Transformer and Large Language Models

Instructor: Lei Wu [1]

Mathematical Introduction to Machine Learning

Peking University, Fall 2023

[1]School of Mathematical Sciences; Center for Machine Learning Research

Transformer

Transformers

- introduced in *Attention is all you need* (Vaswani et al., NeurIPS 2017);
- have revolutionized NLP, CV, robotics and many applications;
- have enabled the creation of powerful LLMs such as GPT-4;
- hold the promise of unlocking the potential for AGI (artificial general intelligence).

Sequence Modeling

Define a natural nonlinear map of

$$X := (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n) \mapsto Y := (\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_n)$$

- Recurrence

$$\boldsymbol{y}_i = f(\boldsymbol{x}_i, \boldsymbol{y}_{i-1}).$$

- Convolution

$$\boldsymbol{y}_i = f(\boldsymbol{x}_{i-1}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}).$$

- Attention (simplified)

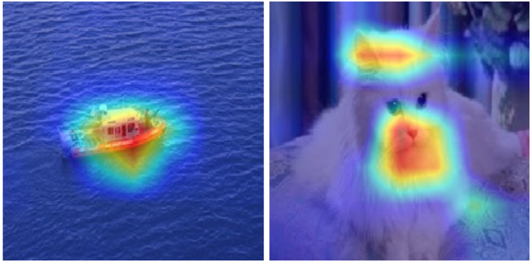
$$\boldsymbol{y}_i = f\left(\sum_{j=1}^{n} P_{i,j} \boldsymbol{x}_j\right),$$

where $P \in \mathbb{R}^{n \times n}$ is a stochastic matrix and may depend on X.

Attention Mechanism

Pre-transformer Attention:

- Attention in vision modeling.
- Attention in machine translation.



Attention Mechanism

Self-attention:

- Let $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$ be our input sequence. A self-attention

$$\mathbb{A} : \mathbb{R}^{d \times n} \mapsto \mathbb{R}^{n \times n}$$

  outputs an attention matrix $P = \mathbb{A}(X)$. The most popular choice is

$$\mathbb{A}(X) = \sigma\left((W_K X)^\top (W_Q X)\right) \in \mathbb{R}^{n \times n},$$

  where
  - $W_K, W_Q \in \mathbb{R}^{d_{\text{key}} \times d}$ are the key and query weight matrices, which are learned from data.
  - σ denotes the softmax normalization performed in a column-wise manner.
- In this case, the attention weights are determined by the **second-order correlation** among tokens. In principle, one can also propose other alternatives.

A Transformer Block

- A transformer block defines a sequence-to-sequence map

$$X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{d\times n} \mapsto Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n) \in \mathbb{R}^{d\times n}.$$

- This maps consists of two blocks:

$$Y = \texttt{FF}(X + \texttt{MHA}(X)),$$

where

  - **Multi-head attention** (MHA)

$$\texttt{MHA}(X) := \sum_{h=1}^{H} W_{O,h} W_{V,h}^T X \mathbb{A}_h(X),$$

  where $W_{O,h}, W_{v,h} \in \mathbb{R}^{d\times r}$ are learnable output and value matrices, respectively.
  - **Positional-wise feed-forward networks** (FFN):

$$\texttt{FF}(Z) := (h(\boldsymbol{z}_1), h(\boldsymbol{z}_2), \ldots, h(\boldsymbol{z}_n)) \in \mathbb{R}^{d\times n}.$$

  In practice, $h : \mathbb{R}^d \mapsto \mathbb{R}^d$ is often chosen to be a two-layer MLP with hidden size d_{FF}.

$$h(\boldsymbol{z}) = W_1^T \mathrm{ReLU}(W_2 \boldsymbol{z} + \boldsymbol{b}),$$

  where $W_1, W_2 \in \mathbb{R}^{d_{\mathrm{FF}}\times d}$ and $\boldsymbol{b} \in \mathbb{R}^d$.

Transformer

- **Input:** Linear embedding to change the dimension of each token.

$$X^{(0)} = VX \text{ with } V \in \mathbb{R}^{d_{\text{model}} \times d}.$$

- **Main block**:

$$X^{\ell} = \texttt{FF}^{(\ell)}(X^{(\ell-1)} + \texttt{MHA}^{(\ell)}(X^{(\ell-1)})), \quad 1 \leq \ell \leq L.$$

- **Ouput:** The output format depends on the tasks. In classification, we may

$$f(X) = p(\boldsymbol{x}_1^{(L)}),$$

where p can be either a linear layer or small MLP.

- Architecure hyperparameters: d_{model}, H, L, d_{key} , d_{FF}. In practice, a common choice $d_{\text{FF}} = 4d_{\text{model}}, d_{\text{key}} = d_{\text{model}}/H$.

Positional Encoding (PE)

Transformers are still inherently permutationally invariant and we need to modify transformers by injecting position information.

