深度之眼
deepshare.net

参数高效微调(PEFT)实用讲解

导师：机器猫

本课内容

Pre-knowledge reserve

1. 背景与定义

参数高效微调技术的定义及其意义

2. 基本理论

常见的参数高效微调算法及其基本原理

3. 模型微调基本流程

面向特定目标的大模型微调基本流程

深度之眼
deepshare.net

目 录

研究背景

Research background

Google与OpenAI在LLM领域的发展时间线：https://hub.baai.ac.cn/view/23989

研究背景

Research background

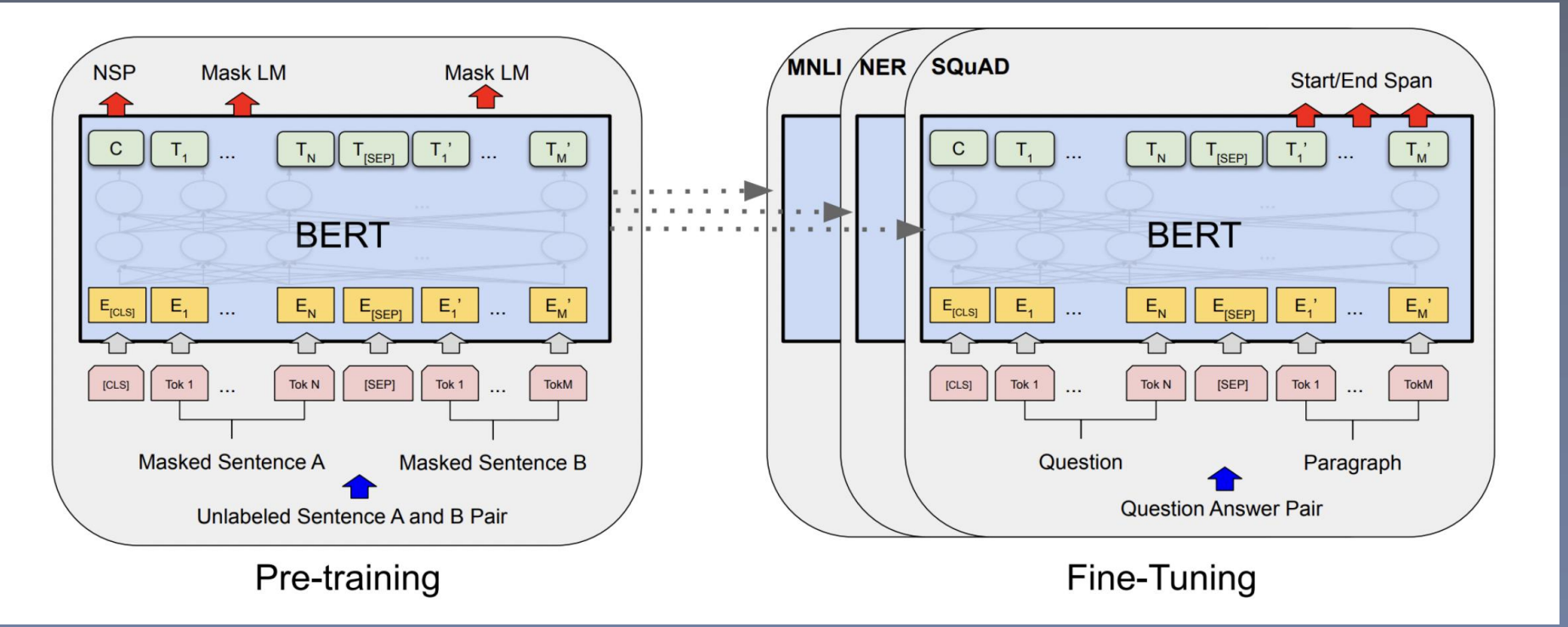
- **怎样使预训练语言模型使用不同下游任务？**

- Automatic speech recognition
- CCG
- Common sense
- Constituency parsing
- Coreference resolution
- Data-to-Text Generation
- Dependency parsing
- Dialogue
- Domain adaptation
- Entity linking
- Grammatical error correction
- Information extraction
- Intent Detection and Slot Filling
- Language modeling
- Lexical normalization
- Machine translation
- Missing elements
- Multi-task learning
- Multi-modal
- Named entity recognition
- Natural language inference
- Part-of-speech tagging
- Paraphrase Generation
- Question answering
- Relation prediction
- Relationship extraction
- Semantic textual similarity
- Semantic parsing
- Semantic role labeling
- Sentiment analysis
- Shallow syntax

