

# AI Security

Transformer and Its Friends

**Sangdon Park**

POSTECH

Many contents from [Bishop and Bishop, 2023]

# Outline

- 1 Introduction
- 2 Transformer
- 3 Large Language Models
- 4 Large Reasoning Models
- 5 Retrieval-Augmented Generation
- 6 Conclusion

# Why Transformers: Next Token Prediction

I swam across the river to get to the other bank

## Key Question

What's the next token of the "other"?

- To this end, we need to model context, *i.e.*, correlation between words.
- The context was modeled by other models, including CNN and RNN, but they have many limitations

# Why Transformers: Next Token Prediction

I swam across the river to get to the other bank

## Key Question

What's the next token of the "other"?

- To this end, we need to model context, *i.e.*, correlation between words.
- The context was modeled by other models, including CNN and RNN, but they have many limitations

## Other Reasons:

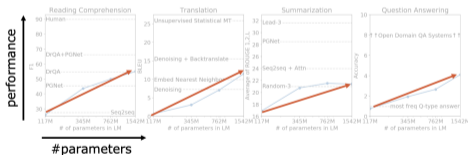
- Parallelizable – enables to use GPUs efficiently
- Scalable – enable to increase model parameters (as long as you have money)

# Outline

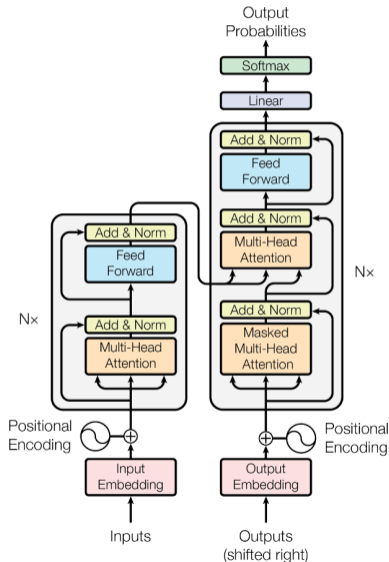
- 1 Introduction
- 2 Transformer**
- 3 Large Language Models
- 4 Large Reasoning Models
- 5 Retrieval-Augmented Generation
- 6 Conclusion

# Birth of Transformer

- A ground-breaking architecture that transforms a set of vectors in one space to a set of vectors in another space with the same dimensionality, proposed by Google [Vaswani et al., 2017].
- Good Properties:
  - ✓ Follows neural scaling law (e.g., GPT-2 paper)



- ✓ Easy for parallel processing
- ✓ Easy to learn (self-supervised learning)
- We focus on “decoding-only” in this class.



# Key Concept: Attention

## 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  
**Lukasz Kaiser\*** Google Brain lukasz.kaiser@google.com

**Illia Polosukhin\* ‡**  
illia.polosukhin@gmail.com

### Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Published as a conference paper at ICLR 2015

## NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

**Dzmitry Bahdanau**  
Jacobs University Bremen, Germany

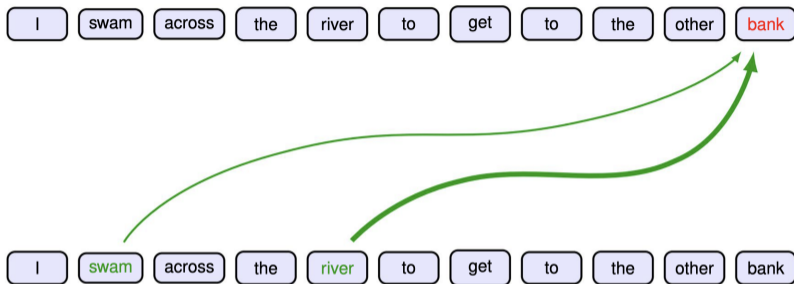
**KyungHyun Cho** **Yoshua Bengio\***  
Université de Montréal

### 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.

- The concept of attention is originated from Bahdanau, Cho, and Bengio (2014).

## Attention with Examples



- The attention is a mechanism to select a proper part of a context.
  - ▶ Attention changes depending on previous tokens (e.g., swam → bank v.s. walked → bank).