- **Absolute positional encoding** (APE): Let $\boldsymbol{r}_i \in \mathbb{R}^d$ denote the information for token i:

$$\boldsymbol{x}_i \to \boldsymbol{x}_i + \boldsymbol{r}_i,$$

  - Learnable APE.
  - Sinusoidal APE: $\boldsymbol{r}_i = \Big(\sin(i), \cos(i), \sin(i/c), \cos(i/c), \ldots, \sin(i/c^{2i/d}), \cos(i/c^{2i/d})\Big) \in \mathbb{R}^d$, where c is constant, e.g. 1000.

- **Relative positional encoding** (RPE): Let $E = (W_K X)^T (W_Q X) \in \mathbb{R}^{n \times n}$.

$$\mathbb{A}(X) = \sigma(E + P),$$

where $P = (h(j-i))_{i,j} \in \mathbb{R}^{n \times n}$. In T5 RPE chooses

$$h(t) = \begin{cases} t & \text{if } t \le B/2 \\ \frac{B}{2} + \frac{B}{2} \left\lfloor \frac{\log(\frac{D}{B/2})}{\log(\frac{D}{B/2})} \right\rfloor & \text{if } \frac{B}{2} \le t \le D \\ B - 1 & \text{if } t \ge D \end{cases}$$

Tokenization in NLP for Transformers

Before we do one-hot embedding, we need to tokenize natural language.

- **Definition:** Process of converting text into tokens (small units) before feeding it into a model.
- **Purpose:** Makes the text interpretable for the model, facilitating further processing like embedding and sequence modeling.

Tokenization in NLP for Transformers

Before we do one-hot embedding, we need to tokenize natural language.

- **Definition:** Process of converting text into tokens (small units) before feeding it into a model.
- **Purpose:** Makes the text interpretable for the model, facilitating further processing like embedding and sequence modeling.

Types of Tokenization

- **Word-level Tokenization:** Splits text into individual words.

  `"Transformers are amazing!"" -> ["Transformers", "are", "amazing", "!"]`

Tokenization in NLP for Transformers

Before we do one-hot embedding, we need to tokenize natural language.

- **Definition:** Process of converting text into tokens (small units) before feeding it into a model.
- **Purpose:** Makes the text interpretable for the model, facilitating further processing like embedding and sequence modeling.

Types of Tokenization

- **Word-level Tokenization:** Splits text into individual words.

  ```
  "Transformers are amazing!"" -> ["Transformers", "are", "amazing", "!"]
  ```

- **Subword-level Tokenization:** Breaks words into smaller pieces (subwords) for efficient handling of unknown words and morphemes. Popular in transformers.

  ```
  "Transformers are amazing!" ->
              ["Trans", "##form", "##ers", "are", "amaz", "##ing", "!"]
  ```

Tokenization in NLP for Transformers

Before we do one-hot embedding, we need to tokenize natural language.

- **Definition:** Process of converting text into tokens (small units) before feeding it into a model.
- **Purpose:** Makes the text interpretable for the model, facilitating further processing like embedding and sequence modeling.

Types of Tokenization

- **Word-level Tokenization:** Splits text into individual words.

  ```
  "Transformers are amazing!"" -> ["Transformers", "are", "amazing", "!"]
  ```

- **Subword-level Tokenization:** Breaks words into smaller pieces (subwords) for efficient handling of unknown words and morphemes. Popular in transformers.

  ```
  "Transformers are amazing!" ->
              ["Trans", "##form", "##ers", "are", "amaz", "##ing", "!"]
  ```

- Many other tokenizations. Libraries like NLTK, spaCy provide basic tokenization. `transformers` library by Hugging Face for transformer-specific tokenization.

Cost Analysis

$$\texttt{MHA}(X) = X + \sum_{h=1}^{H} W_{O,h} W_{V,h}^T X \mathbb{A}_h(X),$$
$$\texttt{FF}(\boldsymbol{x}) = W_1 \mathrm{ReLU}(W_2 \boldsymbol{x} + \boldsymbol{b}).$$

In practice, it is often choose

$$d_H = d_{\text{model}}/H, \quad d_{\text{FF}} = 4d_{\text{model}}.$$

- **Storage:** $4d_{\text{model}}^2 + 8d_{\text{model}}^2$
- **Computation:**
  - MHA: $4nd_{\text{model}}^2 + d_{\text{model}}n^2$
  - FF: $8d_{\text{model}}^2 n$.

  Note that the tokenwise operations can be parallelized. The total cost depends on the sequence length **qudratically**. This is especially bad for inference!!

Training

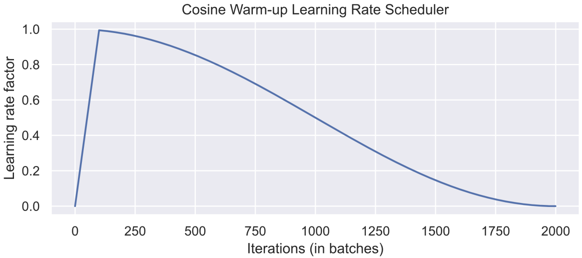
- Scaled dot-product attention

$$\mathbb{A}(X) = \sigma\left(\frac{1}{\sqrt{d_k}}(W_K X)^\top (W_Q X)\right) \in \mathbb{R}^{n\times n},$$

- Layer normalization:

$$\tilde{X}^{(\ell-1)} = \mathrm{LN}(X^{(\ell-1)})$$
$$X^{\ell} = \mathtt{FF}\left(\tilde{X}^{(\ell-1)} + \mathtt{MHA}(\tilde{X}^{(\ell-1)})\right)$$

- Residual connection.
- AdamW optimizer with $(\beta_1 = 0.9, \beta_2 = 0.98)$ and gradient clipping.
- Learning rate Warmup.