研究背景

Research background

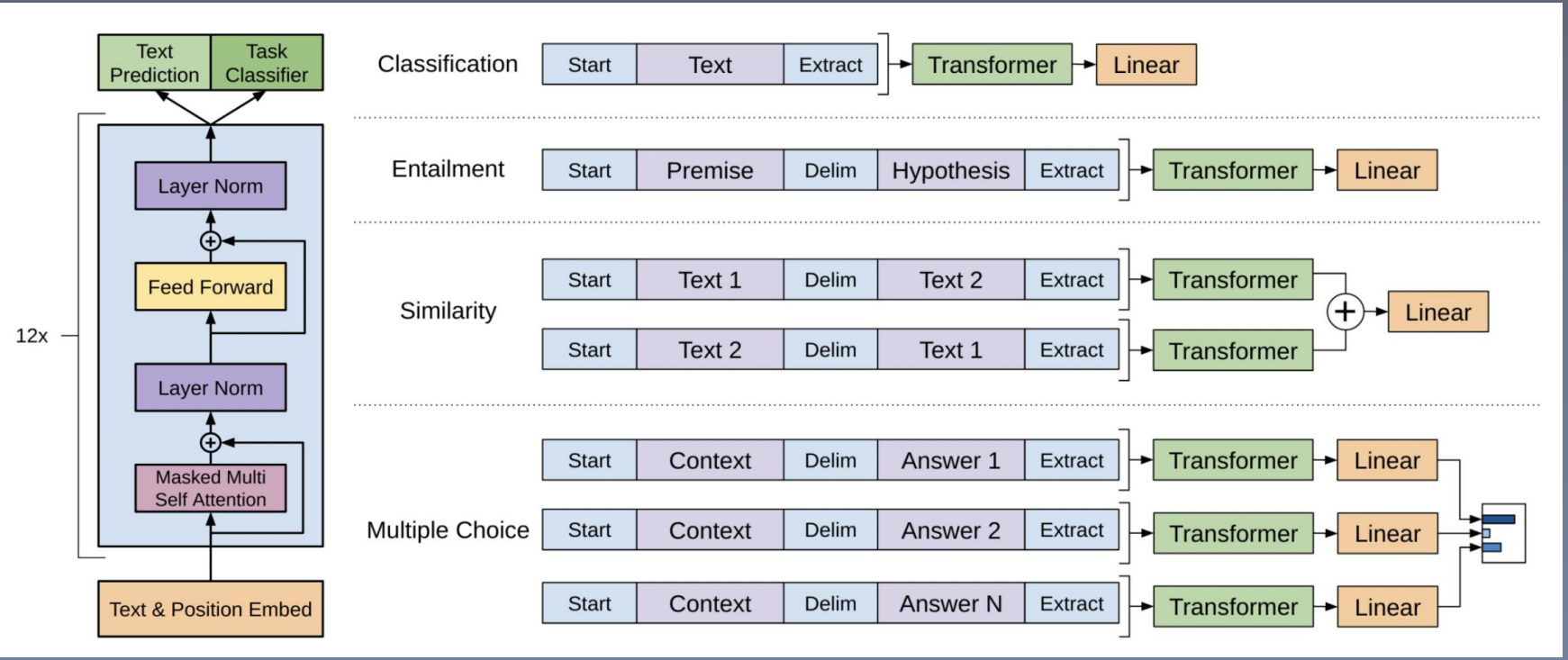
- 怎样使预训练语言模型使用不同下游任务？



研究背景

Research background

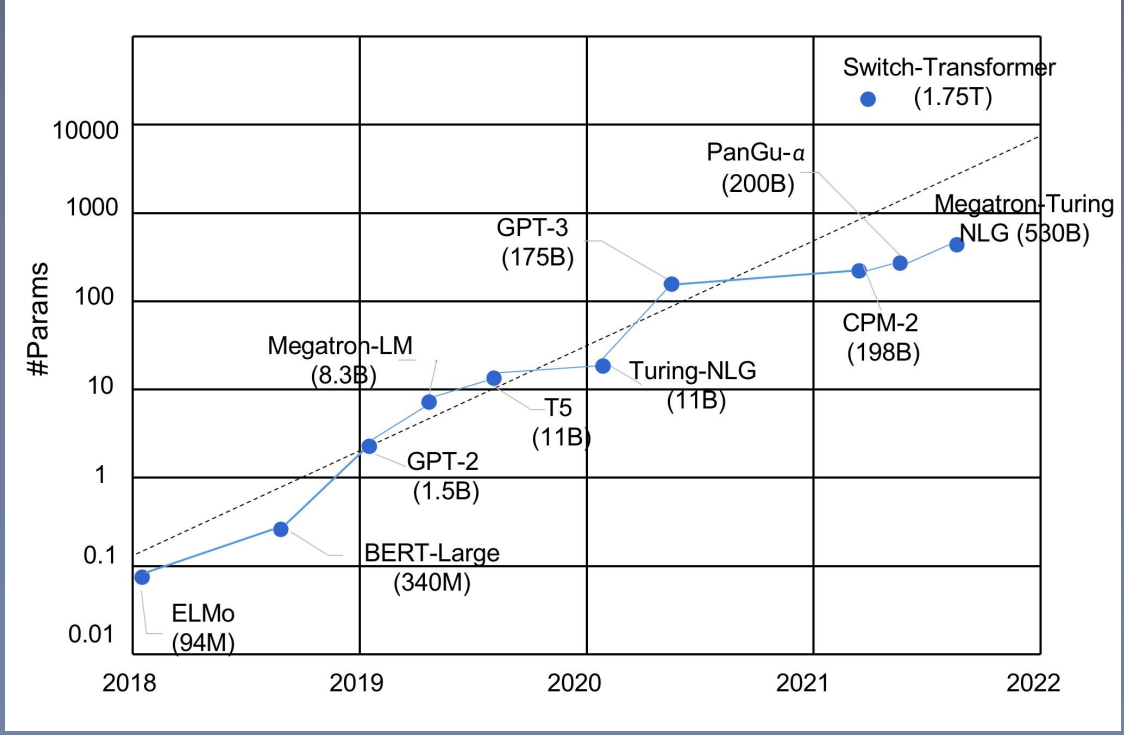
- 怎样使预训练语言模型使用不同下游任务？



研究背景

Research background

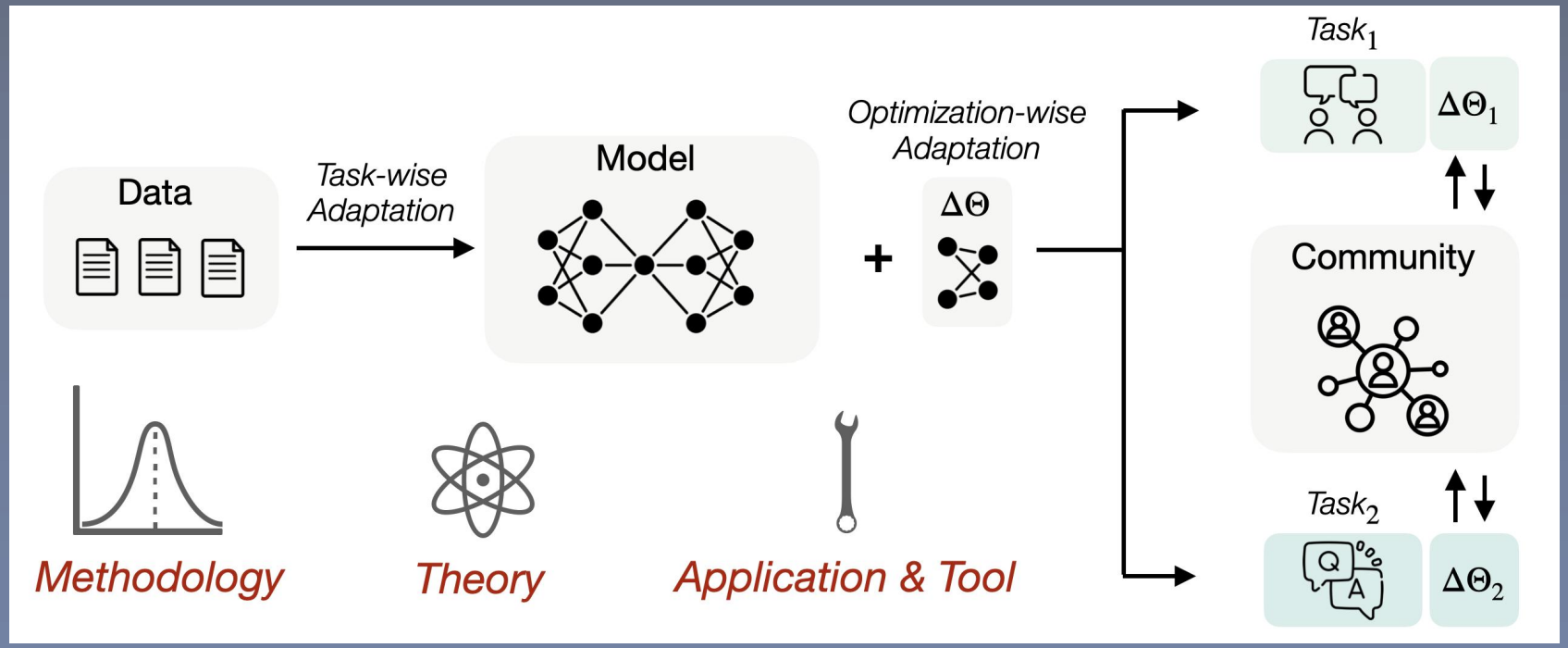
- 当模型不断增大，对于消费级硬件来说，对所有参数进行完全微调已经不再可行



研究背景