I swam across the river to get to the other **bank**.

I walked across the road to get cash from the **bank**.

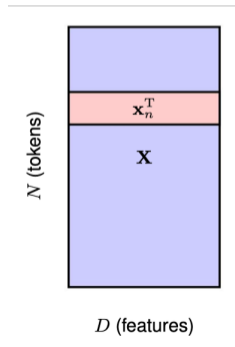
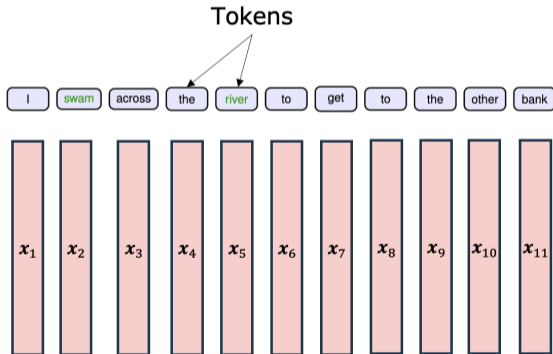
# Transformer Layer

A transformer (layer) maps a set of vectors to a set of vectors, *i.e.*,

$$\tilde{\mathbf{X}} = \text{TransformerLayer}[\mathbf{X}],$$

where it consists of two key components:

- attention mechanism
- feature transform



# Attention Coefficient

## Attention Mechanism 1

$$\mathbf{y}_n = \sum_{m=1}^N a_{nm} \mathbf{x}_m,$$

where  $a_{nm}$  is an attention coefficient. Here,  $\mathbf{x}_m$  can be “river” representation and  $\mathbf{y}_n$  can be “bank” representation.

- An output of the attention is modeled by the convex combination of inputs, *i.e.*,

- ▶  $a_{nm} \geq 0$
- ▶  $\sum_{m=1}^N a_{nm} = 1$

✓ Attend to related inputs to represent the output.

✗ Attention coefficients are not adaptive.

# Self-attention

## Attention Mechanism 2

$$\mathbf{y}_n = \sum_{m=1}^N a_{nm} \mathbf{x}_m, \quad \text{where} \quad a_{nm} = \frac{\exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'})}$$

where  $a_{nm}$  is an attention coefficient.

- Construct weighting coefficients from inputs.
- ✓ Attention coefficients are now adaptive.
  - ▶ The inner product measures correlation between two input tokens.
- ✗ Attention coefficients are not learnable.
- Note that we will use the vector representation of the attention, *i.e.*,

$$\mathbf{y}_n = \sum_{m=1}^N a_{nm} \mathbf{x}_m \quad \rightarrow \quad \mathbf{Y} = \text{Softmax}[\mathbf{X}\mathbf{X}^T] \mathbf{X},$$

where  $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{N \times D}$ .

# Parameterization

## Attention Mechanism 3

The parameterization  $\tilde{\mathbf{X}} := \mathbf{X}\mathbf{U}$  converts

$$\mathbf{Y} = \underbrace{\text{Softmax}[\mathbf{X}\mathbf{X}^T]}_{\text{attention coefficient}} \mathbf{X} \quad \text{into} \quad \mathbf{Y} = \underbrace{\text{Softmax}[\mathbf{X}\mathbf{U}\mathbf{U}^T\mathbf{X}^T]}_{\text{attention coefficient}} \mathbf{X}\mathbf{U}.$$

- Apply a parameterized linear transformation of inputs.
- ✓ Attention coefficients are learnable.
- ✗ Attention coefficients are symmetric.
  - ▶ (the attention from  $i$ -th token to the  $j$ -th token)  $\approx$  (the attention from the  $j$ -th token to the  $i$ -th token)  $\rightarrow$  strong assumption or less degree of freedom.

