Attention Mechanisms

Bùi Tiến Lên

2023

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents

Notation

symbol	meaning
$a, b, c, N \ldots$	scalar number
$\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \ldots$	column vector
$\boldsymbol{X}, \boldsymbol{Y} \ldots$	matrix
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{R}^D$	set of vectors
$\mathcal{X}, \mathcal{Y}, \ldots$	set
$\mathcal{A}$	algorithm

operator	meaning
$\boldsymbol{w}^\top$	transpose
$\boldsymbol{X}\boldsymbol{Y}$	matrix multiplication
$\boldsymbol{X}^{-1}$	inverse
$\boldsymbol{X} \odot \boldsymbol{Y}$	an element-wise matrix-vector multiplication

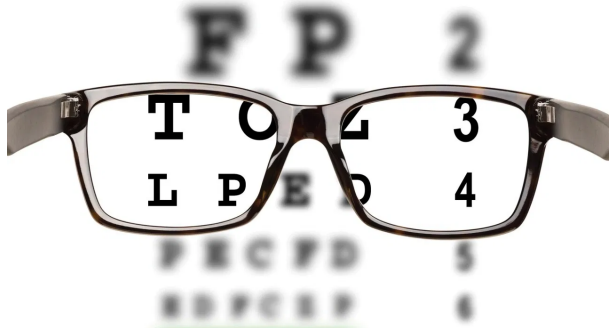
Attention Functions

- Attention Functions
- Type of attention

What is Attention?

Concept 1

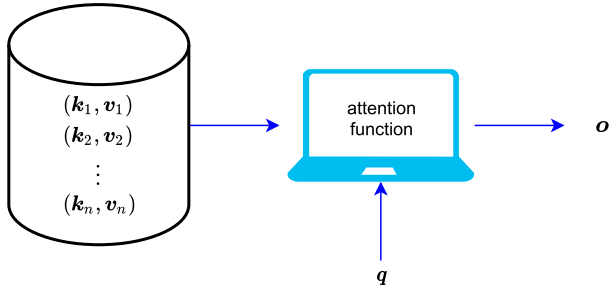
Attention is, to some extent, motivated by how we pay visual attention to different regions of an image or correlate words in one sentence.



Attention Function

Concept 2

An **attention function** can be described as mapping a **query** $\boldsymbol{q}$ and a set of **key-value** pairs $\{(\boldsymbol{k}_i, \boldsymbol{v}_i)\}$ to **an output** $\boldsymbol{o}$



Attention Function (cont.)

Concept 3

The output is computed as a *weighted sum* of the *values*, where the weight assigned to each value is computed by **a compatibility function** α of the *query* with the corresponding *key*.

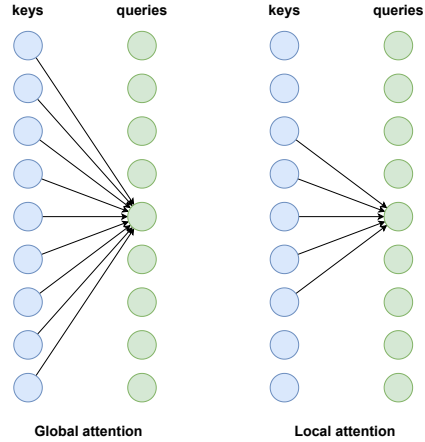
$$\boldsymbol{o} = \text{Attention}(\boldsymbol{q}, \boldsymbol{k}_{1\ldots n}, \boldsymbol{v}_{1\ldots n}) = \sum_{i=1}^{n} \alpha(\boldsymbol{q}, \boldsymbol{k}_i)\boldsymbol{v}_i \tag{1}$$

where $\sum_{i=1}^{n} \alpha(\boldsymbol{q}, \boldsymbol{k}_i) = 1$

Type of attention

Concept 4

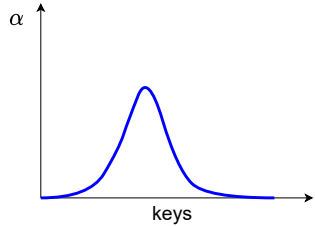
- In **global** attention, we consider all the keys.
- In **local** attention, we consider only a subset of the keys.