Research background

- 高效参数微调：仅微调一小部分或额外的模型参数，而将大部分预训练模型（LLM）参数保持固定，从而显著降低了计算和存储成本，并且能够实现与全量参数微调相当的性能水平。



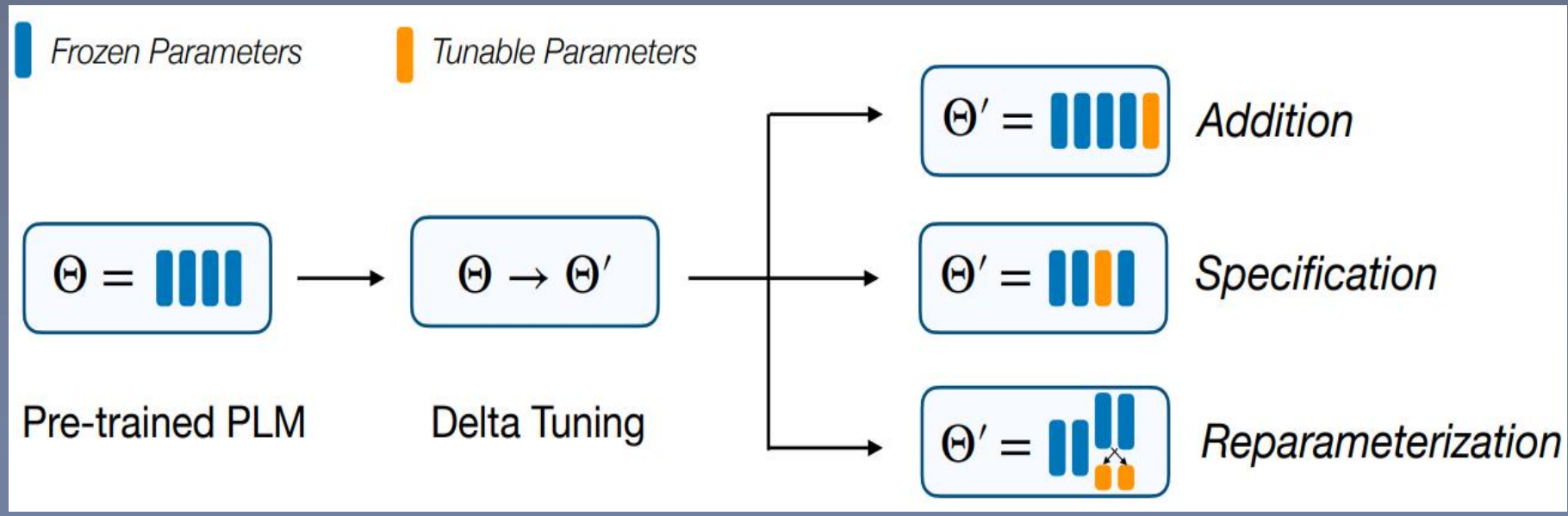
深度之眼
deepshare.net

目 录

算法分类

Basic Theory

- 高效参数微调算法分类
  - 增量参数微调（Addition-based）
  - 指定参数微调（Specification-based）
  - 重参数化（Reparameterization）

