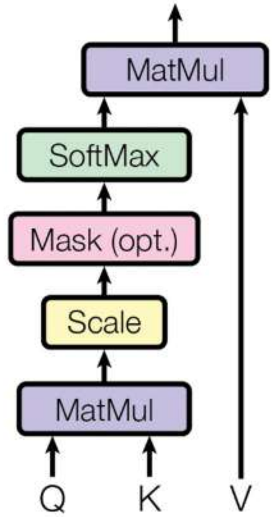
There are many variations used in the Transformer:

1. Self Attention
2. Encoder-decoder Attention
3. Masked Self Attention
4. Multi-headed Self Attention



Scaled Dot-Product Attention

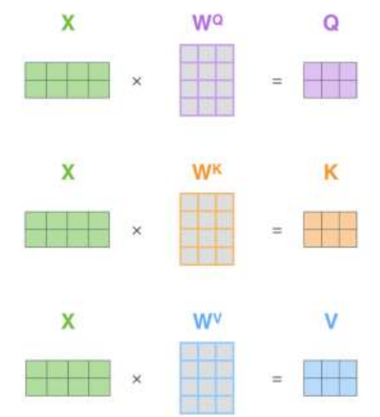



# 1) Self Attention

Q, K, and V are all from the same input.

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

where the query, keys, values, and output are all vectors and $d_k$ is a number.

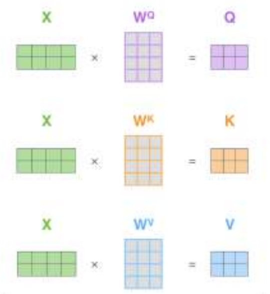


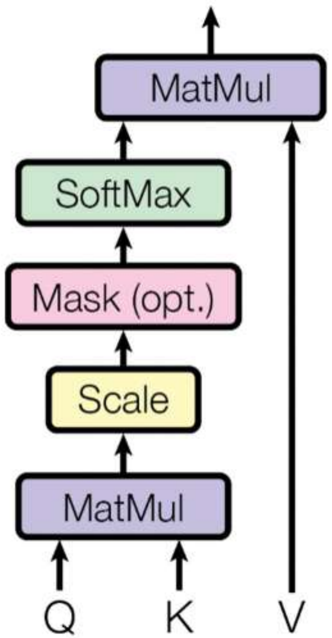

Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

My name is Sam → ฉัน ชื่อ แซม

# 2) Encoder-Decoder Attention

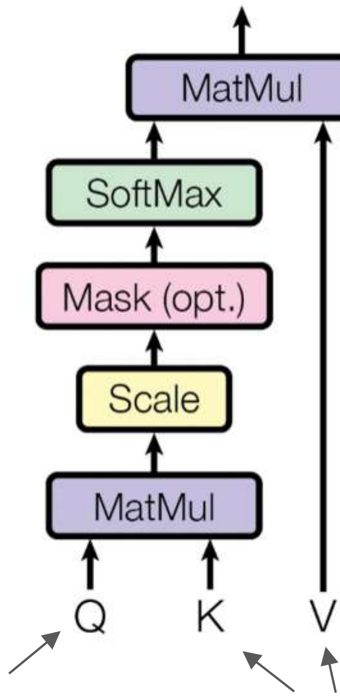
Self Attention

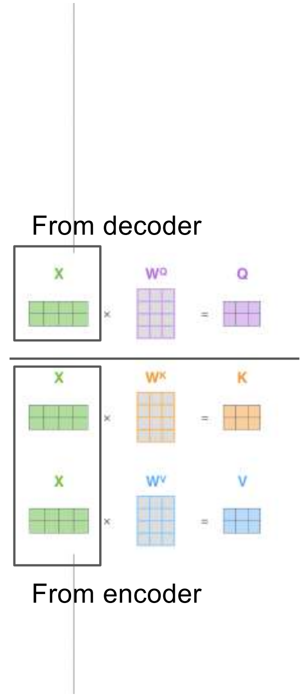





ALL from same input

Encoder-Decoder Attention
(or Cross Attention)




From decoder

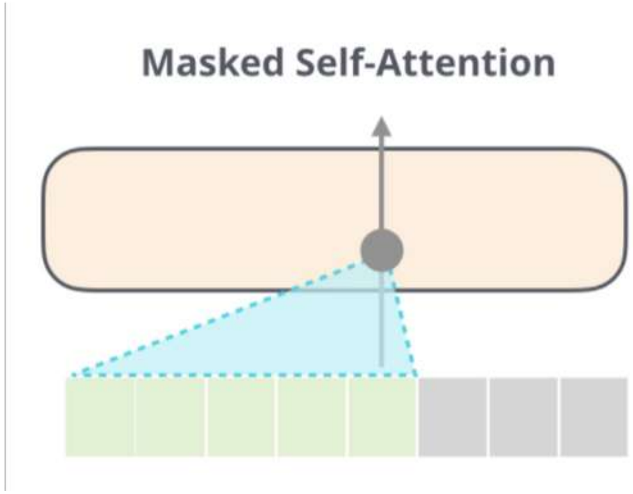
From encoder

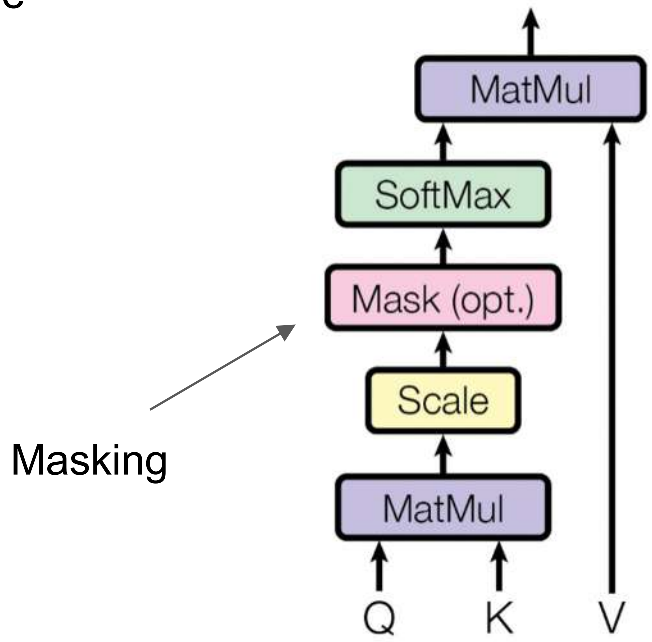
My name is Sam → ฉัน ชื่อ แซม

# 3) Masked Self Attention

**Prevent** the model from **seeing into the future** to preserve the autoregressive property.

Used in *decoder* only (so the model can't see the answer).



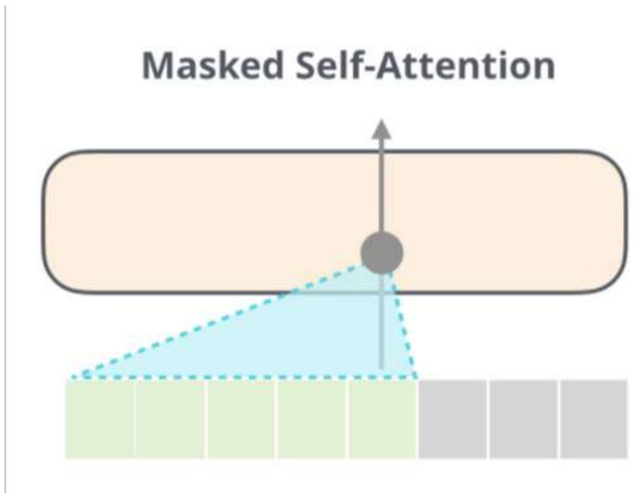



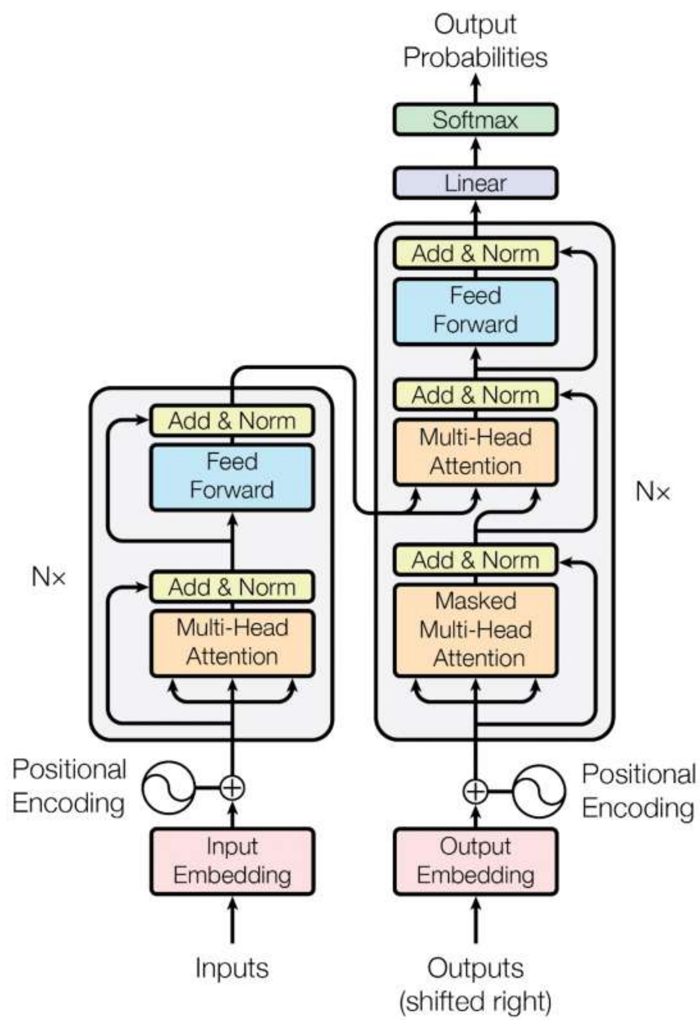

ALL from same input

# Masked Self Attention

**Prevent** the model from **seeing into the future** to preserve the autoregressive property.

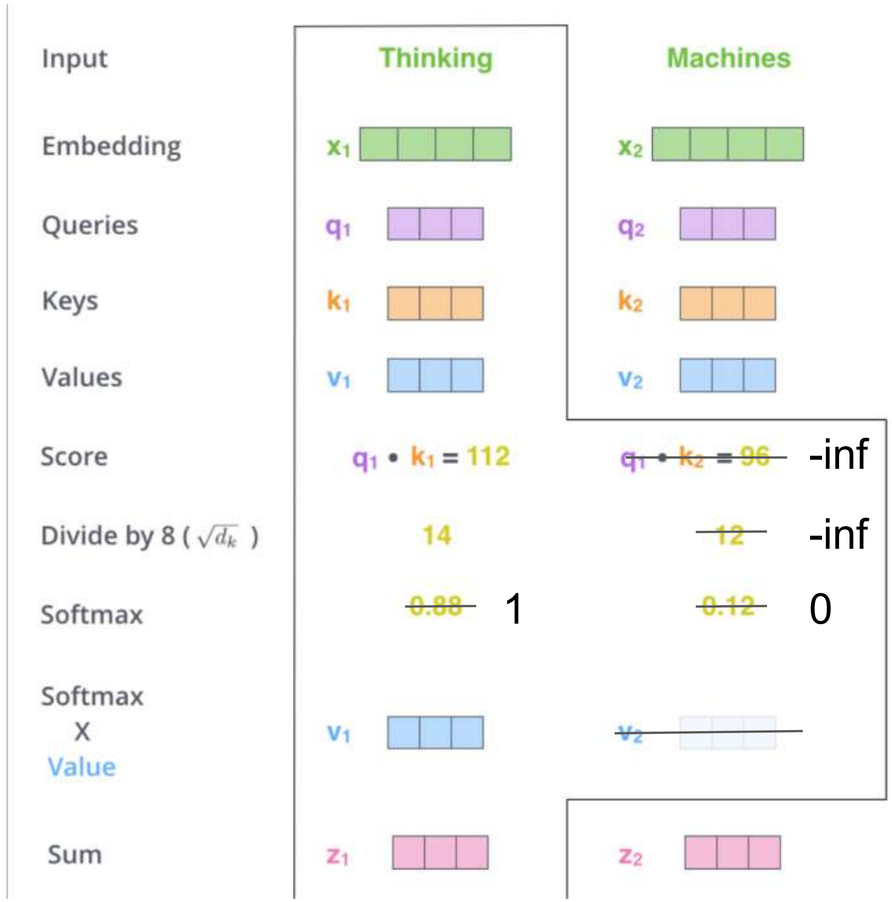
Used in *decoder* only (so the model can't see the answer).




I like dogs <eos> <bos> ฉัน ชอบ ~~สุนัข~~

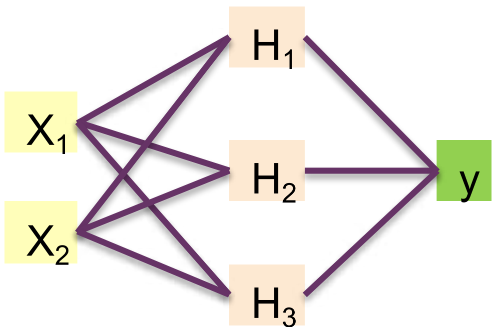
# Masked Self Attention

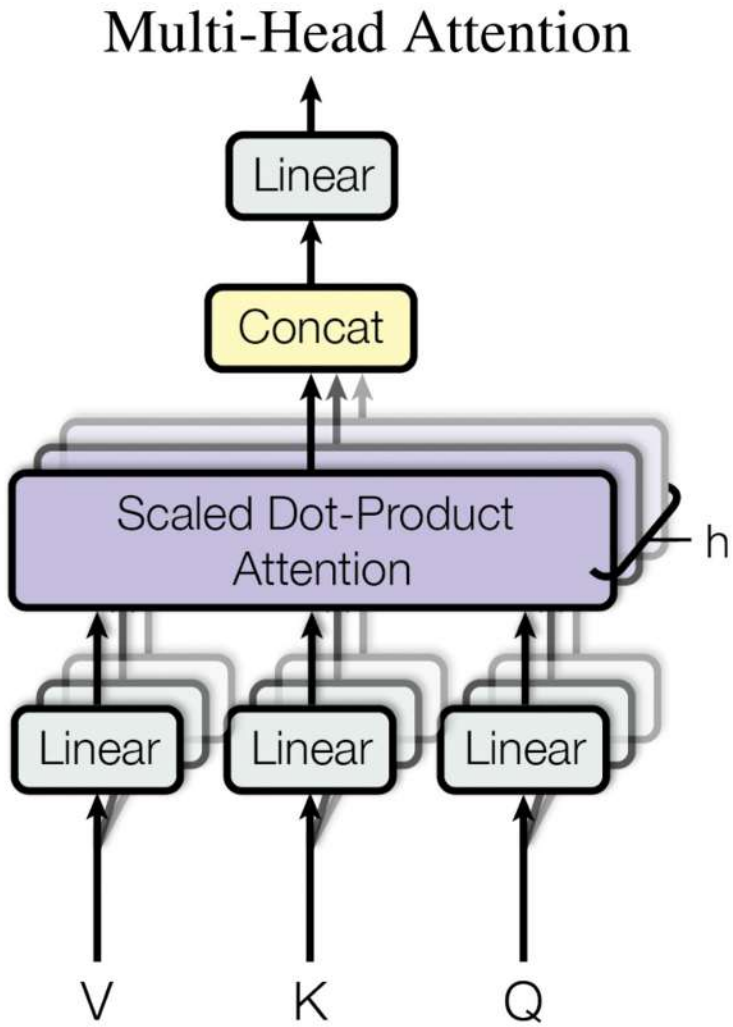


# 4) Multi-head Self Attention

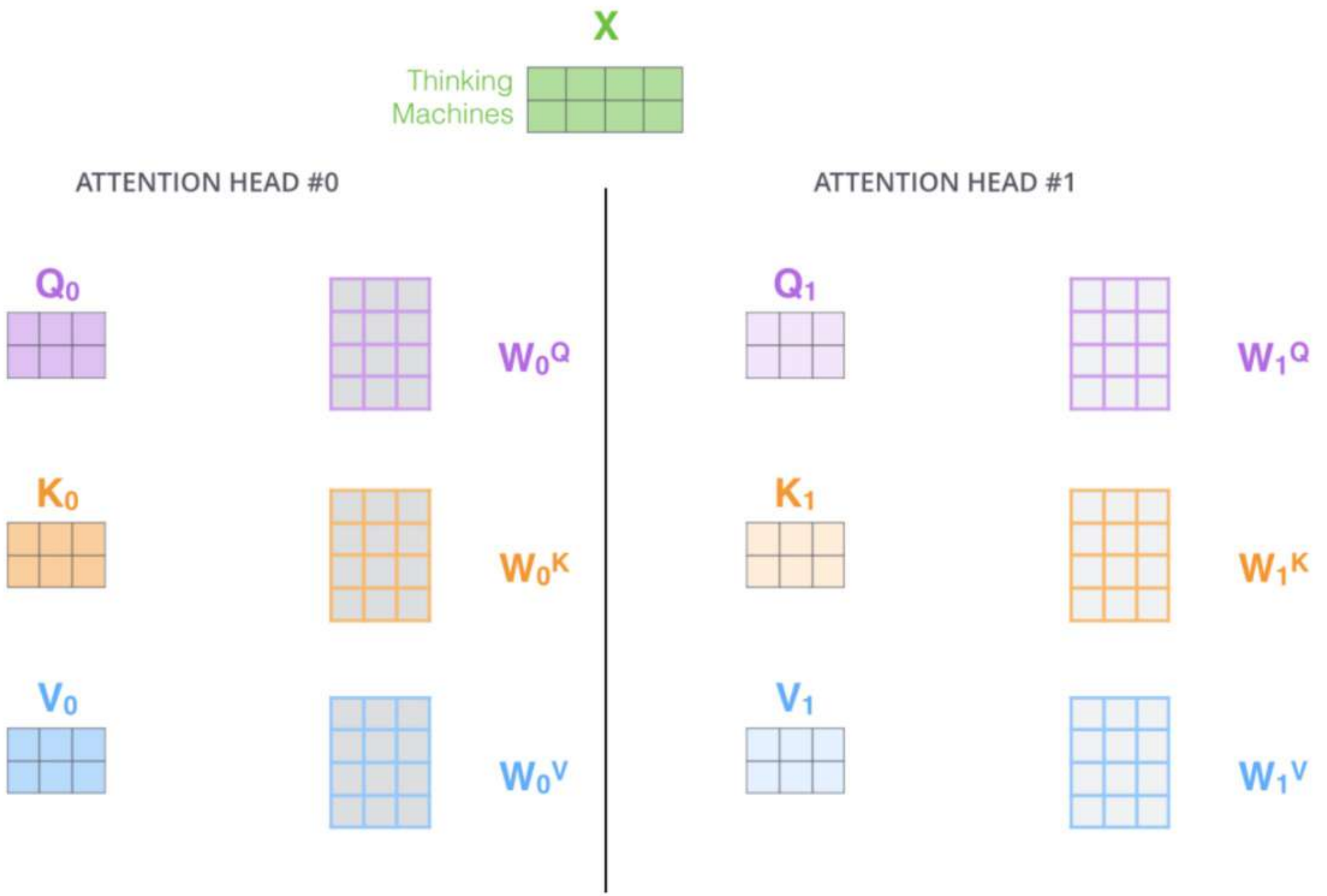
Multi-head attention allows the model to jointly attend to information from different representations.

