

大模型系列 - 微调

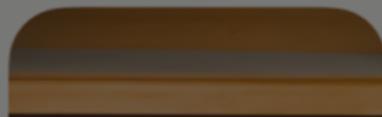
# 微调系列介绍

in the Acropolis



ZOMI

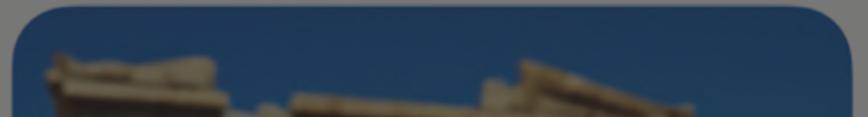
in a doghouse



in a bucket



getting a haircut



príqseelz

prímmíewz



# 大模型业务全流程



# 关于大模型 – 微调系列

## • 具体内容

- **大模型微调基础**：微调基本介绍 -- 大模型训练流程 -- 微调使用场景
- **微调算法 I**：迁移学习 -- Adapters Tuning 算法 -- LoRA 算法
- **微调算法 II**：IA3 算法 -- P-Tuning 微调 -- Prefix Tuning 算法
- **微调算法 III**：Prompt Tuning 提示微调 -- Instruction Tuning 指令微调
- **微调算法比较**：Instruction vs Prompt Tuning -- 低参微调 vs 全参微调

# 关于本内容

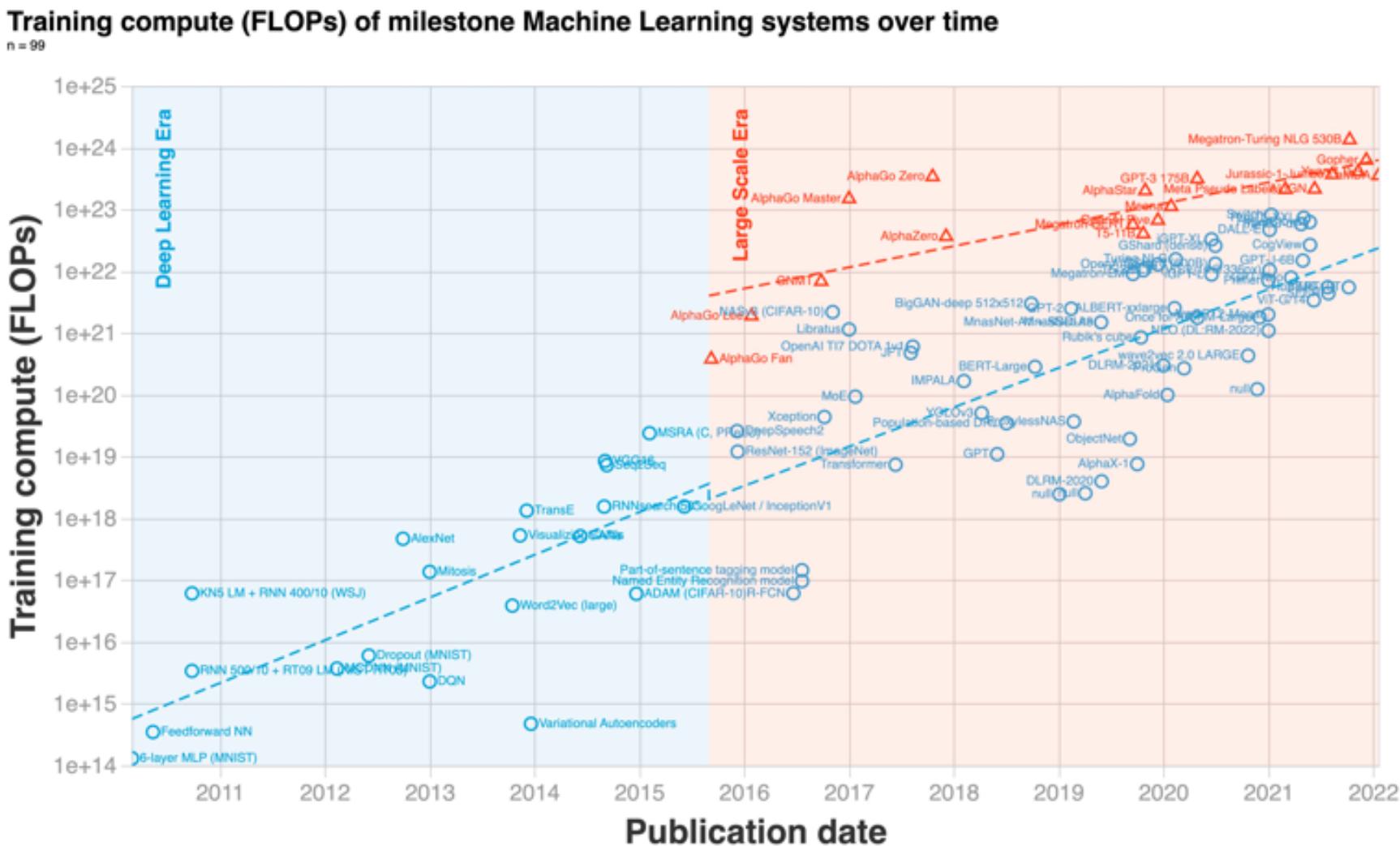
- **大模型微调基础**

- 大模型与微调基本介绍 - Basic Introduction to LLM and fine-tuning
- 微调技术的区别 - Full parameter vs Parameter Efficient Fine-Tuning
- 大模型训练流程 - Large model training process
- 微调使用场景 - Fine-tuning usage scenarios

# 1. 大模型与微调

## 基本介绍

# 模型参数与数据集规模增长



# 模型参数与数据集规模增长

- 从趋势看 LLM 和 LVM 的大模型参数量只会越来越大。
- 训练更大的模型需要更多的数据集，随着模型的增长，训练过程必须调整更多的参数。
- 对于AI集群和算力的消耗会更多，也会不断提升成本。

Date	Model	Model size	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ <sup>+</sup> 21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ <sup>+</sup> 21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH <sup>+</sup> 23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH <sup>+</sup> 23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND <sup>+</sup> 22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ <sup>+</sup> 23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK <sup>+</sup> 23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX <sup>+</sup> 23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX <sup>+</sup> 23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ <sup>+</sup> 23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ <sup>+</sup> 23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF <sup>+</sup> 23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG <sup>+</sup> 23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG <sup>+</sup> 23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG <sup>+</sup> 23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ <sup>+</sup> 23]	16B	1T	57.3%	51.8%
2023 Jun	<b>phi-1</b>	1.3B	7B	50.6%	55.5%

Table 1: We use self-reported scores whenever available. Despite being trained at vastly smaller scale, **phi-1** outperforms competing models on HumanEval and MBPP, except for GPT-4 (also WizardCoder obtains better HumanEval but worse MBPP).

# 模型参数与数据集规模增长：引入微调

## 模型微调：

- 允许少量地重新调整预训练大模型的权重参数，或者大模型的输入和输出
- 有助于降低训练大模型的复杂性，降低重新进行训练的成本。

Date	Model	Model size	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ <sup>+</sup> 21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ <sup>+</sup> 21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH <sup>+</sup> 23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH <sup>+</sup> 23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND <sup>+</sup> 22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ <sup>+</sup> 23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK <sup>+</sup> 23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX <sup>+</sup> 23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX <sup>+</sup> 23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ <sup>+</sup> 23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ <sup>+</sup> 23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF <sup>+</sup> 23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG <sup>+</sup> 23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG <sup>+</sup> 23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG <sup>+</sup> 23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ <sup>+</sup> 23]	16B	1T	57.3%	51.8%
2023 Jun	<b>phi-1</b>	1.3B	7B	50.6%	55.5%

Table 1: We use self-reported scores whenever available. Despite being trained at vastly smaller scale, **phi-1** outperforms competing models on HumanEval and MBPP, except for GPT-4 (also WizardCoder obtains better HumanEval but worse MBPP).