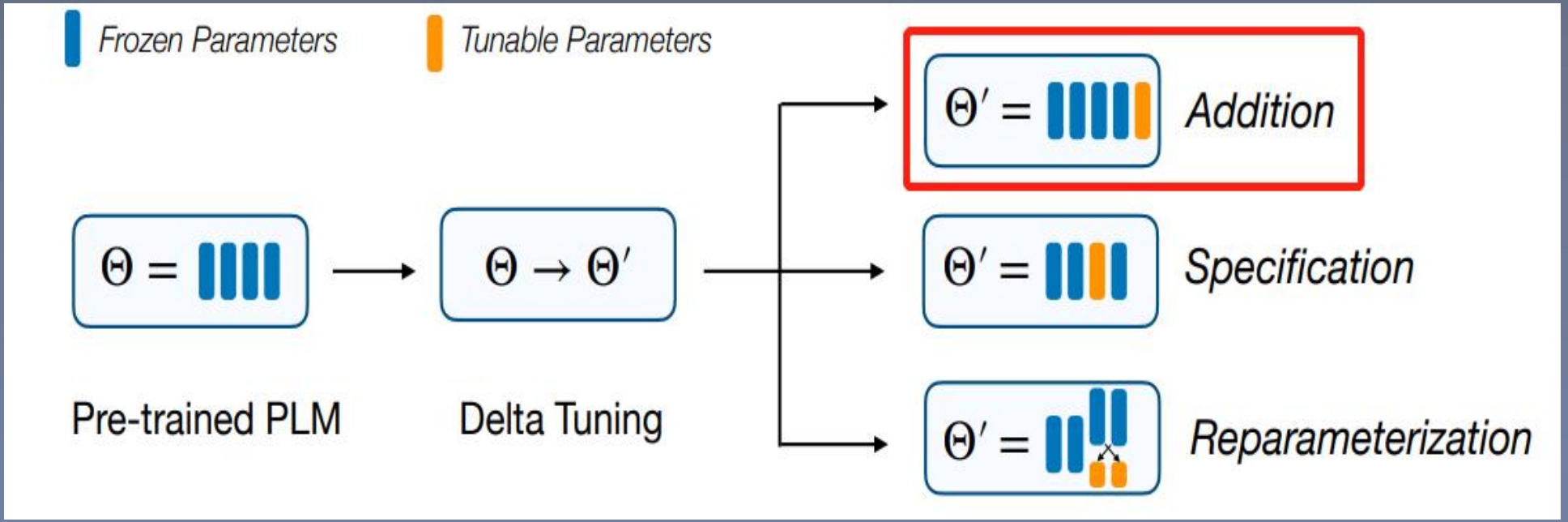


Ding et al.(2022) Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

基本理论

Basic Theory

- 增量参数微调
  - 增量参数微调（Addition-based）:引入额外的参数进行模型微调
  - Adapter-tuning, Prefix-tuning, Prompt-tuning

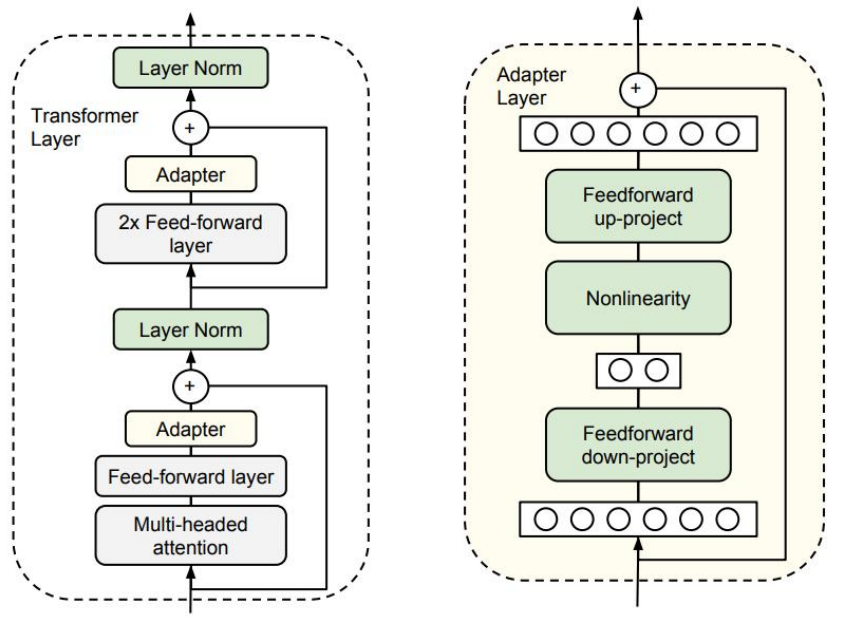


Ding et al.(2022) Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

增量参数微调

Basic Theory

- Adapter-tuning
  - 在Transformer子层间引入小规模的神经网络模块(Adapter)作为微调参数
  - 针对每一个Transformer层，增加了两个Adapter结构(分别是多头注意力的投影之后和第二个feed-forward层之后)



