大模型（LLMs）参数高效微调(PEFT) 面

来自： AiGC面试宝典

宁静致远

2023年09月18日 20:55

1. 微调方法是啥？如何微调？

fine-tune，也叫全参微调，bert微调模型一直用的这种方法，全部参数权重参与更新以适配领域数据，效果好。

prompt-tune, 包括p-tuning、lora、prompt-tuning、adaLoRA等delta tuning方法，部分模型参数参与微调，训练快，显存占用少，效果可能跟FT（fine-tune）比会稍有效果损失，但一般效果能打平。

链家在BELLE的技术报告《A Comparative Study between Full-Parameter and LoRA-based Fine-Tuning on Chinese Instruction Data for Instruction Following Large Language Model》中实验显示：FT效果稍好于LoRA。

Table 4: Main results. In this table, LLaMA-13B + LoRA(2M) represents a model trained on 2M instruction data using LLaMA-13B as base model and LoRA training method, and LLaMA-7B + FT(2M) represents a model trained using full-parameters fine-tuning. LLaMA-7B + FT(2M) + LoRA(math_0.25M) represents a model trained on 0.25M mathematical instruction data using LLaMA-7B + FT(2M) as the base model and LoRA training method, and LLaMA-7B + FT(2M) + FT(math_0.25M) represents a model trained using incremental full-parameters fine-tuning. About the training time, all these experiments were conducted on 8 NVIDIA A100-40GB GPUs.

Model	Average Score	Additional Param.	Training Time (Hour/epoch)
LLaMA-13B + LoRA(2M)	0.648	28M	10
LLaMA-7B + LoRA(4M)	0.624	17.9M	14
LLaMA-7B + LoRA(2M)	0.609	17.9M	7
LLaMA-7B + LoRA(0.6M)	0.589	17.9M	5
LLaMA-7B + FT(2M)	0.710	-	31
LLaMA-7B + FT(0.6M)	0.686	-	17
LLaMA-7B + FT(2M) + LoRA(math_0.25M)	0.729	17.9M	12
LLaMA-7B + FT(2M) + FT(math_0.25M)	0.738	-	4

peft的论文《ADAPTIVE BUDGET ALLOCATION FOR PARAMETER- EFFICIENT FINE-TUNING》显示的结果：AdaLoRA效果稍好于FT。

Table 1: Results with DeBERTaV3-base on GLUE development set. The best results on each dataset are shown in **bold**. We report the average correlation for STS-B. *Full FT*, *HAdapter* and *PAdapter* represent full fine-tuning, Houlsby adapter, and Pfeiffer adapter respectively. We report mean of 5 runs using different random seeds.

Method	# Params	MNLI m/mm	SST-2 Acc	CoLA Mcc	QQP Acc/F1	QNLI Acc	RTE Acc	MRPC Acc	STS-B Corr	All Ave.
Full FT	184M	89.90/90.12	95.63	69.19	**92.40/89.80**	94.03	83.75	89.46	91.60	88.09
BitFit	0.1M	89.37/89.91	94.84	66.96	88.41/84.95	92.24	78.70	87.75	91.35	86.02
HAdapter	1.22M	90.13/90.17	95.53	68.64	91.91/89.27	94.11	84.48	89.95	91.48	88.12
PAdapter	1.18M	90.33/90.39	95.61	68.77	92.04/89.40	94.29	85.20	89.46	91.54	88.24
LoRA$_{r=8}$	1.33M	90.65/90.69	94.95	69.82	91.99/89.38	93.87	85.20	89.95	91.60	88.34
AdaLoRA	1.27M	**90.76/90.79**	**96.10**	**71.45**	**92.23/89.74**	**94.55**	**88.09**	**90.69**	**91.84**	**89.31**
HAdapter	0.61M	90.12/90.23	95.30	67.87	91.65/88.95	93.76	85.56	89.22	91.30	87.93
PAdapter	0.60M	90.15/90.28	95.53	69.48	91.62/88.86	93.98	84.12	89.22	91.52	88.04
HAdapter	0.31M	90.10/90.02	95.41	67.65	91.54/88.81	93.52	83.39	89.25	91.31	87.60
PAdapter	0.30M	89.89/90.06	94.72	69.06	91.40/88.62	93.87	84.48	89.71	91.38	87.90
LoRA$_{r=2}$	0.33M	90.30/90.38	94.95	68.71	91.61/88.91	94.03	85.56	89.71	**91.68**	88.15
AdaLoRA	0.32M	**90.66/90.70**	**95.80**	**70.04**	**91.78/89.16**	**94.49**	**87.36**	90.44	91.63	**88.86**

2. 为什么需要 PEFT?

在面对特定的下游任务时，如果进行Full FineTuning（即对预训练模型中的所有参数都进行微调），太过低效；而如果采用固定预训练模型的某些层，只微调接近下游任务的那几层参数，又难以达到较好的效果。

3. 介绍一下 PEFT?

