AiDD AI+研发数字峰会
AI+ Development Digital summit
第5届 5

模型轻量化技术与端侧落地

宋晓辉 | OPPO

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 · 敦煌站
K+ 思考周®研习社
时间：2025.08.29-30

K+峰会 · 上海站
K+ 金融专场
时间：2025.10.17-18

K+峰会 · 香港站
K+ 思考周®研习社
时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 · 上海站
AI+研发数字峰会
时间：2025.05.17-18

AiDD峰会 · 北京站
AI+研发数字峰会
时间：2025.08.08-09

AiDD峰会 · 深圳站
AI+研发数字峰会
时间：2025.11.28-29

AiDD峰会详情

演讲嘉宾

宋晓辉

OPPO AI中心-高级NLP算法工程师

中国科学院信息工程研究所工学博士，现就职于OPPO AI中心，负责大模型轻量化技术体系建设和端侧文本算法业务。

目录
CONTENTS

PART 01

大模型端侧化背景

模型端侧化的时代基础

端侧算力的高速发展

苹果**A Bionic**系列芯片
谷歌Edge TPU
高通/MTK 带NPU的SoC芯片

开发工具与推理技术

TensorFlow Lite
PyTorch Mobile
ONNX
量化推理
模型压缩

大模型的优秀效果

OpenAI GPT
Claude
Qwen
Deepseek
...

模型端/云部署的优势和劣势

端侧部署面临的主要挑战

针对硬件环境优化，实现高效、稳定、节省资源的端侧模型

01

快：构建高效的端侧模型

如何针对硬件环境，构建小而精的端侧模型，保证编解码速度快，同时体验不输于云侧模型。

02

稳：鲁棒的模型效果和推理环境

如何优化端侧模型的存储空间占用，实现低成本、高效率的部署和更新，提高算法功能的鲁棒性。

03

省：优化推理方案

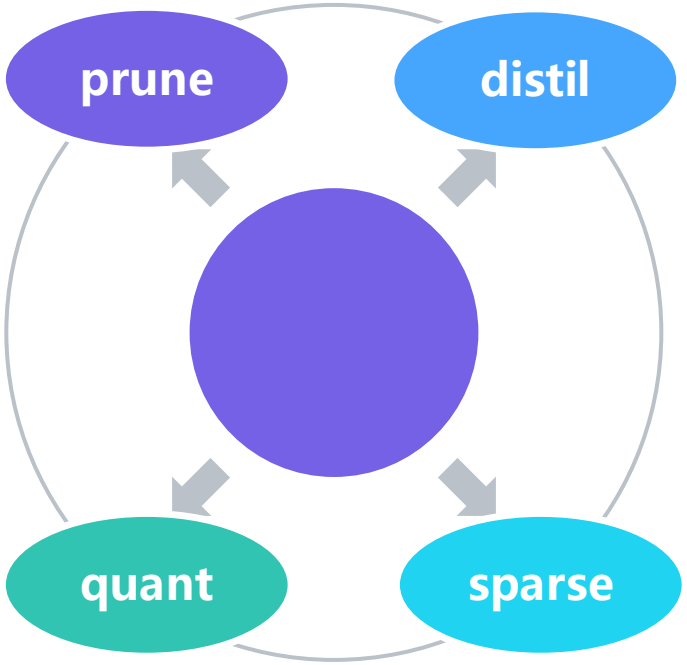
如何在保证延迟低的基础上，考虑内存和功耗的限制，实现推理方案的优化，提升端侧模型的资源使用效率。

PART 02
模型轻量化技术

模型轻量化技术

模型剪枝技术

- 模型剪枝通过移除网络中不重要的权重或神经元，来减少模型的计算量和存储要求。



知识蒸馏技术

- 知识蒸馏通过将大型（教师模型）模型的知识传递给较小的（学生模型），使较小模型在较少参数下也能表现良好。

量化压缩技术

- 量化通过降低模型参数和计算的精度（如将32位浮点数转换为8位整数），减少存储和计算资源。

权重稀疏化技术

- 通过将部分权重置零来减少模型中活跃连接的数量，来降低计算复杂度和存储需求。

2.1知识蒸馏

知识蒸馏

离线蒸馏

教师模型往往和学术模型体量差异巨大，存在词表、架构上的差异，不能实时参与到训练过程中，通常用于以下场景：

1. **数据构造**
   - 自动标注，常见的依赖无标签数据加提示词工程，通过超大规模模型构建场景训练数据
   - 数据合成，在指令微调阶段通过超大规模模型构建大规模通用指令集

2. **数据增强**
   - 在现有数据上通过在超大规模模型上进行推理和调优，进行数据清洗或者增强工作

在线蒸馏

教师和学生模型使用相同架构或者在关键组件上存在相似性，例如attention，词表等，可以直接参与训练。通常用于提升参数效率的场景。技术较为丰富和多样：

1. **基于中间特征/注意力的蒸馏**
2. **基于采样策略的序列蒸馏**

⋮

3. **基于回复/logits的蒸馏**

2.1知识蒸馏-LLM的logits知识蒸馏

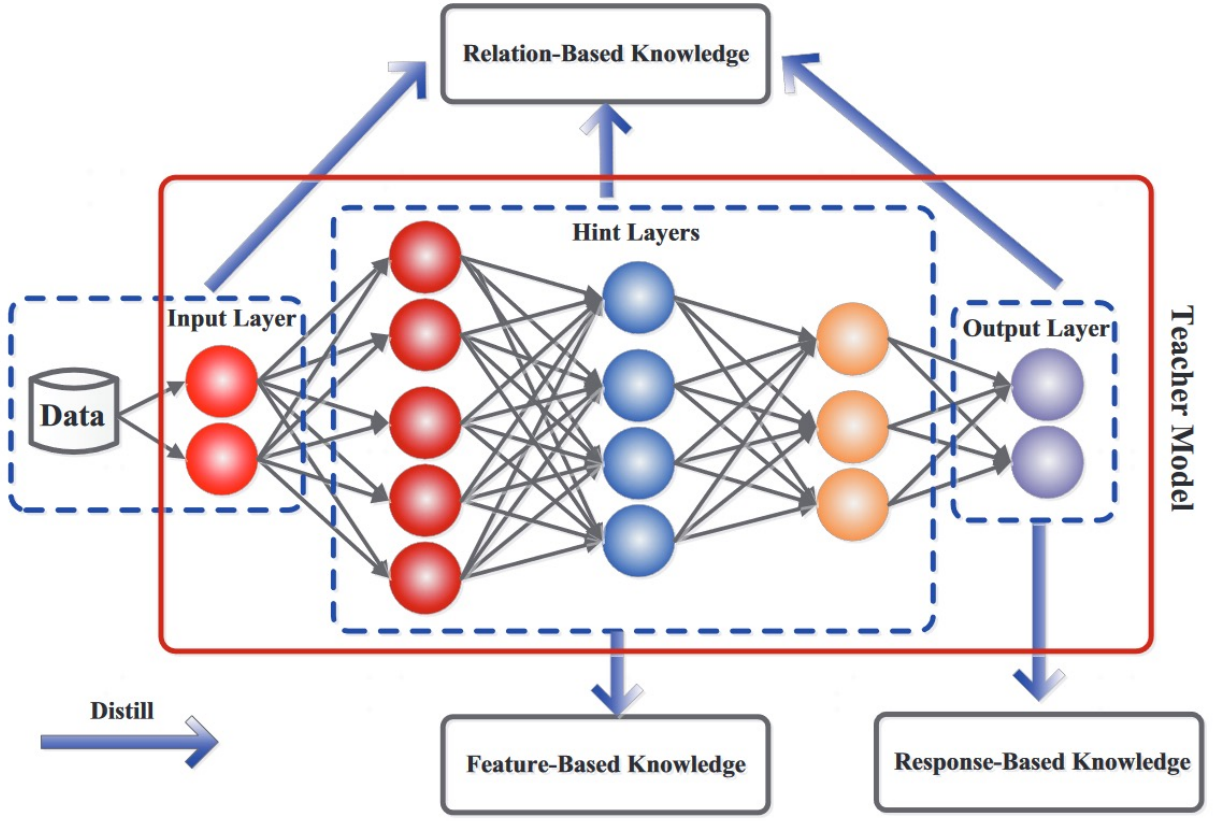
为什么使用logits蒸馏

1. 预训练蒸馏成本过高，通常是微调阶段的指令蒸馏

2. 教师和学生模型可以使用相同架构，但超参数不同，特征维度不同，但词表不会随模型规模变化，logits/回复蒸馏最直观

3. 数据稀缺/垂域场景下，logits蒸馏比较高效

4. 实践发现Logits蒸馏可以强化模型泛化能力和减少量化损失



[1] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," arXiv:2006.05525 [cs, stat], Oct. 2020, Accessed: Nov. 25, 2020. [Online]. Available: http://arxiv.org/abs/2006.05525