# 2. 微调技术的区别

# 概念区分

- PEFT Parameter Efficient Fine-Tuning , 参数高效微调，通常俗称低参微调
- Full Parameter Fine-Tuning , 全参微调，俗称普通微调
- Supervised Fine-Tuning , SFT , 有监督微调
- Instruction Tuning , 指令微调
- RLHF , Reinforcement Learning Human Feedback , 人类反馈强化学习

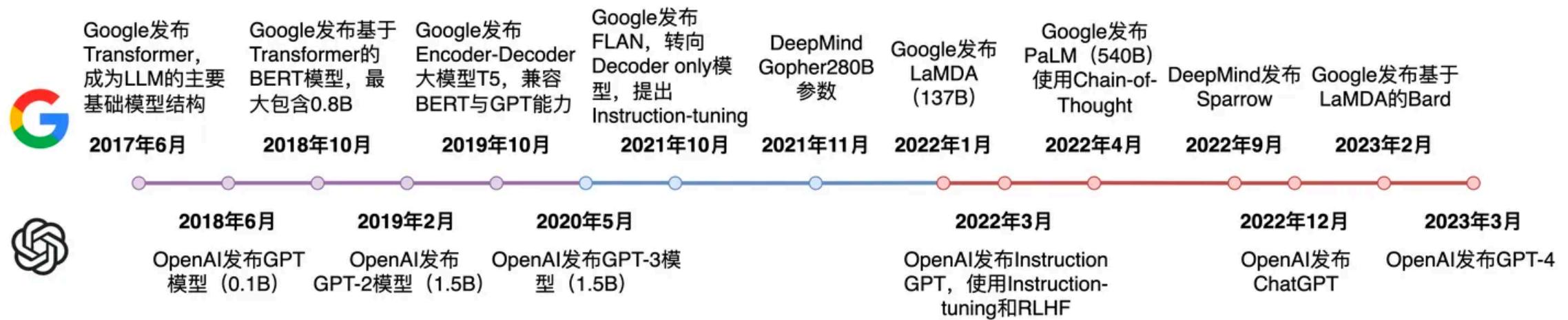
按微调参数规模划分

按训练的流程来划分

按训练的方式来划分

# 大模型与微调的发展

- LLM and LVM 预训练大模型的训练成本非常高昂，需要庞大的计算资源（AI集群）和大量的数据，一般客户难以承受。



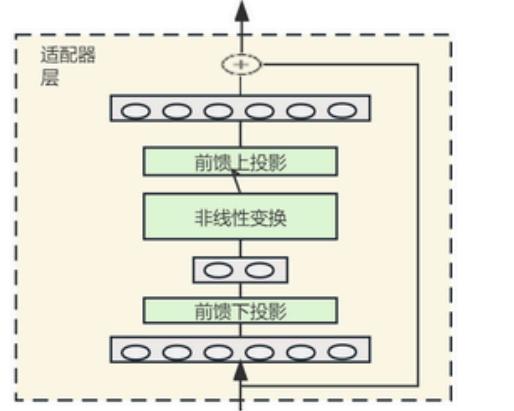
# 1. 按微调参数规模划分

- Full Parameter Fine-Tuning 全参数微调 和 Parameter Efficient Fine-Tuning PEFT , 参数高效微调：
  - **FPFT** : 用预训练模型作为初始化权重，在特定数据集上继续训练，全部参数都更新的方法
  - **PEFT** : 用更少的计算资源完成模型参数的更新，包括只更新一部分参数，或者通过对参数进行某种结构化约束，例如稀疏化或低秩近似来降低微调的模型参数量。

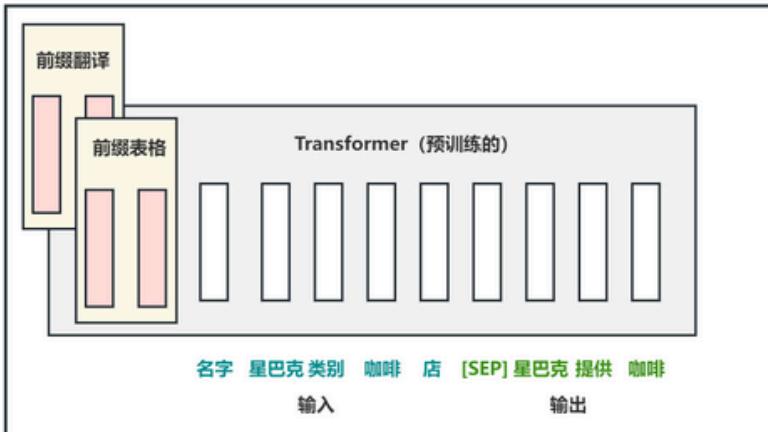
# 参数高效微调 PEFT

- **微调引入**：研究者开始研究 Parameter-Efficient Fine-Tuning ( PEFT ) ，旨在通过最小化微调网络模型中的参数数量和降低计算复杂度，来提高预训练模型在新任务上的性能，从而缓解大型预训练模型的训练成本。
- **好处**：即使计算资源受限，也可以利用预训练模型的知识来迅速适应新任务，实现高效的迁移学习 Transfer Learning。因此，PEFT 技术可以在提高模型效果的同时，缩短模型训练时间和计算成本，让更多研究者参与到 AI 研究中。

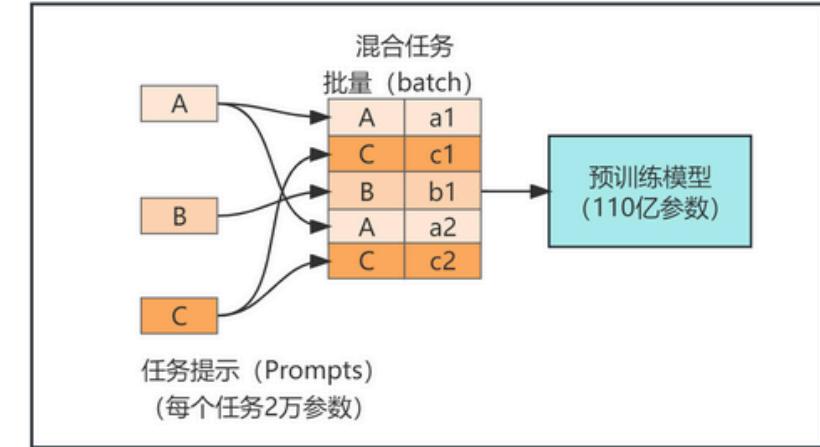
Adapter (谷歌2019)



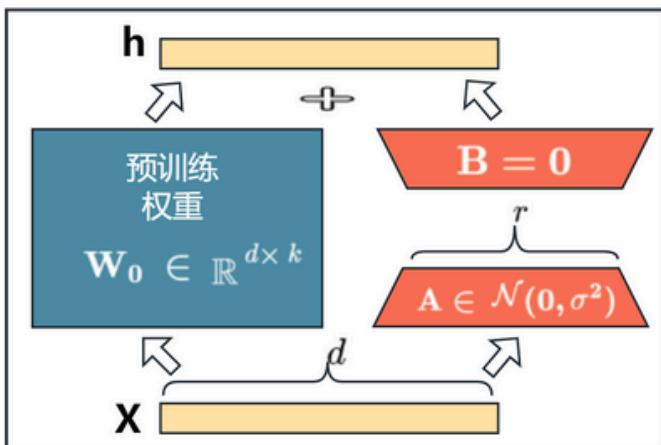
Prefix Tuning (斯坦福2021)