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



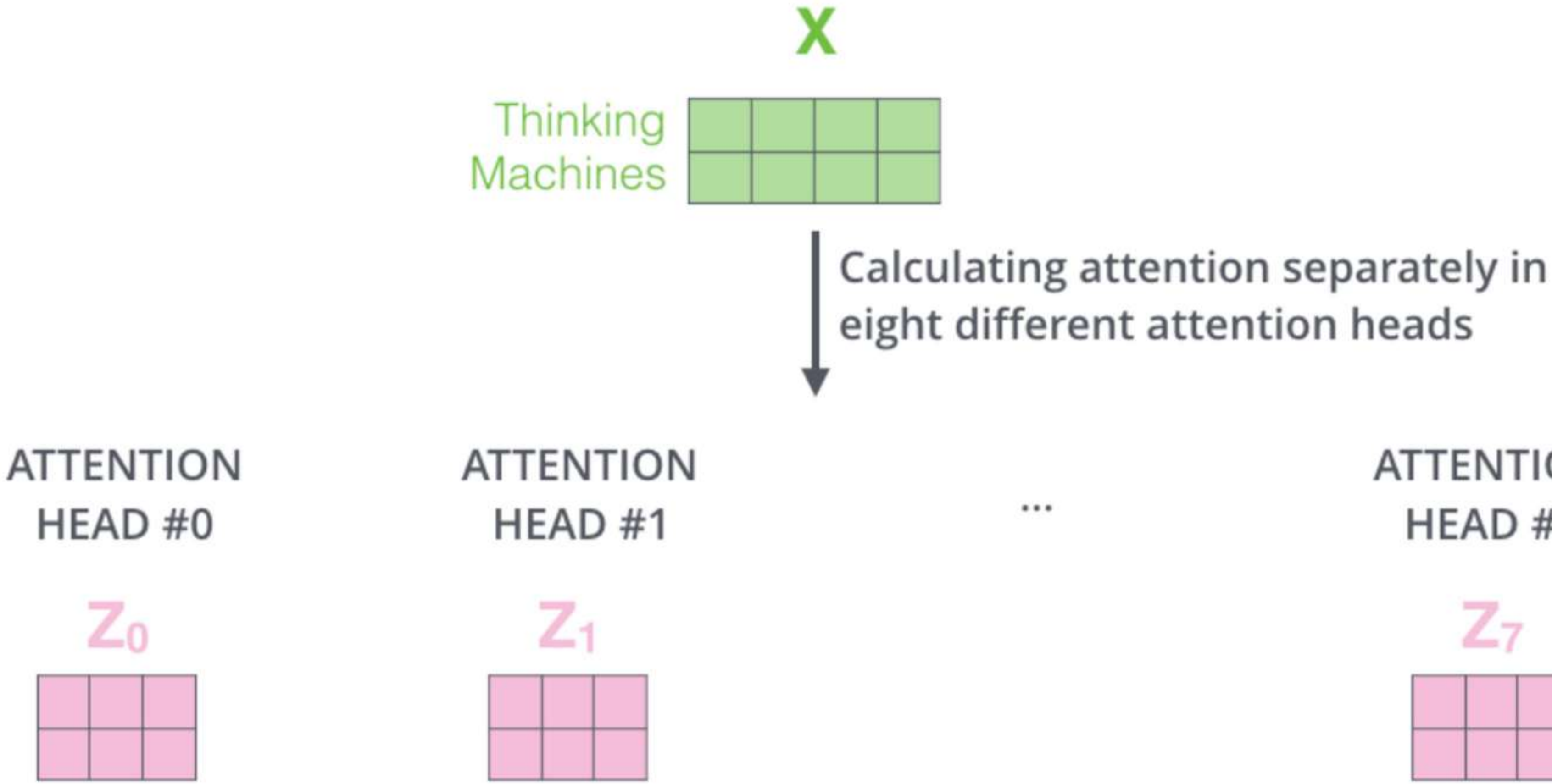

Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



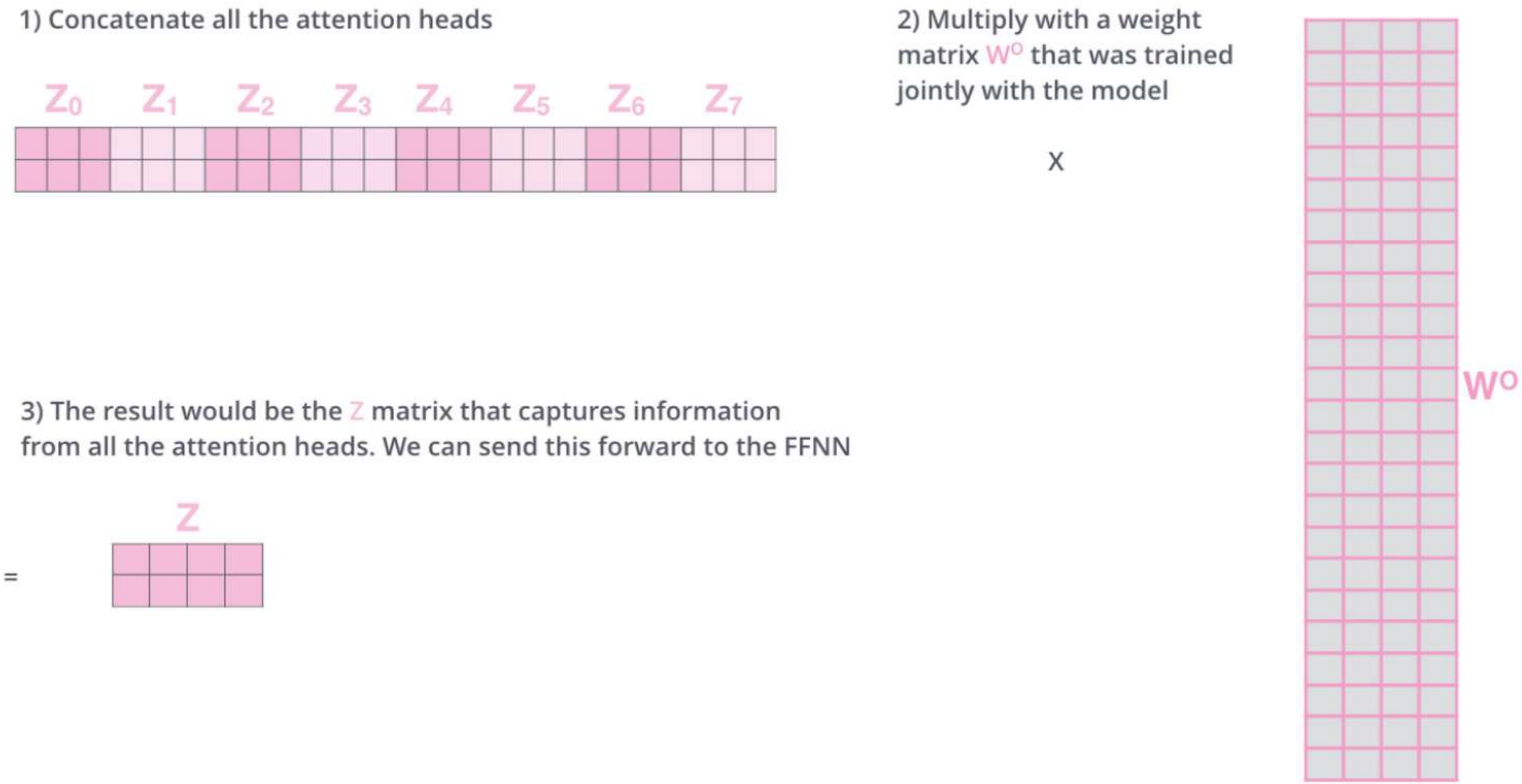

With multi-headed attention, we maintain separate Q/K/V weight matrices for each head resulting in different Q/K/V matrices. As we did before, we multiply X by the WQ/WK/WV matrices to produce Q/K/V matrices.



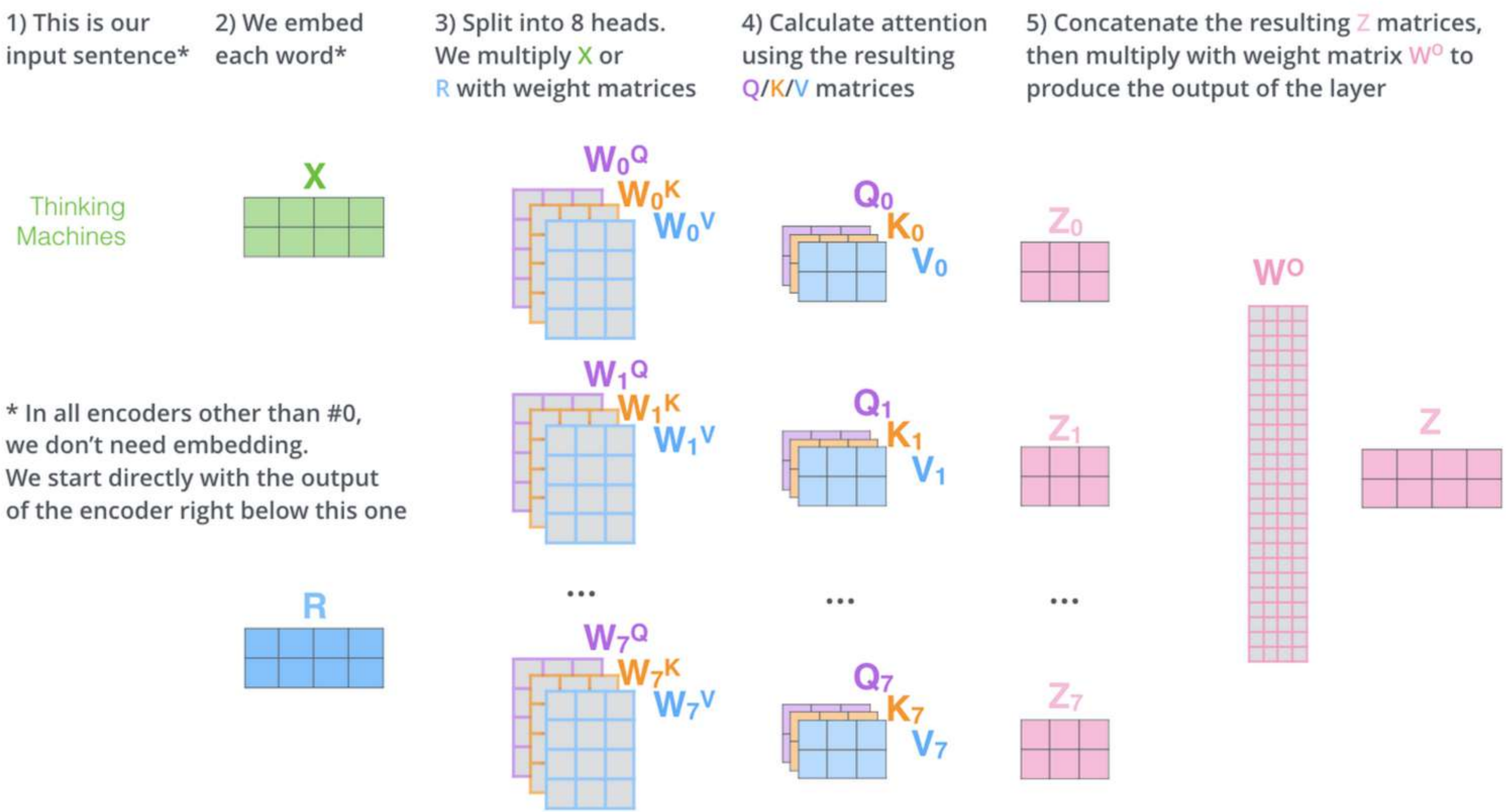
http://jalammar.github.io/illustrated-transformer/




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



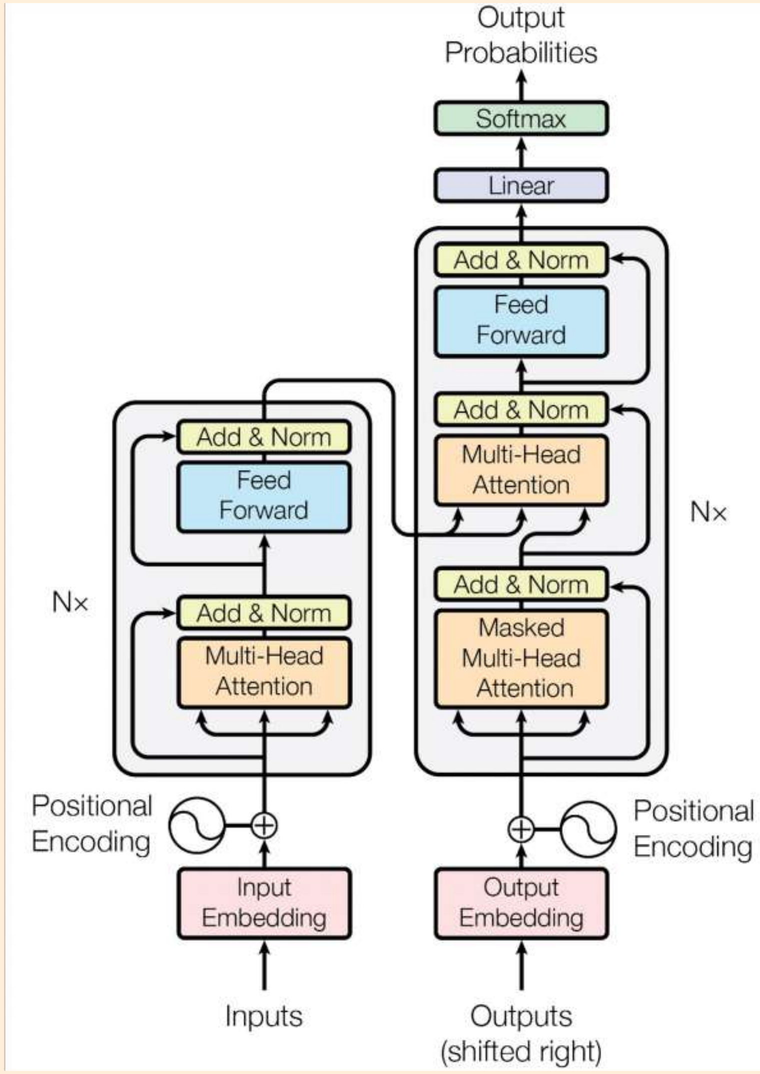

http://jalammar.github.io/illustrated-transformer/




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

# Positional Encodings

The black cat fights the white cat.



# Positional Encodings

Without RNN, the model **cannot** make use of the *order* of the input sequence, e.g. the first, second, or third token.

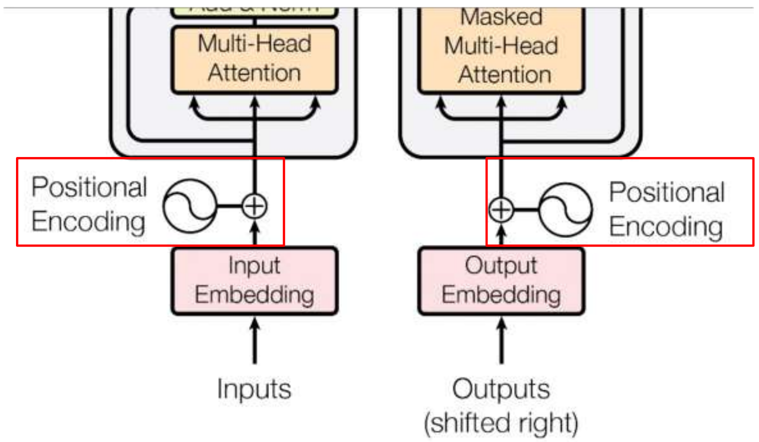



Figure 1: The Transformer - model architecture.