## Query, Key, and Value

$$\mathbf{Y} = \text{Softmax} \left\{ \mathbf{QK}^T \right\} \times \mathbf{V}$$

$N \times D_v$                        $N \times N$                        $N \times D_v$

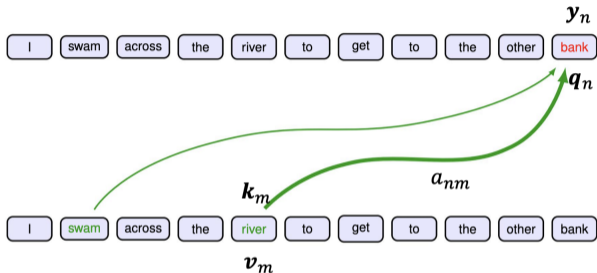
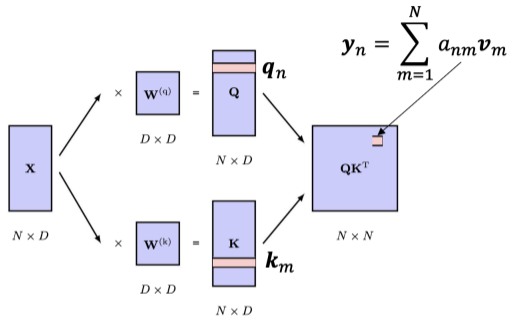
### Attention Mechanism 3

Convert  $\mathbf{Y} = \text{Softmax}[\mathbf{XUU}^T\mathbf{X}^T]\mathbf{XU}$  into  $\mathbf{Y} = \text{Softmax}[\mathbf{QK}^T]\mathbf{V}$ ,

where  $\mathbf{W}^{(q)}, \mathbf{W}^{(k)} \in \mathbb{R}^{D \times D_k}$ ,  $\mathbf{W}^{(v)} \in \mathbb{R}^{D \times D_v}$ ,  $\mathbf{Q} := \mathbf{XW}^{(q)}$ ,  $\mathbf{K} := \mathbf{XW}^{(k)}$ , and  $\mathbf{V} := \mathbf{XW}^{(v)}$ .

- Consider three parameterized linear transformations of inputs (often  $D_v = D$ ).
- ✓ Attention coefficients are asymmetric.
- ✗ The attention module has a gradient vanishing problem.

# Intuition behind the Query, Key, and Value Naming



- They are terms from Information Retrieval.

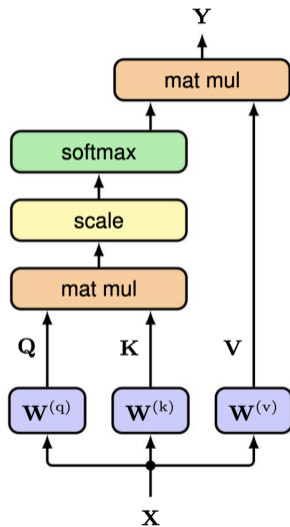
- ▶ Key  $\mathbf{K}$  or  $\mathbf{k}_m$ : a transformed input  $\mathbf{x}_m$  to summarize the input  $\mathbf{x}_m$
- ▶ Value  $\mathbf{V}$  or  $\mathbf{v}_m$ : a transformed input  $\mathbf{x}_m$  to construct the output  $\mathbf{y}_n$
- ▶ Query  $\mathbf{Q}$  or  $\mathbf{q}_n$ : a transformed input  $\mathbf{x}_n$  to summarize the input  $\mathbf{x}_n$

# Scaled Self-attention

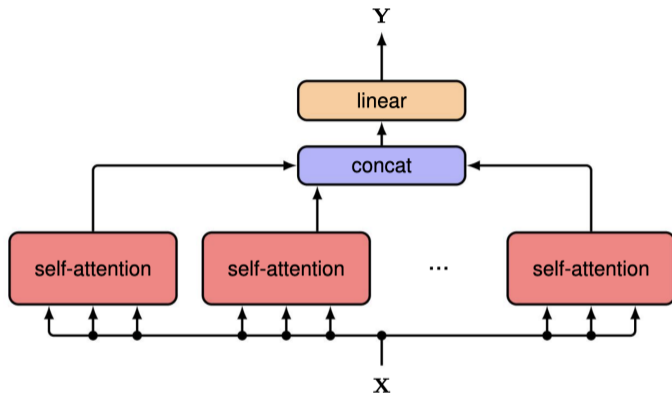
## Attention Mechanism 4

$$\mathbf{Y} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) := \text{Softmax} \left[ \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}} \right] \mathbf{V}$$

- Rescale the product of query and key vectors.
- ✓ Partially avoid gradient vanishing – make the attention stable (= not sharp = not overconfident)
- ✗ Attend to a **single data-dependent** pattern.