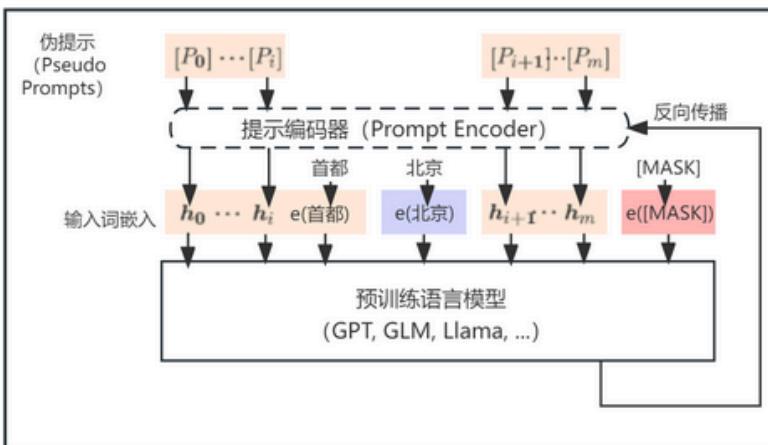
Prompt Tuning (谷歌2021)



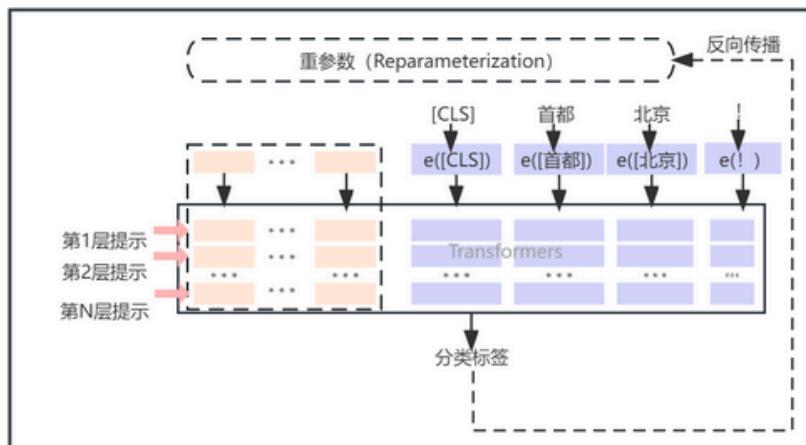
LORA (2021)



P-Tuning (清华2022)

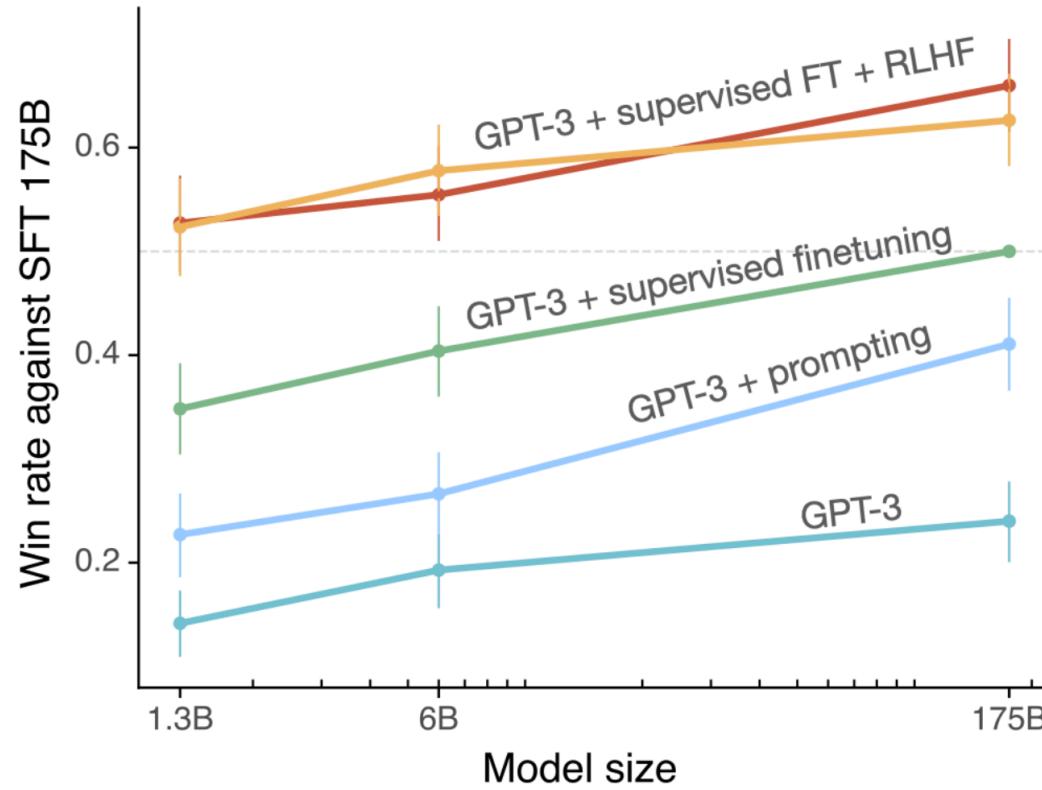


P-Tuning v2 (清华2022)