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

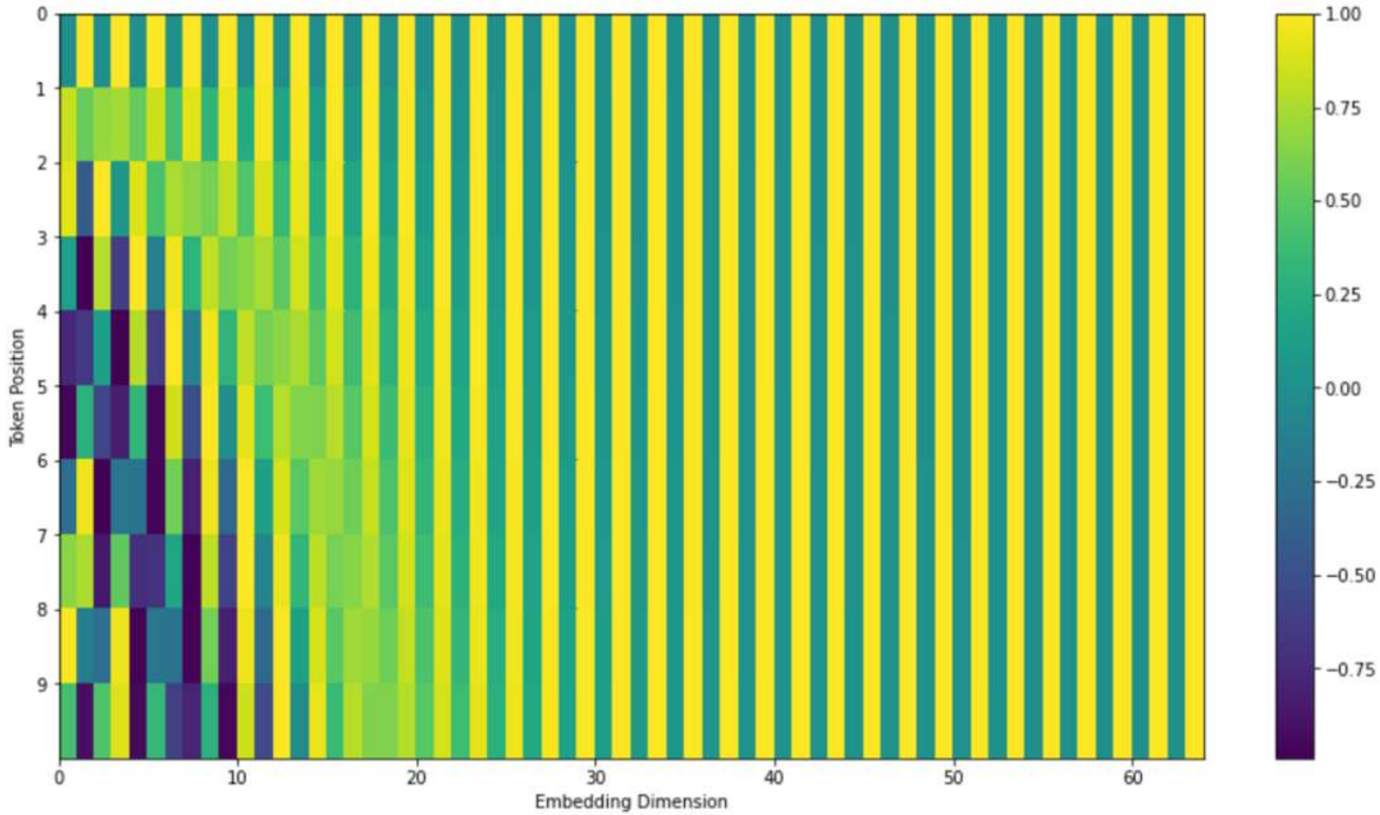
*pos* is the token position, *i* is the dimension

The black cat fights the white cat.

Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

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

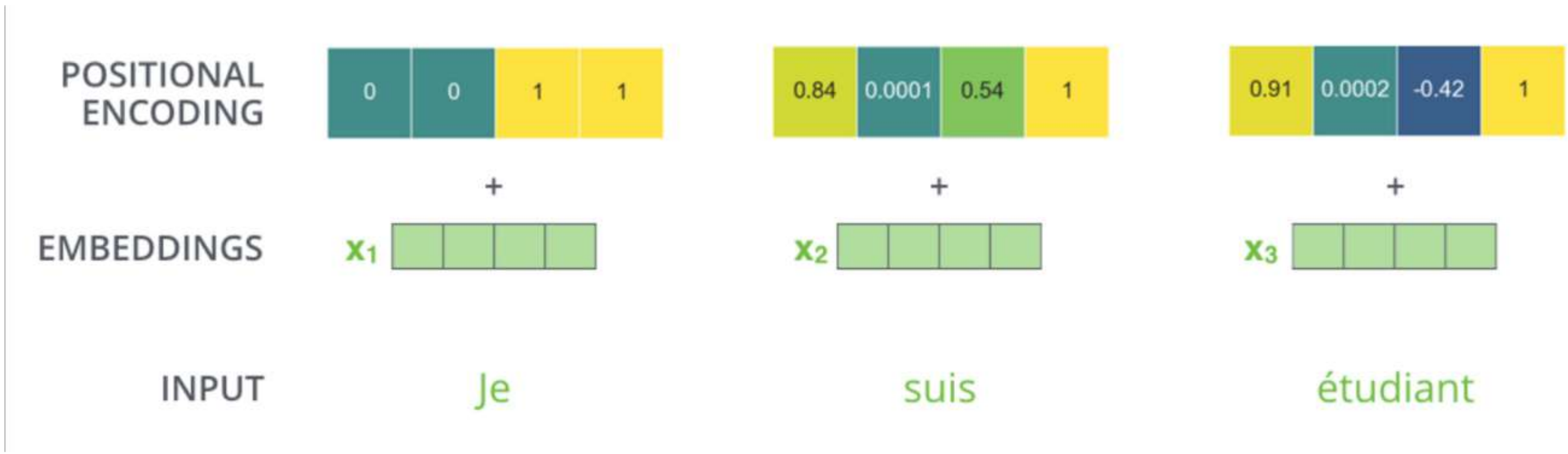
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

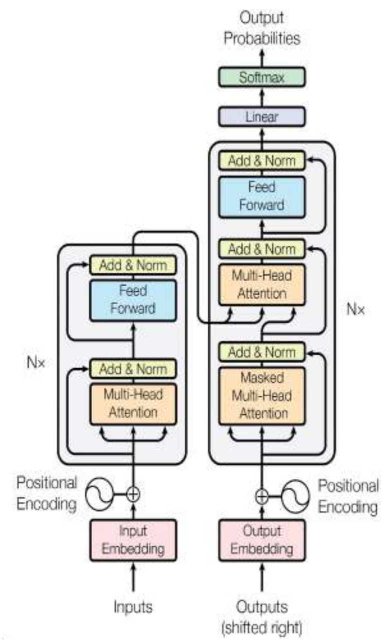





Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

i = 4




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

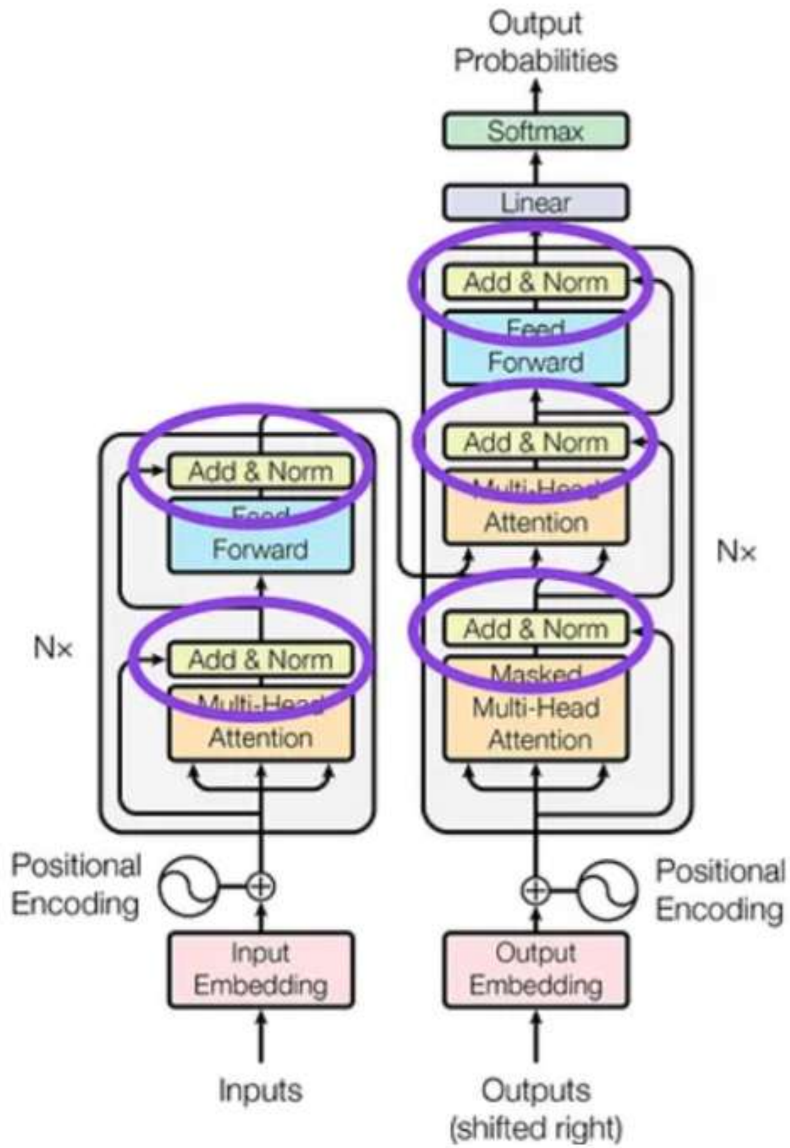
# Types of positional encoding

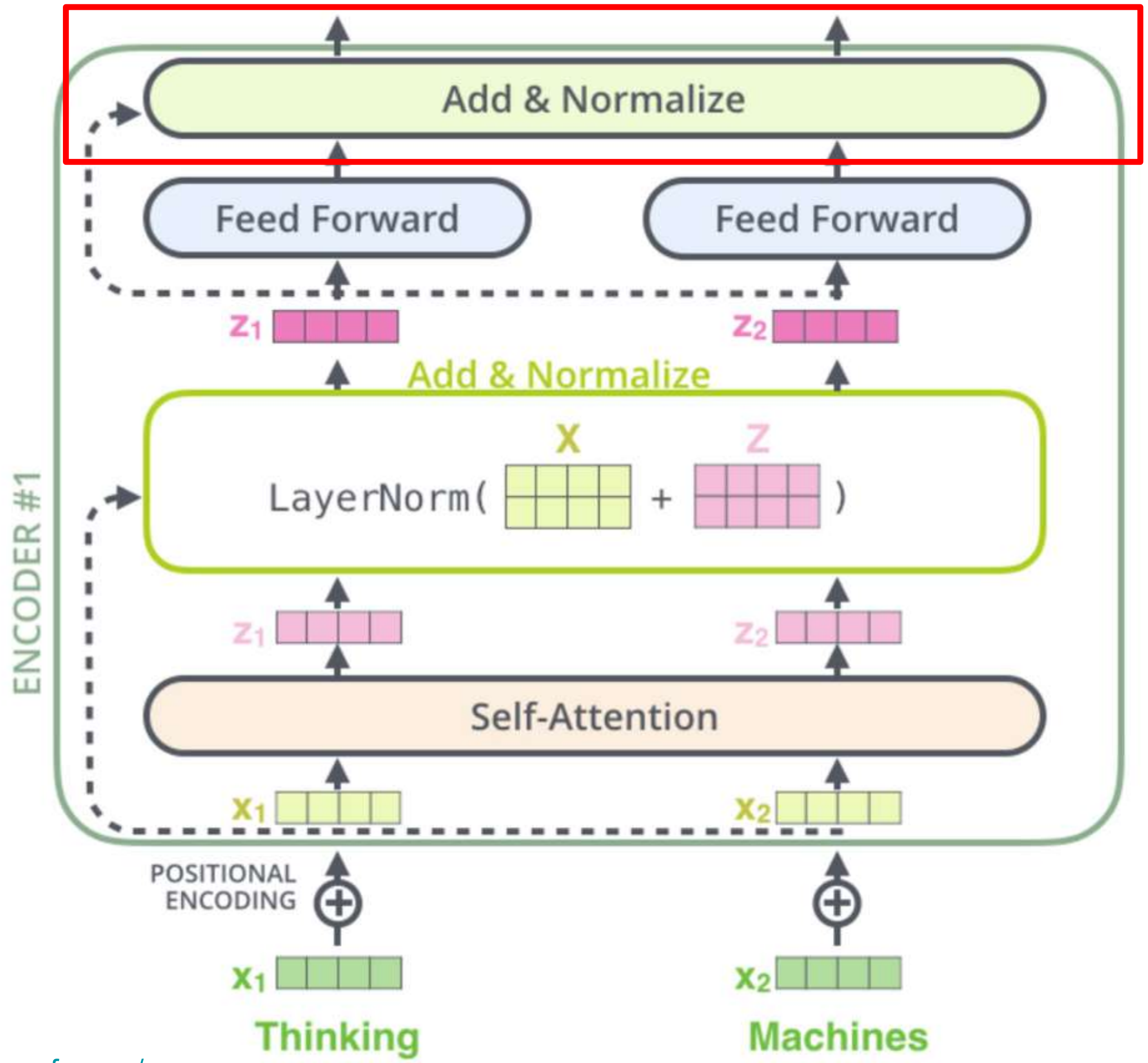
1. Absolute Positional encoding
   - 1.1) Fixed encoding (Transformer)
     - E.g. sinusoidal forms
   - 1.2) Learned encoding (GPT)

2. Relative Positional encoding (GPT NeoX)

| | Token 1 | Token 2 | Token 3 | Token 4 | Token 5 | Token 6 |
|---|---|---|---|---|---|---|
| Absolute | 1 | 2 | 3 | 4 | 5 | 6 |
| Relative | -2 | -1 | 0 | 1 | 2 | 3 |

# Layer Normalization

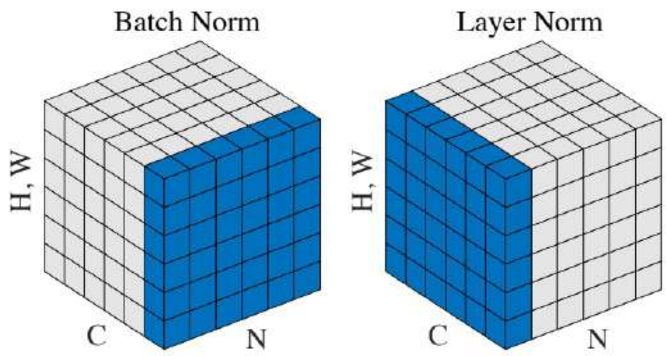





http://jalammar.github.io/illustrated-transformer/

# Batch norm. (each feature (channel)) vs Layer norm. (each word)

https://medium.com/@sachinsoni600517/layer-normalization-in-transformer-1a2efbff8b85




$$\text{Normalized Value} = \frac{\text{Original Value} - \mu}{\sigma}$$

$$\text{Final Value} = (\text{Normalized Value} \times \gamma) + \beta$$

| $f_1$ | $f_2$ | $z_1$ | $z_2$ | $z_3$ |
|---|---|---|---|---|
| 2 | 3 | 7 | 5 | 4 |
| 1 | 1 | 2 | 3 | 4 |
| 5 | 4 | 1 | 2 | 3 |
| 6 | 1 | 7 | 5 | 6 |
| 7 | 1 | 3 | 3 | 4 |

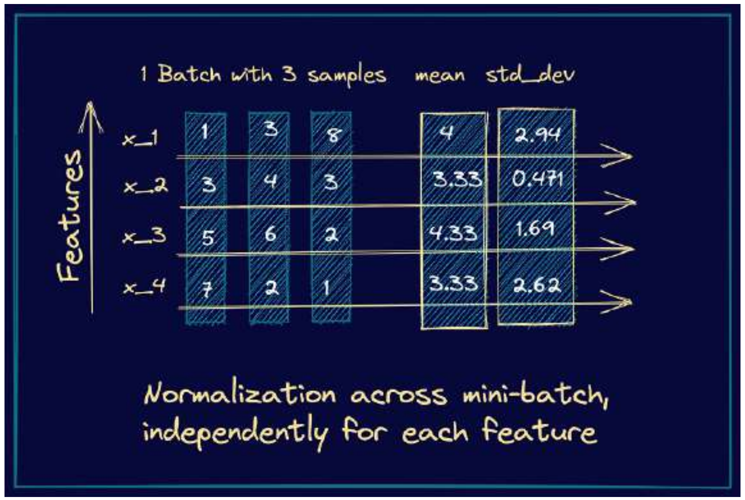
$\mu_1$ $\sigma_1$ $\mu_2$ $\sigma_2$ $\mu_3$ $\sigma_3$