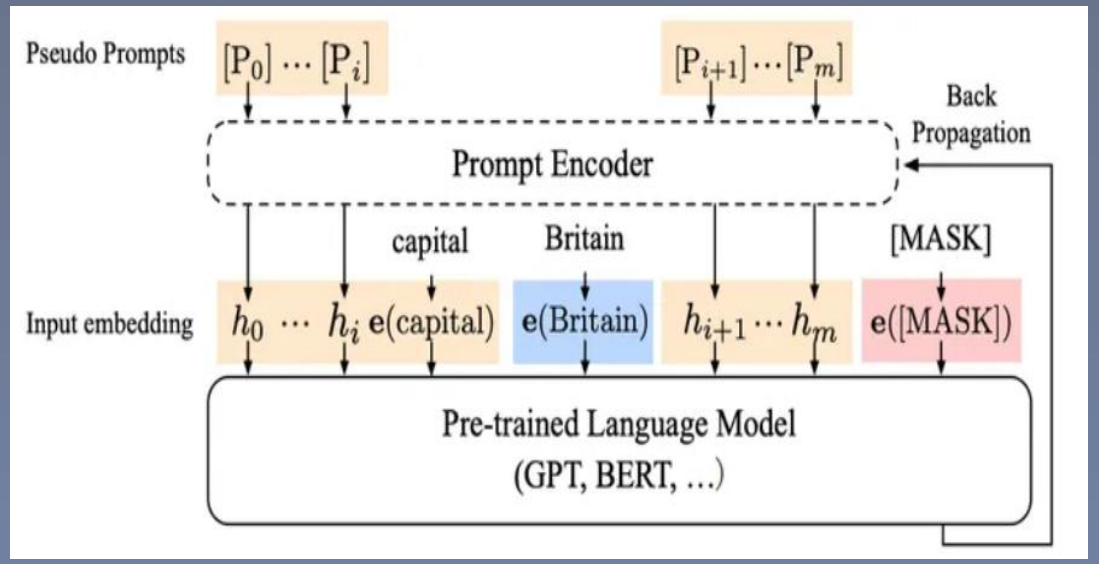

```
def transformer_block_with_adapter(x):
        residual = x
        x = SelfAttention(x)
        x = FFN(x) # adapter
        x = LN(x + residual)
        residual = x
        x = FFN(x) # transformer FFN
        x = FFN(x) # adapter
        x = LN(x + residual)
        return x
```

Houlsby et al. (2019), Parameter-Efficient Transfer Learning for NLP

增量参数微调

Basic Theory

- Prompt-tuning
  - 通过可训练参数化的提示语进行模型微调
  - The Power of Scale for Parameter-Efficient Prompt Tuning
  - GPT Understands, Too(P-tuning)

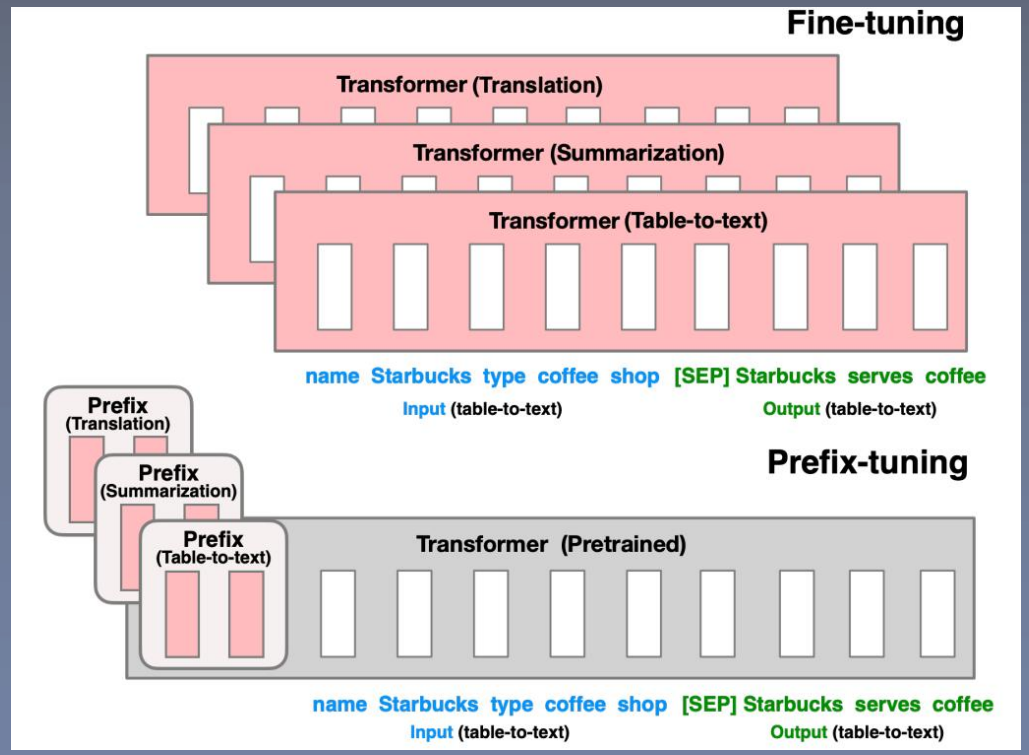



```
def soft_prompted_model(input_ids):
        x = Embed(input_ids)
        x = concat([soft_prompt, x], dim=1)
        return model(x)
```