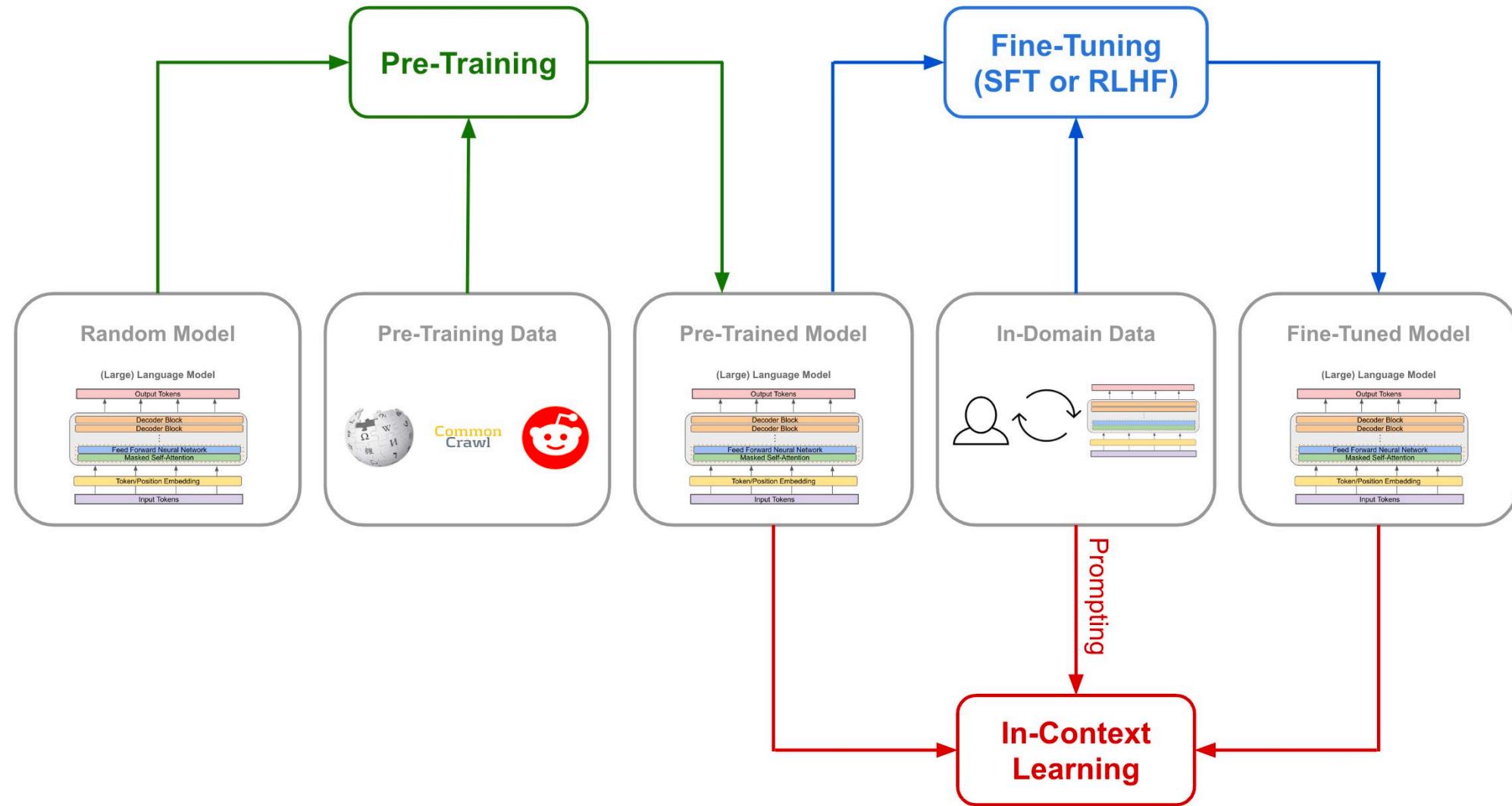


## 2. 按训练的流程来划分

- 按照大模型训练阶段进行微调，或者根据大模型微调的目标来区分，从提示微调 Prompt Tuning、有监督微调 SFT、RLHF 人类反馈强化学习的方式来划分。



## 2. 按训练的流程来划分



# In-Context Learning 上下文学习

- **ICL ( In-Context learning ) 上下文学习**：区别于普通微调 Fine-Tuning，不对 LLMs 执行任何的微调，直接将模型的输入输出拼接起来作为一个 prompt，引导模型根据输入的数据结构 demo，给出任务的预测结果。
- ICL 能够基于无监督学习的基础上取得更好模型效果，并且不需要根据特定的任务重新微调 Fine-Tuning 更新模型参数，避免不同任务要重新进行真正的微调。

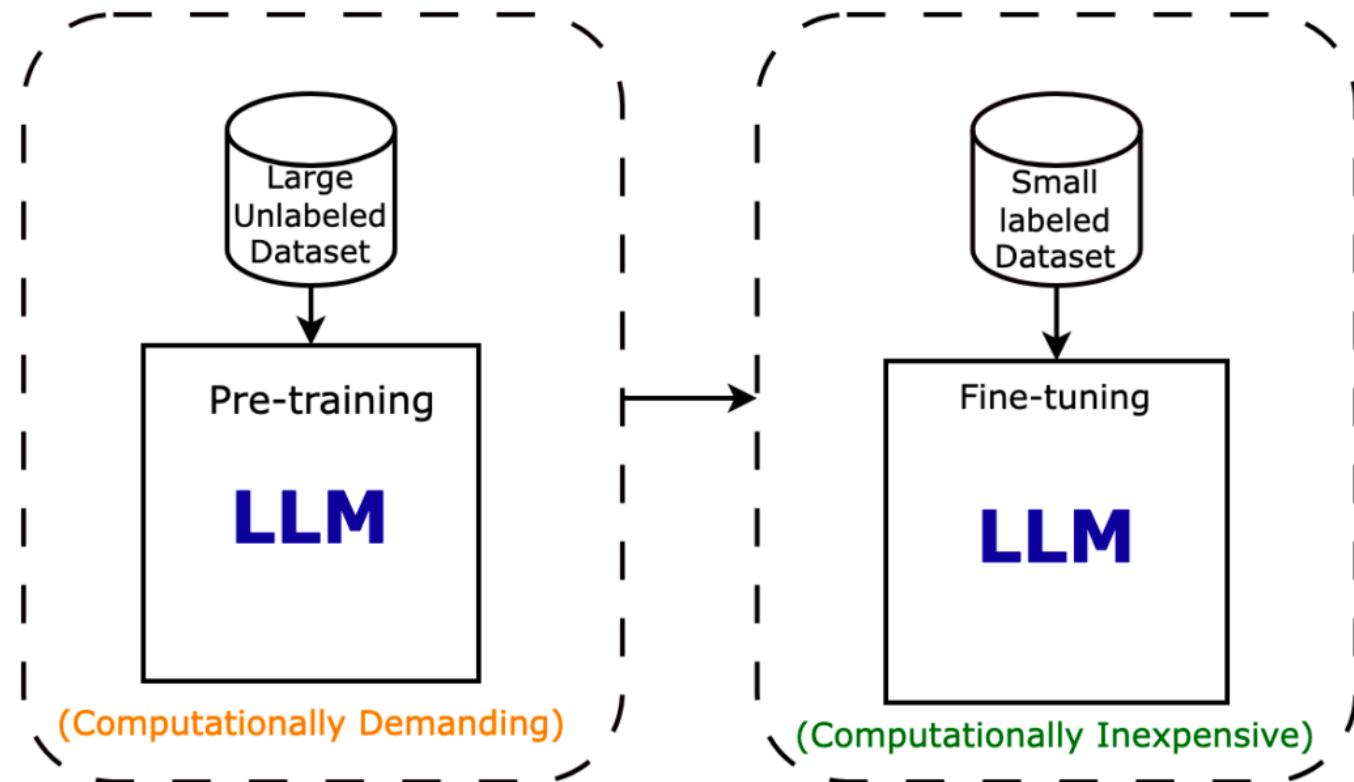
### 3. 按训练的方式来划分

#### 预训练 Pre-Training :

- LLMs 预训练过程是无监督的，但微调过程往往是有监督的。当进行有监督微调时，模型权重会根据与真实标签的差异进行调整。

#### Supervised fine-tuning , SFT :

- 有监督微调使用有标签的数据 (Label Data) 来调整已经预训练的 LLMs，使其更适应某一特定场景任务。



# 指令微调 Instruction Tuning

- 指令微调 Instruction Tuning 可以被视为有监督微调 SFT 的一种特殊形式
  - SFT：使用标记数据对预训练模型进行微调的过程，以便模型能够更好地执行特定任务。
  - IT：通过 <指令，输出> 对的数据集上进一步训练 LLMs 的过程，以增强 LLMs 能力和可控性。
- 特殊之处在于其数据集的结构，由人类指令和期望的输出组成进行配对。这种数据结构使得指令微调专注于让模型理解和遵循人类指令。作为**有监督微调的一种特殊形式，专注于通过理解和遵循人类指令来增强大型语言模型的能力和可控性。**

# 概念区分

按微调参数规模划分

按训练的流程来划分

按训练的方式来划分

- **PEFT Parameter Efficient Fine-Tuning** , 参数高效微调，通常俗称低参微调
- Full Parameter Fine-Tuning , 全参微调，俗称普通微调
- **Supervised Fine-Tuning , SFT** , 有监督微调
- Instruction Tuning , 指令微调
- RLHF , Reinforcement Learning Human Feedback , 人类反馈强化学习

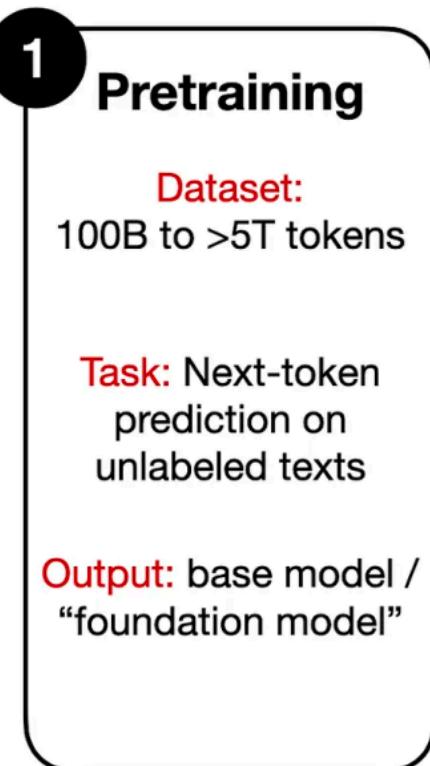
# 3. 大模型训练流程

# 经典 LLMs 训练流程

- 目前基于 Transformer 的大型语言模型，例如 ChatGPT 或 Llama 2，大体都包括三个训练步骤：  
**预训练 Pre-training，有监督微调 Supervised finetuning、对齐 Alignment。**
- 1 ) 预训练阶段，模型会学习来自海量、无标注文本数据集的知识；
- 2 ) 然后使用有监督微调的方式来细化模型，以便后期在推理的过程中更好地遵守特定指令；
- 3 ) 最后使用对齐技术使 LLMs 可以更有用且更安全地响应用户的提示 Prompt；