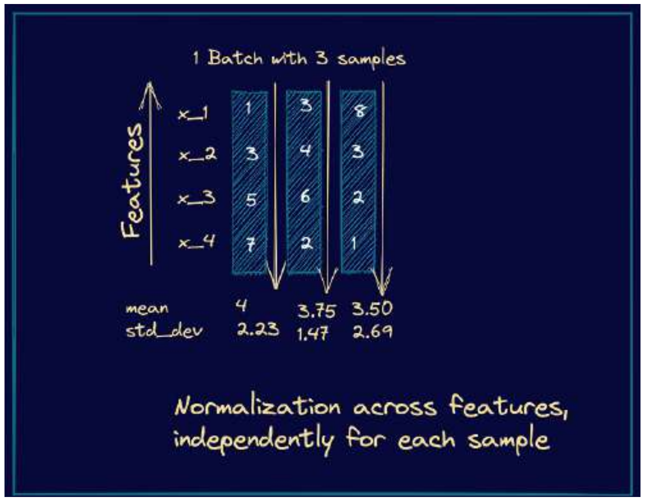
$$\frac{7-\mu_1}{\sigma_1} = \underline{0.86}\,\underset{(1)}{\gamma_1} + \underset{(0)}{\beta_1} = 0.86$$

$$\frac{2-\mu_1}{\sigma_1} = 0.71\,\gamma_1 + \beta_1 = 0.71$$

$$\frac{5-\mu_2}{\sigma_2} = -0.21\,\gamma_2 + \beta_2 = -0.21$$

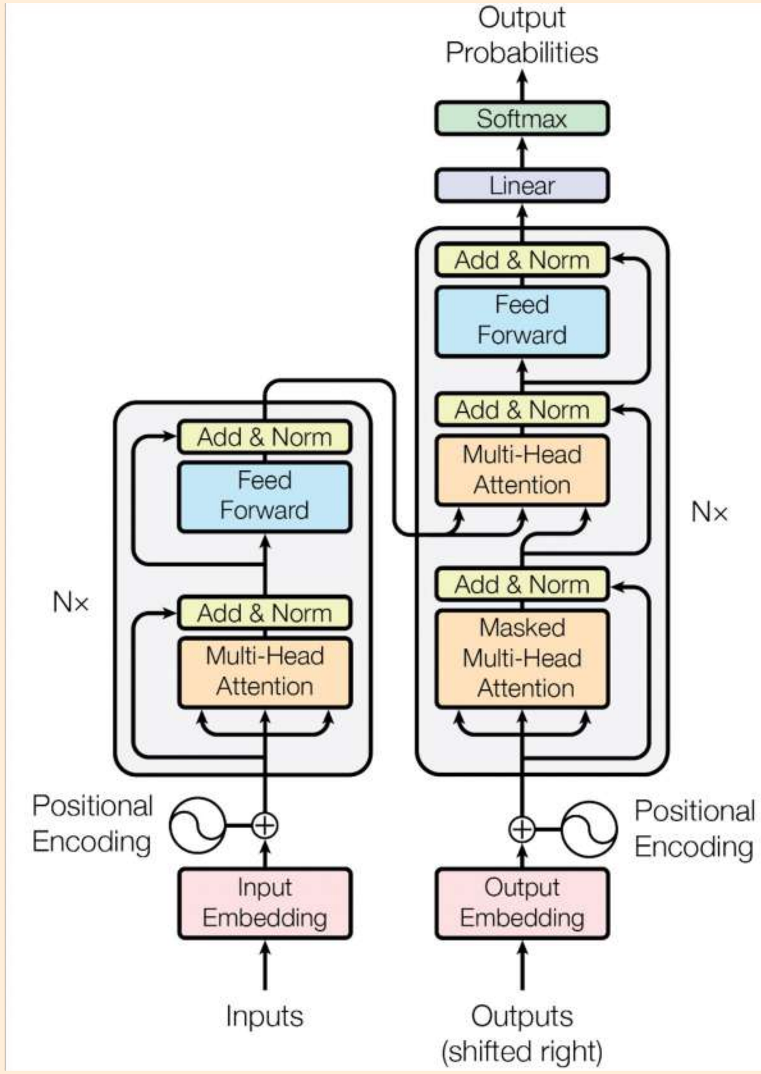
$$\frac{4-\mu_3}{\sigma_3} = 0.12\,\gamma_3 + \beta_3 = 0.12$$



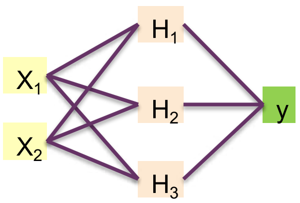



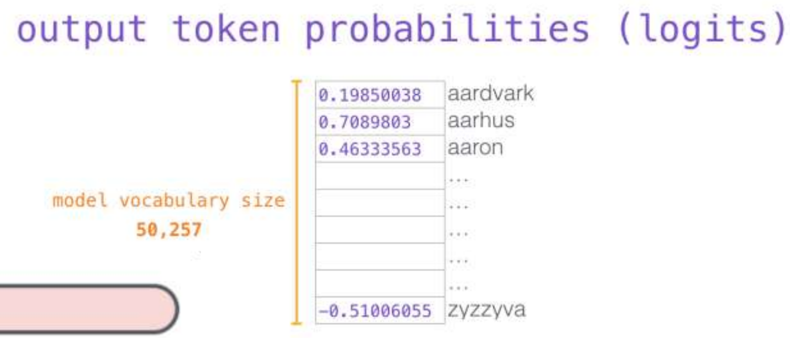
Done:

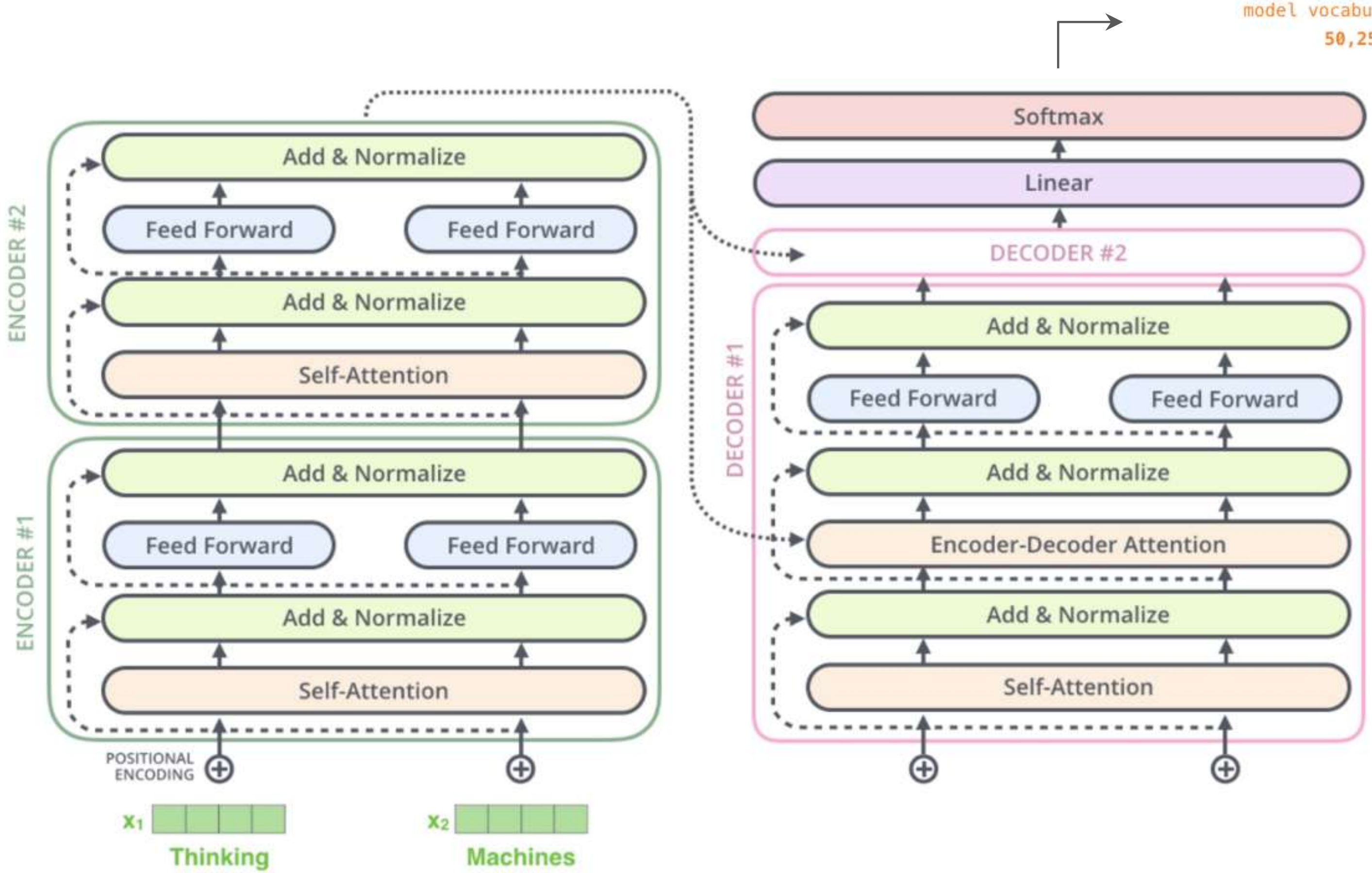
- Let's link encoder to decoder
- Then, generate final output

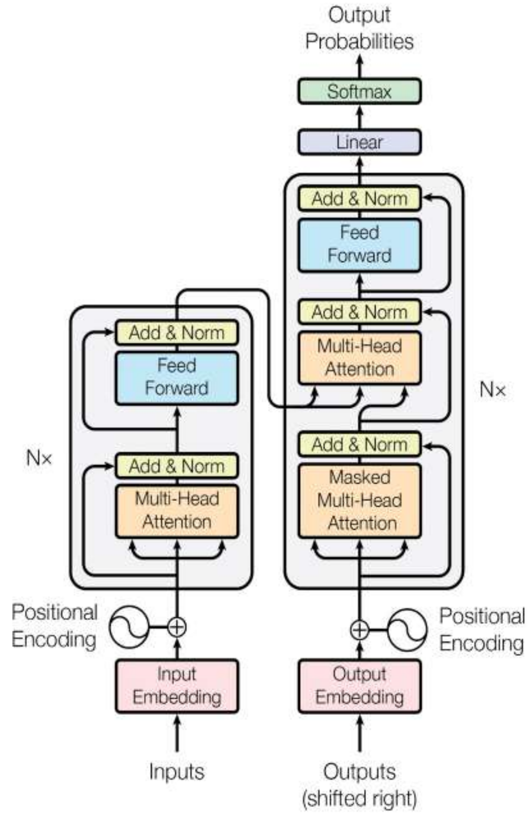


# Final output



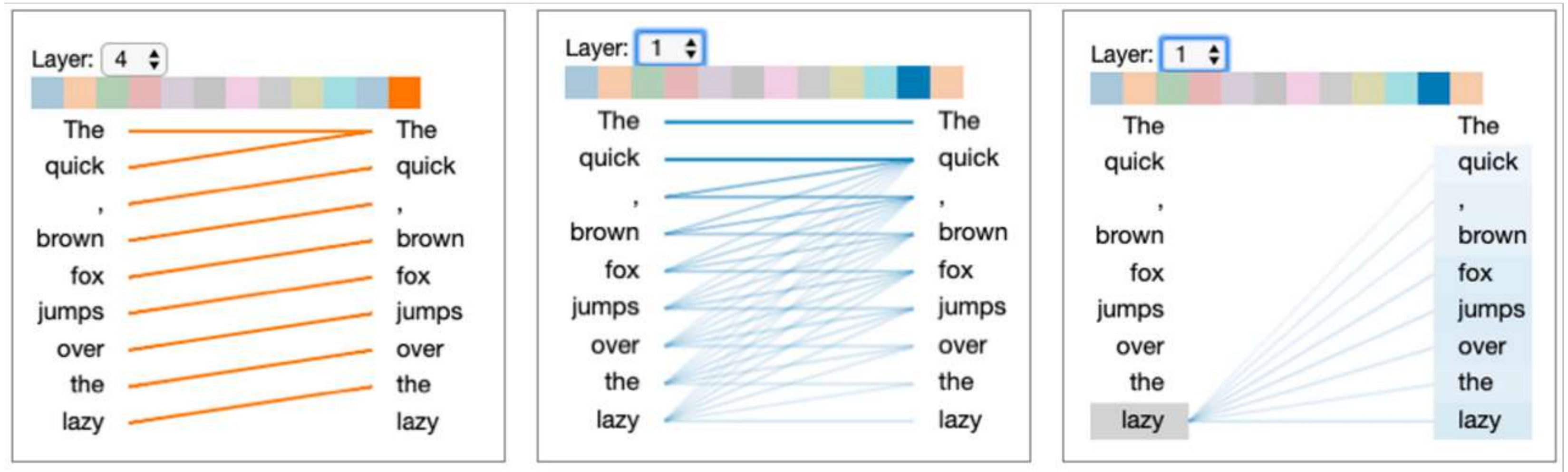



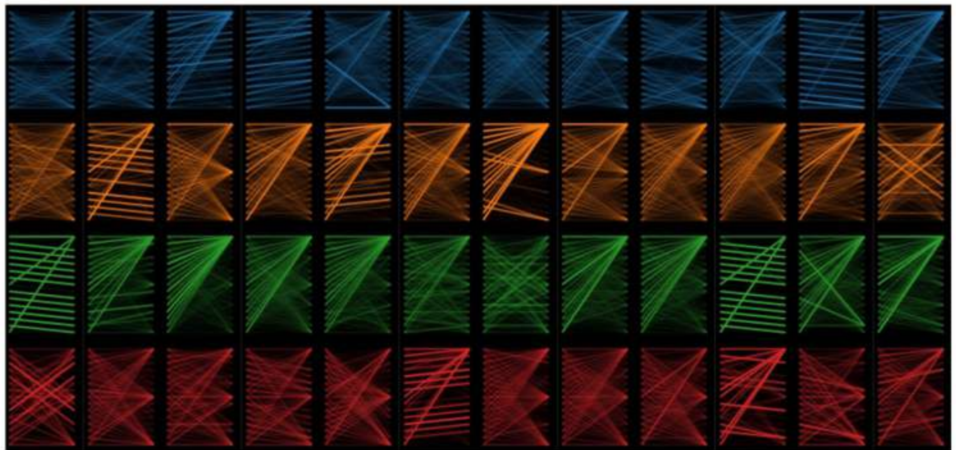



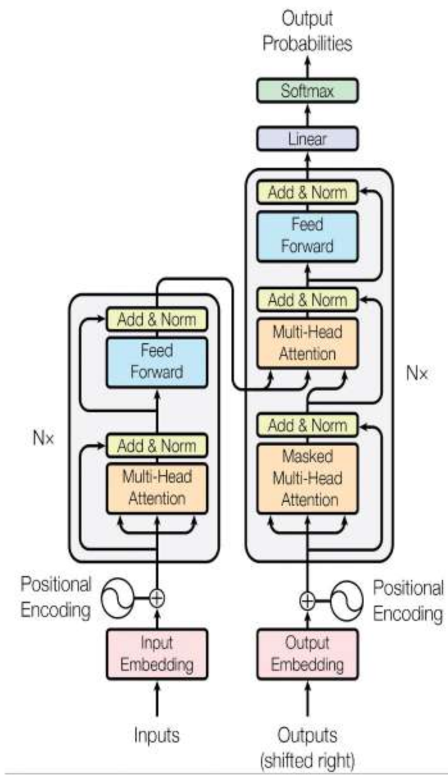




http://jalammar.github.io/illustrated-transformer/

# Visualizing Attention (N encoders & N decoders)





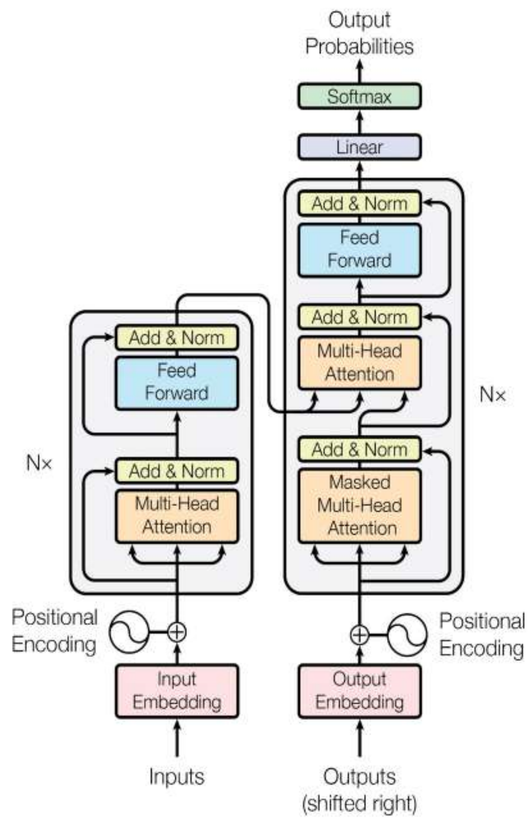

https://github.com/jessevig/bertviz

# Transformer-Based Models

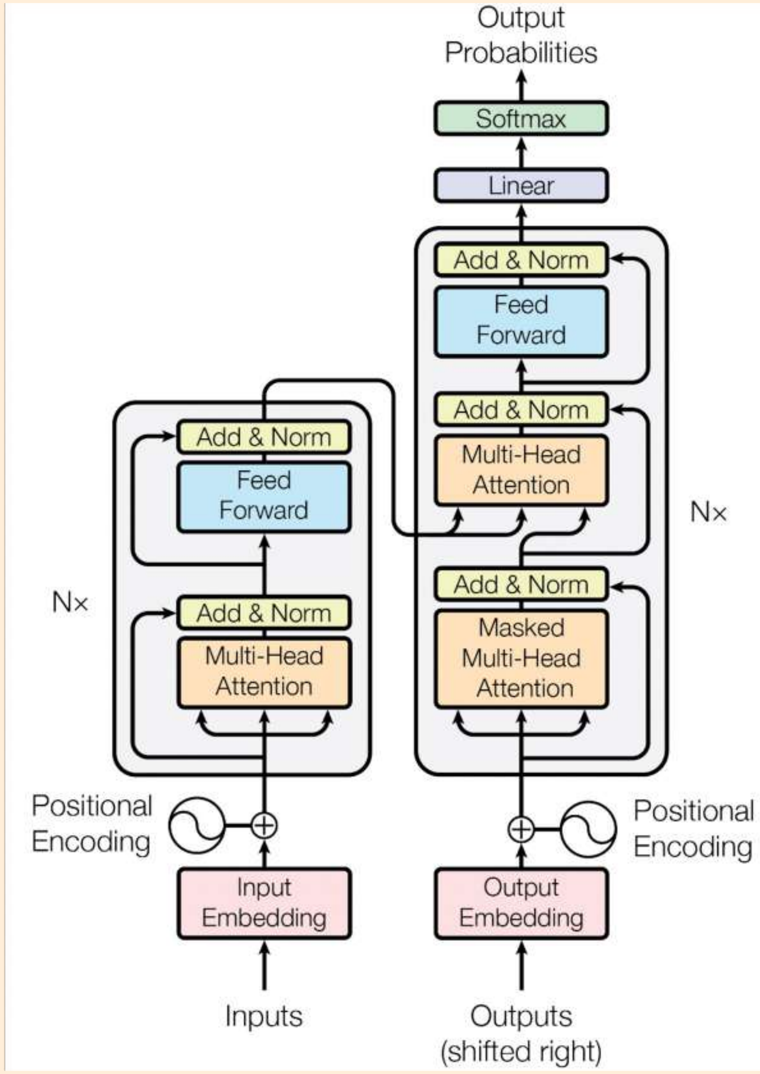
# Transformer-Based Models

All Transformer based

1. Decoder-based model: GPT
2. Encoder-based model: BERT
3. Encoder and Decoder: BART



# 1) Generative Pre-Training (OpenAI GPT)



# OpenAI GPT (Generative Pre-Training) [Radford, 2018]

## Improving Language Understanding by Generative Pre-Training


**Alec Radford**
OpenAI
alec@openai.com

**Karthik Narasimhan**
OpenAI
karthikn@openai.com

**Tim Salimans**
OpenAI
tim@openai.com

**Ilya Sutskever**
OpenAI
ilyasu@openai.com



### Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

# Generative Pre-Training (OpenAI GPT)

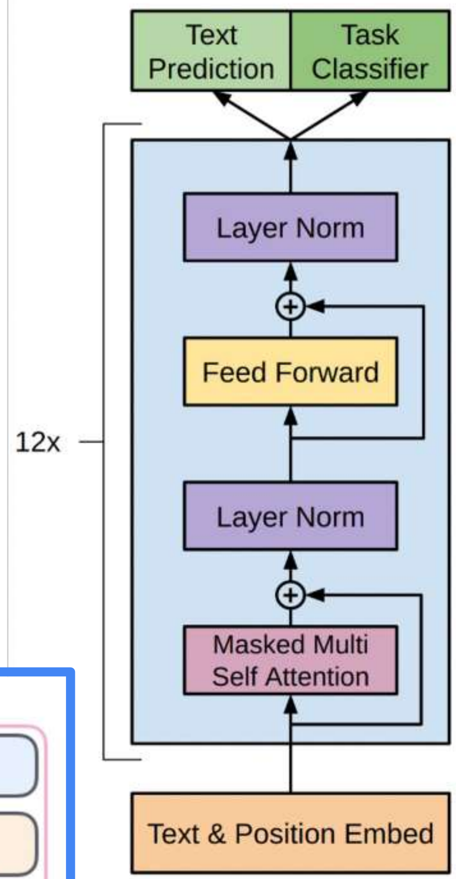
A model comprises of Transformer decoders only.