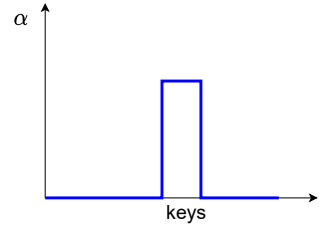
Type of attention (cont.)

Concept 5

- In **soft** attention, the function α varies smoothly over its domain and, as a result, it is **differentiable**.
- In **hard** attention, the function α determine whether to attend to a region or not, which means that the function has many abrupt changes over its domain.



soft attention



hard attention

Attention Pooling

- NonParametric Kernel Regression
- Parametric Kernel Regression

Kernel Regression

- **Problem**: Given a dataset of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, how to learn f to predict the output $\hat{y} = f(\boldsymbol{x})$ for any new input $\boldsymbol{x}$?
- **Solution**: Consider $(\boldsymbol{x}_i, y_i)$ as a pair of key-value and x as query

key	value
$\boldsymbol{x}_1$	y_1
$\vdots$	$\vdots$
$\boldsymbol{x}_N$	y_N

$$\hat{y} = \sum_{i=1}^{N} \alpha(\boldsymbol{x}, \boldsymbol{x}_i) y_i, \tag{2}$$

Kernel Regression (cont.)

- We define α using a Gaussian kernel

$$\alpha(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{\exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|^2\right]}{\sum_{j=1}^{n} \exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_j\right\|^2\right]}. \tag{3}$$

and plug it into equation (17)

$$\begin{aligned} \hat{y} &= \sum_{i=1}^{N} \alpha(\boldsymbol{x}, \boldsymbol{x}_i) y_i \\ &= \sum_{i=1}^{N} \frac{\exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|^2\right]}{\sum_{j=1}^{N} \exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_j\right\|^2\right]} y_i \end{aligned} \tag{4}$$

Kernel Regression (cont.)

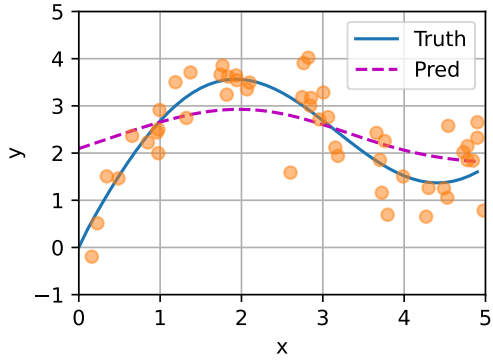
- A key $\boldsymbol{x}_i$ that is closer to the given query $\boldsymbol{x}$ will get more attention via a larger attention weight assigned to the key's corresponding value y_i.

Example 1

- Generate an artificial dataset including 50 training examples and 50 testing examples according to the following nonlinear function with the noise term $\epsilon \sim \mathcal{N}(0, 0.5)$

$$y = 2\sin(x) + x^{0.8} + \epsilon$$

- Find the kernel regression



Parametric Kernel Regression

- Kernel regression enjoys the consistency benefit: given enough data this model converges to the optimal solution.
- Nonetheless, we can easily integrate learnable parameters.
- In the following the distance between the query $\boldsymbol{x}$ and the key $\boldsymbol{x}_i$ is multiplied a learnable parameter w:

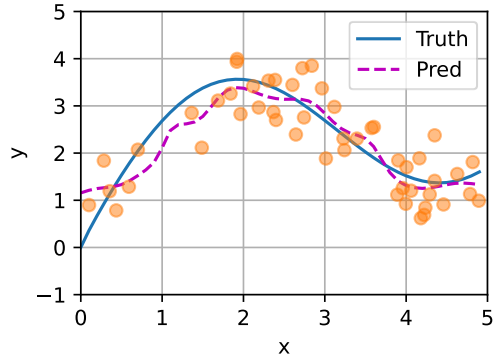
$$\hat{y} = \sum_{i=1}^{N} \frac{\exp\left[-\frac{1}{2}\left(\|\boldsymbol{x} - \boldsymbol{x}_i\| \, w\right)^2\right]}{\sum_{j=1}^{N} \exp\left[-\frac{1}{2}\left(\|\boldsymbol{x} - \boldsymbol{x}_j\| \, w\right)^2\right]} y_i \tag{5}$$

Example 2

- Generate an artificial dataset including 50 training examples and 50 testing examples according to the following nonlinear function with the noise term $\epsilon \sim \mathcal{N}(0, 0.5)$

$$y = 2\sin(x) + x^{0.8} + \epsilon$$

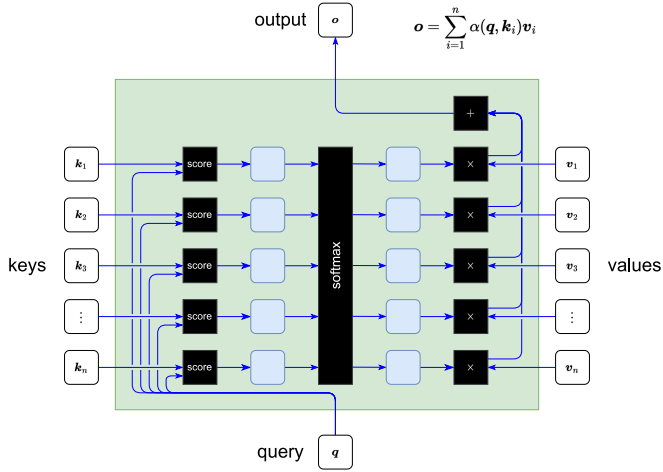
- Find the parametric kernel regression



Attention Scoring Functions

Introduction

- We use an **attention scoring function** and **softmax function** to compute the attention function



Introduction (cont.)

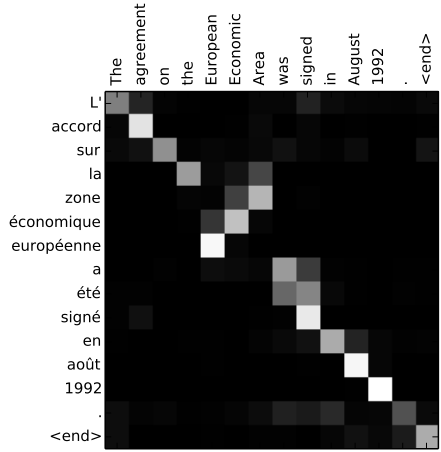
Concept 6

An **attention scoring function** is used to measure the relevance between a query and a key

Introduction (cont.)

Attention scores from different sources

- **English**: "The agreement on the European Economic Area was signed in August 1992"
- **French**: "L'accord sur la zone économique européenne a été signé en août 1992."



Some Attention Scoring Functions

- Additive score (Bahdanau et al. 2015)

$$\text{score}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w}_v^\top \tanh(W_q \boldsymbol{q} + W_k \boldsymbol{k}) \tag{6}$$

- Multiplicative score (Luong et al. 2015)

$$\text{score}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\top W \boldsymbol{k} \tag{7}$$

- Dot-product score (Luong et al. 2015)

$$\text{score}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\top \boldsymbol{k} \tag{8}$$

- Scaled dot-product score (Vaswani et al. 2017)

$$\text{score}(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^\top \boldsymbol{k}}{\sqrt{dim(\boldsymbol{k})}} \tag{9}$$