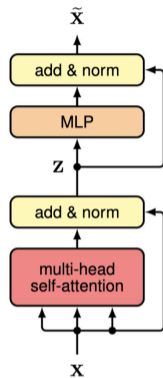
# Multi-head Attention



- Consider multiple attention patterns by stacking.
- ✓ Attend to **multiple data-dependent** patterns.
  - ▶ e.g., “straight edge”, “curve”, or “stripe” patterns in CNNs
- ✗ Still (mostly) linear model and still suffers from gradient vanishing.

# Transformer Layer

$$\tilde{\mathbf{X}} = \text{TransformerLayer}[\mathbf{X}]$$



$$\mathbf{Z} = \text{LayerNorm}[\mathbf{Y}(\mathbf{X}) + \mathbf{X}]$$

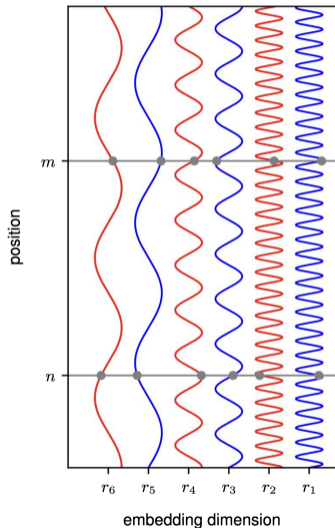
- Add a non-linear layer, residual connections, and layer norm.
- ✓ Non-linear model robust to gradient vanishing.
- ✗ A transformer is “equivariant” with respect to input tokens.
  - ▶ (The food was bad, not good at all)  $\stackrel{\text{Transformer}}{=}$  (The food was good, not bad at all)

# Positional Encoding

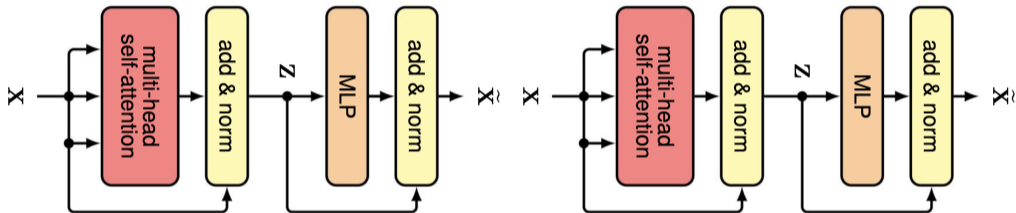
## positional encoding

$$\tilde{\mathbf{x}}_n = \mathbf{x}_n + \mathbf{r}_n$$

- Add position encoding vectors.
- ✓ Now our transformer is position-aware.
  - ▶ (The food was bad, not good at all)  $\neq_{\text{Transformer}}$  (The food was good, not bad at all)
- ? Can this encoding corrupt an input?
  - ▶ probably not as  $r_n$  is likely to be orthogonal to the input  $x_n$  in a high dimensional space.



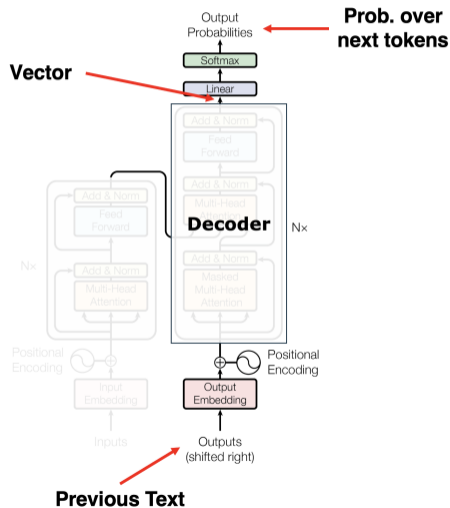
# Scaling Up



- Just stack transformer layers
  - ▶ Possible? Yes as (input dim.)=(output dim.)