Readings

- The original paper https://arxiv.org/abs/1706.03762
- Annotated Transformer https://jalammar.github.io/illustrated-transformer/
- Illustrated Transformer https://nlp.seas.harvard.edu/annotated-transformer/

BERT

- Developed by Google.
- Bidirectional: Unlike traditional models that read text unidirectionally, BERT reads the entire sequence of words at once.
- Layers: Typically 12 layers (BERT Base) or 24 layers (BERT Large).

BERT

- Developed by Google.
- Bidirectional: Unlike traditional models that read text unidirectionally, BERT reads the entire sequence of words at once.
- Layers: Typically 12 layers (BERT Base) or 24 layers (BERT Large).

Pre-training Tasks:

- Masked Language Model (MLM): Randomly masks words in the sentence and predicts them.
- Next Sentence Prediction (NSP): Predicts if a given sentence logically follows another.

BERT

- Developed by Google.
- Bidirectional: Unlike traditional models that read text unidirectionally, BERT reads the entire sequence of words at once.
- Layers: Typically 12 layers (BERT Base) or 24 layers (BERT Large).

Pre-training Tasks:

- Masked Language Model (MLM): Randomly masks words in the sentence and predicts them.
- Next Sentence Prediction (NSP): Predicts if a given sentence logically follows another.

Fine-tuning: Adapts pre-trained BERT for various downstream tasks like question answering, sentiment analysis, etc.

GPT

- Next-token prediction (autoregressive model):

$$\max \sum_{i=1}^{n} \log P(x_i | x_1, \dots, x_{i-1}).$$

GPT

- Next-token prediction (autoregressive model):

$$\max \sum_{i=1}^{n} \log P(x_i | x_1, \ldots, x_{i-1}).$$

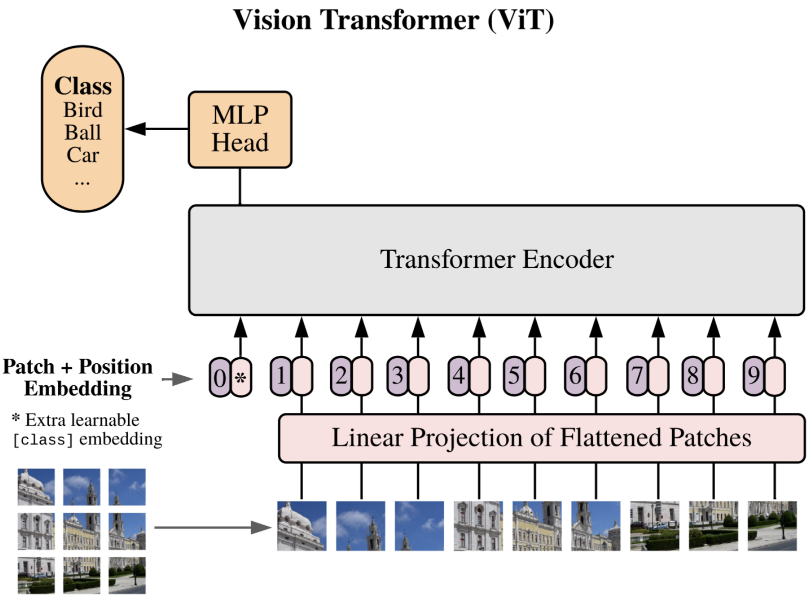
- Text Generation:

$$
\begin{aligned}
&\text{text} = [\langle bos \rangle] \text{ or } [\text{some context}]\\
&\text{while True:}\\
&\quad \text{logit} = \text{decoder}(\text{embed}(\text{text}))\\
&\quad \text{index} = \text{top}(\text{logit}[-1])\\
&\quad \text{token} = \text{vocabulary}(\text{index})\\
&\quad \text{if token} == \langle eos \rangle :\\
&\qquad \text{break}\\
&\quad \text{text.append}(\text{token})\\
&\text{return text}
\end{aligned}
$$

Practice

- Pre-train models in large dataset. Fine-tune models on down-stream tasks.
- Fine-tuning needs to retrain our model, which is not user-friendly.
- Next-token prediction enables capability of doing **in-context learning**.
  https://chat.openai.com/share/75d354d5-5a4d-4877-8aa2-04093506ca20
- Prompt!

Vision Transformer (ViT)



Summary

- Transformers or attention-based models are versitile in many applications.
- Next-token prediction is powerful and **it implicitly performs multi-task learning**. The latter might be the major reason of why GPT is so successful.