增量参数微调

Basic Theory

- Prefix-tuning
  - 为每个Transformer层的输入和隐藏状态拼接由可训练参数构成的前缀



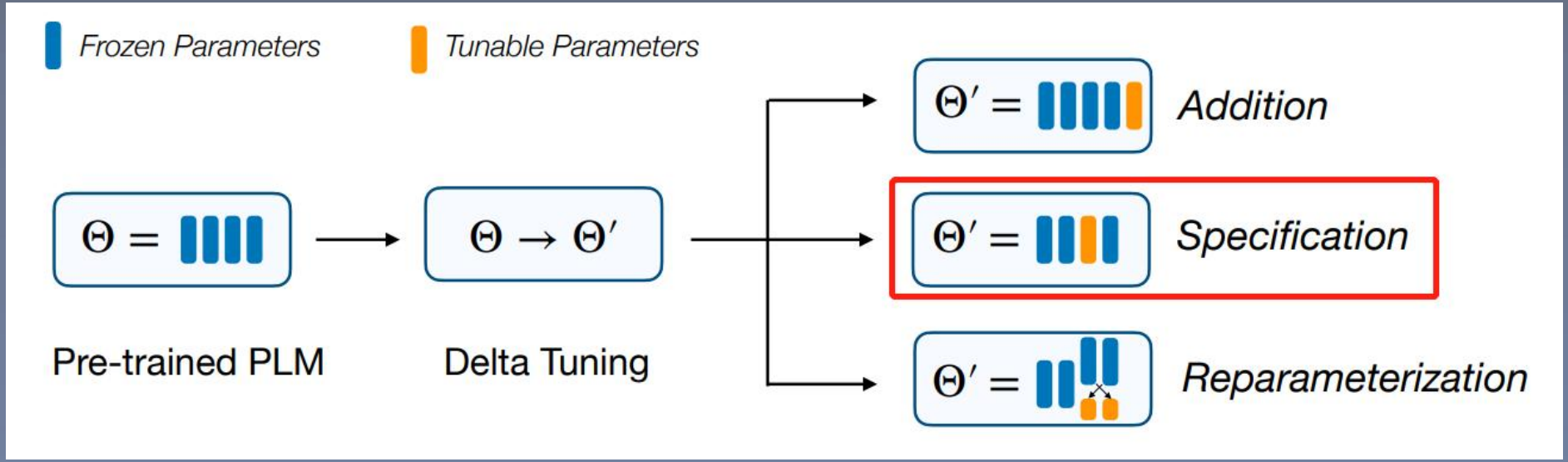

```
def transformer_block_for_prefix_tuning(x):
        soft_prompt = FFN(soft_prompt)
        x = concat([soft_prompt, x], dim=seq)
        return transformer_block(x)
```

Li et al. (2021), Prefix-Tuning: Optimizing Continuous Prompts for Generation

指定参数微调

Basic Theory

- 指定参数微调算法
  - 指定参数微调（Specification-based）：指定原始模型中的特定的某些参数变得可训练，其他参数则被冻结



.(2022) Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

指定参数微调

Basic Theory

- BitFit
  - 只更新bias-terms相关参数（train only the bias-terms and the task-specific classification layer）

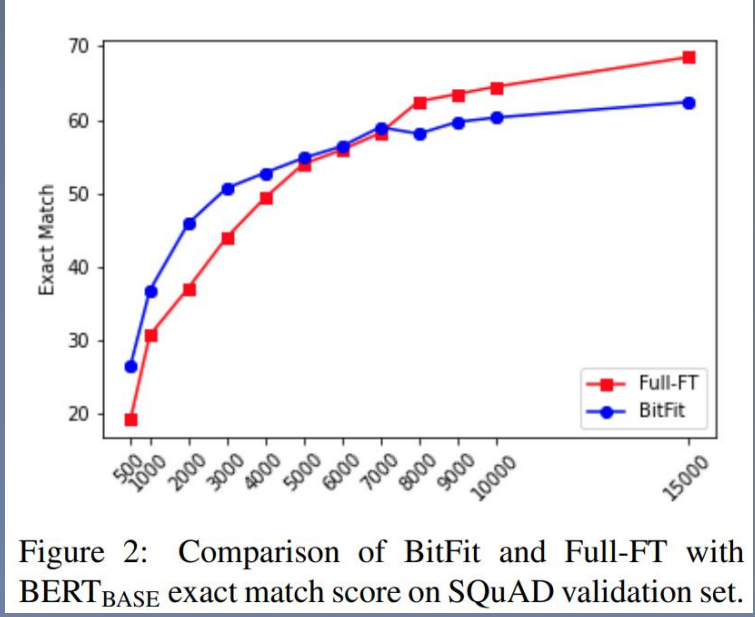


Figure 2: Comparison of BitFit and Full-FT with $\text{BERT}_{\text{BASE}}$ exact match score on SQuAD validation set.