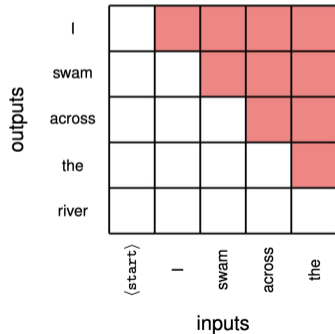
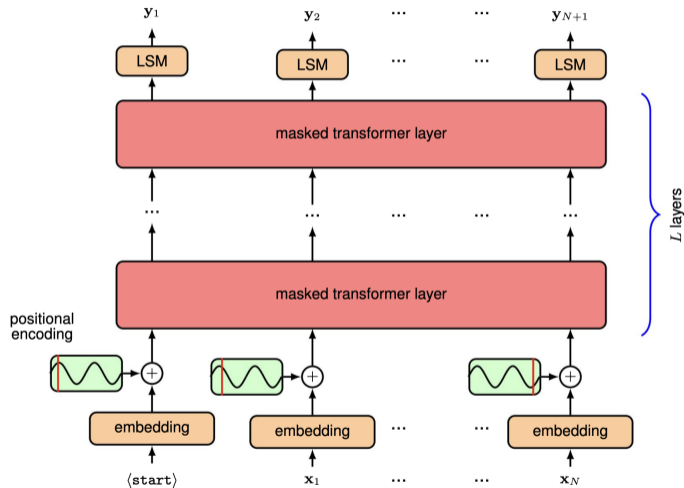
# Decoder-only Models

We focus on decoder-only models.



# Decoder-only Models: Implementation

Use masked transformer layers.



# Outline

- 1 Introduction
- 2 Transformer
- 3 Large Language Models**
- 4 Large Reasoning Models
- 5 Retrieval-Augmented Generation
- 6 Conclusion

# Large Language Models

The transformer is the key part in understanding large language models (LLMs), including the following components:

- Tokenizer – how to convert words or tokens into vectors?
- Decoding Methods – how to generate tokens from LLMs?
- Next token prediction – how to learn LLMs?
- Low-Rank Adapter – how to learn LLMs efficiently?

# Tokenizer

## Question

How to split text into a set of tokens? Or equivalently, how to build

- 1 an encoder (*i.e.*, a map from text to token ids) and
  - 2 a decoder (*i.e.*, a map from token ids to text)?
- “token”: a minimal unit to represent text.
  - What could be a good tokenizer? We can split text by space characters.
  - But, this is not a necessarily best way.
    - ▶ a sub-string in a word may (*e.g.*, “re”) reappear in other words.
    - ▶ thus, split-by-word may not minimize the number of total tokens.
  - What are good necessary properties?
    - 1 reversible: convert tokens back into the original text
    - 2 universal: works for any text
    - 3 compressible: a tokenized sequence should be smaller than the original text in bytes

## tiktoken

tiktoken is a fast [BPE](#) tokeniser for use with OpenAI's models.

```
import tiktoken
enc = tiktoken.get_encoding("o200k_base")
assert enc.decode(enc.encode("hello world")) == "hello world"

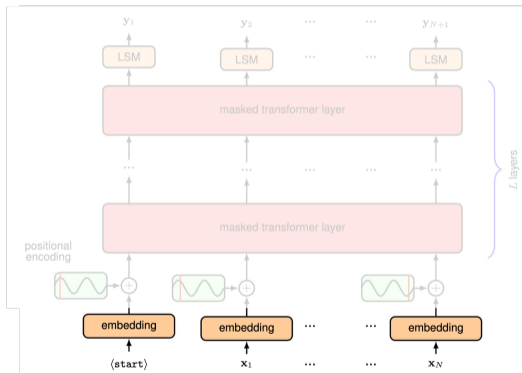
# To get the tokeniser corresponding to a specific model in the OpenAI API:
enc = tiktoken.encoding_for_model("gpt-4o")
```