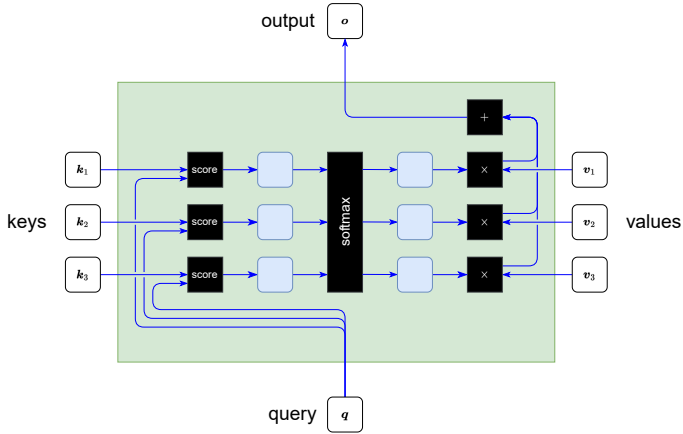
- Cosine score (Graves et al. 2014)

$$\text{score}(\boldsymbol{q}, \boldsymbol{k}) = \text{cosine}(\boldsymbol{q}, \boldsymbol{k}) \tag{10}$$

Example 1

- Using **dot-product score** to calculate the output $\boldsymbol{o}$ given that

$$\boldsymbol{q} = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}, \boldsymbol{k}_1 = \begin{pmatrix} 2 \\ 1 \\ -2 \end{pmatrix}, \boldsymbol{v}_1 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \boldsymbol{k}_2 = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}, \boldsymbol{v}_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \boldsymbol{k}_3 = \begin{pmatrix} 3 \\ -2 \\ 0 \end{pmatrix}, \boldsymbol{v}_3 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$



Attention Layer

Attention Layer

Input:

- Query vectors: Q
- Memory vectors: X

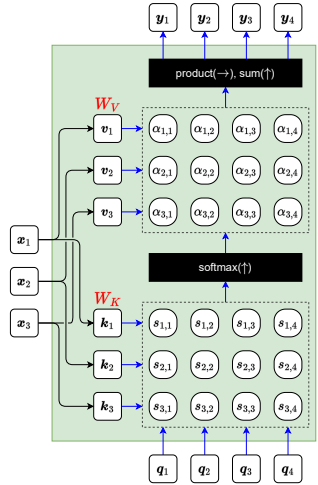
Output:

- Output vectors: Y

Parameters: W_K, W_V

Function:

$$Y = \text{AttentionLayer}(X, Q; W_K, W_V)$$



Self-Attention Layer

Input:

- Input vectors: X

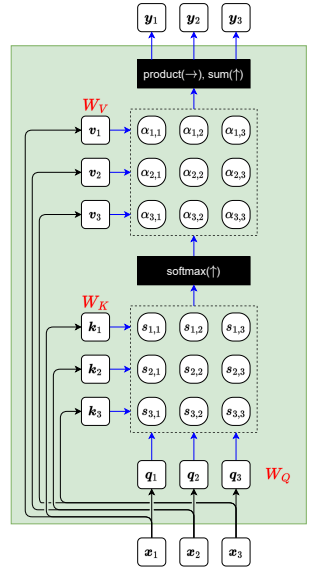
Output:

- Output vectors: Y

Parameters: W_K, W_V, W_Q

Function:

$$Y = \text{SelfAttentionLayer}(X; W_K, W_V, W_Q)$$



Example 2

- Calculate the output of the self-attention layer given that

$$X = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix}$$

$$W_K = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, W_Q = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}, W_V = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 2 & 3 & 0 & 1 \end{pmatrix}$$