2.1知识蒸馏-LLM的logits知识蒸馏

关键观察

- 微调后的 LLM 的 logits 表现出**极端的长尾分布**，其中的重要信息集中于非长尾部分
- 采样策略(top-k/top-p)下，大模型**logits的内部排序**对生成结果有着重要影响

方法-BiLD

- 对logits分布的长尾部分进行过滤，减少概率极小值参与计算
- 将蒸馏过程从logits的概率分布对齐，转换为logits的差值分布对齐，将logits之间的排序信息进行放大
- 构建双向损失

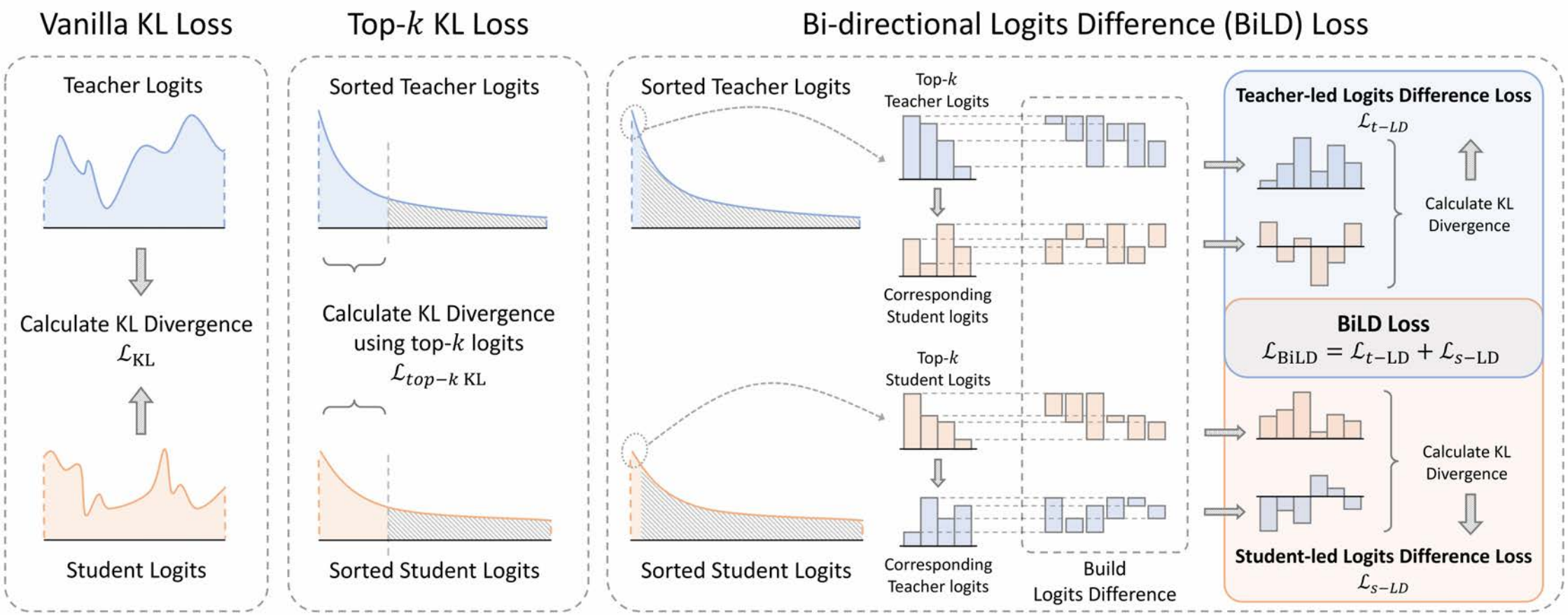
结果

在13个数据集上的大量实验证明，BiLD仅使用 top-8 位 logits 就可以获得 SOTA 的效果。

M. Li, F. Zhou, and X. Song, "BiLD: Bi-directional Logits Difference Loss for Large Language Model Distillation," Sep. 11, 2024, *arXiv*. arXiv:2406.13555.