# 1. 预训练 Pretraining

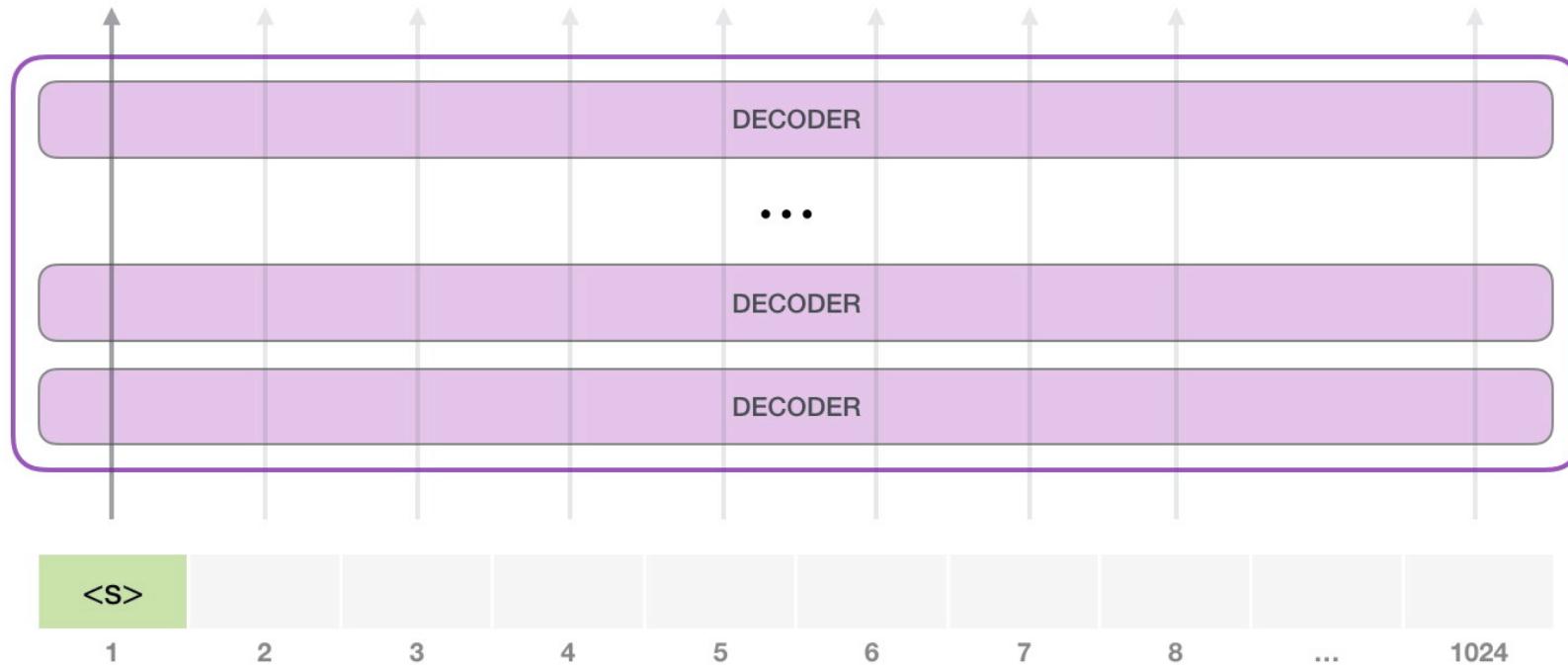
- 预训练阶段通常需要数十到百亿 Token 的文本语料库，但训练目标只是简单的「下一个单词预测」（next word prediction）任务。



Project Gutenberg (PG) is a volunteer effort to digitize and archive cultural works, as well as to "encourage the creation and distribution of eBooks." It was founded in 1971 by American writer Michael S. Hart and is the oldest digital library. Most of the items in its collection are the full texts of books or individual stories in the public domain. All files can be accessed for free under an open format layout, available on almost any computer. As of 3 October 2015, Project Gutenberg had reached 50,000 items in its collection of free eBooks.