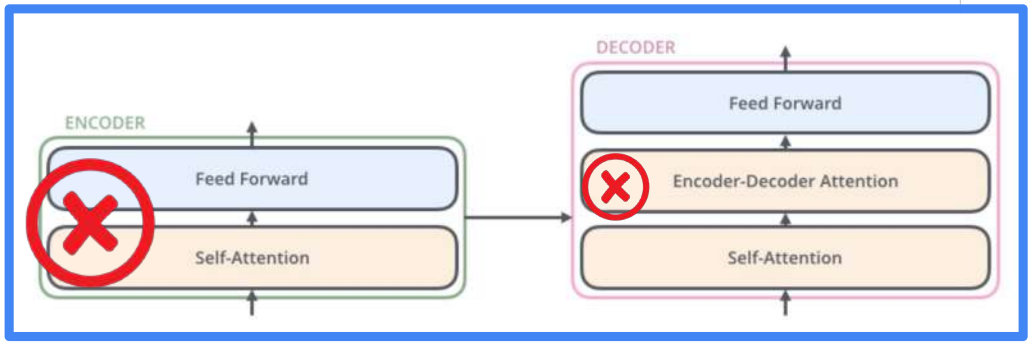
The framework consists of two stages:

1. Unsupervised pre-training
2. Supervised finetuning

Excels at text generation tasks

Try it out:
https://transformer.huggingface.co/



# Generative Pre-Training (OpenAI GPT)

1) Unsupervised pre-training

Given a large unlabeled corpus $\mathcal{U} = \{u_1, \ldots, u_n\}$ the model's objective is to maximize the following likelihood (also called "Language Modelling").

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

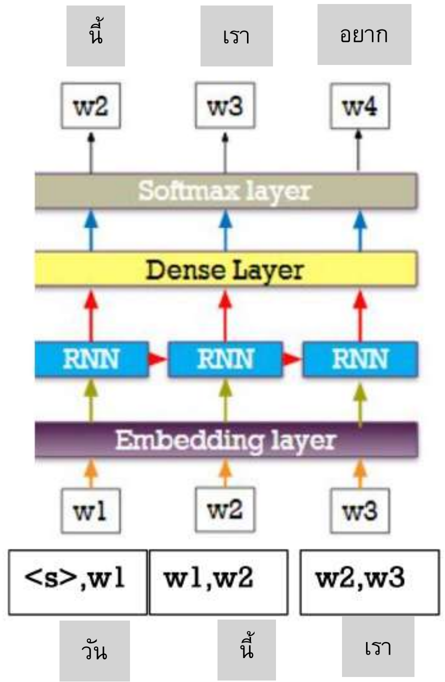
where *k* is the context length and the conditional probability *P* is modeled using a neural network with parameters $\boldsymbol{\Theta}$.

# Language Model (LM)

- It is the model that aims to predict next word based on the given previous words.
- So, the model can understand grammar & context.

วันนี้เราอยากจะกินข้าวมัน____

วัน นี้ เรา อยาก จะ กิน ข้าว มัน



# Generative Pre-Training (OpenAI GPT)

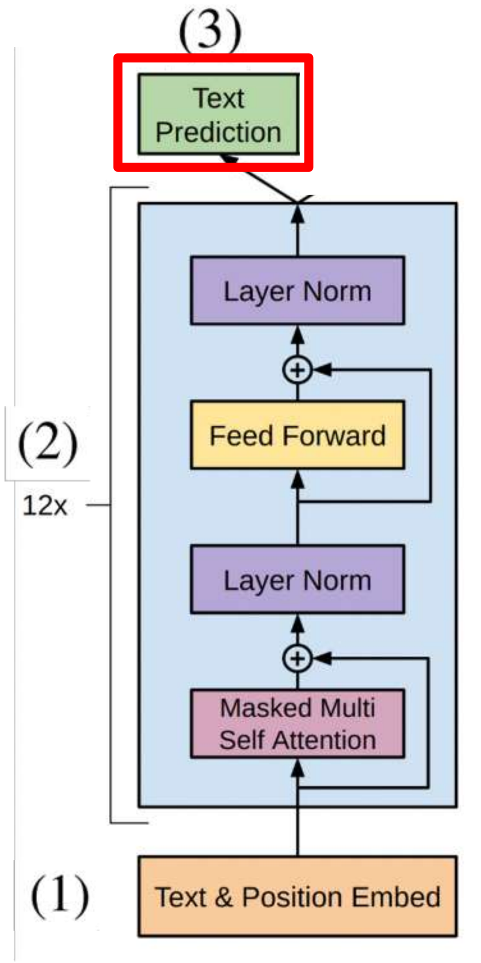
1) Unsupervised pre-training (next word prediction)

(1) $$h_0 = UW_e + W_p$$

(2) $$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$

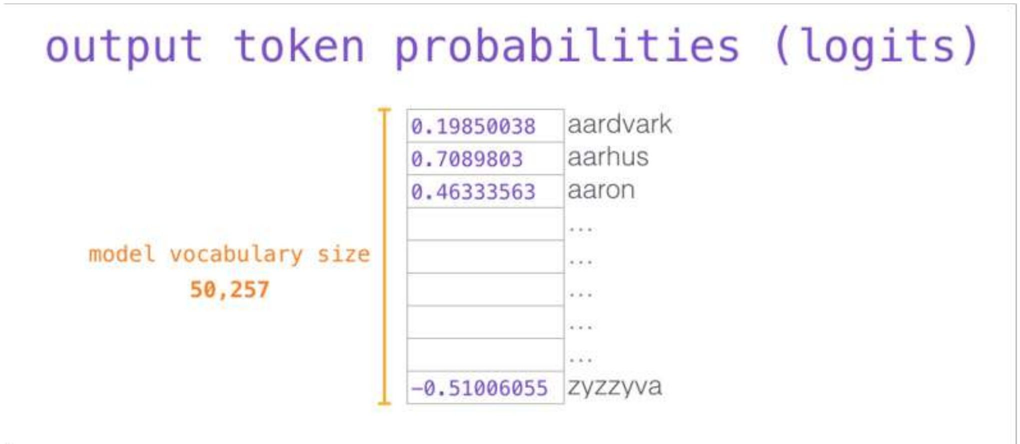
(3) $$P(u) = \texttt{softmax}(h_n W_e^T)$$

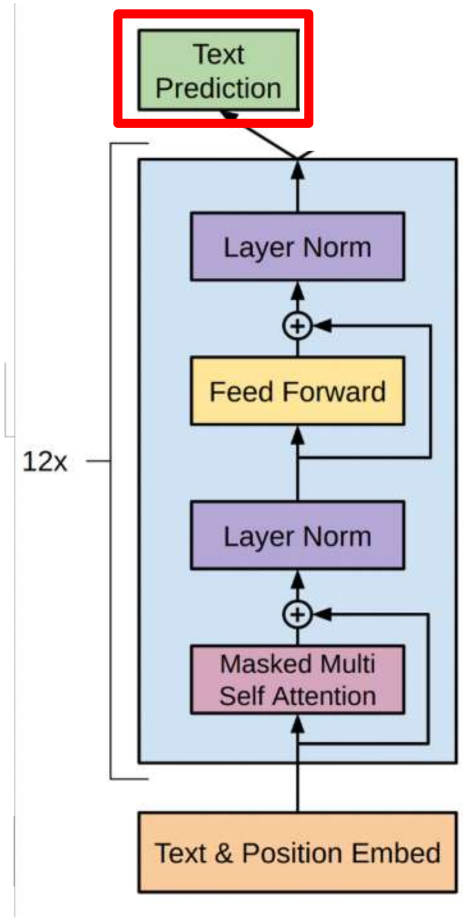
where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, $n$ is the number of layers, $W_e$ is the token embedding matrix, and $W_p$ is the position embedding matrix.



# Generative Pre-Training (OpenAI GPT)

1) Unsupervised pre-training (next word prediction)



# Generative Pre-Training (OpenAI GPT)

2) Supervised finetuning

We assume a labeled dataset C, where each instance consists of a sequence of input tokens ($x^1, \ldots, x^m$), along with a label y.

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

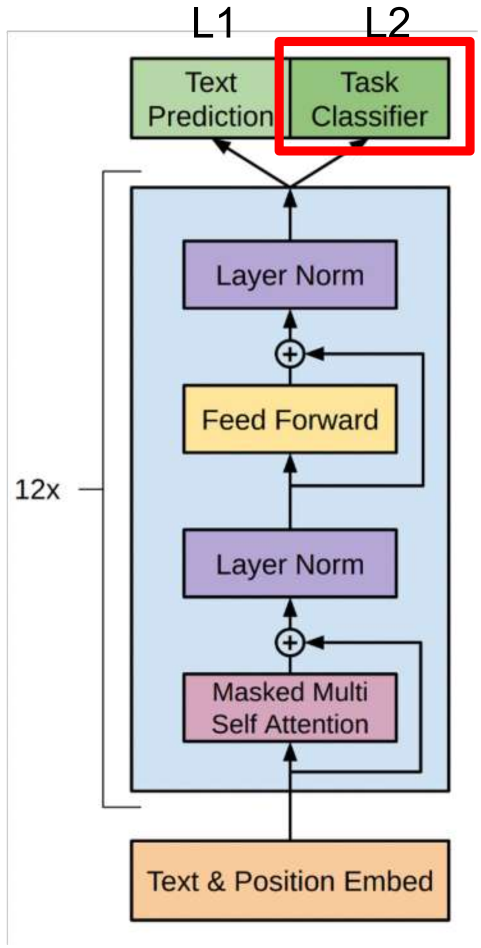
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

Total finetuning loss

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

classification

LM (unsupervised)



# Generative Pre-Training (OpenAI GPT)

2) Supervised finetuning

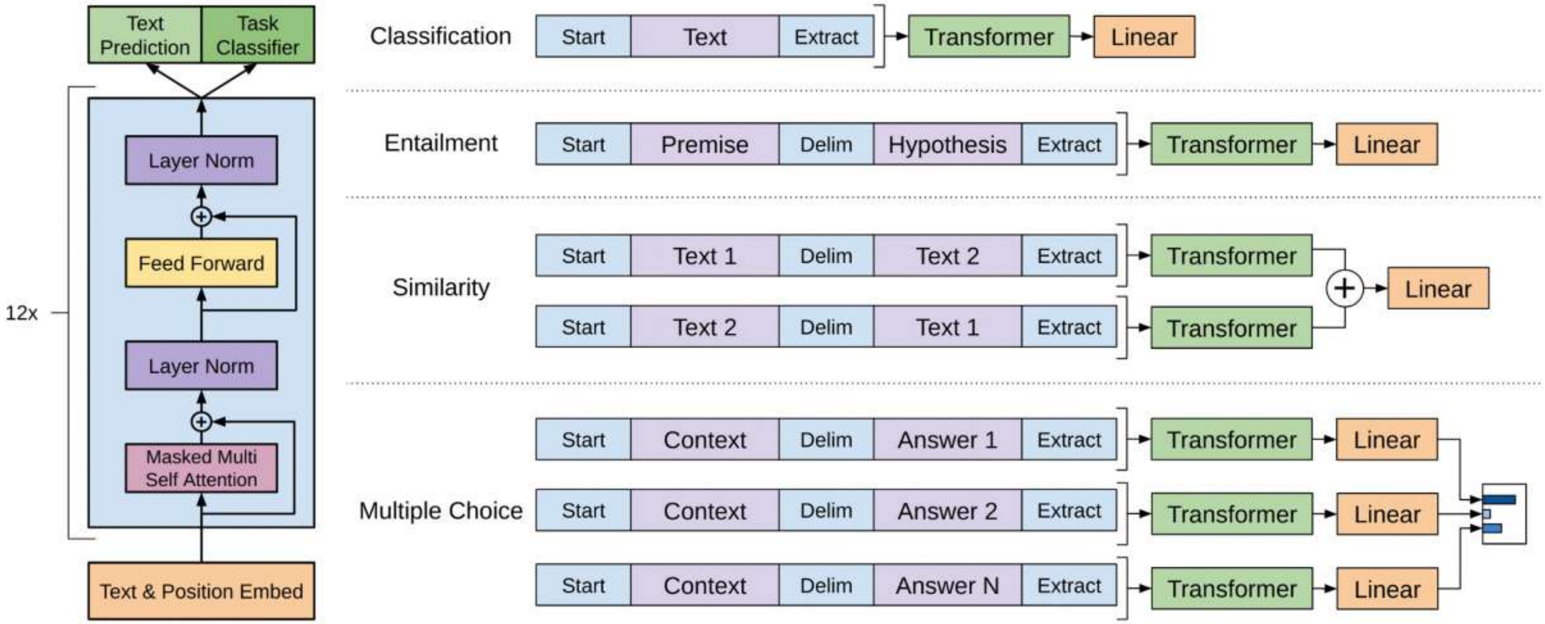
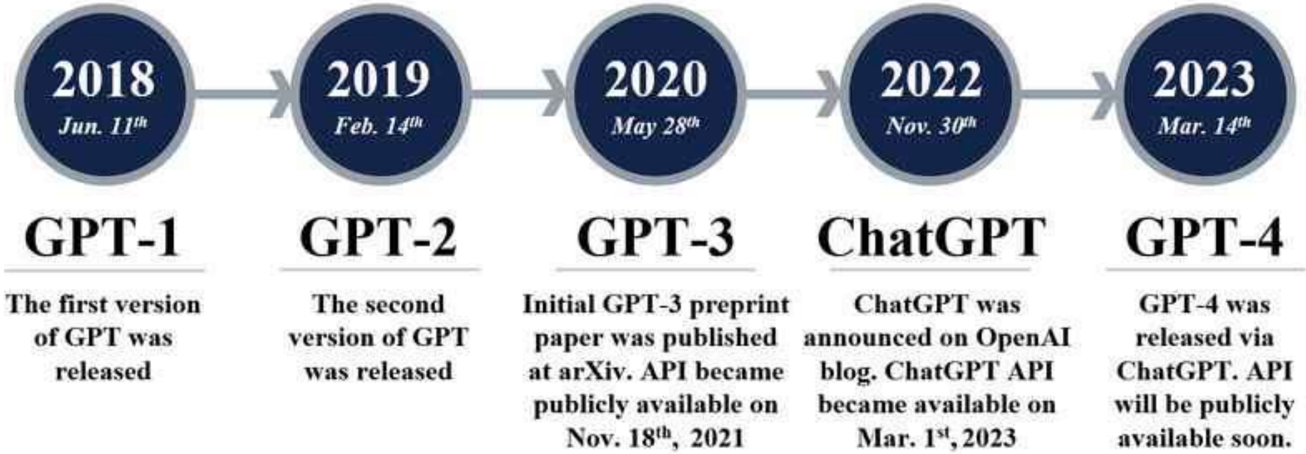
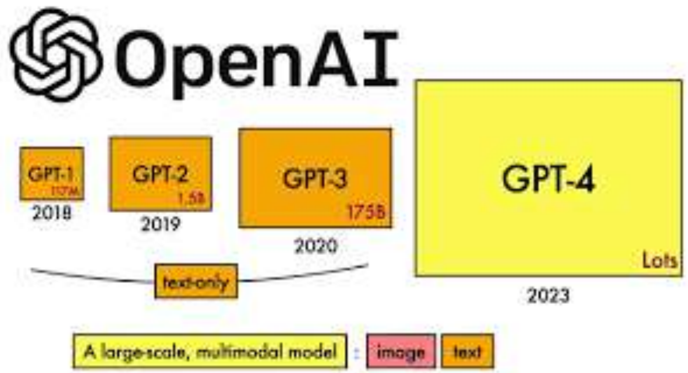