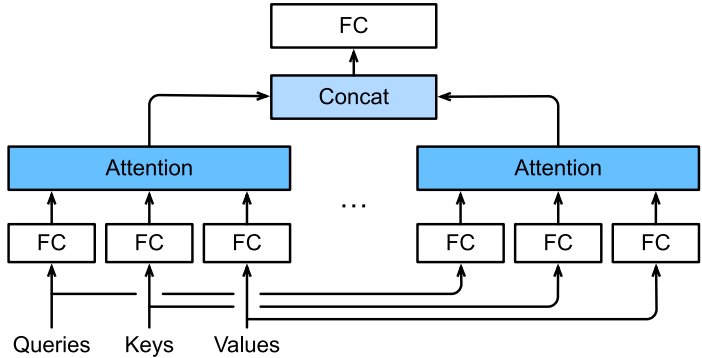
Positional Encoding

- Unlike RNNs that recurrently process tokens of a sequence one by one, self-attention ditches sequential operations in favor of parallel computation.
- To use the sequence order information, we can inject absolute or relative positional information by adding positional encoding to the input representations.
- Positional encodings can be either learned or fixed.
- Suppose that the input representation $X \in \mathbb{R}^{n \times d}$ contains the d-dimensional embeddings for n tokens of a sequence. The positional encoding outputs $X + P$ using a positional embedding matrix $P \in \mathbb{R}^{n \times d}$ of the same shape, whose element on the i^{th} row and the $(2j)^{\text{th}}$ or the $(2j+1)^{\text{th}}$ column

$$\begin{aligned} p_{i,2j} &= \sin\left(\frac{i}{10000^{2j/d}}\right), \\ p_{i,2j+1} &= \cos\left(\frac{i}{10000^{2j/d}}\right). \end{aligned} \tag{11}$$

Multi-Head Attention

- In practice, it may be beneficial to use different representation subspaces of queries, keys, and values.



Multi-Head Attention (cont.)

Input:

- Query vectors: Q
- Key vectors: K
- Value vectors: V

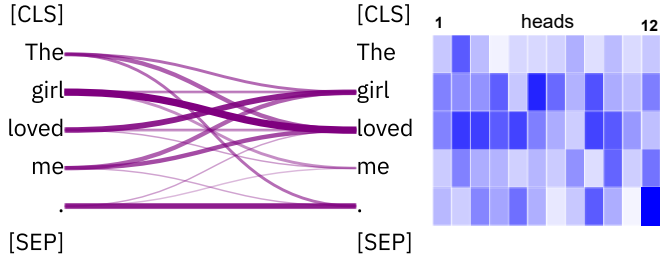
Output:

$$\begin{aligned} \text{MultiHeadAttention}(Q,K,V) &= [\text{head}_1;\ldots;\text{head}_h]W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), i = 1...h \end{aligned} \tag{12}$$

where W_i^Q, W_i^K, W_i^V, and W^O are parameter matrices to be learned.

Multi-Head Attention (cont.)

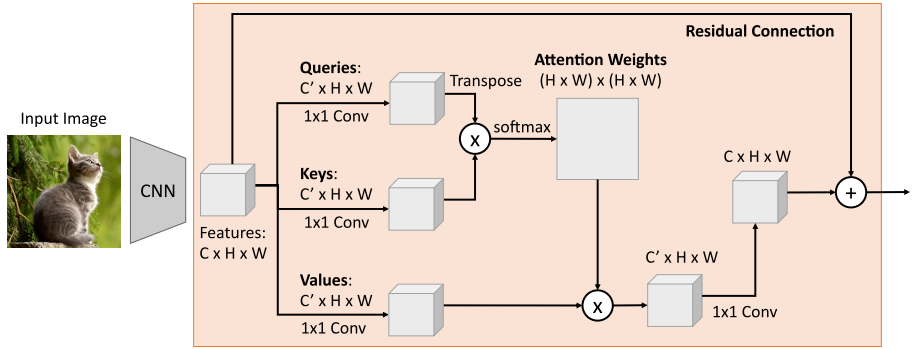
- Multi-head self-attention for the sentence “the girl loved me.” with 12 heads