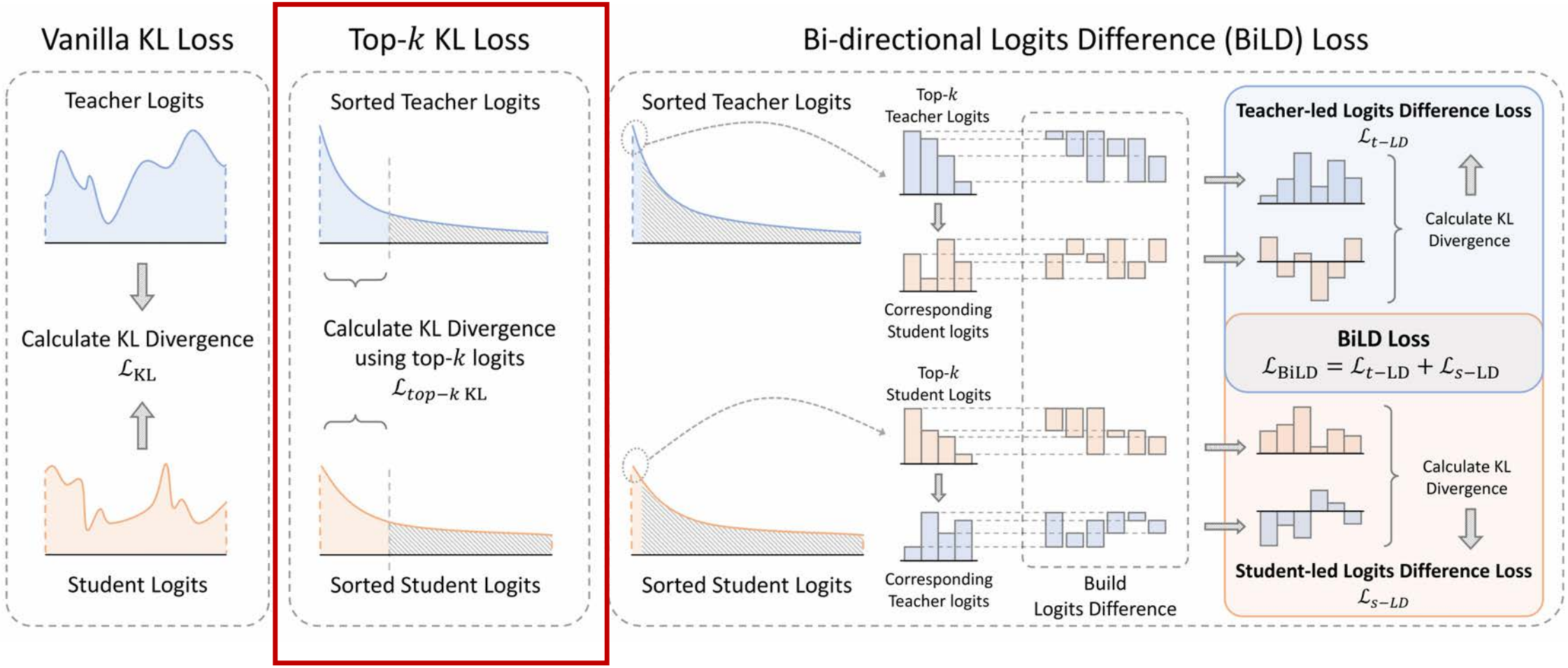
2.1知识蒸馏-LLM的logits知识蒸馏

BilD损失函数示意图



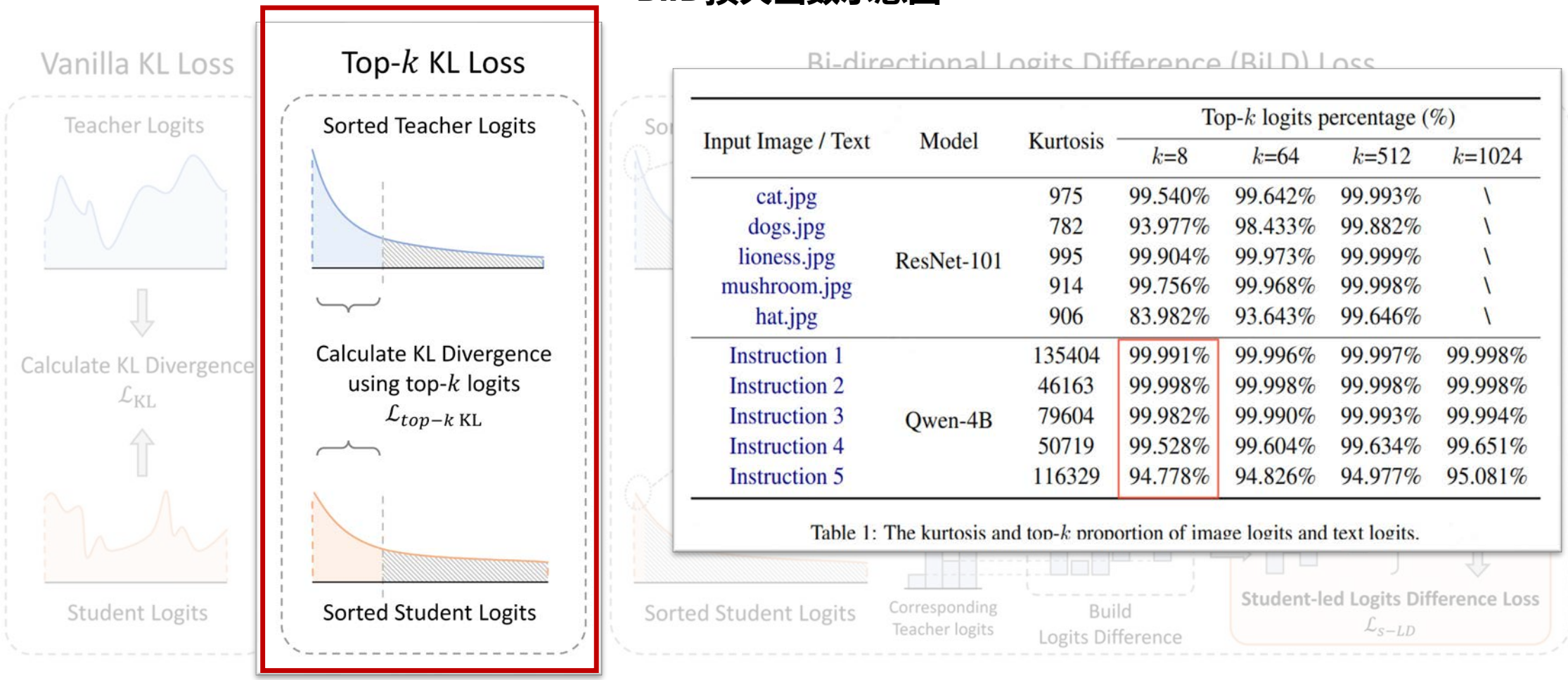
2.1知识蒸馏-为什么只用Top-K?

BilD损失函数示意图



2.1知识蒸馏-为什么只用Top-K?

BilD损失函数示意图



Input Image / Text	Model	Kurtosis	Top-k logits percentage (%)			
			k=8	k=64	k=512	k=1024
cat.jpg	ResNet-101	975	99.540%	99.642%	99.993%	\
dogs.jpg		782	93.977%	98.433%	99.882%	\
lioness.jpg		995	99.904%	99.973%	99.999%	\
mushroom.jpg		914	99.756%	99.968%	99.998%	\
hat.jpg		906	83.982%	93.643%	99.646%	\
Instruction 1	Qwen-4B	135404	99.991%	99.996%	99.997%	99.998%
Instruction 2		46163	99.998%	99.998%	99.998%	99.998%
Instruction 3		79604	99.982%	99.990%	99.993%	99.994%
Instruction 4		50719	99.528%	99.604%	99.634%	99.651%
Instruction 5		116329	94.778%	94.826%	94.977%	95.081%

Table 1: The kurtosis and top-k proportion of image logits and text logits.

2.1知识蒸馏-为什么要使用差值?

先介绍两种蒸馏方式，正向蒸馏和反向蒸馏，记教师模型分布为P，学生模型分布为Q：

正向蒸馏(forward KL)

$$D_{KL}(P\|Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

反向蒸馏(reverse KL)

$$D_{KL}(Q\|P) = \sum_i Q(i) \log\left(\frac{Q(i)}{P(i)}\right)$$

均值寻求(mean-seeking)

- 正向蒸馏中，当Q(i)<< P(i)时会产生很大损失，迫使学生模型去尽力模仿教师的整体分布，这样会让学生模型的整体分布较为平滑，在NLG任务上体现为**多样性强，但置信度低**

模式寻求(mode-seeking)

- 反向蒸馏中，损失会强调二者在教师低概率部分的对齐程度，反而促成了学生模型更关注教师的高概率部分，在NLG任务上体现为**置信度高，但多样性差**

2.1知识蒸馏-为什么要使用差值?

将正反两个损失函数线性结合起来是一个比较直观的想法，在此之上，我们通过在top-k logits上面构造差值矩阵，有以下几点收益:

- Top-k的选择过滤掉了绝大多数概率极小值
- 使用差值矩阵的上三角矩阵展开计算交叉熵，客观上扩大了softmax的分母个数，弱化极大概率值对蒸馏过程的影响
- 模型拟合概率差异的分布难度高于直接学习原始尖锐的概率分布，收敛效果更好

我们提出使用学生和教师之间的Overlap@k来评估学生和教师之间，在生成场景下的相似程度:

- Overlap@1: logits的top1是否相同，衡量模型在贪心解码的情况下和teach教师er的相似程度
- Overlap@k: logits 的top-k个位置的重合程度，衡量模型在采样生成策略下和教师的相似程度

BiLD在均值寻求和模式寻求之间取得了不错的均衡效果。

Model	Method	Overlap@1	Overlap@8
BLOOM-3B	SFT	74.89	44.61
	Vanilla KL	**82.51**	54.64
	RKL	82.31	54.64
	DKD	74.00	52.39
	NKD	82.11	53.25
	NormKD	48.80	36.95
	Top-k KL	81.67	55.73
	BiLD	81.72	**56.57**
BLOOM-1B	SFT	74.40	40.71
	Vanilla KL	80.82	51.91
	RKL	80.71	51.58
	DKD	75.44	48.83
	NKD	79.59	50.01
	NormKD	73.56	42.70
	Top-k KL	80.20	50.87
	BiLD	**81.21**	**52.86**
Qwen-1.8B	SFT	93.30	53.28
	Vanilla KL	94.35	68.02
	RKL	94.31	67.93
	DKD	94.09	67.01
	NKD	94.02	65.01
	NormKD	94.26	68.32
	Top-k KL	**94.43**	67.55
	BiLD	94.39	**70.97**
Qwen-0.5B	SFT	91.67	47.29
	Vanilla KL	92.72	61.81
	RKL	92.54	61.65
	DKD	91.50	56.62
	NKD	92.88	59.11
	NormKD	91.76	58.16
	Top-k KL	93.11	64.00
	BiLD	**93.23**	**68.58**