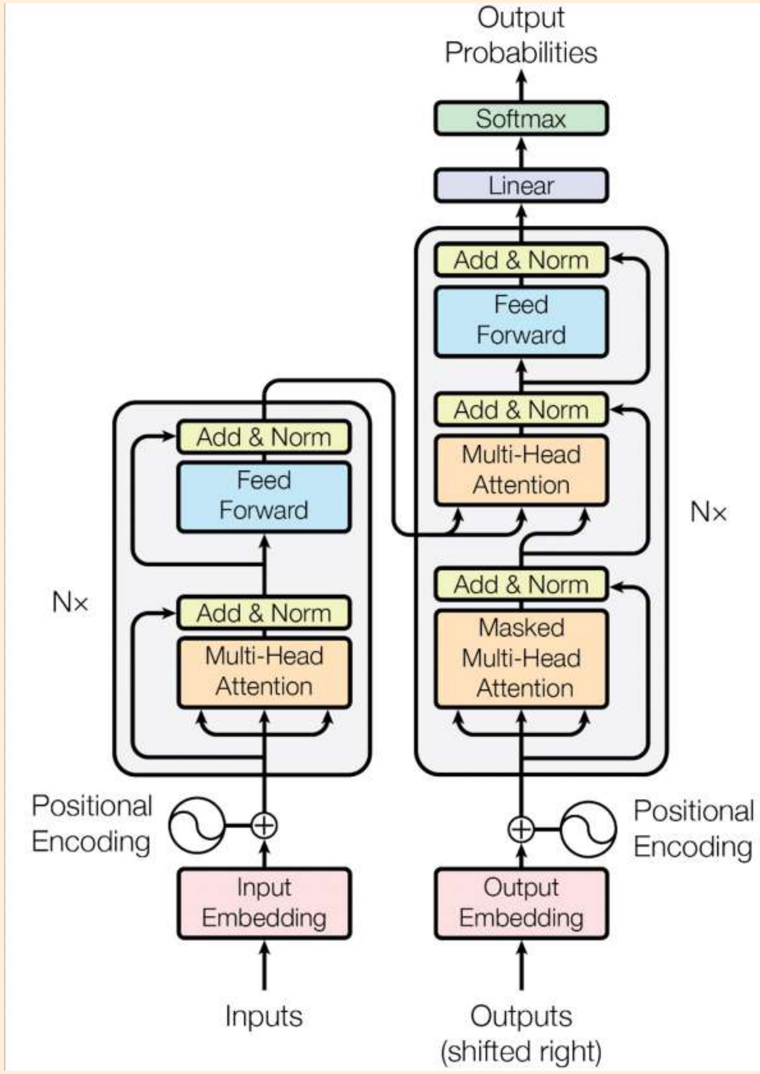

Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.


2018
Jun. 11th
GPT-1
The first version of GPT was released
2019
Feb. 14th
GPT-2
The second version of GPT was released
2020
May 28th
GPT-3
Initial GPT-3 preprint paper was published at arXiv. API became publicly available on Nov. 18th, 2021
2022
Nov. 30th
ChatGPT
ChatGPT was announced on OpenAI blog. ChatGPT API became available on Mar. 1st, 2023
2023
Mar. 14th
GPT-4
GPT-4 was released via ChatGPT. API will be publicly available soon.


OpenAI
GPT-1
117M
2018
GPT-2
1.5B
2019
GPT-3
175B
2020
GPT-4
Lots
2023
text-only
A large-scale, multimodal model
image
text


175B parameters
1.5B
117M
GPT-1
GPT-2
GPT-3
12 decoders
48 decoders
96 decoders
512 token size
1024 token size
2048 token size

# 2) Bidirectional Encoder Representation from Transformers (BERT)



# BERT [Devlin, et al, 2018]:
**B**idirectional **E**ncoder **R**epresentation from **T**ransformers

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

**Jacob Devlin** **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

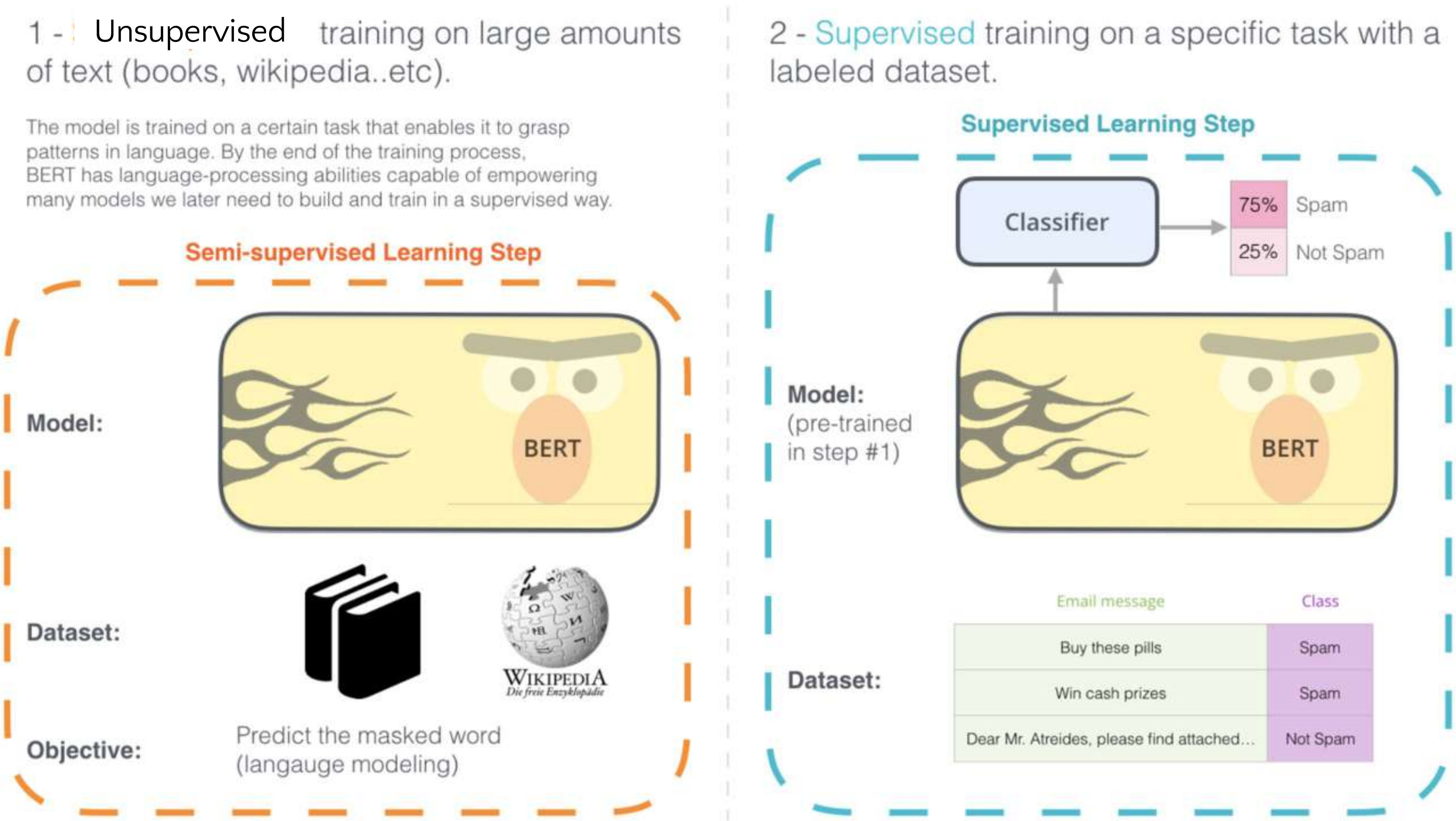
24 May 2019

### Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-

# BERT [Devlin, et al, 2018]:
**B**idirectional **E**ncoder **R**epresentation from **T**ransformers




Devlin et al., **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**, arXiv 2018, https://arxiv.org/pdf/1810.04805.pdf

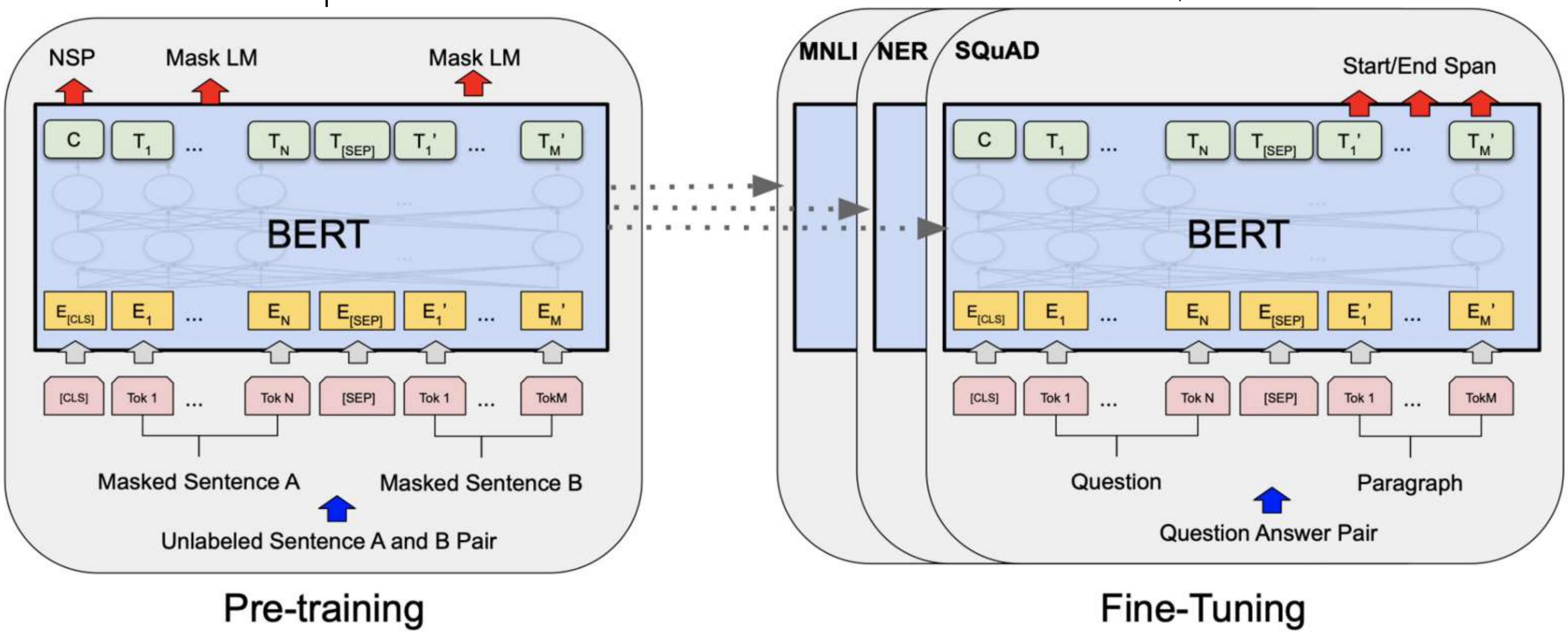
# BERT (cont.): Overall idea

**Phase1: Unsupervised Learning**

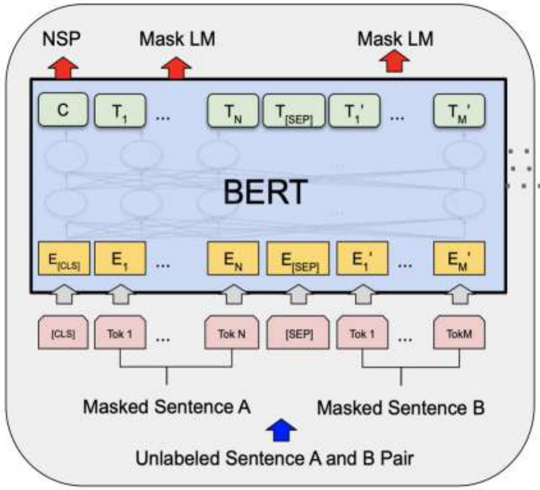
- Mask LM
- Next sentence prediction

**Phase2: Supervised learning**

- **E.g.** Finetune on QA, Text classification, etc.



# Phase1: Unsupervised Phase



Pre-training

Semi-supervised training, using 2 prediction tasks:

### 1.1) Mask language modeling

- represents the word using both its left and right context, so called deeply bidirectional.
- Mask out 15% of the words in the input, run the entire sequence through a deep bidirection encoder, and then predict only the masked words.

```
Input: the man went to the [MASK1] . he bought a [MASK2] of milk.
Labels: [MASK1] = store; [MASK2] = gallon
```

### 1.2) Next sentence prediction (NSP)

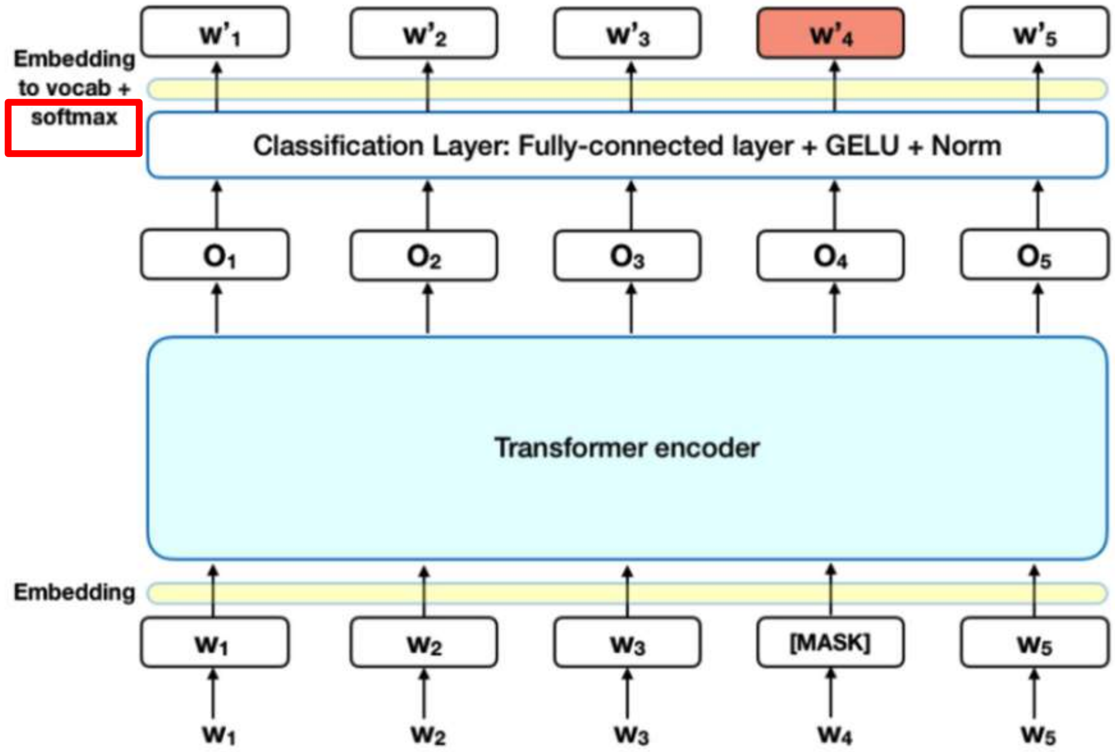
- to learn relationships between sentences.