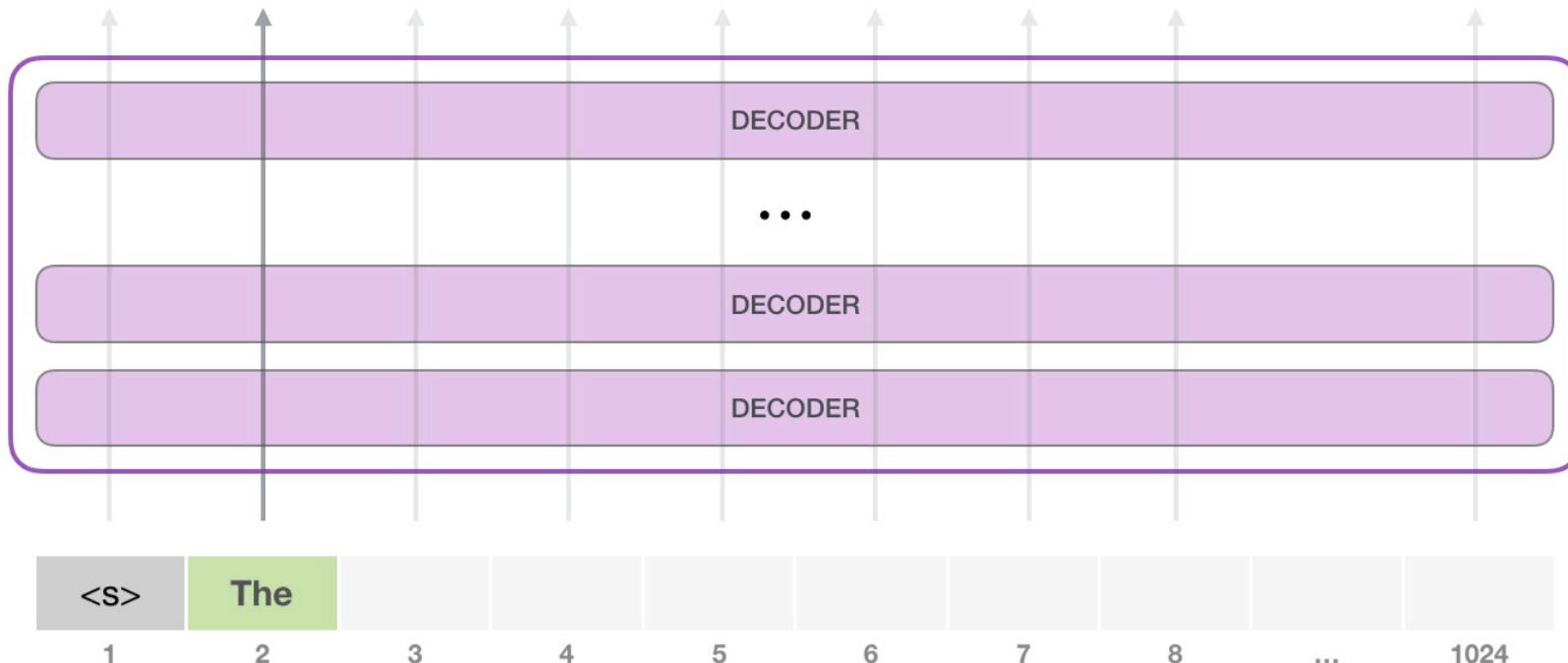
# 1. 预训练 Pretraining

- 预训练阶段通常需要数十到百亿 Token 的文本语料库，但训练目标只是简单的「下一个单词预测」（next word prediction）任务。



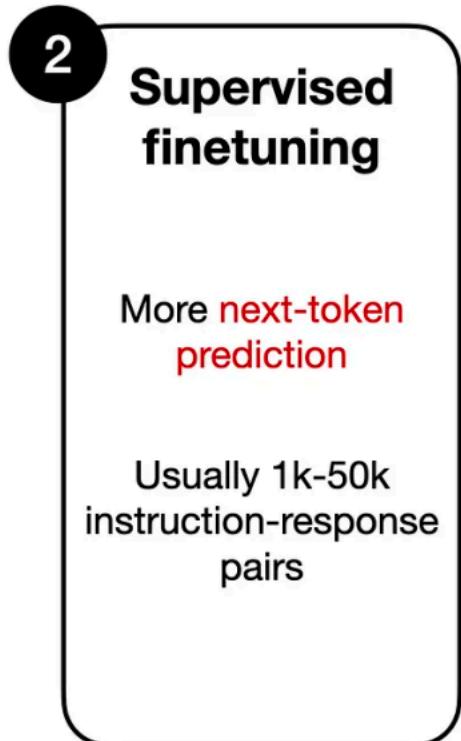
# 1. 预训练 Pretraining

- 自监督预训练（无监督学习）：让大模型从大规模数据中学习，不依赖人工标注完成训练，因为训练/学习的标签（Label Data）是文本的后续单词，已经暗含在训练数据集中。



## 2. 有监督微调 Supervised finetuning

- 第二阶段仍然是「next token prediction」任务，区别在于数据集，需要人工标注的指令数据集，模型输入是一个指令或者特殊的数据结构，输出为期望大模型的回复内容。



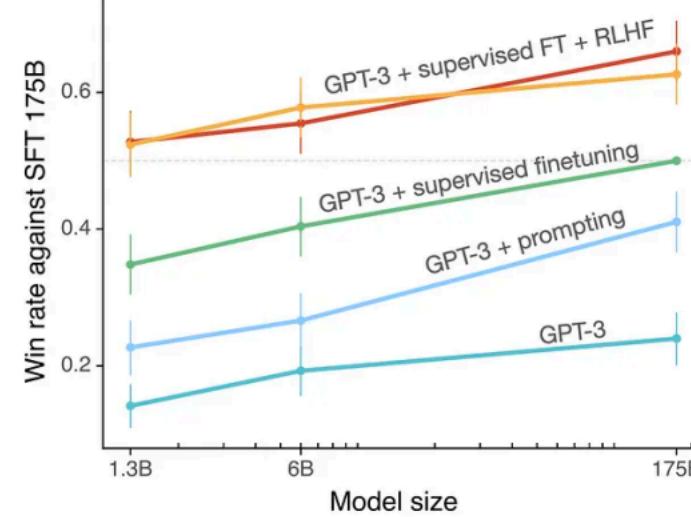
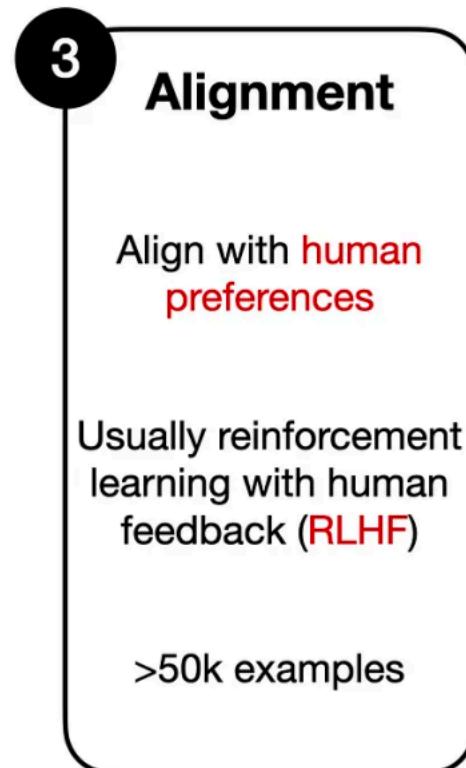
```
{  
    "instruction": "Write a limerick about a  
                    pelican.",  
    "input": "",  
    "output": "There once was a pelican so fine,  
             \nHis beak was as colorful as  
             sunshine,\nHe would fish all day,\nIn  
             a very unique way,\nThis pelican was  
             truly divine!\n\n"  
,  
  
{  
    "instruction": "Identify the odd one out from  
                    the group.",  
    "input": "Carrot, Apple, Banana, Grape",  
    "output": "Carrot\n\n"  
},
```

## 2. 有监督微调 Supervised finetuning

- 数据形式类似于：
  - Instruction: "Write a about a pelican."
  - Output: "There once was a pelican so fine..."
- 程序会将指令文本作为大模型的输入（Prompt and/or Instruction），并逐个 Token 输出，训练目标是与预期输出相同。
- 虽然 1 阶段和 2 阶段都采用「next token prediction」训练方式，但 SFT 的数据集通常比预训练数据小得多，指令数据集需要提供标注结果（如 RLHF），所以无法大规模应用。