2.1知识蒸馏-实验结果

我们在**13**个数据集上验证了BiLD的效果，在全部**4**组实验设置上均分达到了SOTA的水平。

Code:
https://github.com/fpcsong/BiLD

Paper:
https://arxiv.org/pdf/2406.13555

Model	Method	Arc-C (Acc.)	Arc-E (Acc.)	boolQ (Acc.)	CB (Acc.)	COPA (Acc.)	HellaSwag (Acc.)	MultiRC (F1a/EM)	PIQA (Acc.)	ReCoRD (F1/Acc.)	RTE (Acc.)	WiC (Acc.)	WinoGrande (Acc.)	WSC (Acc.)	Avg.
BLOOM-7B	Teacher	50.84	68.95	85.26	89.29	81.00	76.08	81.36/40.82	74.92	79.87/78.50	83.03	72.41	71.51	65.38	72.15
BLOOM-3B	SFT	44.15	61.75	84.04	87.50	67.00	57.00	77.09/36.20	70.84	76.05/74.59	78.34	69.75	69.69	64.42	66.56
	Vanilla KL	49.50	68.07	84.50	87.50	76.00	72.60	78.89/36.52	74.27	79.81/78.32	81.59	71.94	70.96	**74.04**	71.21
	RKL	**50.50**	68.42	84.62	87.50	**80.00**	72.20	78.95/36.41	74.48	79.63/78.13	82.31	72.57	71.35	68.27	71.29
	DKD	49.50	**69.82**	85.26	91.07	**80.00**	71.54	77.84/35.68	73.01	79.09/77.65	79.42	**73.20**	70.96	66.35	71.04
	NKD	50.17	67.19	84.01	**92.86**	79.00	72.68	79.69/37.67	73.50	78.50/77.09	81.23	71.32	**72.06**	66.35	71.16
	NormKD	48.16	67.54	**85.35**	89.29	79.00	70.57	77.19/35.57	71.82	78.44/76.98	80.87	72.88	70.48	68.27	70.52
	Top-k KL	47.49	68.25	84.19	87.50	77.00	**72.75**	79.39/37.67	**74.59**	79.40/78.01	**82.67**	72.10	70.80	64.42	70.57
	BiLD (ours)	49.83	67.54	84.86	91.07	**80.00**	72.10	**79.49/37.78**	73.61	**79.96/78.57**	**82.67**	72.88	71.98	71.15	**71.85**
BLOOM-1B	SFT	34.78	53.86	80.76	87.50	64.00	37.39	73.18/30.12	65.72	72.04/70.59	73.65	67.71	67.40	64.42	61.38
	Vanilla KL	45.48	64.39	83.67	87.50	73.00	65.31	77.66/33.37	70.95	77.11/75.67	77.62	68.03	68.43	68.27	67.82
	RKL	45.48	**65.44**	83.43	85.71	74.00	65.70	76.63/32.95	70.78	**77.51/76.10**	79.42	70.69	68.27	64.42	67.88
	DKD	42.47	64.56	**84.10**	85.71	72.00	63.72	75.49/31.79	69.48	75.78/74.46	79.78	**71.79**	68.98	**69.23**	67.55
	NKD	43.14	60.88	82.75	89.29	68.00	63.53	76.94/34.84	70.73	75.31/73.87	77.62	69.44	**69.30**	61.54	66.53
	NormKD	42.81	61.05	83.82	83.93	69.00	62.80	74.13/30.75	67.74	74.49/72.95	77.62	69.91	67.80	65.38	65.81
	Top-k KL	**49.50**	62.11	83.06	89.29	74.00	**65.72**	78.30/34.73	71.22	77.28/75.89	77.98	70.22	**69.30**	60.58	67.97
	BiLD (ours)	44.48	62.98	83.39	**91.07**	**77.00**	64.84	**78.37/35.78**	**72.20**	77.23/75.93	**80.14**	70.53	**69.30**	68.27	**68.92**
Qwen-4B	Teacher	68.23	81.40	87.43	96.43	89.00	86.30	85.85/51.63	82.10	82.59/81.10	87.73	72.73	80.82	74.04	79.92
Qwen-1.8B	SFT	52.17	73.86	83.88	91.07	86.00	72.58	79.95/39.66	75.90	77.37/76.05	84.12	71.79	72.06	61.54	72.36
	Vanilla KL	**55.52**	74.74	85.60	96.43	86.00	77.74	79.46/36.52	76.66	79.24/36.52	85.56	69.59	75.14	64.42	73.98
	RKL	50.84	76.14	85.14	94.64	87.00	77.85	79.52/39.14	76.39	79.49/77.98	84.48	71.47	76.64	69.23	74.38
	DKD	51.84	**77.02**	85.75	**98.21**	85.00	76.90	80.56/39.77	74.54	77.91/76.18	84.48	71.16	76.56	67.31	74.21
	NKD	51.84	73.33	84.53	92.86	**88.00**	77.49	81.98/42.18	76.61	79.03/77.58	84.12	70.85	74.98	66.35	73.90
	NormKD	52.84	76.49	85.26	96.43	85.00	77.24	80.81/40.50	74.76	78.22/76.48	**85.92**	70.53	**76.87**	**70.19**	74.50
	Top-k KL	53.85	76.14	**85.93**	96.43	82.00	**77.99**	81.81/41.03	76.71	**80.08/78.71**	83.39	71.32	75.85	67.31	74.36
	BiLD (ours)	54.85	73.16	84.53	96.43	**88.00**	77.56	**81.49/42.92**	**77.97**	79.87/78.56	85.56	**72.10**	76.01	68.27	**75.07**
Qwen-0.5B	SFT	37.46	62.11	80.40	87.50	77.00	46.71	74.24/28.54	68.44	71.19/69.79	77.26	66.30	69.38	59.62	63.88
	Vanilla KL	43.14	63.68	81.74	85.71	78.00	66.73	75.97/29.07	71.87	72.55/70.91	79.78	70.53	71.35	60.58	67.16
	RKL	**46.49**	64.39	81.53	87.50	**79.00**	67.06	75.37/29.38	71.16	71.46/69.55	**82.31**	69.91	70.80	58.65	67.52
	DKD	40.80	62.98	82.66	82.14	77.00	61.03	72.35/26.55	66.87	65.68/63.20	81.59	70.06	70.64	61.54	65.16
	NKD	41.14	63.86	82.42	94.64	78.00	68.30	79.33/36.20	73.01	74.81/73.35	**82.31**	67.40	72.22	71.15	69.54
	NormKD	41.14	61.40	82.72	83.93	77.00	62.31	74.13/29.07	68.55	67.17/64.79	**82.31**	**71.16**	71.43	62.50	66.02
	Top-k KL	43.14	65.79	82.39	94.64	77.00	68.58	78.83/35.89	71.82	74.30/72.95	**82.31**	69.28	**73.24**	62.50	69.19
	BiLD (ours)	41.81	**67.54**	**83.43**	**96.43**	78.00	**68.99**	**79.72/37.78**	**73.34**	**75.22/73.94**	81.59	69.75	72.22	**74.04**	**70.68**

2.2目标结构化剪枝


模型剪枝
权重稀疏化/weight
计算成本低，不改变模型结构
缺点：加速和存储收益较少，需要特定硬件才能获得收益
半结构化剪枝/block
计算成本低，效果损失较少
破坏模型结构，导致推理端需要额外适配
目标结构化剪枝/slice
效果损失较少，不破坏模型结构，可以自定义模型形状
计算成本高，优化难度大

2.2目标结构化剪枝-L0正则化方法

基本原理

L0正则化剪枝方法的核心思想是减少模型中非零参数的数量，通过在模型的损失函数中引入L0范数来实现参数的稀疏化。**由于L0范数的非凸性，直接优化L0范数是困难的。**通常采用近似方法来实现L0正则化剪枝，下面介绍经典的基于Hard Concrete Distribution实现L0剪枝的原理：

1. 首先为需要剪枝的权重/块初始化一个**mask(0~1)**，初始全1，为避免过多的公式介入，直接记为**logα**.
2. 在训练过程中，每个mask都会在一个**均匀分布**中采样**噪声**，获得一个动态的mask z 参与模型计算，以此来通过lm_loss逐渐评估出权重/块的重要程度。

$$u \sim \mathcal{U}(0,1), \quad s = \text{Hard-Sigmoid}\left((\log u - \log(1-u) + \log\alpha)/\beta\right), \quad \bar{s} = s(\zeta - \gamma) + \gamma$$
$$z = \min(1, \max(0, \bar{s}))$$

3. 每一个mask都相当于一个0-1之间的概率分布，通过一个**概率累积分布函数**来控制一批mask的分布，当收敛完成后，该累积分布函数可以确保这批mask中目标比例一定为0，以此来达到剪枝的目的。

$$\mathcal{L}_C = \sum_{j=1}^{|\theta|}\left(1 - Q_{\bar{s}_j}(0 \mid \phi)\right) = \sum_{j=1}^{|\theta|}\text{Hard-Sigmoid}\left(\log\alpha_j - \beta\log\frac{-\gamma}{\zeta}\right)$$
$$\hat{\mathbf{z}} = \min(1, \max(0, \text{Sigmoid}(\log\alpha)(\zeta - \gamma) + \gamma)), \quad \boldsymbol{\theta}^* = \tilde{\boldsymbol{\theta}}^* \odot \hat{\mathbf{z}}$$