- tiktoken is the OpenAI's main tokenizer library.
- tikotken is used in other models, including Llama 3, Mistral, Qwen.
- This is a variant of byte-pair encoding (BPE).
  - ▶ What is BPE? An algorithm for encoding strings of text into smaller strings by creating and using a translation table – check out Wikipedia.
- This provides the encoder and decoder:
  - ▶ encoder : str → list[int]
  - ▶ decoder : list[int] → str

# Token Embedding Map I

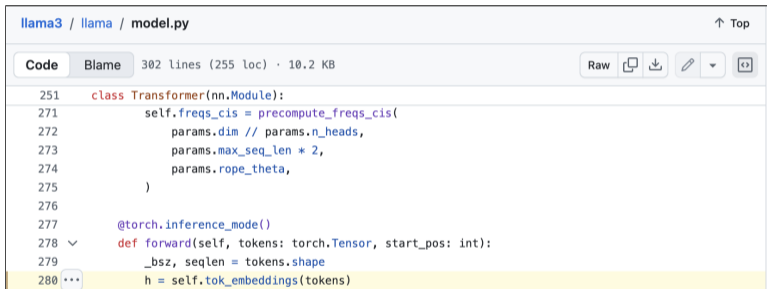
## Question

How does each token represented in a vector space (or a “model” space)?



# Token Embedding Map II

- Token-Embedding-Map :  $\text{int} \rightarrow \mathbb{R}^D$ : a map from token id to a vector.
- The token embedding map is part of model definitions.



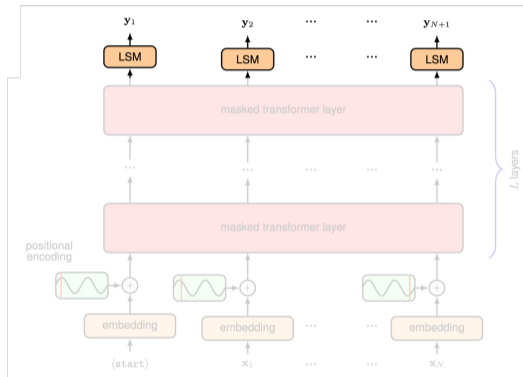
```
llama3 / llama / model.py ↑ Top  
  
Code Blame 302 lines (255 loc) · 10.2 KB Raw Copy Download Edit Search  
  
251 class Transformer(nn.Module):  
271     self.freqs_cis = precompute_freqs_cis(  
272         params.dim // params.n_heads,  
273         params.max_seq_len * 2,  
274         params.rope_theta,  
275     )  
276  
277     @torch.inference_mode()  
278     def forward(self, tokens: torch.Tensor, start_pos: int):  
279         _bsz, seqlen = tokens.shape  
280         h = self.tok_embeddings(tokens)
```

- Here, the token embedding vectors (*i.e.*, a set of vectors from each token id) are learnable parameters.

# Language Model Head

## Question

How to generate tokens?



- The language model(LM) head is a softmax classifier header with tokens as class labels.
- How to choose one token from the distribution over tokens?

# Decoding Methods

Let  $p(y_t | x, y_{1:t-1})$  be a distribution over next tokens from the LM head given previous tokens  $(x, y_{1:t-1})$ , where  $f(x, y_{1:t-1})$  returns a logit vector over tokens and

$$p(y_t | x, y_{1:t-1}) := \text{softmax}(f(y | x, y_{1:t-1}))$$

## 1 Stochastic Methods

### 1 Multinomial Sampling:

$$y_t \sim \text{Multi}(p(y_t | x, y_{1:t-1}))$$

### 2 Multinomial Sampling with temperature scaling:

$$y_t \sim \text{Multi}\left(\text{softmax}\left(\frac{f(y | x, y_{1:t-1})}{T}\right)\right),$$