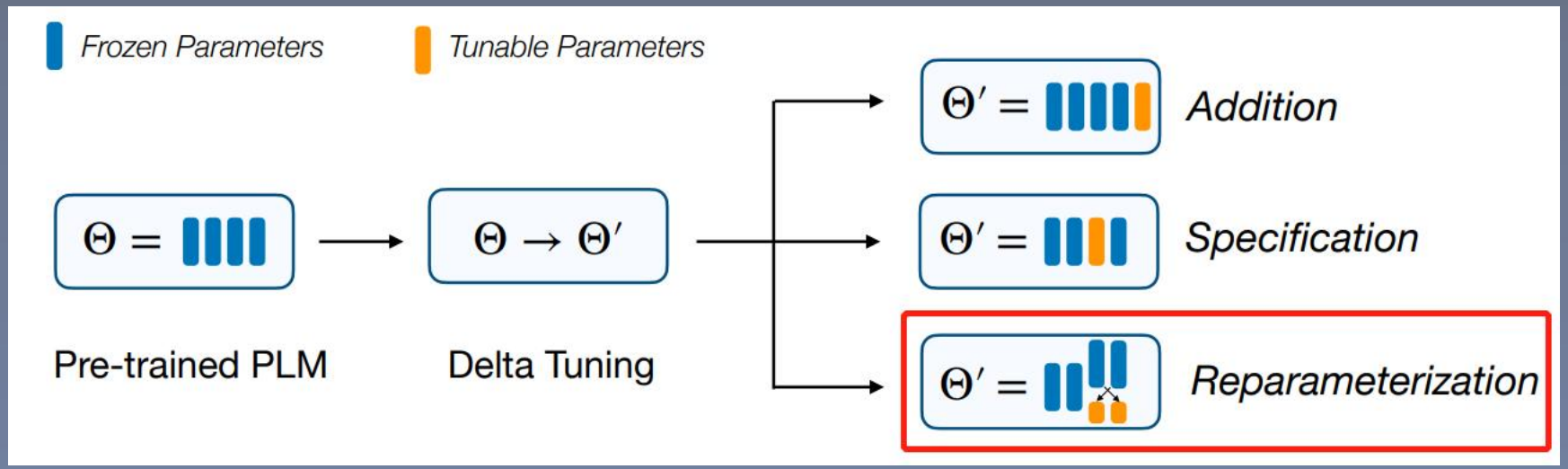
```
params = (p for n, p
          in model.named_parameters()
          if "bias" in n)
optimizer = Optimizer(params)
```

Elad et al.(2021) BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models

基于重参数的微调

Basic Theory

- 基于重参数的算法
  - 重参数化微调（Reparameterization-based）：将模型中参数转化为高效微调的形式



.(2022) Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

基于重参数的微调

Basic Theory

- LoRA(Low-Rank Adaption)
  - 秩矩阵表示来减少可训练参数数量，从而以小参数量实现大模型的高效微调
  - 在涉及到矩阵相乘的模块，引入两个矩阵A,B相乘，第一个矩阵A负责降维，第二个矩阵B负责升维，中间层维度为r，从而来模拟所谓的本征秩
  - r << d 是low-rank的体现，极大地减少待训练的参数量

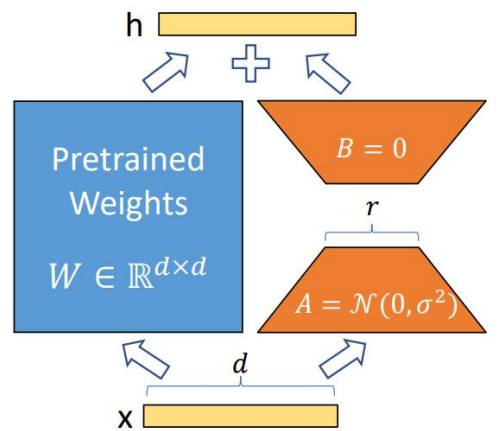


Figure 1: Our reparametrization. We only train A and B.

转化前：d x d

转化后：d x r + r x d （r << d）

Edward et al.(2021) LoRA: Low-Rank Adaptation of Large Language Models

基于重参数的微调

Basic Theory

- LoRA算法改进
  - AdaLoRA：将关键的增量矩阵分配高秩以捕捉更精细和任务特定的信息，而将较不重要的矩阵的秩降低，以防止过拟合并节省计算预算
  - QLoRA：通过冻结的、4Bit量化(Quantization)的预训练语言模型来做 LoRA
    - 一种新的数据类型4位NormalFloat（NF4）；
    - 双重量化以减少平均内存占用；
    - 分页优化器来管理内存峰值。

Zhang et al. (2023) Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning
Tim et al. (2023) QLoRA: Efficient Finetuning of Quantized LLMs

深度之眼
deepshare.net

目 录

实践步骤

Practical Steps

- 专用模型微调的基本步骤
  - Step1：构建指令数据
  - Step2：选择基座模型
  - Step3：模型高效微调

实践步骤

Practical Steps

- 指令数据构建
  - 数据收集：精标任务数据、弱监督数据、公开通用数据
  - 数据去噪：清洗任务数据集，对齐输入与输出
  - 指令构建：目的清晰，内容流畅

instruction="已知候选的关系列表：['父母','配偶']，请你根据关系列表，从以下输入中抽取出可能存在的头实体(Subject)与尾实体(Object)，并给出对应的关系三元组。请按照(Subject,Relation,Object)的格式回答。"
input="2008年6月23日，刘德华与朱丽倩在美国拉斯维加斯注册结婚。2012年5月9日，朱丽倩在香港养和医院产下女儿刘向蕙。刘向蕙出生后，少数看过刘向蕙面貌的圈内人士称，女儿的耳朵像刘德华，长得眉清目秀。"
output="(刘德华,配偶,朱丽倩),(刘向蕙,父母,刘德华),(刘向蕙,父母,朱丽倩)"