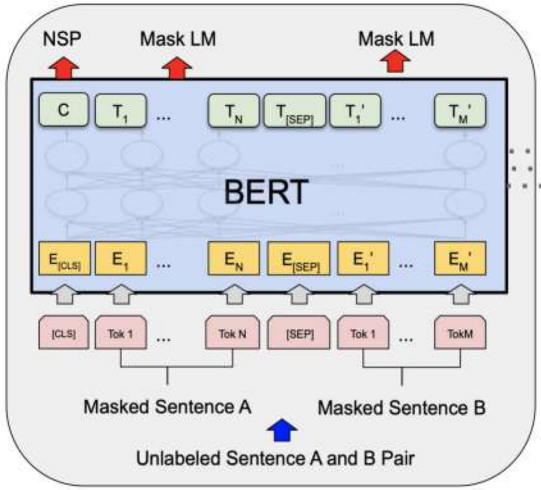
```
Sentence A: the man went to the store .
Sentence B: he bought a gallon of milk .
Label: IsNextSentence
```

```
Sentence A: the man went to the store .
Sentence B: penguins are flightless .
Label: NotNextSentence
```

# Phase1: Unsupervised Phase (1.1. Masked LM)

Mask language modeling:

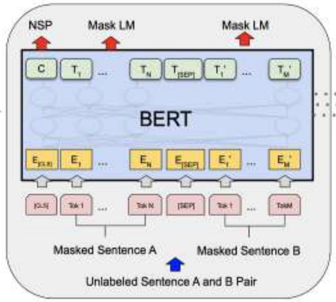


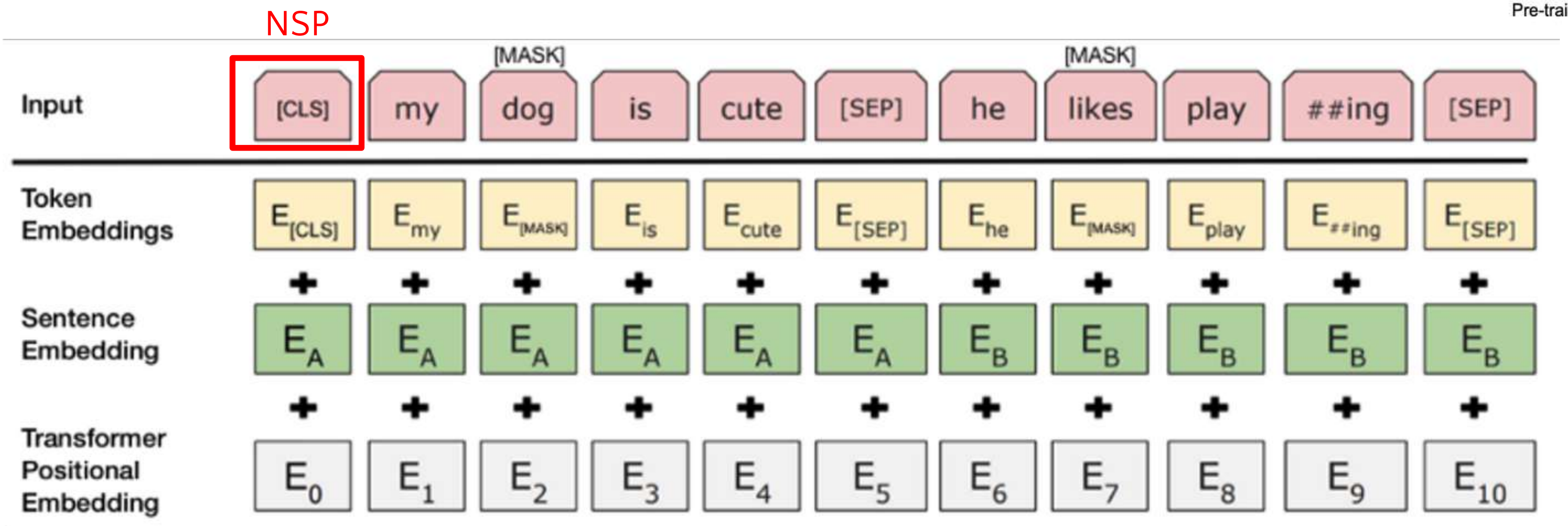
Pre-training

Source: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

# Phase1: Unsupervised Phase
# (1.2. Next Sentence Prediction)




Pre-training



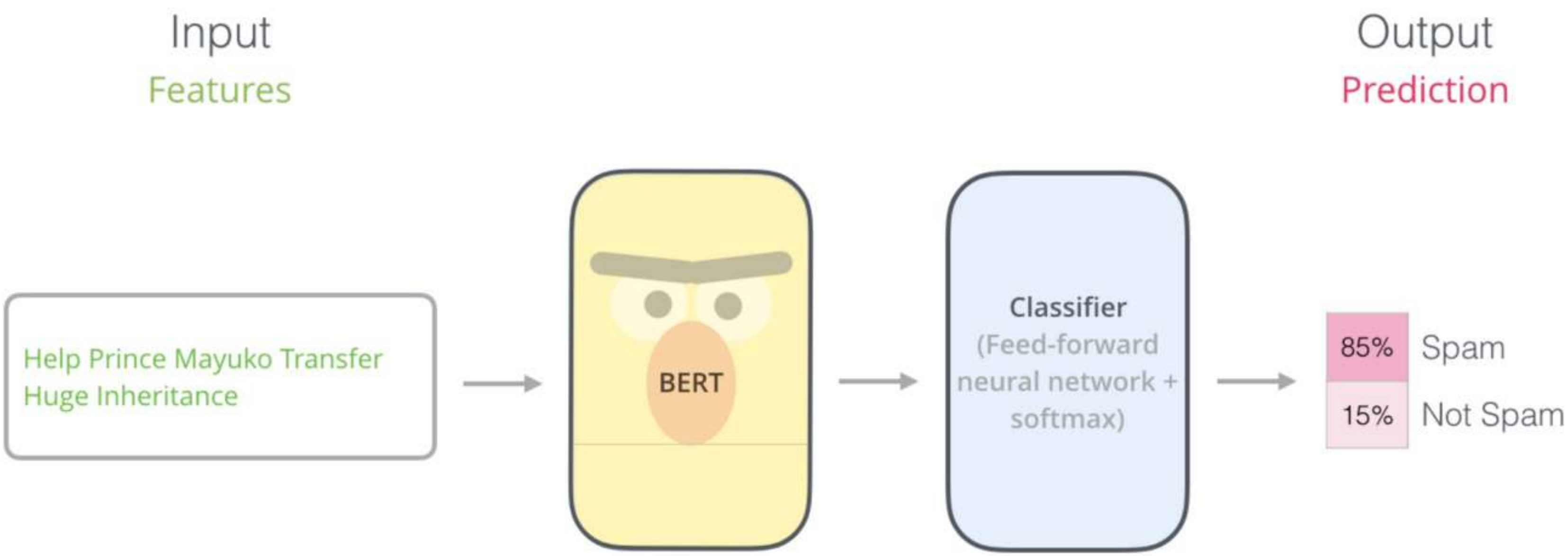

**CLS = Classification, which is used for NSP (Next Sentence Prediction) during this phase**
SEP = a special separator token (e.g. separating questions/answers).

Source:https://arxiv.org/pdf/1810.04805.pdf

# Phase2: Supervised Phase

Supervised training (e.g. Email sentence classification)

- you mainly have to train the classifier, with minimal changes happening to the pre-trained model during the training phase (fine-tuning approach).

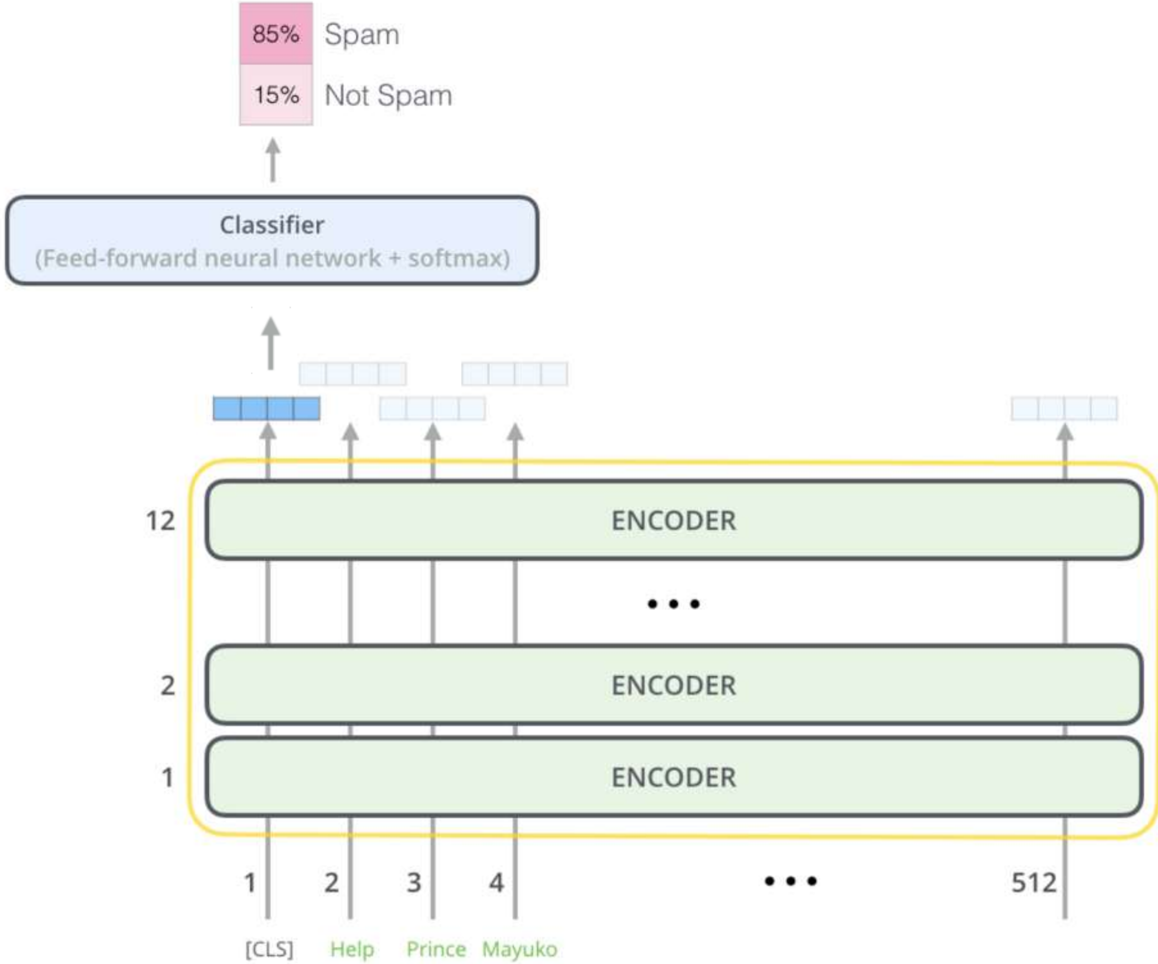


# Phase2: Supervised Phase
# E.g., Spam Email Classification

- BERT is basically a trained Transformer Encoder[1] stack.

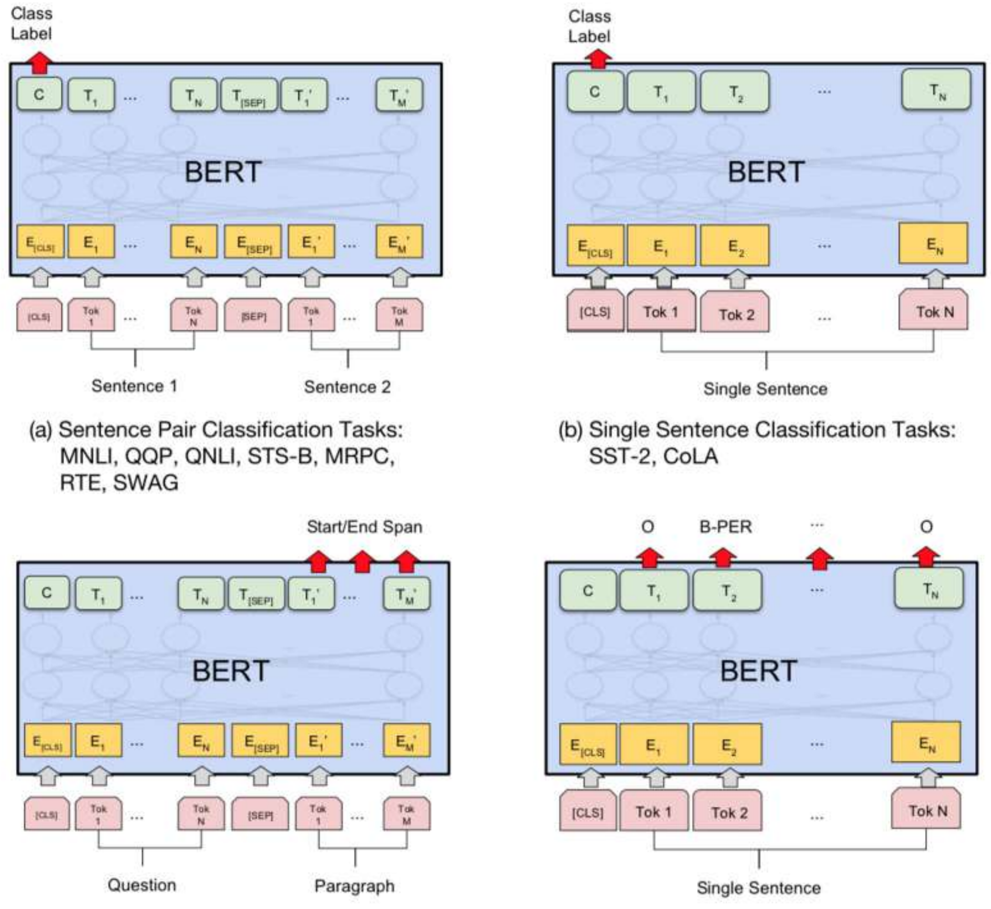
- The first input token is supplied with a special (classification embedding) [CLS] token for classification task to represent the entire sentence.

- The output is generated from only this special [CLS] token.




[1] Vaswani et al., **Attention Is All You Need**, NIPS 2017, https://arxiv.org/pdf/1706.03762.pd

# Phase2: Supervised Phase
# E.g., other tasks




(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Devlin et al., **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**, arXiv 2018, https://arxiv.org/pdf/1810.04805.pdf

# Pretrained BERT (English)

https://github.com/google-research/bert

## BERT

******* New March 11th, 2020: Smaller BERT Models *******

This is a release of 24 smaller BERT models (English only, uncased, trained with WordPiece masking) referenced in Well-Read Students Learn Better: On the Importance of Pre-training Compact Models.

We have shown that the standard BERT recipe (including model architecture and training objective) is effective on a wide range of model sizes, beyond BERT-Base and BERT-Large. The smaller BERT models are intended for environments with restricted computational resources. They can be fine-tuned in the same manner as the original BERT models. However, they are most effective in the context of knowledge distillation, where the fine-tuning labels are produced by a larger and more accurate teacher.

Our goal is to enable research in institutions with fewer computational resources and encourage the community to seek directions of innovation alternative to increasing model capacity.

You can download all 24 from here, or individually from the table below:

| | H=128 | H=256 | H=512 | H=768 |
|---|---|---|---|---|
| L=2 | **2/128 (BERT-Tiny)** | 2/256 | 2/512 | 2/768 |
| L=4 | 4/128 | **4/256 (BERT-Mini)** | **4/512 (BERT-Small)** | 4/768 |
| L=6 | 6/128 | 6/256 | 6/512 | 6/768 |
| L=8 | 8/128 | 8/256 | **8/512 (BERT-Medium)** | 8/768 |
| L=10 | 10/128 | 10/256 | 10/512 | 10/768 |
| L=12 | 12/128 | 12/256 | 12/512 | **12/768 (BERT-Base)** |

# Pretrained BERT (Multilingual)

https://github.com/google-research/bert/blob/master/multilingual.md

## Models

There are two multilingual models currently available. We do not plan to release more single-language models, but we may release `BERT-Large` versions of these two in the future:

- `BERT-Base, Multilingual Cased (New, recommended)` : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Multilingual Uncased (Orig, not recommended)` : 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Chinese` : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

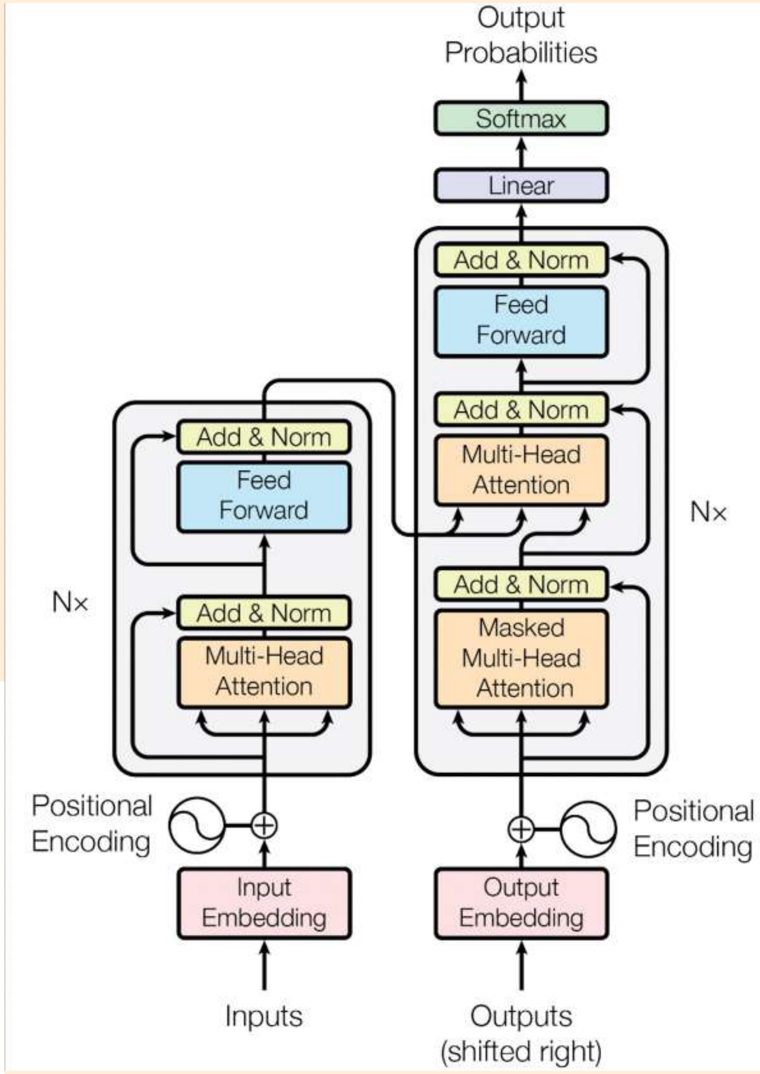
# BART [2019]

**BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension**

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer

We present BART, a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text. It uses a standard Tranformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pretraining schemes. We evaluate a number of noising approaches, finding the best performance by both randomly shuffling the order of the original sentences and using a novel in-filling scheme, where spans of text are replaced with a single mask token. BART is particularly effective when fine tuned for text generation but also works well for comprehension tasks. It matches the performance of RoBERTa with comparable training resources on GLUE and SQuAD, achieves new state-of-the-art results on a range of abstractive dialogue, question answering, and summarization tasks, with gains of up to 6 ROUGE. BART also provides a 1.1 BLEU increase over a back-translation system for machine translation, with only target language pretraining. We also report ablation experiments that replicate other pretraining schemes within the BART framework, to better measure which factors most influence end-task performance.