2.2目标结构化剪枝-L0正则化方法

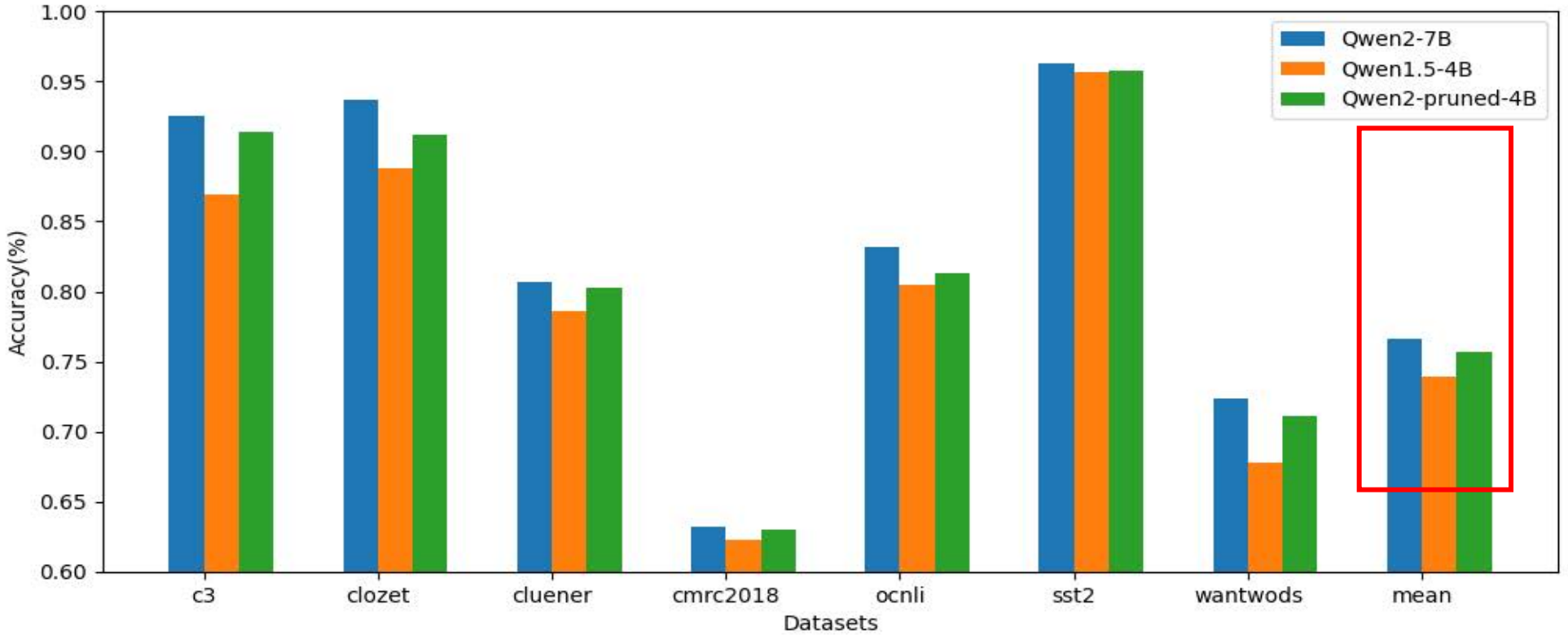
缺点	优化方向	核心优化思路	具体改进措施
收敛速度慢，对数据规模要求比较高，mask的优化速度和模型权重的优化速度不好均衡	如何让lm_loss通过mask对权重的评估更快、更准确	把L0正则化剪枝直接通过梯度下降获得0-1二值mask转换为两阶段问题： • **排序**：通过梯度下降评估参数重要程度，体现在mask数值的排序上 • **剪枝**：通过soft top-k mask将排序结果渐进的转化为0-1二值序列	加入**梯度缩放因子**，将mask从0-1映射到更小的范围，e.g. (0,1e-3]，并通过**伪输入**技巧将mask引入到lm_loss的优化过程中，让mask对模型效果感知更强，提升优化效率。
均匀分布的噪声均值太高，噪声波动影响的参数量较大（类似dropout），无法适用于剪枝比例比较大的情况(例如90%以上)	Mask的噪声分布设计和实现		使用加入**直通估计**的hardtanh，更加充分的利用mask的梯度信息。
为了达成剪枝目标，会产生较多0-1之间的mask，剩余参数存在浪费，影响剪枝后模型的效果	概率累积分布函数的设计和实现		重写**噪声采样逻辑**，转变噪声分布，并限制噪声的绝对值的上限，仅用于评估重要性，不致力于产生二值mask。 整合了Hard Concrete Distribution和soft-topk-mask的思路，设计了一些列辅助函数，保证剪枝目标达成并且不浪费参数。

排序和剪枝两个阶段在训练过程中动态交替进行，实现了较为平缓的剪枝过程。

2.2目标结构化剪枝-L0正则化方法效果

基于L0的改进算法，从Qwen2-7B剪枝到4B，超过了Qwen1.5-4B的效果。

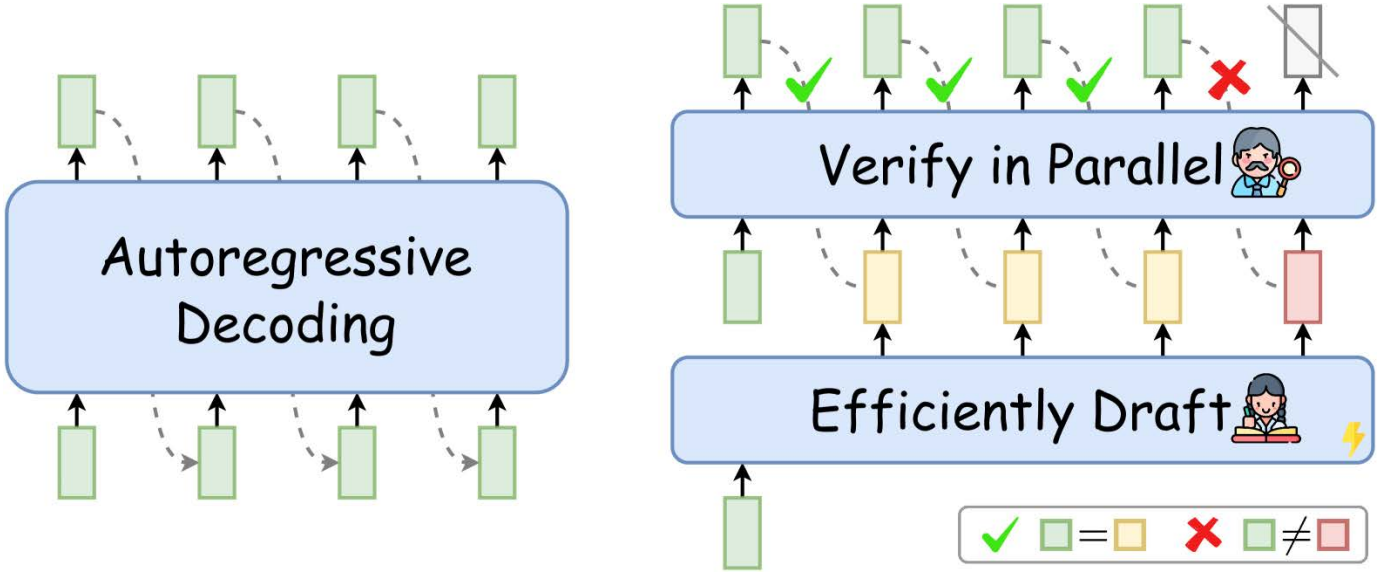
L0正则化剪枝在Qwen7B->4B的结果



2.3目标结构化剪枝应用-低成本draft model构建

投机解码基础

投机解码（Speculative Decoding）是一种用于加速大型语言模型（LLM）推理过程的方法。它通过“**先推测后验证**”（Draft-then-Verify）的策略，提高解码效率和并行性。在每个解码步骤中，投机解码首先利用一个小型的**草稿模型**生成多个候选token，然后使用目标大型语言模型并行验证这些候选token，一旦候选被接受，则可以减少大模型的decoding次数，以加速整个推理过程。



Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., & Sui, Z. (2024). Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding. Retrieved from https://arxiv.org/abs/2401.07851

2.3目标结构化剪枝应用-低成本draft model构建

自生成方法(self-drafting)

通过模型自身构建轻量的draft模型。例如跳过模型的某些层(Layer Skipping)，添加额外的层或者其他组件(eagle/eagle2)，添加额外的Lm head(medusa)等。

- **优点：**基于大模型自有的能力，通常能达成**更高的接受率**，在采样策略下也能和基模型有更高的风格一致性和回复质量。
- **缺点：**训练出来的draft权重和目标模型完全耦合，需要更改目标模型的推理逻辑，**不够灵活**，需要随着目标模型一起迭代，并且通常自生成的单次推理计算量更高。

独立生成方法(Independent Drafting)

通过一个额外的轻量小模型来完成猜测过程。

- **优点：****高度灵活**，和目标模型完全解耦，只要模型的任务不变，大小模型可以分开优化。对推理框架改动很小，不影响大模型推理逻辑。
- **缺点：**需要额外的编码时间，小模型和大模型的行为对齐较难优化，小模型的训练成本较高。

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., & Sui, Z. (2024). Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding. Retrieved from https://arxiv.org/abs/2401.07851

