GPT Structure

Changshi Li

Wuhan University

2024.3.6

Contents

Training datasets

OpenAI has made significant contributions to the field of AI research and development, especially in the area of language models. Its flagship product, GPT, has revolutionized the way people use AI technology and has paved the way for future development in AI.
Setting Training sequence $T = 3$, then training datasets are as follows:

Table: Training datasets.

Meaning	Input	Goal output
tokens	OpenAI has made	has made significant
code	1 2 3	2 3 4
tokens	language models.	models. Its
code	12 13 14	13 14 15
tokens	, has revolutionized	revolutionized the
code	16 2 17	2 17 18
	...	...

Table: The code is the number of the token in the directory.

Optimization goals

Optimization goals in GPT[1]:
Let dataset be $\mathcal{X}$, then

$$\prod_{i}^{N} P(x_i|x_{i-T:i-1})$$

Optimization goals in reality:
Let dataset be $\mathcal{X} = \{X_1, \cdots, X_N\}$, where $X_i = \{x_1^{(i)}, \cdots, x_T^{(i)}\}$, then

$$\prod_{i=1}^{N} P(X_i|x_1^{(i)}),$$

where

$$\begin{aligned} P(X_i|x_1^{(i)}) &= \prod_{j=1}^{T} P(x_j^{(i)}|x_{1:j-1}^{(i)}) \\ &= P(x_T^{(i)}|x_{1:j-T}^{(i)})P(x_{T-1}^{(i)}|x_{1:T-2}^{(i)}) \cdots P(x_2^{(i)}|x_1^{(i)}) \end{aligned}$$

[1] Language Models are Unsupervised Multitask Learners. OpenAI blog. 2019.

Contents

Math format

Let $X \in \{0,1\}^{T\times d}$ for input, where d is the dimension of tokens and T is the length of inputs.Then the **embedding layer** is defined as:

$$\mathrm{Enc}(X) = XE_1, \tag{1}$$

where $E \in \mathbb{R}^{d\times D}$.
The Enc module plus **positional encoding** reconstruct a new Enc module:

$$\mathrm{Enc}(X) = XE_1 + PE_2, \tag{2}$$

where $P \in \{0,1\}^{T\times T}$, $P_{i,i} = 1, P_{i\neq j} = 0$ and $E_2 \in \mathbb{R}^{T\times D}$
Remark: $\mathrm{Enc}(X) : \{0,1\}^{T\times d} \to \mathbb{R}^{T\times D}$.
Let $X \in \mathbb{R}^{T\times D}$ for the middle input, where D is the dimension of middle layer.
The **norm layer** is defined as:

$$\mathrm{Norm}(X) := \frac{X - \mathbb{E}X}{\mathrm{Var}(X)}, \tag{3}$$

where $\mathrm{E}(X), \mathrm{Var}(X) \in \mathbb{R}^{T\times D}$.
Norm: $\mathrm{ATT} : \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times D}$.

Math format

Let H be the head number of multi-head attention, for the h-th head $W_{K,h} \in \mathbb{R}^{D\times D_K}$, $W_{Q,h} \in \mathbb{R}^{D\times D_K}$, $W_{V,h} \in \mathbb{R}^{D\times D_V}$ and $W_O \in \mathbb{R}^{HD_V\times D}$. Then the **multi-head attention** is defined as:

- Single head attention:

$$S_h = \mathrm{Softmax}(\frac{XW_{Q,h}(XW_{K,h})^T}{\sqrt{D_K}})XW_{V,h}, \tag{4}$$

- Masked single head attention:

$$S_h = \mathrm{Softmax}(\frac{\mathrm{Mask}(XW_{Q,h}(XW_{K,h})^T)}{\sqrt{D_K}})XW_{V,h}, \tag{5}$$

  where $\mathrm{Mask}(X)$ is an assignment operator, defined as $X_{i>j} = -\infty$.
  Remark: $\mathrm{S_h} : \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times D_V}$.