### 3. 对齐 Alignment

- 第三阶段依然是微调，不过其主要目标在于将大模型与人类的偏好、价值观进行对齐，也是 RL HF 机制发挥作用的时候。



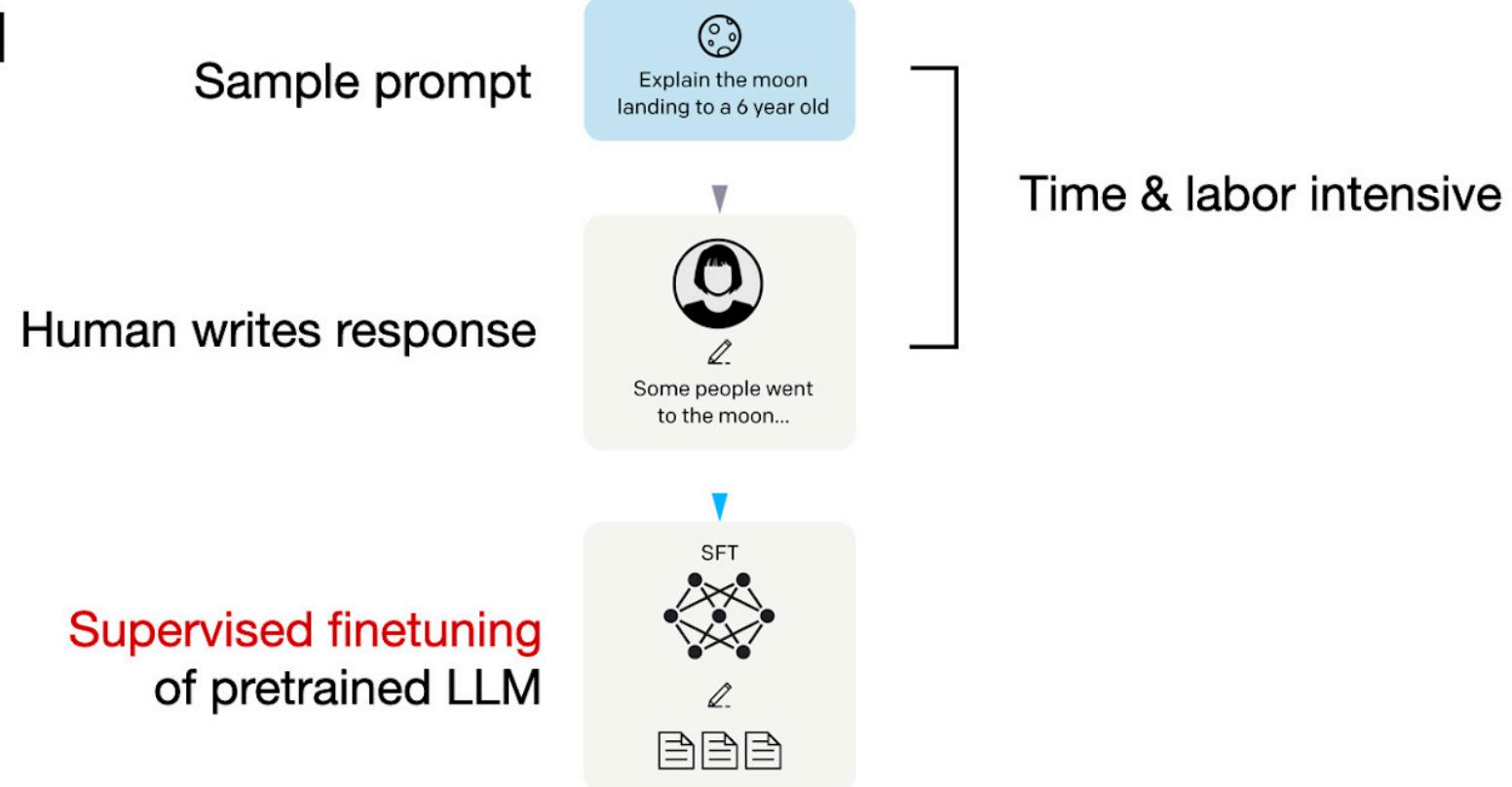
# RLHF 主要包括步骤

- Step 1 : 预训练模型的有监督微调 Supervised finetuning of the pretrained model
- Step 2 : 创建奖励模型 Creating a reward model
- Step 3 : PPO 进行微调 Finetuning via proximal policy optimization

# RLHF Step 1：预训练模型的有监督微调

- 收集提示词集合，并要求 Label 人员写出高质量的答案，然后使用该数据集以监督的方式微调预训练模型。

## RLHF Step 1

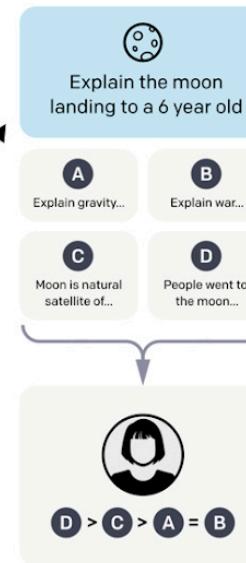
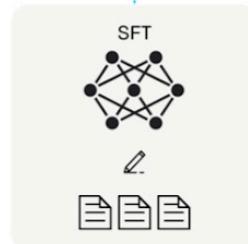


## RLHF Step 2：创建奖励模型

- 对于每个提示 Prompt，要求微调后的LLMs生成多个回复，再由标注人员根据真实的偏好对所有回复进行排序。接着训练奖励模型 RM 来学习人类的偏好，用于后续优化。

### RLHF Step 2

LLM finetuned in step 1:



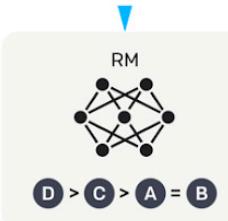
Sample prompt

Collect model responses

Human ranks responses

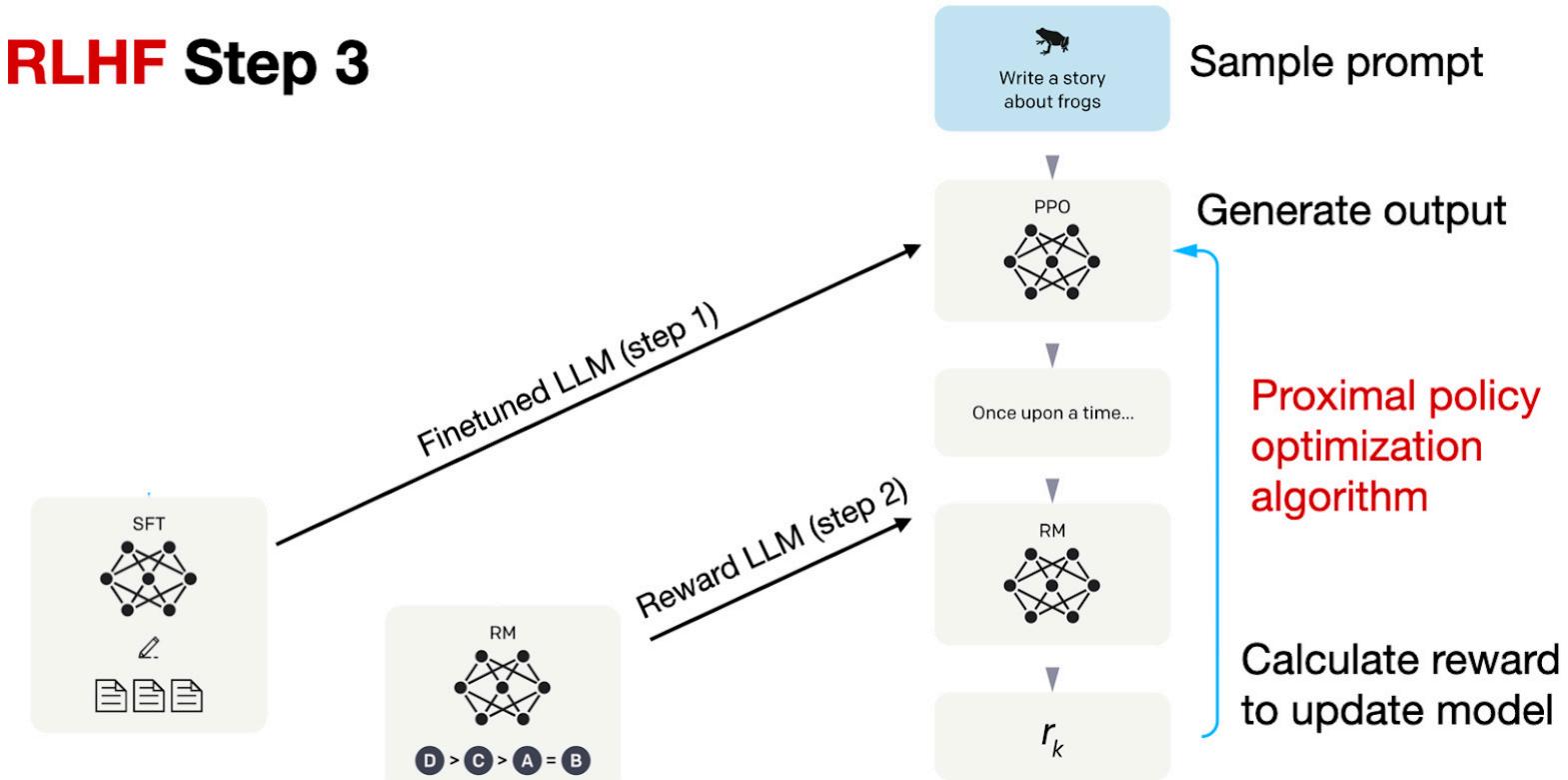
Time & labor intensive

Train reward model  
(Another LLM)



# RLHF Step 3 : PPO 进行微调

- 使用强化学习的算法（如 PPO , proximal policy optimization ）, 根据奖励模型 RM 提供的奖励分数，对 SFT 模型进一步优化用于后续的推理（文字生成）。