实践步骤

Practical Steps

深度之眼
deepshare.net

- 基座模型：经海量数据预训练（Pre-train）后具备一定通用能力的语言模型

	Position embedding	Self-attention	Activation	Norm Type	Num Layers	Num Heads	Vocab Size	Public Time
GPT3-175B	Absolute	Standard	ReLU	-	96	96	50,257	2020.05.28
CodeGen-16B	RoPE	Standard	GeLU	ParallelLayer	34	24	51,200	2022.03.25
PaLM-540B	RoPE	Multi-Query	SwiGLU	ParallelLayer	118	48	256,000	2022.04.05
OPT-175B	Absolute	Standard	ReLU	PreNorm	96	96	50,272	2022.05.02
GLM-130B	RoPE	Standard	GeGLU	PostNorm	70	96	150,528	2022.10.05
BLOOM-176B	ALiBi	Standard	GeLU	PreNorm	70	112	250,680	2022.11.09
LLaMA-65B	RoPE	Standard	SwiGLU	PreNorm	80	64	32,000	2023.02.24
CodeGen2-16B	RoPE	Standard	GeLU	ParallelLayer	34	24	50,400	2023.05.03
MPT-7B	ALiBi	Standard	ReLU	PreNorm	32	32	50,432	2023.05.05
StarCoder	Absolute	Multi-Query	GeLU	PreNorm	40	48	49,152	2023.05.09

Ding et al.(2022) Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

实践步骤

Practical Steps

- **常见中文基座模型**

1. GLM： https://github.com/THUDM/GLM-130B

2. MOSS： https://github.com/OpenLMLab/MOSS

3. OpenChineseLLaMA： https://github.com/OpenLMLab/OpenChineseLLaMA

4. Chinese-LLaMA-Alpaca： https://github.com/ymcui/Chinese-LLaMA-Alpaca

6. BiLLa： https://github.com/Neutralzz/BiLLa

7 CPM-Bee： https://github.com/OpenBMB/CPM-

实践步骤

Practical Steps

- 基于PEFT框架的高效微调实现
  - https://github.com/huggingface/peft

今天，我们很高兴地介绍 🤗 PEFT 库。它提供了最新的参数高效微调技术，与 🤗 Transformers 和 🤗 Accelerate 无缝集成。这使得能够使用来自 Transformers 的最流行和高性能的模型，以及 Accelerate 的简单性和可扩展性。以下是目前支持的 PEFT 方法，即将推出更多：

1. LoRA: LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS
2. Prefix Tuning: P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks
3. Prompt Tuning: The Power of Scale for Parameter-Efficient Prompt Tuning
4. P-Tuning: GPT Understands, Too

实践步骤

Practical Steps

- 基于PEFT框架的高效微调实现（针对核心代码进行介绍）
  - Step 1：参数设置

```
from transformers import AutoModelForSeq2SeqLM
+ from peft import get_peft_model, LoraConfig,
TaskType

model_name_or_path = "bigscience/mt0-large"
 tokenizer_name_or_path = "bigscience/mt0-large"
```

```
peft_config = LoraConfig(
    task_type=TaskType.SEQ_2_SEQ_LM,
inference_mode=False, r=8, lora_alpha=32,
lora_dropout=0.1
)
```

```
class PeftType(str, enum.Enum):
    PROMPT_TUNING = "PROMPT_TUNING"
    MULTITASK_PROMPT_TUNING = "MULTITASK_PROMPT_TUNING"
    P_TUNING = "P_TUNING"
    PREFIX_TUNING = "PREFIX_TUNING"
    LORA = "LORA"
    ADALORA = "ADALORA"
    ADAPTION_PROMPT = "ADAPTION_PROMPT"
    IA3 = "IA3"


class TaskType(str, enum.Enum):
    SEQ_CLS = "SEQ_CLS"
    SEQ_2_SEQ_LM = "SEQ_2_SEQ_LM"
    CAUSAL_LM = "CAUSAL_LM"
    TOKEN_CLS = "TOKEN_CLS"
    QUESTION_ANS = "QUESTION_ANS"
    FEATURE_EXTRACTION = "FEATURE_EXTRACTION"
```

实践步骤

Practical Steps

- 基于PEFT框架的高效微调实现
  - Step 2：加载模型

```
model = AutoModelForSeq2SeqLM.from_pretrained(model_name_or_path)
+ model = get_peft_model(model, peft_config)
+ model.print_trainable_parameters()
# output: trainable params: 2359296 || all params: 1231940608 || trainable%:
0.191510531001182
```

  - Step3：加载数据

```
dataset = load_dataset("financial_phrasebank", "sentences_allagree")
dataset = dataset["train"].train_test_split(test_size=0.1)

dataset = dataset.map(     lambda x: {"text_label": [classes[label] for label in x["label"]]},     batched=True,
    num_proc=1)
```

实践步骤

Practical Steps

- 基于PEFT框架的高效微调实现
  - Step 4: 模型训练

```
optimizer = torch.optim.AdamW(model.parameters(), lr=lr)
lr_scheduler = get_linear_schedule_with_warmup(
    optimizer=optimizer,
    num_warmup_steps=0,
    num_training_steps=(len(train_dataloader) * num_epochs),
```

```
for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    for step, batch in enumerate(tqdm(train_dataloader)):
        batch = {k: v.to(device) for k, v in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        total_loss += loss.detach().float()
        loss.backward()
        optimizer.step()
        lr_scheduler.step()
        optimizer.zero_grad()
```

深度之眼
deepshare.net

本课总结

讨论论文中存在的问题，总结本阶段所学内容

总结

Summary

关键点

- 为什么要进行高效微调
- 常见的高效微调算法有哪些
- 特定任务专用模型微调的基本步骤

下节课程课前准备

Preview of next lesson

- 了解工具学习相关知识

我 说

在其他任何地方的投资，都没有投资自己回报率高，你是赚不到超出你认知以外的钱的

愿你成为一名终身学习者。

如果有一天学累了，坚持不下去了，告诉自己，如果我现在觉得学习痛苦，是因为我还不知道无知的代价更大

深度之眼
deepshare.net

联系我们：

电话：18001992849

邮箱：service@deepshare.net

Q Q：2677693114

公众号

客服微信