2.3目标结构化剪枝应用-低成本draft model构建

基于剪枝的draft模型构建

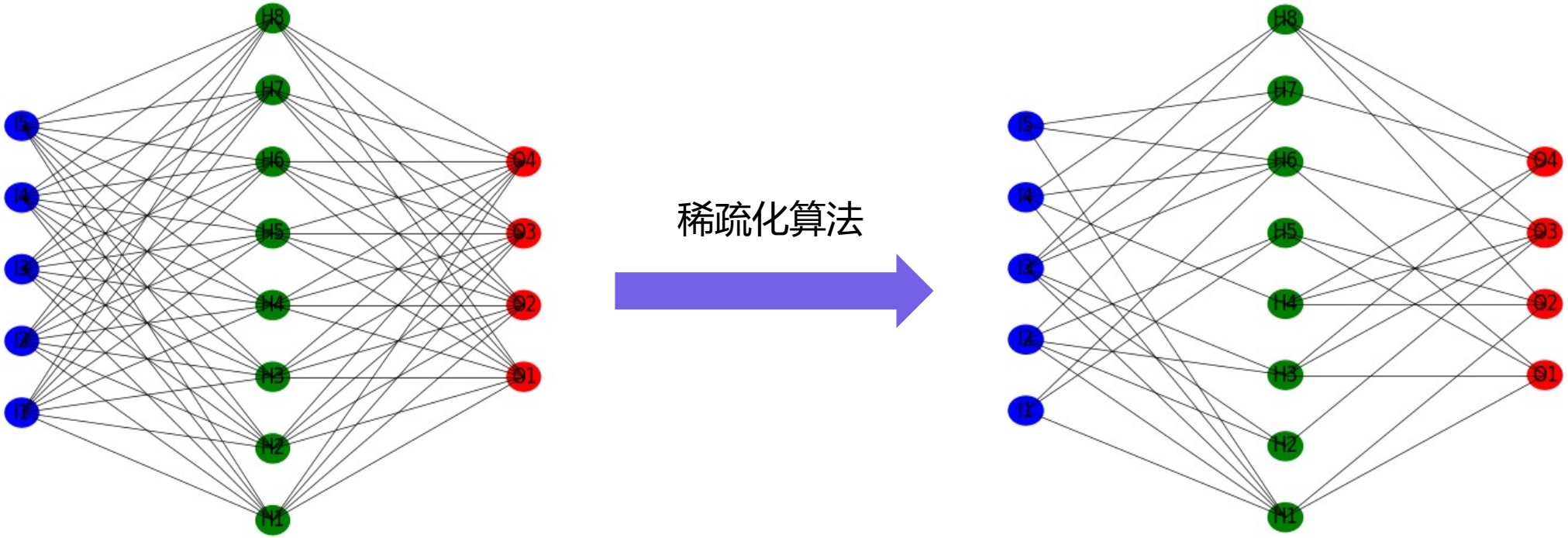
利用前述改进的L0剪枝+BiID蒸馏，将目标模型从4B直接剪枝到200M，作为draft模型。

1. 使用相同训练集训练（而非针对基模型进行数据蒸馏），保证更高的接受率。
2. 针对**极端稀疏度**设计的稀疏度调度策略和噪声分布，保证剪枝完成和效果收敛。
3. 剪枝完成后，针对小模型进行二阶段QAT训练。
4. 整个过程可以在一台8*A100上3天以内完成。相比于直接训练一个200M模型（类比2个bert-base）极大的节省了机器成本。

当前模型经过量化后部署在手机端侧，在OPPO端侧系统/三方通话摘要功能上可以实现1.4~2倍的加速比。

2.4权重稀疏化

权重稀疏化（Weight Sparsification）是一种在深度学习模型中减少参数数量的技术，通过将模型中的某些权重置为零，从而实现模型的压缩和加速。稀疏化方法可以显著减少模型的内存占用，并依赖硬件相关的稀疏化算子获得一定程度的加速效果。



2.4权重稀疏化-开源方法

大模型时代的稀疏化方法：

- post-training成为主流，关注剪枝效率，争取不做/少做权重更新
- 主要依赖ppl测试
- 均支持结构化稀疏模式（硬件友好）

稀疏化方法对比

方法	是否更新权重	是否需要校准数据	剪枝度量方法
Magnitude (Han et al., 2016a)	X	X	$\|\mathbf{W}\|$
SparseGPT (Frantar & Alistarh, 2023)	✓	✓	$\left[\|\mathbf{W}\|^2 / \operatorname{diag}\left[\mathbf{H}^{-1}\right]\right]_{ij}$
Wanda (Sun et al., 2024)	X	✓	$\|\mathbf{W}_{ij}\| \cdot \|\mathbf{X}_j\|_2$
Pruner-Zero(Dong et al., 2024)	X	✓	$\|\|\mathbf{W}\| \times \|\mathbf{W}\|\| \times \sigma(\|\mathbf{G}\|)$
RIA(Zhang et al., 2024)	✓	✓	$(\|W_{ij}\| \sum_k \|W_{kj}\| + \|W_{ij}\| \sum_k \|W_{ik}\|) \times \|X_i\|^a$

2.4权重稀疏化-开源方法对比

在Qwen2.5-3B和llama2-7B上的实验得到如下结论：

- PPL不是一个好的衡量指标，下游任务的损失和PPL不成比例
- Sparse算法当前还不成熟，损失都比较大，不能像量化一样部署
- 开源算法恢复前的差异较大，恢复后都很接近，mask的质量差异不大

50%稀疏开源方法效果对比

方法	结果(8个数据集平均)	lora性能恢复
Dense	0.7569	N/A
SparseGPT	**0.7341(-2.28)**	**0.7413(-1.56)**
Wanda	0.7079(-4.9)	0.737(-1.99)
Prune-Zero	0.7013(-5.56)	0.739(-1.79)
RIA	0.7060(-5.09)	0.738(-1.89)

50%稀疏开源方法PPL对比

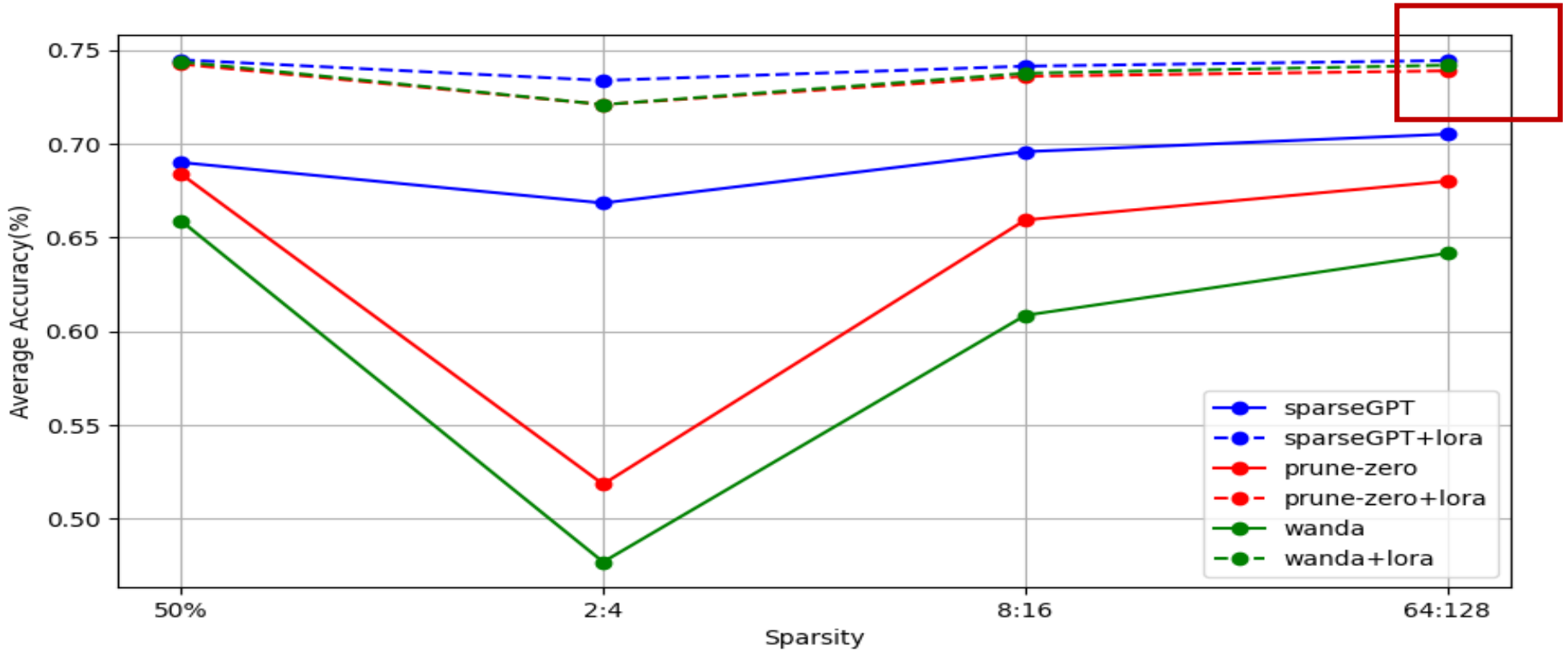
模型/方法	wikitext2-PPL
Llama2-7B	5.47
SparseGPT	7.004
Wanda	6.919
Prune-Zero	**6.7059**
RIA	6.8068