Applications

- Vision
- Transformer

CNN with Self-Attention



Self-Attention Module

Image Captioning with RNNs and Attention

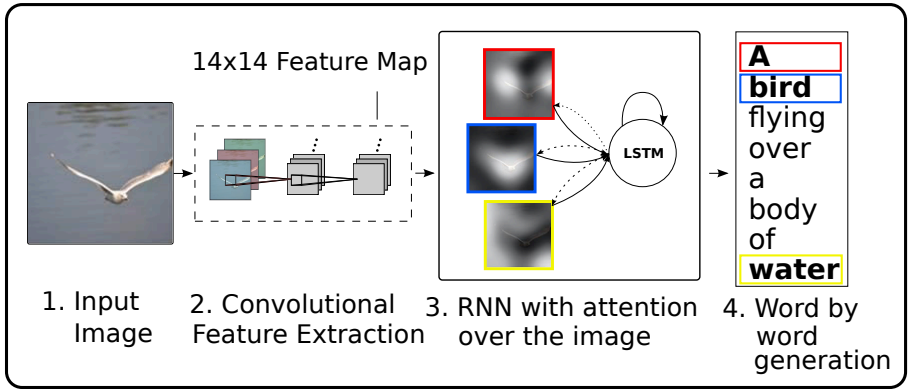
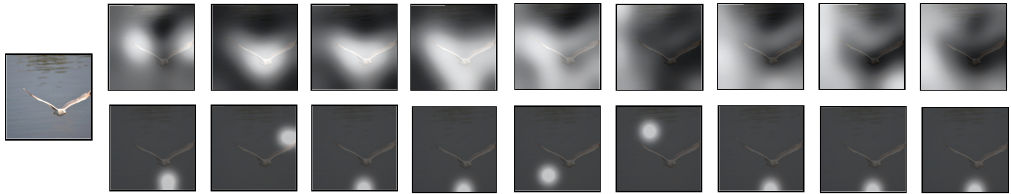


Image Captioning with RNNs and Attention (cont.)

- Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention.



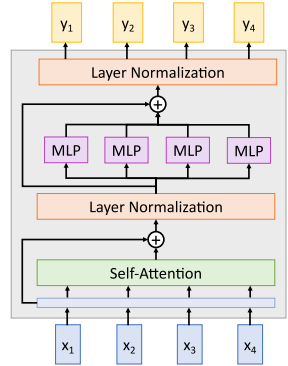
The Transformer

Concept 7

Transformers are a new neural network model that only uses attention!

Transformer Block

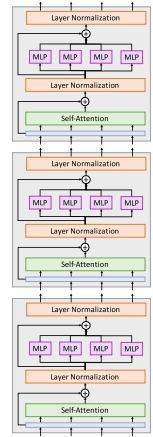
- **Input**: Set of vectors $\boldsymbol{x}$
- **Output**: Set of vectors $\boldsymbol{y}$
- Self-attention is the only interaction between vectors!
- Layer norm and MLP work independently per vector
- Highly scalable, highly parallelizable



The Transformer (cont.)

Concept 8

A Transformer is a sequence of transformer blocks



Transfer Learning

"ImageNet Moment for (Natural) Language Processing"

Pretraining:

- Download a lot of text from the internet
- Train a giant Transformer model for language modeling

Finetuning:

- Fine-tune the Transformer on your own language task

Scaling up Transformers

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13GB	
BERT-Large	24	1024	16	340M	13GB	
XLNet-Large	24	1024	16	~340M	126GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40GB	
Megatron-LM	72	3072	32	8.3B	174GB	512x V100 GPU (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100 GPU
GPT-3	96	12288	96	175B	694GB	?

References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

Lê, B. and Tô, V. (2014).
Cở sở trí tuệ nhân tạo.
Nhà xuất bản Khoa học và Kỹ thuật.

Russell, S. and Norvig, P. (2021).
Artificial intelligence: a modern approach.
Pearson Education Limited.