- Concatenate & Residual

$$\mathrm{ATT}(X) := X + [S_1, \ldots, S_h, \ldots, S_H]W_O, \tag{6}$$

where $\mathrm{Softmax}(X)$ is row-wise.
Remark: $\mathrm{ATT} : \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times D}$.

Math format

The **MLP** layer is defined as:

$$\mathrm{MLP}(X) := X + f_L(X) \circ \sigma \circ \cdots \sigma \circ f_1(X), \tag{7}$$

where $f_i(X) := XW_i + b_i$, $b_i \in \mathbb{R}^{D_i}$, $W_1 \in \mathbb{R}^{D \times D_1}$, $W_i \in \mathbb{R}^{D_{i-1} \times D_i}$ and $W_L \in \mathbb{R}^{D_{L-1} \times D}$.

Remark: $\mathrm{ATT} : \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$.

For each element x, the **active function** σ is defined as:

$$\begin{aligned} \mathrm{GeLU}(x) &:= xP(X \leq x) \\ &= x \int_{-\infty}^{x} \frac{\exp(\frac{-(t-\mu)^2}{2\sigma^2})}{\sqrt{2\pi}\sigma} dt \\ &\simeq 0.5x(1 + \tanh(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3))). \end{aligned} \tag{8}$$

Math format

The **block** in GPT is defined as:

$$\mathrm{Block}(X) := \mathrm{MLP} \circ \mathrm{Norm} \circ \mathrm{ATT} \circ \mathrm{Norm}(X) \tag{9}$$

The **embedding** is defined as:

$$X^e := \mathrm{Norm} \circ \mathrm{Block}_M \cdots \mathrm{Block}_1 \circ \mathrm{Enc}(X). \tag{10}$$

The **GPT** architecture is defined as:

$$\mathrm{GPT}(X) := \arg\max_{\mathrm{index}} X^l = X^e W_{\mathrm{head}}, \tag{11}$$

where $W_{\mathrm{head}} \in \mathbb{R}^{D \times d}$.
Remark: $\mathrm{ATT} : \{0,1\}^{T \times d} \to \mathbb{R}^{T \times d}$.

GPT parameters

Table: Parameter setting.

Parameter	GPT-2(125M)	GPT-3/3.5(175B)	GPT-4(1800B)
d (vocab_size)	50304	*	*
T (block_size)	1024	2048	8000(p.t.)->32000(f.t.)
D (n_embd)	768	12288	*
D_V (n_embd)	768/12=64	12288/96 =128	*
D_K (n_embd)	768/12=64	12288/96 =128	*
H (n_head)	12	96	*
L (MLP layer)	2	2	*
W_1 (first layer)	$\mathbb{R}^{4D\times D}$	$\mathbb{R}^{4D\times D}$	*
W_2 (second layer)	$\mathbb{R}^{D\times 4D}$	$\mathbb{R}^{D\times 4D}$	*
M (n_layer)	12	96	120
N(data_number)	40G	570G	*

Loss functions

Then the loss function is:

$$-\sum_{i=1}^{T}\sum_{j=1}^{d} P_{ij} \log \mathrm{Softmax}(X^l)_{ij} \tag{12}$$

, where P_i is the probability of the next token. or

$$-\sum_{i=1}^{T} \log \mathrm{Softmax}(X^l)_{ij_{\mathrm{true}}} \tag{13}$$

, where j_{true} is the next token in the directory.
How to train GPT?
Training with or without $< /s >$? The influence is small.

Why decoder can reflect the optimization goals

GPT can predict the next token based on former tokens.