2.4权重稀疏化-稀疏化模式

在不同稀疏化模式下进行测试：

- 结构化稀疏损失更大，但随着n,m的值增加会逐渐好转，整体来说损失大于量化
- n,m值增加更有利于性能恢复

不同N:M效果和恢复结果



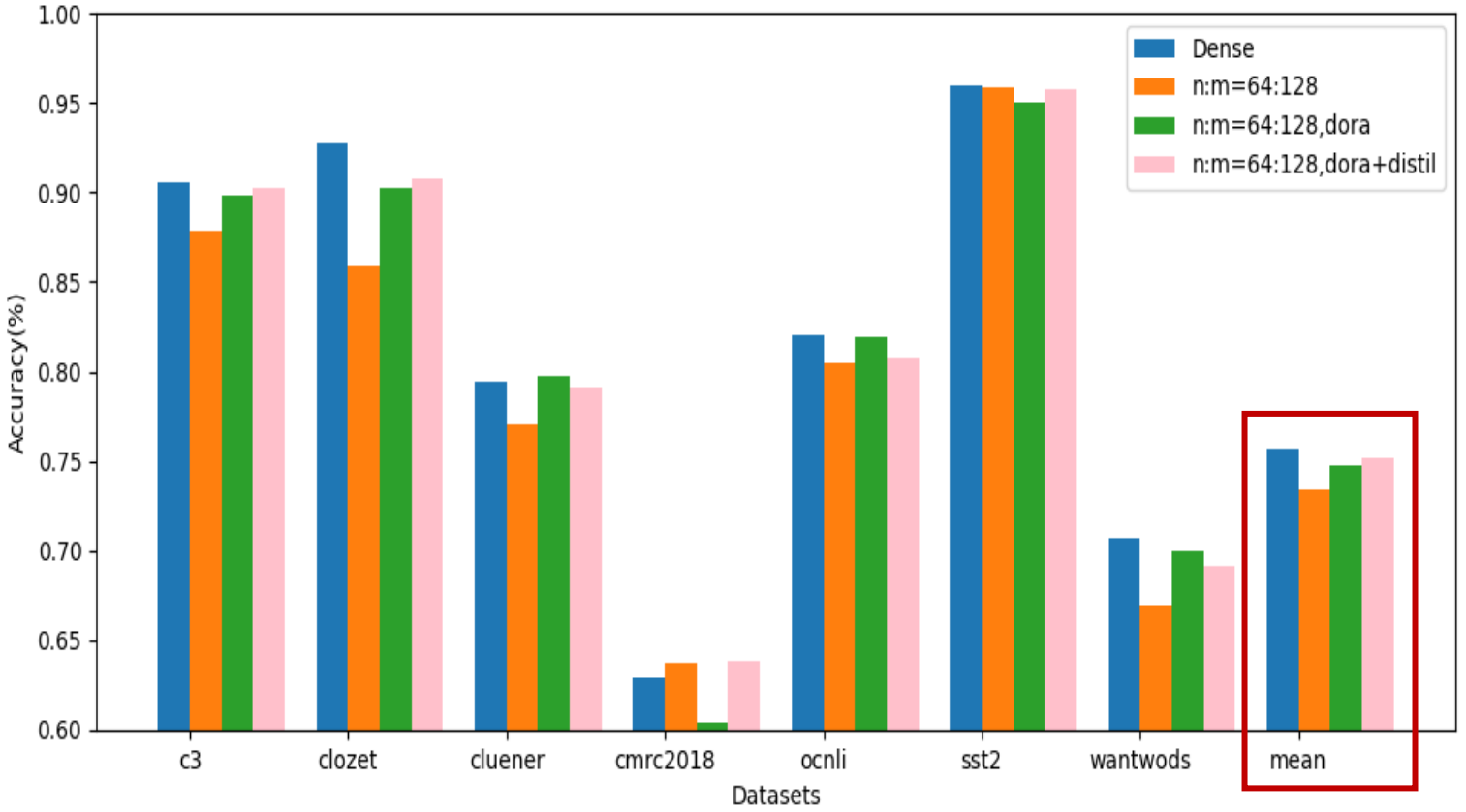
2.4权重稀疏化-性能恢复

基于BilD设计了细粒度蒸馏恢复方案，显著降低sparse性能损失，基本达到了可落地的水平。

与lora恢复效果的对比

方法	结果(8个数据集平均)
Dense	0.7569
SparseGPT(50%)	0.7341(-2.28)
+lora	0.7413(-1.56)
Ours	**0.7513(-0.56)**

8个任务上的性能恢复对比



PART 03

量化感知训练

3.1量化感知训练-量化基础

量化(quantization)是将神经网络中的高精度浮点数（如float32/16）转换为低精度整数（如int8/4）表示的过程。其目的是减少模型的存储空间和推理过程的显存占用，从而加大吞吐量并降低设备功耗。

由于端侧的内存功耗等方面限制，当前量化已经是模型在端侧部署过程中必须的一环。

量化的优点

- 减少模型大小，部署和更新更便捷；在芯片量化算子的加持下加速推理过程；
- 更低的功耗和内存占用，相比其他压缩方法计算成本更低；

量化的缺点

- **算法性能损失。对生成式模型而言，会给模型行为带来一定程度的不确定性。**

损失的来源

- 权重分布不够均匀
- 激活需要进行对称量化（为了保持0点不变，如果和sparse结合则权重也需要），导致异常值的影响被放大

3.3量化感知训练-方法

基于直通估计的伪量化算子实现

基础的线性量化是不可微的，因此需要引入直通估计(Straight-Through Estimator, STE)来完成梯度回传。一个基础的线性量化算子如下：

$$Q(x) = \text{clip}\left(\text{round}\left(\frac{x}{\Delta}\right), \quad Z_{\min}, Z_{\max}\right)$$

STE部分需要设计Q(x)的梯度估计形式，也是各种QAT算法的差异之一，我们使用了最朴素的STE实现。

$$\frac{\partial Q(x)}{\partial x} = \begin{cases} 1 & \text{if } x \text{ is close to a quantized value} \\ 0 & \text{otherwise} \end{cases}$$

激活敏感层评估

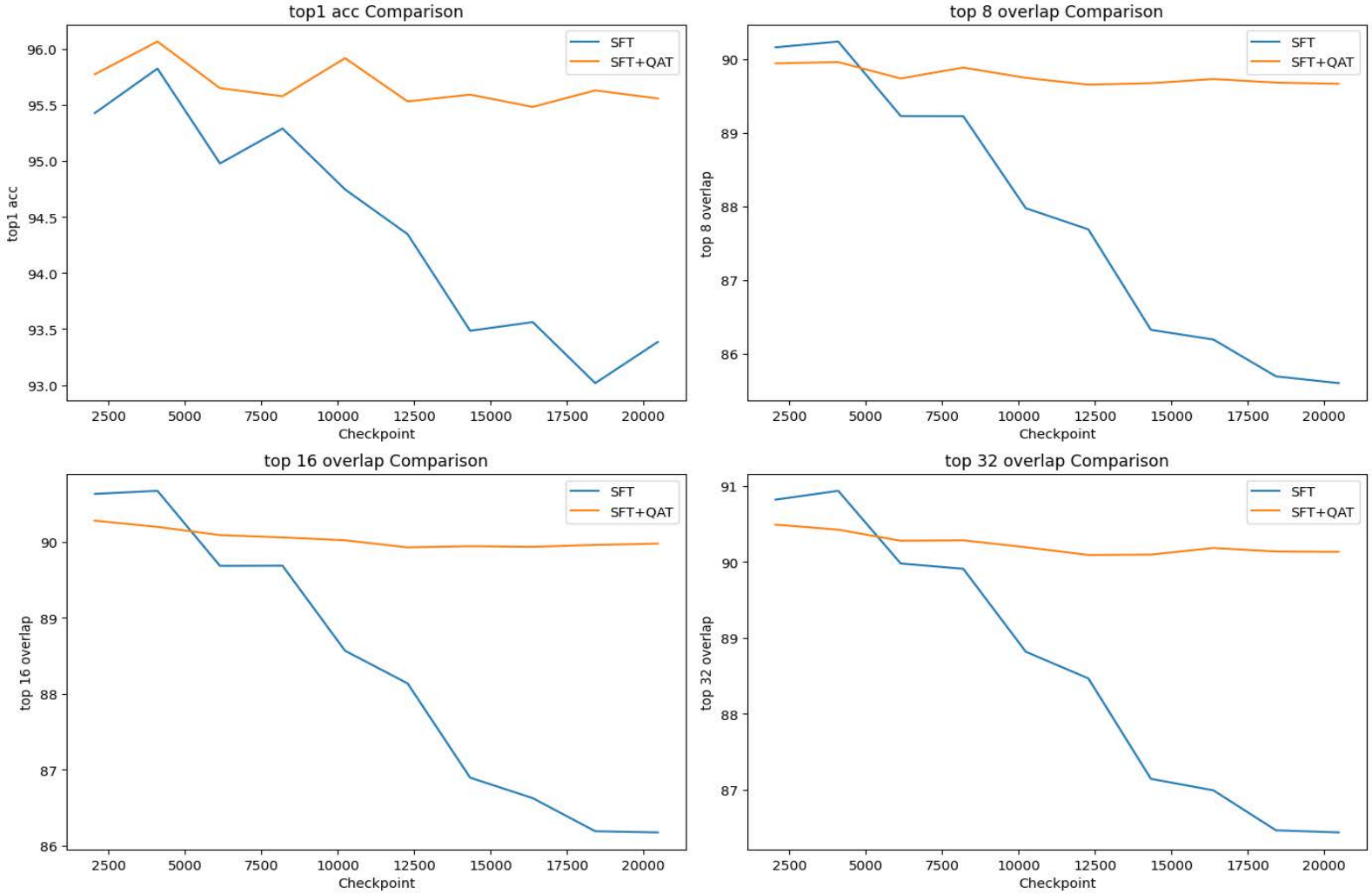
加入伪量化算子是会影响训练效果的，我们会预先评估出离群点较多的层（通常在topk层）仅对这些层加入QAT算子进行训练。