Subjects: **Computation and Language (cs.CL)**; Machine Learning (cs.LG); Machine Learning (stat.ML)
Cite as: arXiv:1910.13461 **[cs.CL]**
(or arXiv:1910.13461v1 **[cs.CL]** for this version)
https://doi.org/10.48550/arXiv.1910.13461

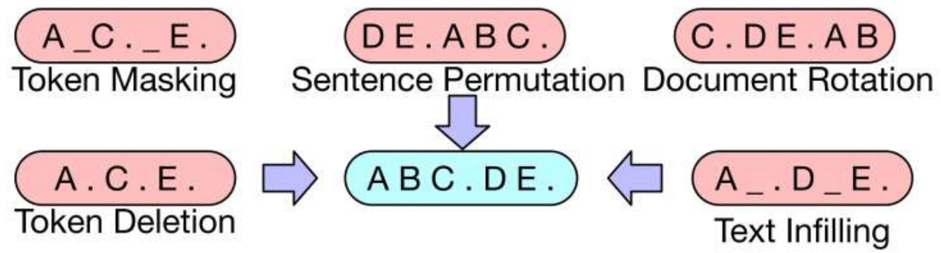


# BART

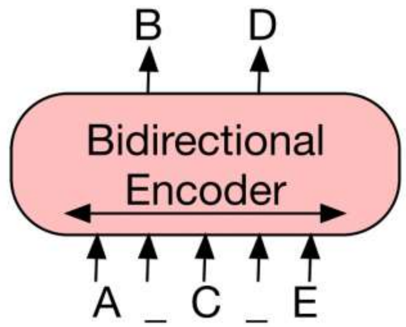
A model comprises of both Transformer encoders *and* decoders

BART optimizes on denoising corrupted inputs

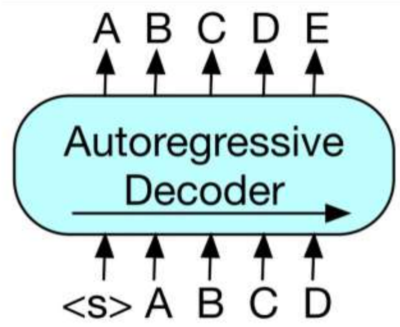
By using both encoders and decoders, BART combines the bidirectionality of BERT and autoregressive ability of GPT.



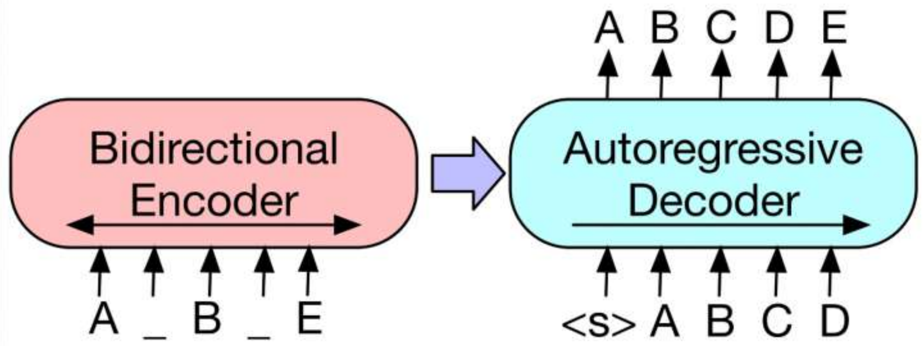

Input noising scheme




BERT




GPT

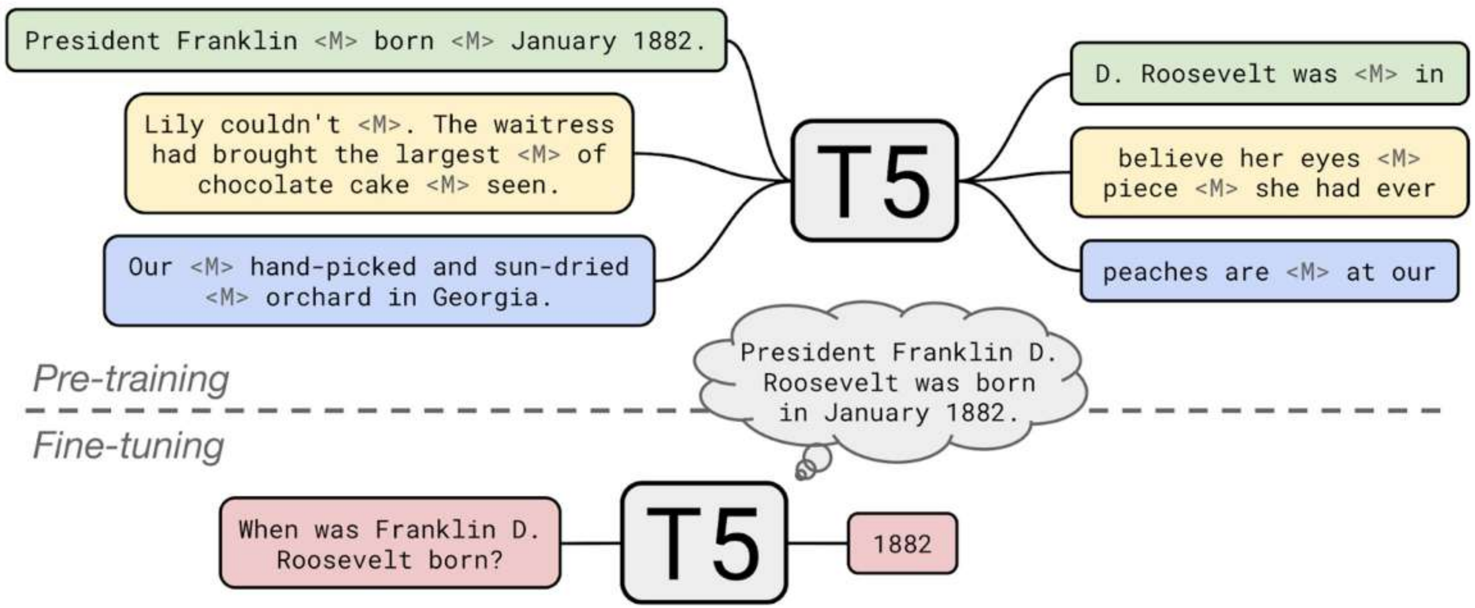



BART

# Text-to-text Transfer Transformer (T5)

T5 is also an encoder-decoder model

It pretrains on the span-corruption objective.



# When to use which?

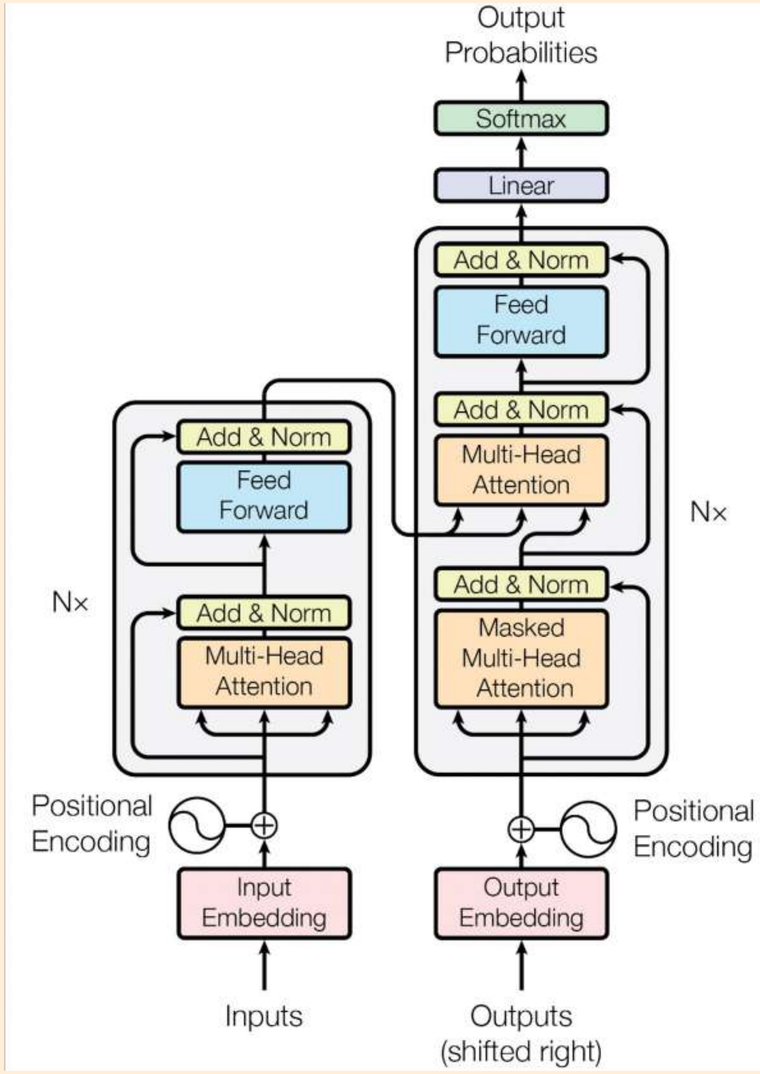
Text Generation: GPT models

Sequence-to-sequence (e.g. text summarization, translation): BART, T5, or similar

Text classification/ Token classification: BERT (easier to use) or BART

| Model | data | bsz | steps | SQuAD (v1.1/2.0) | MNLI-m | SST-2 |
|---|---|---|---|---|---|---|
| RoBERTa | | | | | | |
| with BOOKS + WIKI | 16GB | 8K | 100K | 93.6/87.3 | 89.0 | 95.3 |
| + additional data (§3.2) | 160GB | 8K | 100K | 94.0/87.7 | 89.3 | 95.6 |
| + pretrain longer | 160GB | 8K | 300K | 94.4/88.7 | 90.0 | 96.1 |
| + pretrain even longer | 160GB | 8K | 500K | **94.6/89.4** | **90.2** | **96.4** |

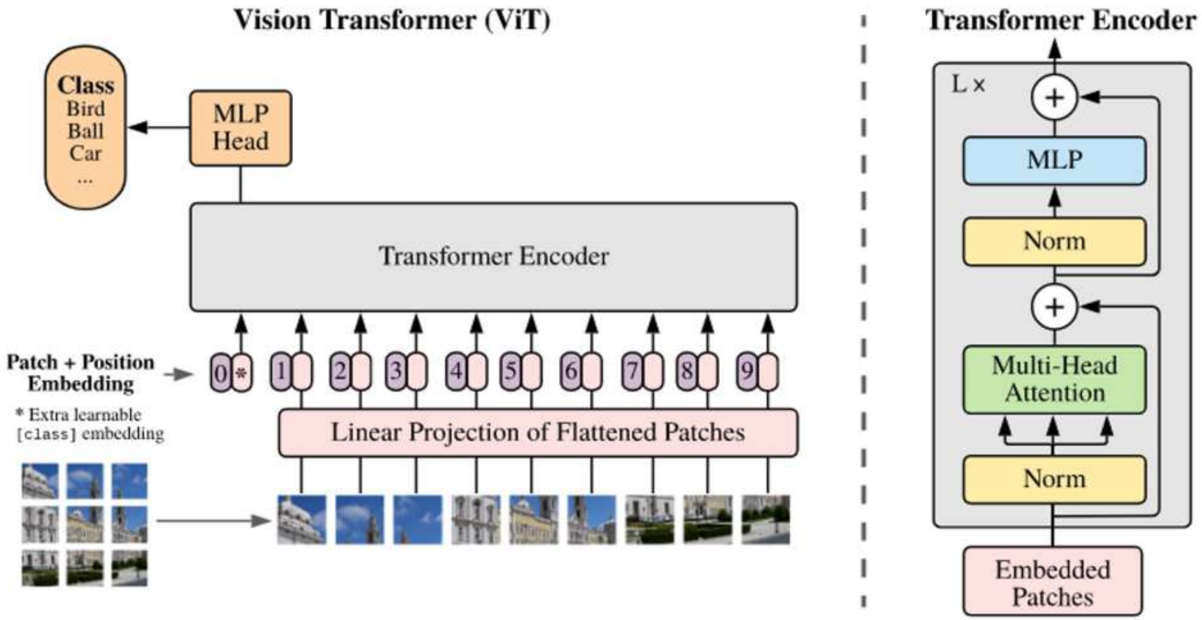
# Beyond NLP

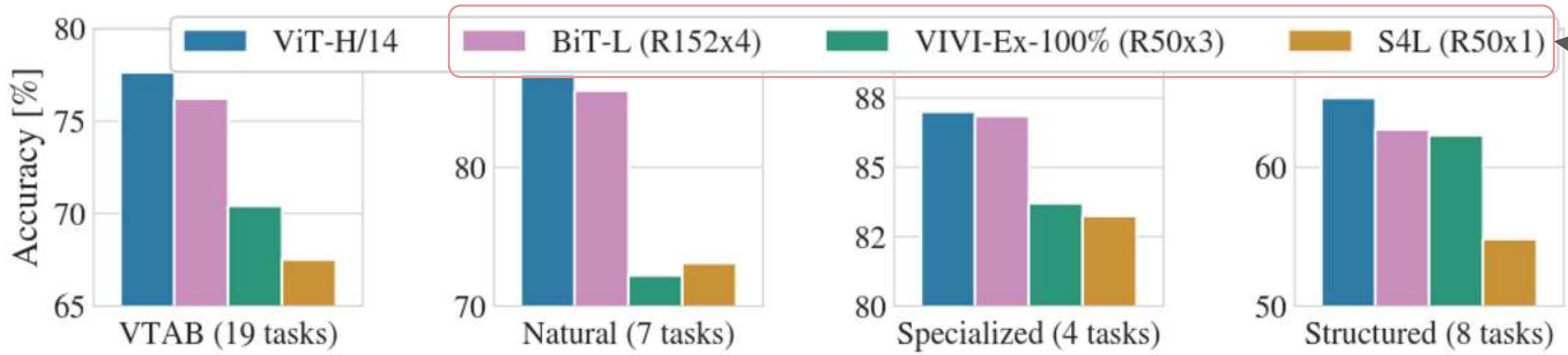

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

# Beyond NLP

1) Computer Vision

ViT uses only encoder.




Baselines are ResNet



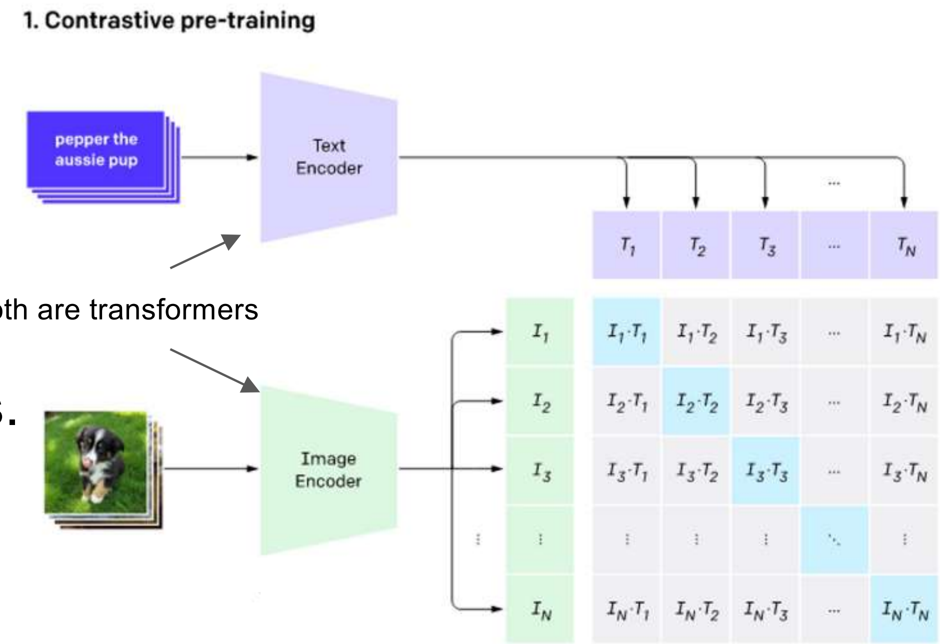
AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

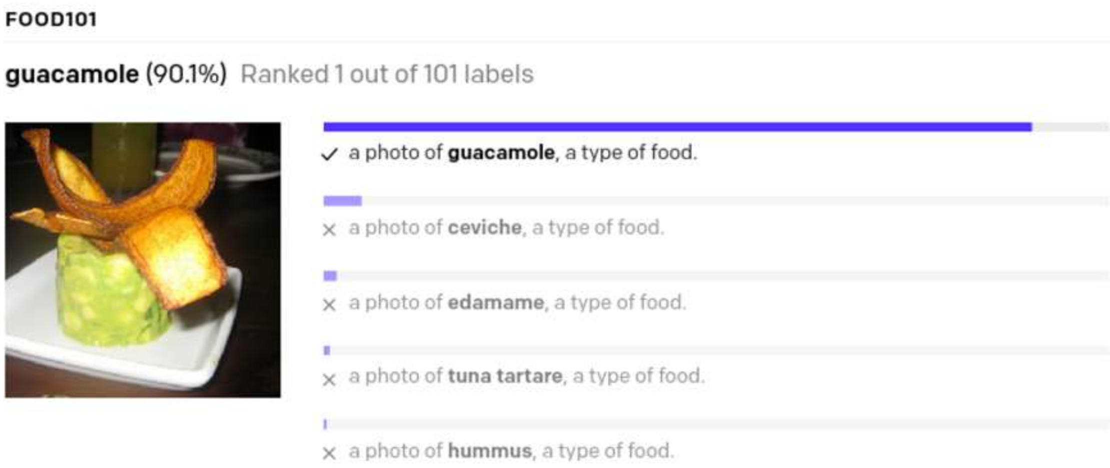
# Beyond NLP

2) Text-Image (Multimodal)

CLIP maps images into categories.

Both are transformers




**DALL.E and CLIP: How Open AI's New Models are Defining the Future of AI Yet Again**

By [x]cube LABS Published: Jan 07 2021

**OpenAI** @OpenAI · Follow

We've developed two neural networks which have learned by associating text and images. CLIP maps images into categories described in text, and DALL-E creates new images, like this, from text.

https://openai.com/blog/clip/

# Beyond NLP

2) Text-to-Image

DALL-E creates new images.

TEXT PROMPT an illustration of a baby daikon radish in a tutu walking a dog

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT PROMPT a store front that has the word 'openai' written on it. . . .

AI-GENERATED IMAGES


Edit prompt or view more images↓

https://openai.com/blog/dall-e/