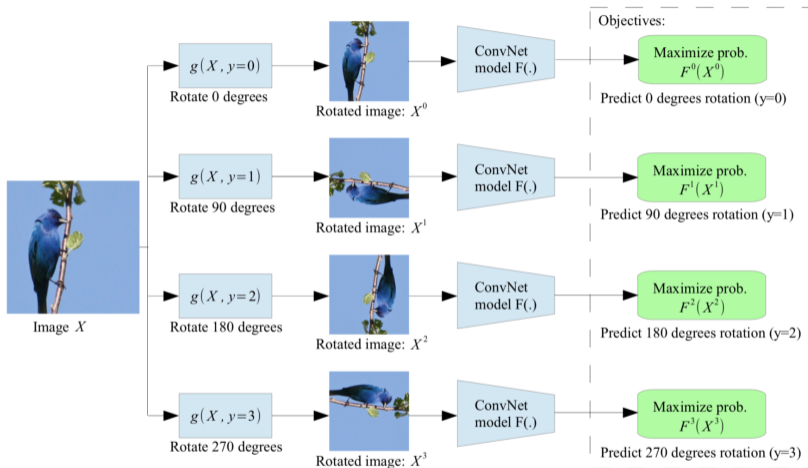
where  $T$  is a temperature parameter.

## 2 Deterministic Methods

### 1 Greedy Decoding:

$$y_t = \arg \max_y p(y | x, y_{1:t-1})$$

# Self-Supervised Learning: Overview



- Self-Supervised Learning is unsupervised learning where it could “self-generate” labels and cast the task into supervised learning (e.g., [Gidaris et al., 2018])

# Learning to Predict Next Tokens

## Next Token Prediction (NTP)

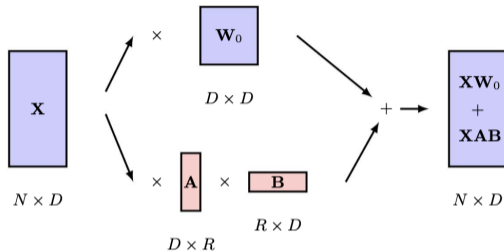
Given a sequence  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$ , suppose that we observe a sub-sequence  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t$ . The goal of the next token prediction is to predict the next token  $\hat{\mathbf{x}}_{t+1}$  where the supervision  $\mathbf{x}_{t+1}$  is obtained from the original sequence.

- Essentially NTP is classification.
  - ▶ Predict  $\hat{\mathbf{x}}_1$  given nothing.
  - ▶ Predict  $\hat{\mathbf{x}}_2$  given  $\mathbf{x}_1$ .
  - ▶ Predict  $\hat{\mathbf{x}}_3$  given  $(\mathbf{x}_1, \mathbf{x}_2)$ .
  - ▶ ...
- Scalable
  - ▶ NTP is applicable to many modalities, including text, image, and video.
- Learning objective (on a single text sequence):

$$\min_{\theta} - \sum_{t=1}^T \ln p_{\theta}(x_t | x_{<t})$$

# Low-Rank Adaptation (LoRA)

Full-finetuning is memory intensive; how to efficiently tune LLMs?



- LoRA is special parameterization of Transformer.
- It adds adapters to attention layers (e.g., key, query, and value weights):

$$\hat{W} = W_0 + AB$$

- Memory usages for learning:
  - ▶ full-finetuning:  $W_0$  is updated, thus need  $D^2$  space for one forward pass.
  - ▶ LoRA:  $AB$  is updated, thus need  $2DR$  space for one forward pass.
  - ▶  $2DR \ll D^2$  if  $R \ll D$ ; thus LoRA is memory-efficient than full-finetuning.

# Outline

- 1 Introduction
- 2 Transformer
- 3 Large Language Models
- 4 Large Reasoning Models**
- 5 Retrieval-Augmented Generation
- 6 Conclusion

# Large Reasoning Models

## Question

How to recall relevant knowledge from parametric knowledge?

- LLMs have deep and wide knowledge.
- How to recall question-related knowledge?
- We will briefly see the following two seminal papers.
  - ① LLMs have reasoning capabilities! – Chain-of-Thought
  - ② The magic secret is revealed by DeepSeek – DeepSeek-R1-Zero
- AI security view: reasoning may **help** or **mitigate** to generate harmful responses.