# 4. 微调使用场景

# 微调的优势

## 1. 定制化模型

- 通过微调大模型，可以根据用户自身的需求定制模型，从而提高准确性和性能。

## 2. 提高资源利用率

- 通过减少从头开始构建新模型的方式进行预训练，从而节省时间、算力资源和其他带来的成本。

## 3. 性能提升

- 微调的过程，可以使用用户的独特数据集，来增强预训练模型的性能。

## 4. 数据优化

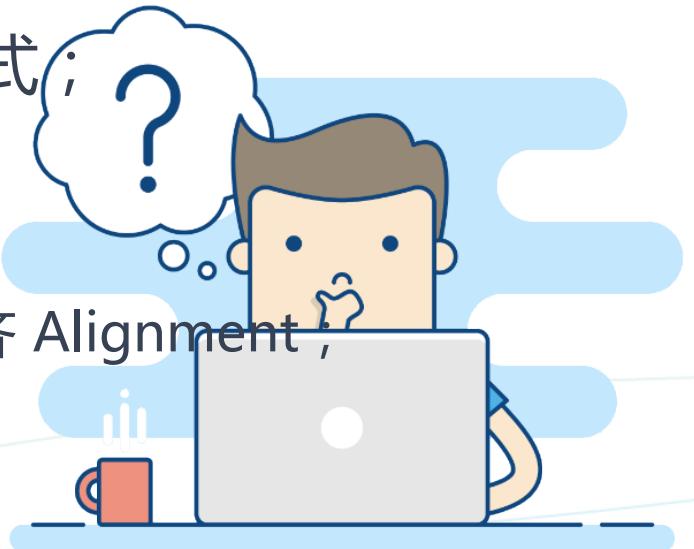
- 可以充分利用客户的数据，调整大模型以更好地适应用特定数据场景，甚至在需要时合并新数据。

# 小结 & 思考



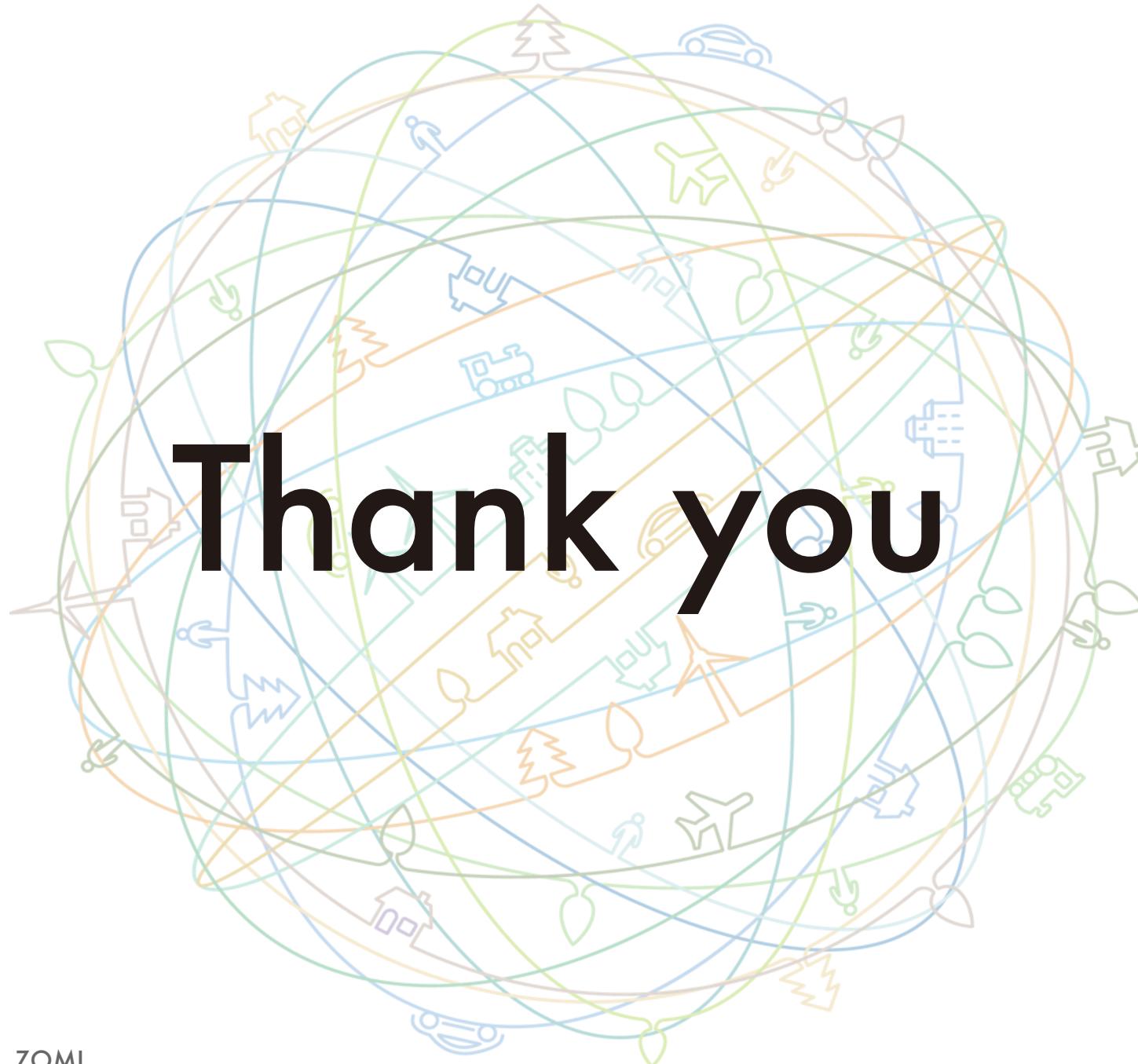
# 小结

1. 了解到了通过微调可以更好地提升大模型的效果；
2. 微调按照按微调参数规模划分，可以分为全参微调和低参微调 PEFT；
3. 微调按照训练的流程，可以划分为微调（SFT & RLHF）和上下文学习；
4. 指令微调（IT）属于有监督微调（SFT）的一种特殊方式；
5. 大模型训练流程主要包括三大步骤：
  - 预训练 Pre-training，有监督微调 Supervised finetuning、对齐 Alignment；



# Reference 引用&参考

1. [Asynchronous Methods for Deep Reinforcement Learning](#) (2016) by Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, and Kavukcuoglu introduces policy gradient methods as an alternative to Q-learning in deep learning-based RL.
2. [Proximal Policy Optimization Algorithms](#) (2017) by Schulman, Wolski, Dhariwal, Radford, and Klimov presents a modified proximal policy-based reinforcement learning procedure that is more data-efficient and scalable than the vanilla policy optimization algorithm above.
3. [Fine-Tuning Language Models from Human Preferences](#) (2020) by Ziegler, Stiennon, Wu, Brown, Radford, Amodei, Christiano, Irving illustrates the concept of PPO and reward learning to pretrained language models including KL regularization to prevent the policy from diverging too far from natural language.
4. [Learning to Summarize from Human Feedback](#) (2020) by Stiennon, Ouyang, Wu, Ziegler, Lowe, Voss, Radford, Amodei, Christiano introduces the popular RLHF three-step procedure that was later also used in the [InstructGPT paper](#).
5. <https://magazine.sebastianraschka.com/p/llm-training-rlhf-and-its-alternatives>
6. <https://www.mercity.ai/blog-post/fine-tuning-llms-using-peft-and-lora>
7. <https://mp.weixin.qq.com/s/oWmMglkaqkxDfoyDRrQ4Q>
8. <https://zhuanlan.zhihu.com/p/627642632>



把AI系统带入每个开发者、每个家庭、  
每个组织，构建万物互联的智能世界

Bring AI System to every person, home and  
organization for a fully connected,  
intelligent world.

Copyright © 2023 XXX Technologies Co., Ltd.  
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. XXX may change the information at any time without notice.



Course [chenzomi12.github.io](https://chenzomi12.github.io)

GitHub [github.com/chenzomi12/DeepLearningSystem](https://github.com/chenzomi12/DeepLearningSystem)