- MLP: $XW = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} W = \begin{pmatrix} X_1 W \\ X_2 W \end{pmatrix}$.
- ATT1: $XW_Q(XW_K)^T = \begin{pmatrix} X_1 W_Q \\ X_2 W_Q \end{pmatrix} \begin{pmatrix} (X_1 W_K)^T & (X_2 W_K)^T \end{pmatrix} =$
  $\begin{pmatrix} X_1 W_Q (X_1 W_K)^T & X_1 W_Q (X_2 W_K)^T \\ X_2 W_Q (X_1 W_K)^T & X_2 W_Q (X_2 W_K)^T \end{pmatrix} := \begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix}$.
- ATT2: $\mathrm{Mask}(\begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix}) = \begin{pmatrix} \mathrm{Mask}(s_{11}) & -\infty \\ s_{21} & \mathrm{Mask}(s_{22}) \end{pmatrix}$.
- ATT3:
  $\mathrm{Softmax}(\mathrm{Mask}(\begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix})) = \begin{pmatrix} \mathrm{SM}(s_{11}) & 0 \\ s_{21} & \mathrm{SM}(s_{22}) \end{pmatrix} := \begin{pmatrix} p_{11} & 0 \\ p_{21} & p_{22} \end{pmatrix}$.
- ATT4: $\begin{pmatrix} p_{11} & 0 \\ p_{21} & p_{22} \end{pmatrix} \begin{pmatrix} X_1 W \\ X_2 W \end{pmatrix} = \begin{pmatrix} p_{11} X_1 W \\ p_{21} X_1 W + p_{22} X_2 W \end{pmatrix}$.
- Norm: Normalized for a token not among tokens.

Contents

Prediction process

Set $T = 5$.
Input: OpenAI has made
Output: has made **significant**
Combine the last word of Output to Input, obtain the next Input.
Input: OpenAI has made significant
Output: has made significant contributions
Input: OpenAI has made significant contributions
Output: has made significant contributions to
Input: OpenAI has made significant contributions to
The sequence context is growing too long, which longer than the inputs length (5) in training, we must crop it at T.
Croped input: has made significant contributions to
Output: made significant contributions to the
Croped input: made significant contributions to the
Output: significant contributions to the field
...
Complete the sequence T(max_new_tokens) times.

Random prediction

Set Top_K = 3, temperature.
Input: OpenAI
Output: X^{l} =$\{0.001, 10, 6, 1, 4, \cdots\}$
1 (0.001) 2 (10) 3(6) 4(1) 5 (4)
Select Top_K: 2(10), 3(6), 5(4)
Div temperature and calculate new probability based on the Top_K tokens:
2(10/temperature), 3(6/temperature), 5(4/temperature)

- If temperature = 1, calculate new probability with Softmax: 2(0.97962921), 3(0.01794253), 5(0.00242826).
- If temperature = 10, calculate new probability with Softmax: 2(0.45062671), 3(0.30206411), 5(0.24730918).

Select new code with their own new probability.

- If temperature = 1, select 3 with probability 0.01794253.
- If temperature = 10, select 3 with probability 0.30206411.

According to the directory get Random Output:
OpenAI (1) has (2) made(3) significant(4) contributions (5).

Contents

RLHF

RLHF[2]

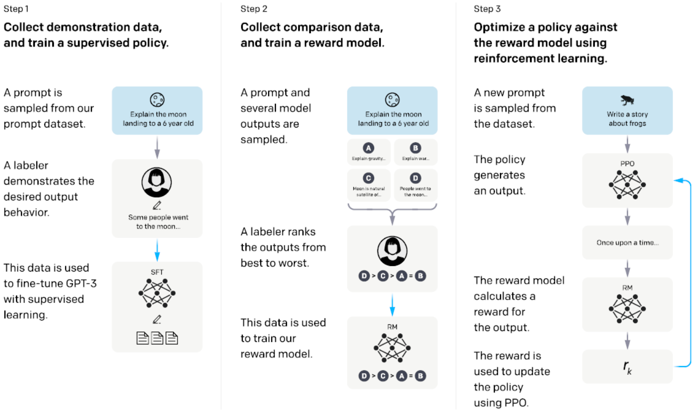


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

[2] Training language models to follow instructions with human feedback. NeurIPS. 2022.