PEFT技术旨在**通过最小化微调参数的数量和计算复杂度，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。**这样一来，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。

4. PEFT 有什么优点?

PEFT技术可以在提高模型效果的同时，大大缩短模型训练时间和计算成本，让更多人能够参与到深度学习研究中来。除此之外，FEFT可以缓解全量微调带来灾难性遗忘的问题。

5. 微调方法批处理大小模式GPU显存速度?

微调方法批处理大小模式GPU显存速度

```
LoRA (r=8) 16 FP16 28GB 8ex/s
LoRA (r=8) 8 FP16 24GB 8ex/s
LoRA (r=8) 4 FP16 20GB 8ex/s
LoRA (r=8) 4 INT8 10GB 8ex/s
LoRA (r=8) 4 INT4 8GB 8ex/s
P-Tuning (p=16) 4 FP16 20GB 8ex/s
P-Tuning (p=16) 4 INT8 16GB 8ex/s
P-Tuning (p=16) 4 INT4 12GB 8ex/s
Freeze (l=3) 4 FP16 24GB 8ex/s
Freeze (l=3) 4 INT8 12GB 8ex/s
```

6. Peft 和 全量微调区别?

所谓的 fune-tine 只能改变风格, 不能改变知识, 是因为我们的 fine-tune, 像是 LoRA 本来就是低秩的, 没办法对模型产生决定性的改变. 要是全量微调, 还是可以改变知识的.

7. 多种不同的高效微调方法对比

像P-Tuning v2、LoRA等都是综合评估很不错的高效微调技术。如果显存资源有限可以考虑QLoRA；如果只是解决一些简单任务场景，可以考虑P-Tuning、Prompt Tuning也行。

下表从参数高效方法类型、是否存储高效和内存高效、以及在减少反向传播成本和推理开销的计算高效五个维度比较了参数高效微调方法。

Method	Type	Storage	Memory	Backprop	Inference overhead
Adapters (Houlsby et al., 2019)	A	yes	yes	no	Extra FFN
AdaMix (Wang et al., 2022)	A	yes	yes	no	Extra FFN
SparseAdapter (He et al., 2022b)	AS	yes	yes	no	Extra FFN
Cross-Attn tuning (Gheini et al., 2021)	S	yes	yes	no	No overhead
BitFit (Ben-Zaken et al., 2021)	S	yes	yes	no	No overhead
DiffPruning (Guo et al., 2020)	S	yes	no	no	No overhead
Fish-Mask (Sung et al., 2021)	S	yes	maybe[5]	no	No overhead
LT-SFT (Ansell et al., 2022)	S	yes	maybe[5]	no	No overhead
Prompt Tuning (Lester et al., 2021)	A	yes	yes	no	Extra input
Prefix-Tuning (Li and Liang, 2021)	A	yes	yes	no	Extra input
Spot (Vu et al., 2021)	A	yes	yes	no	Extra input
IPT (Qin et al., 2021)	A	yes	yes	no	Extra FFN and input
MAM Adapter (He et al., 2022a)	A	yes	yes	no	Extra FFN and input
Parallel Adapter (He et al., 2022a)	A	yes	yes	no	Extra FFN
Intrinsinc SAID (Aghajanyan et al., 2020)	R	no	no	no	No overhead
LoRa (Hu et al., 2021)	R	yes	yes	no	No overhead
UniPELT (Mao et al., 2021)	AR	yes	yes	no	Extra FFN and input
Compacter (Karimi Mahabadi et al., 2021)	AR	yes	yes	no	Extra FFN
PHM Adapter (Karimi Mahabadi et al., 2021)	AR	yes	yes	no	Extra FFN
KronA (Edalati et al., 2022)	R	yes	yes	no	No overhead
KronA$^{B}_{res}$ (Edalati et al., 2022)	AR	yes	yes	no	Extra linear layer
$(IA)^3$ (Liu et al., 2022)	A	yes	yes	no	Extra gating
Attention Fusion (Cao et al., 2022)	A	yes	yes	yes	Extra decoder
LeTS (Fu et al., 2021)	A	yes	yes	yes	Extra FFN
Ladder Side-Tuning (Sung et al., 2022)	A	yes	yes	yes	Extra decoder
FAR (Vucetic et al., 2022)	S	yes	maybe[6]	no	No overhead
S4-model (Chen et al., 2023)	ARS	yes	yes	no	Extra FFN and input

Comparing PEFT methods across storage efficiency, memory efficiency, and computational efficiency in terms of reducing backpropagation costs and having inference overhead. Method types: A – additive, S – selective, R – reparametrization-based.