# Chain-of-Thought [Wei et al., 2023]

## Standard Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

- This paper demonstrates the efficacy of reasoning – open-up possibilities on reasoning!

❌ needs few-shots with reasoning

# DeepSeek-R1-Zero [Guo et al., 2025]

---

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within `<think>...</think>` and `<answer>...</answer>` tags, respectively, i.e., `<think>` reasoning process here `</think>` `<answer>` answer here `</answer>`. User: `prompt`. Assistant:

---

Question: If  $a > 1$ , then the sum of the real solutions of  $\sqrt{a - \sqrt{a+x}} = x$  is equal to

---

Response: `<think>`

To solve the equation  $\sqrt{a - \sqrt{a+x}} = x$ , let's start by squaring both ...

$$(\sqrt{a - \sqrt{a+x}})^2 = x^2 \implies a - \sqrt{a+x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

...

**Wait, wait. Wait. That's an aha moment I can flag here.**

Let's reevaluate this step-by-step to identify if the correct sum can be ...

We started with the equation:

$$\sqrt{a - \sqrt{a+x}} = x$$

First, let's square both sides:

$$a - \sqrt{a+x} = x^2 \implies \sqrt{a+x} = a - x^2$$

Next, I could square both sides again, treating the equation: ...

...

---

- The first performant open source reasoning models.
- From a question, it generates reasoning, followed by answers.
- Your model may have reasoning capabilities – check out reasoning options or start generation with a reasoning tag.

# Outline

- 1 Introduction
- 2 Transformer
- 3 Large Language Models
- 4 Large Reasoning Models
- 5 Retrieval-Augmented Generation**
- 6 Conclusion

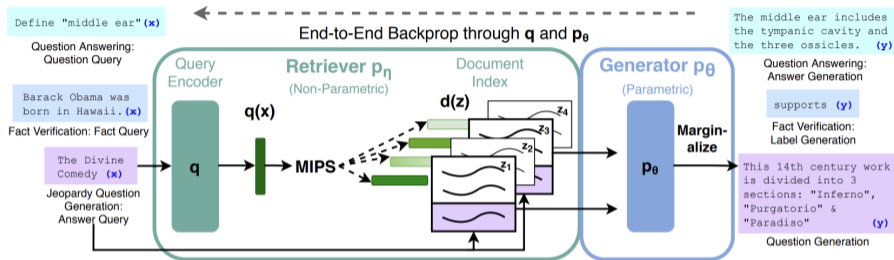
# Retrieval-Augmented Generation (RAG)

## Question

How to generate up-to-date knowledge from LLMs?

- LLMs embed parametric knowledge.
- But, the parametric knowledge is fixed.
- We need ways to exploit new knowledge from LLMs – RAG!
- AI security view: the knowledge can be **contaminated**.

# The Standard RAG [Lewis et al., 2020]



- RAG procedure for question-answering:

- 1 Given a question  $q$
- 2 Retrieve related context passages  $c_1, \dots, c_k$
- 3 Generate an answer given the context passages, *i.e.*,  $p(y_i | q, c_1, \dots, c_k, y_{1:i-1})$

# Outline

- 1 Introduction
- 2 Transformer
- 3 Large Language Models
- 4 Large Reasoning Models
- 5 Retrieval-Augmented Generation
- 6 Conclusion**

# Conclusion

- We overview famous generative models, including
  - ▶ transformers
  - ▶ LLMs
  - ▶ LRMs
  - ▶ RAG
- Injecting adversarial inputs to these generative models leads to safety-critical results (e.g., harmful responses).
- How to attack these generative models?

## Reference I

- C. M. Bishop and H. Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.
- S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.
- D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Xu, H. Ding, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Chen, J. Yuan, J. Tu, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. You, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Zhou, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, T. Yun, T. Pei, T. Sun, T. Wang,

## Reference II

W. Zeng, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, Sept. 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL <http://dx.doi.org/10.1038/s41586-025-09422-z>.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

## Reference III

- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL <https://arxiv.org/abs/2201.11903>.