利用高温蒸馏的均值寻求效应

实验发现logits的分布和topk层的激活分布有同步的倾向，模型训练越充分，logits越尖锐，激活离群点越多，因此我们借助之前蒸馏部分的均值寻求效应，**充分训练**一个teacher模型，再使用较高的温度蒸馏到目标模型，迫使logits分布更平滑，发现能够显著提升QAT的效果。

3.4量化感知训练-效果

通过引入QAT技术，能够让训练全周期的结果对量化更加鲁棒，避免float模型的选型结果和部署效果产生不可控的差异。

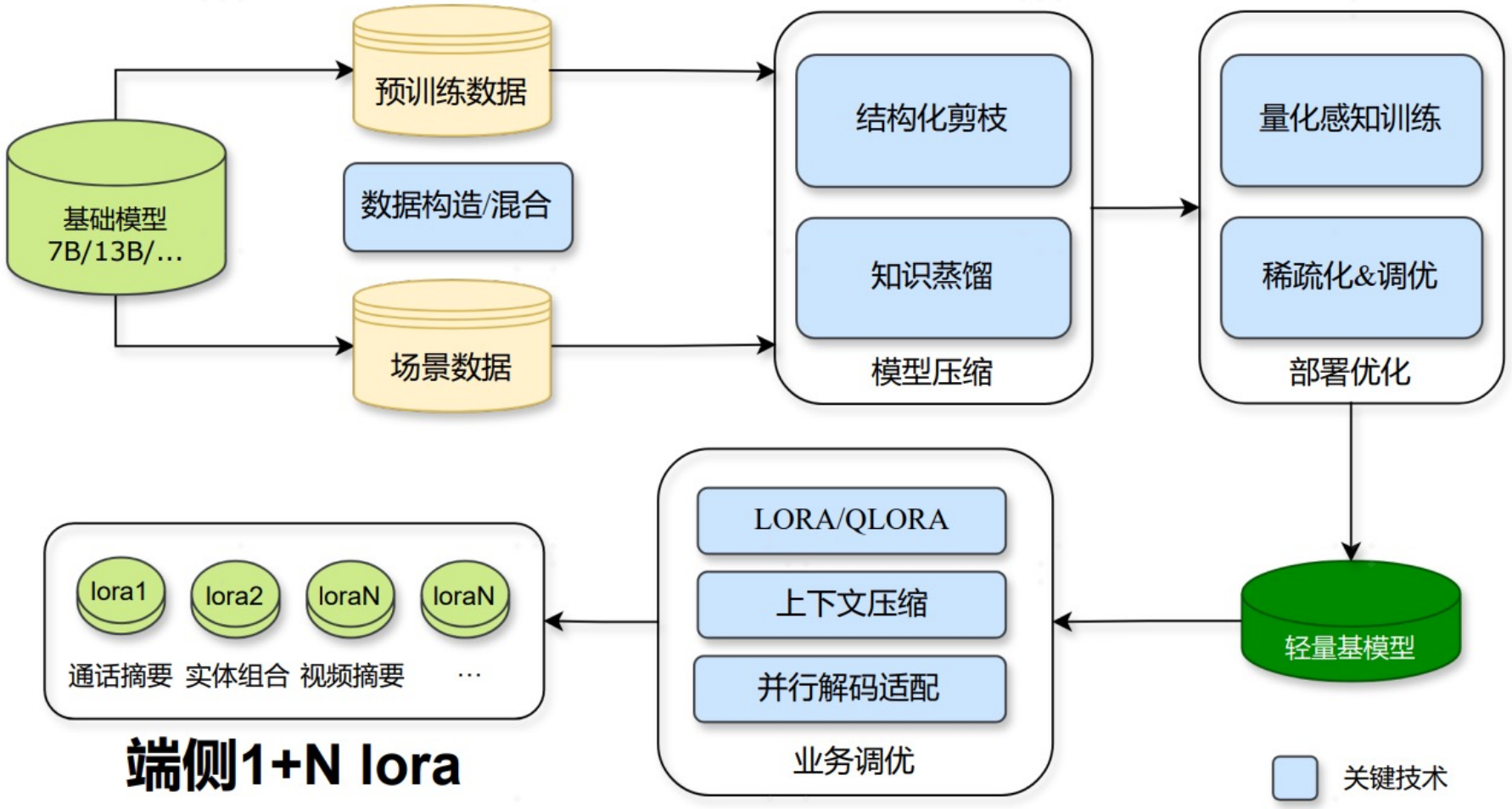


PART 04

案例-OPPO端侧1+NLora架构

1+NLora-算法布局

OPPO端侧1+Nlora算法优化流程



1+NLora案例-通话摘要

项目背景

1.0版本：Find X7 系列上跟随行业首个端侧7B大模型落地。主打隐私安全，全流程在端侧运行。

2.0版本：Find X8 系列，基于1+Nlora架构重构，更轻量，算法效果也显著提升。

难点

1. 单次推理上下文窗口有限
2. 端侧的ASR效果和云侧算法有一定差距，模型输入的噪音更大
3. 并行程度较低，只能使用效果较差的流式长文摘要方案
4. 量化损失在摘要事实性方面体现较为明显

1+NLora案例-通话摘要算法优化

数据构造

1. 通过COT方式**强化摘要实体密度**；
2. 针对数字实体、对话视角、长数字序列等高频错误**构造对抗样本**，显著缓解模型幻觉问题；
3. 通过动态切割长度，切换prompt等方式进行**数据增强**；
4. **混合指令优化**：使用指令数据构造符合预训练数据分布的训练集，提升模型的泛化能力和微调效果；
5. **数据评估规则积累**：从格式、实体密度、可读性等角度积累了一系列标签数据自动化评估规则，通过多次迭代保证训练集数据质量。

微调算法

1. 广泛吸收优秀的开源工作，整合了C-RLFT(Guan et al., 2024)，SPIN(Chen et al., 2024)等微调策略。
2. 针对混合指令优化实现了加权的标签长度归一化损失函数，在混合指令场景下作用显著。

PART 05

总结与展望

端侧模型发展趋势

服务

端云协同会越来越成熟，端侧算力占比会不断提高

隐私和个性化会成为主要卖点

终端上有一个服务多场景的基础模型会是标配

技术

量化位宽会越来越低，量化感知/低精度训练、
推理技术会被重视起来

端侧部署会成为一个研究方向，
例如面向NPU的高性能架构、
极致的提升参数效率或许会成为研究热点

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 · 敦煌站
K+ 思考周®研习社
时间：2025.08.29-30

K+峰会 · 上海站
K+ 金融专场
时间：2025.10.17-18

K+峰会 · 香港站
K+ 思考周®研习社
时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 · 上海站
AI+研发数字峰会
时间：2025.05.17-18

AiDD峰会 · 北京站
AI+研发数字峰会
时间：2025.08.08-09

AiDD峰会 · 深圳站
AI+研发数字峰会
时间：2025.11.28-29

AiDD峰会详情

AiDD AI+研发数字峰会 第5届
AI+ Development Digital summit

利用AI技术深化计算机对现实世界的理解

推动研发进入智能化时代