下表展示了各种参数高效方法的参与训练的参数量、最终模型与原始模型的改变参数（delta值）以及论文中参与评估的模型的范围（<1B、<20B、>20B）。

Method	% Trainable parameters	% Changed parameters	Evaluated on <1B	<20B	>20B
Adapters (Houlsby et al., 2019)	0.1 - 6	0.1 - 6	yes	yes	yes
AdaMix (Wang et al., 2022)	0.1 - 0.2	0.1 - 0.2	yes	no	no
SparseAdapter (He et al., 2022b)	2.0	2.0	yes	no	no
BitFit (Ben-Zaken et al., 2021)	0.05 - 0.1	0.05 - 0.1	yes	yes	yes
DiffPruning (Guo et al., 2020)	200	**0.5**	yes	no	no
Fish-Mask (Sung et al., 2021)	0.01 - 0.5	0.01 - 0.5	yes	yes	no
Prompt Tuning (Lester et al., 2021)	0.1	0.1	yes	yes	yes
Prefix-Tuning (Li and Liang, 2021)	0.1 - 4.0	0.1 - 4.0	yes	yes	yes
IPT (Qin et al., 2021)	56.0	56.0	yes	no	no
MAM Adapter (He et al., 2022a)	0.5	0.5	yes	no	no
Parallel Adapter (He et al., 2022a)	0.5	0.5	yes	no	no
Intrinsinc SAID (Aghajanyan et al., 2020)	0.001 - 0.1	**∼0.1 or 100**	yes	yes	no
LoRa (Hu et al., 2021)	0.01 - 0.5	**∼0.5 or ∼30**	yes	yes	yes
UniPELT (Mao et al., 2021)	1.0	1.0	yes	no	no
Compacter (Karimi Mahabadi et al., 2021)	0.05-0.07	**∼0.07 or ∼0.1**	yes	yes	no
PHM Adapter (Karimi Mahabadi et al., 2021)	0.2	**∼0.2 or ∼1.0**	yes	no	no
KronA (Edalati et al., 2022)	0.07	**∼0.07 or ∼30.0**	yes	no	no
KronA$^{B}_{res}$ (Edalati et al., 2022)	0.07	**∼0.07 or ∼1.0**	yes	no	no
$(IA)^3$ (Liu et al., 2022)	0.02	0.02	no	yes	no
Ladder Side-Tuning(Sung et al., 2022)	7.5	7.5	yes	yes	no
FAR (Vucetic et al., 2022)	6.6-26.4	6.6-26.4	yes	no	no
S4-model (Chen et al., 2023)	0.5	**more than 0.5**	yes	yes	no

What model sizes PEFTmethods have been evaluated on and their typical amount of trainable parameters used in the papers. By trainable parameter count we specifically mean the number parameters that are updated by a gradient optimization algorithm, not the delta between the original and final model which we denote "changed parameters". For reprametrization-based method we report parameters before and after reparametrization. Estimating updated parameter count for S4 is complicated as the model uses different methods at different layers. We report the range at which the methods were evaluated in the published literature.

从表中可以看到，Prompt Tuning、Prefix Tuning、LoRA等少部分微调技术针对不同参数规模的模型进行过评估，同时，这几种方式也是目前应用比较多的高效微调方法。

8. 当前高效微调技术存在的一些问题

当前的高效微调技术很难在类似方法之间进行直接比较并评估它们的真实性能，主要的原因如下所示：

- **参数计算口径不一致**：参数计算可以分为三类：可训练参数的数量、微调模型与原始模型相比改变的参数的数量、微调模型和原始模型之间差异的等级。例如，DiffPruning更新0.5%的参数，但是实际参与训练的参数量是200%。这为比较带来了困难。尽管可训练的参数量是最可靠的存储高效指标，但是也不完美。 Ladder-side Tuning使用一个单独的小网络，参数量高于LoRA或BitFit，但是因为反向传播不经过主网络，其消耗的内存反而更小。
- **缺乏模型大小的考虑**：已有工作表明，大模型在微调中需要更新的参数量更小（无论是以百分比相对而论还是以绝对数量而论），因此（基）模型大小在比较不同PEFT方法时也要考虑到。
- **缺乏测量基准和评价标准**：不同方法所使用的的模型/数据集组合都不一样，评价指标也不一样，难以得到有意义的结论。
- **代码实现可读性差**：很多开源代码都是简单拷贝Transformer代码库，然后进行小修小补。这些拷贝也不使用git fork，难以找出改了哪里。即便是能找到，可复用性也比较差（通常指定某个Transformer版本，没有说明如何脱离已有代码库复用这些方法）。

9. 高效微调技术最佳实践

针对以上存在的问题，研究高效微调技术时，建议按照最佳实践进行实施：

- 明确指出参数数量类型。
- 使用不同大小的模型进行评估。
- 和类似方法进行比较。
- 标准化PEFT测量基准。
- 重视代码清晰度，以最小化进行实现。

10. PEFT 存在问题?

相比全参数微调，大部分的高效微调技术目前存在的两个问题：

1. 推理速度会变慢；
2. 模型精度会变差；

11. 能不能总结一下各种参数高效微调方法?

本文针对之前介绍的几种参数高效微调方法进行了简单的概述，主要有如下几类：

- 增加额外参数，如：Prefix Tuning、Prompt Tuning、Adapter Tuning及其变体。
- 选取一部分参数更新，如：BitFit。
- 引入重参数化，如：LoRA、AdaLoRA、QLoRA。
- 混合高效微调，如：MAM Adapter、UniPELT。

并比较了不同的高效微调方法之间的差异；同时，还指出当前大多数高效微调方法存在的一些问题并给出了最佳实践。