RLHF

Why fine-tune on LLM in Step1 ?
How to calculate the gradient based on the generated sequence ?
Let $\mathcal{G} : \{< /s >, X\} \to \{Y, < /e >\} := \{< /s >, x_1, \cdots, x_T\} \to \{x_{T+1}, \cdots, < /e >\}$. (Without gradient)
Let $\mathcal{F} : \{< /s >, x_1, \cdots, x_t\} \to \{x_1, \cdots, x_{t+1}\}$. (With gradient)
Loss function in Step 2: train RM.

$$\mathrm{loss}(\theta) = -\frac{1}{C_K^2}\mathbb{E}_{(X,Y_w,Y_l)\sim D}\big[\log(\sigma(r_\theta(X, Y_w) - r_\theta(X, Y_l)))\big]. \tag{14}$$

Loss function in Step 3: RLHF(reinforcement learning with human feedback).

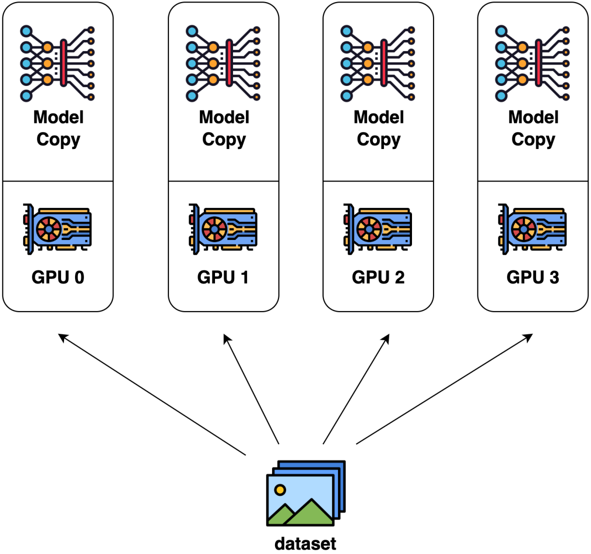
$$\begin{aligned}\mathrm{objective}(\phi) = \mathbb{E}_{(X,Y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(X, Y) - \beta\log(\pi_\phi^{\mathrm{RL}}(Y|X)/\pi^{\mathrm{SFT}}(Y|X))\right] + \\ \gamma\mathbb{E}_{X\sim D_{\mathrm{pretrain}}}\big[\log(\pi_\phi^{\mathrm{RL}}(X))\big].\end{aligned} \tag{15}$$

RLHF

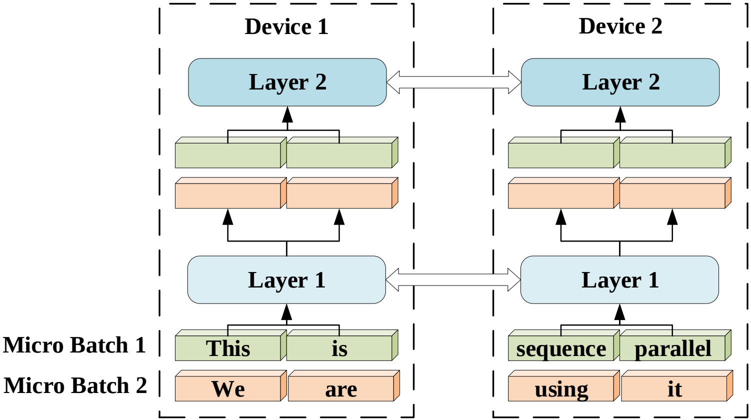
- $\pi_\phi^{\mathrm{RL}}(Y|X)$:
  - $\mathcal{G} : X \to Y$.
  - $\mathcal{F} : [< /s >, X, Y] \to [X, Y, < /e >]$.
  - Obtain the probability of Y.
- $\pi_\phi^{\mathrm{RL}}(X)$:
  - $\mathcal{F} : [< /s >, X] \to [X, < /e >]$.

Contents

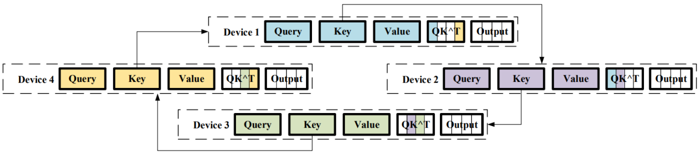
Data Parallel



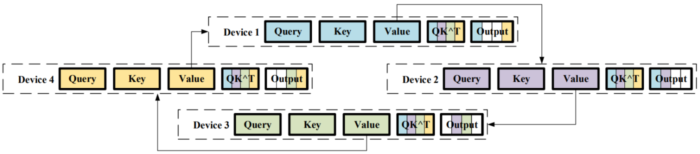
Sequence Parallel



Sequence Parallel

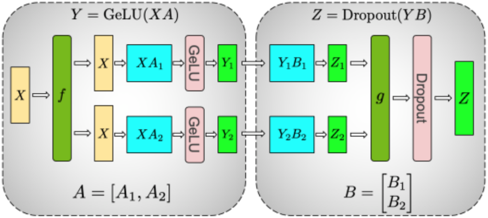


(a) Transmitting key embeddings among devices to calculate attention scores

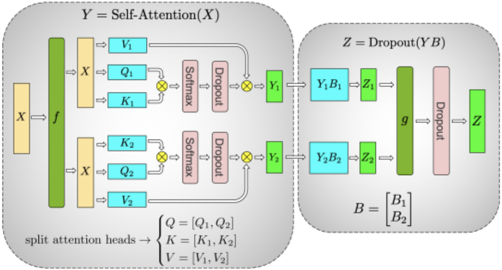


(b) Transmitting value embeddings among devices to calculate the output of attention layers

Tensor Parallel

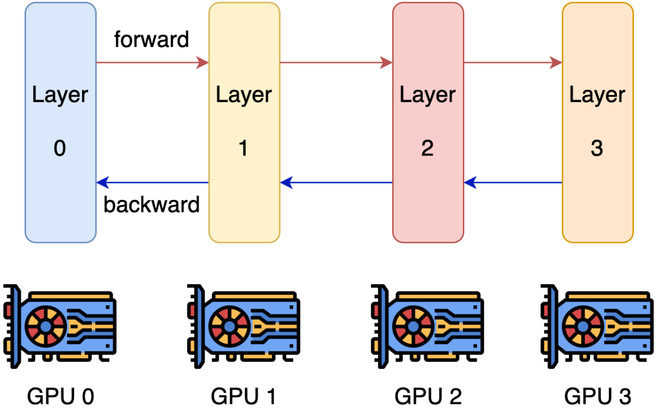


(a) MLP

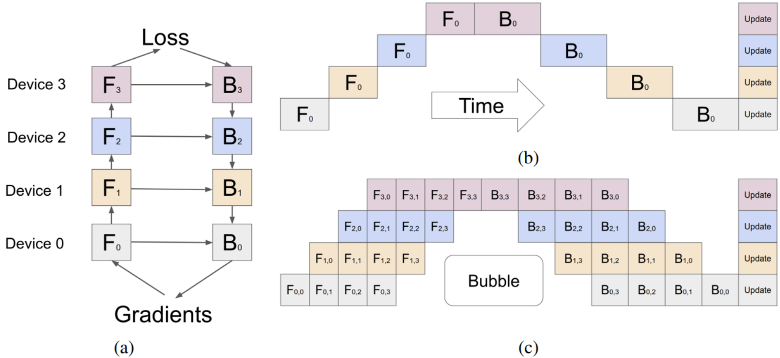


(b) Self-Attention

Model Parallel



Pipeline Parallel



(a) (b) (c)

GPT-4 Training

GPT-4[3]

- 8-way tensor parallelism.
- 15-way pipeline parallelism.
- 25,000 A100 GPU for 90 to 100 days at about 32% to 36% MFU.
- Vision: the architecture is similar to Flamingo.
- ...

[3] Gpt-4 technical report. 2023.
GPT-4 Architecture, Infrastructure, Training Dataset, Costs, Vision, MoE. 2023

THANKS!