QPruner

This is just the beginning of something big.

Abstract

lenges. Structured pruning is an effective approach to reducing model size, but it often results in significant accuracy degradation, necessitating parameter updates to adapt. Unfortunately, such fine-tuning requires substantial memory, which limits its applicability. To address these challenges, we introduce quantization into the structured pruning framework to reduce memory consumption during both fine-tuning and inference. However, the combined

Structured pruning은 모델 크기를 줄이는 데 효과적인 방법이지만 종종 눈에 띄는 정확도 저하를 초래하여 적응을 위한 파라미터 업데이트(미세조정) 를 필요로 한다.

불행히도 이러한 파인튜닝에는 막대한 메모리가 필요하여 적용 가능성을 제한한다.

이 문제를 해결하기 위해, 양자화(quantization) 를 구조적 프루닝 프레임워크에 도입하여 파인튜닝과 추론 단계 모두에서의 메모리 사용량을 줄인다.

그러나 프루닝과 양자화에서 발생하는 오류가 결합되면 파인튜닝 난도가 높아지므로, 더 정교한 양자화 체계가 요구된다.

이를 위해 QPruner 라는 새로운 프레임워크를 제안한다.

Abstract

refined quantization scheme. To this end, we propose QPruner, a novel framework that employs structured pruning to reduce model size, followed by a layer-wise mixed-precision quantization scheme. Quantization precisions are assigned to each layer based on their importance to the target task, and Bayesian optimization is employed to refine precision allocation strategies, ensuring a balance between model accuracy and memory efficiency. Extensive experiments on benchmark datasets demonstrate that QPruner significantly outperforms existing methods in memory savings while maintaining or improving model performance.

이 프레임워크는 먼저 구조적 프루닝으로 모델 크기를 줄인 뒤, 레이어별 혼합정밀(mixed-precision) 양자화 를 적용한다.

각 레이어의 양자화 정밀도는 대상 과제에 대한 중요도 에 따라 할당되며, 베이지안 최적화 를 통해 정밀도 배분 전략을 더 세련되게 다듬어 모델 정확도와 메모리 효율 간 균형을 보장한다.

벤치마크 데이터셋에 대한 광범위한 실험을 통해, QPruner가 기존 방법 대비 메모리를 크게 절감하면서 성능을 유지하거나 향상함을 입증하였다.

Introduction

Structured pruning (Ma et al., 2023; Xia et al., 2023) is a widely used approach that reduces model size by removing less important parameters in a structured manner, preserving the overall architecture compatibility with hardware requirements. However, the disruption of computational graph uniformity and the removal of parameters can significantly reduce the accuracy of LLMs, which are inherently information-dense networks. To mitigate this degradation, fine-tuning is often used to recover the accuracy of pruned models. This fine-tuning step, while effective, is memory-intensive and presents substantial challenges in terms of resource consumption.

Structured Pruning 은 하드웨어 요구사항과의 전체 아키텍처 호환성을 유지하면서 덜 중요한 파라미터를 구조적으로 제거해 모델 크기를 줄이는 널리 쓰이는 접근법이다.

그러나 계산 그래프의 균질성이 깨지고 파라미터가 제거되면, 정보 밀도가 높은 LLM의 정확도가 크게 감소할 수 있다. 이를 완화하기 위해 보통 미세조정을 통해 프루닝된 모델의 정확도를 회복한다.

다만 이 단계는 효과적이긴 하나 메모리 집약적이며, 자원 측면에서 상당한 도전을 야기한다.

Introduction

To further reduce memory usage during the fine-tuning and inference phases, we introduce quantization into the structured pruning framework. Specifically, after performing structured pruning, we quantize the pruned model and then apply different fine-tuning strategies. Quantization effectively reduces the bit-width of model parameters, thereby lowering the resource consumption during both fine-tuning and inference. However, integrating quantization with structured pruning introduces additional complexities. Structured pruning applies different pruning intensities across model layers, which exacerbates the uneven distribution of layer importance, making some layers more critical for maintaining model performance. Moreover, the

이러한 관찰을 바탕으로, 우리는 QPruner라는 새 프레임워크를 제안한다.

QPruner에서는 먼저 Structured pruning으로 모델 크기를 줄이고, 이어서 레이어별 기여도에 따라 서로 다른 양자화 정밀도를 할당한다. 나아가 이 정밀도 배분 전략을 개선하기 위해 베이지안 최적화를 사용해 더 나은 정밀도 구성을 탐색한다.

마지막으로, 파라미터 효율적 미세조정(PEFT) 전략을 적용해 모델 성능을 회복한다. 이 통합 접근은 정확도와 메모리 효율 사이의 최적 균형을 도모하며, 자원 제약 시나리오에 적합하다.

Background

Quantization. Quantization is an essential technique used to reduce the computational and memory overhead of large-scale models by converting high-precision numerical values, such as a 32-bit floating-point number $X^{\mathrm{HP}} \in \mathbb{R}$, into a lower-bit integer representation $X^{\mathrm{INT}} \in \{0, 1, \ldots, 2^N - 1\}$. This process is mathematically expressed as:

$$X^{\mathrm{INT}} = \mathrm{round}\left((2^N - 1)F\left(X^{\mathrm{HP}}\right)\right),$$

양자화는 32-비트 부동소수점 고정밀 수치를 더 낮은 비트의 정수 표현 X으로 변환하여 대규모 모델의 계산·메모리 오버헤드를 줄이는 핵심 기법이다. 수식으로는 좌측과 같다.

Background

$$F(X) = \frac{X - X_{\min}}{X_{\max} - X_{\min}}.$$

여기서 F(·):R→[0,1]는 정규화 함수다. 전형적인 균일 양자화에서는 좌측 식을 사용하고

$$F(X) = \Phi(X/\sigma),$$

QLoRA(Dettmers et al., 2024)가 도입한 4-비트 NormalFloat 양자화(NF4) 는 데이터가 정규분포 X~N(0,σ^2)를 따른다고 가정하고, 좌측 식을 적용한다.

Background

$$\mathcal{T}[i] = F^{-1}\left(\frac{i}{2^N - 1}\right), \quad i = 0, 1, \ldots, 2^N - 1, \tag{2}$$

역양자화(Dequantization) 는 룩업테이블 T 를 사용해 양자화된 값을 고정밀로 복원한다

$$X^{\mathrm{D}} = \mathcal{T}[X^{\mathrm{INT}}].$$

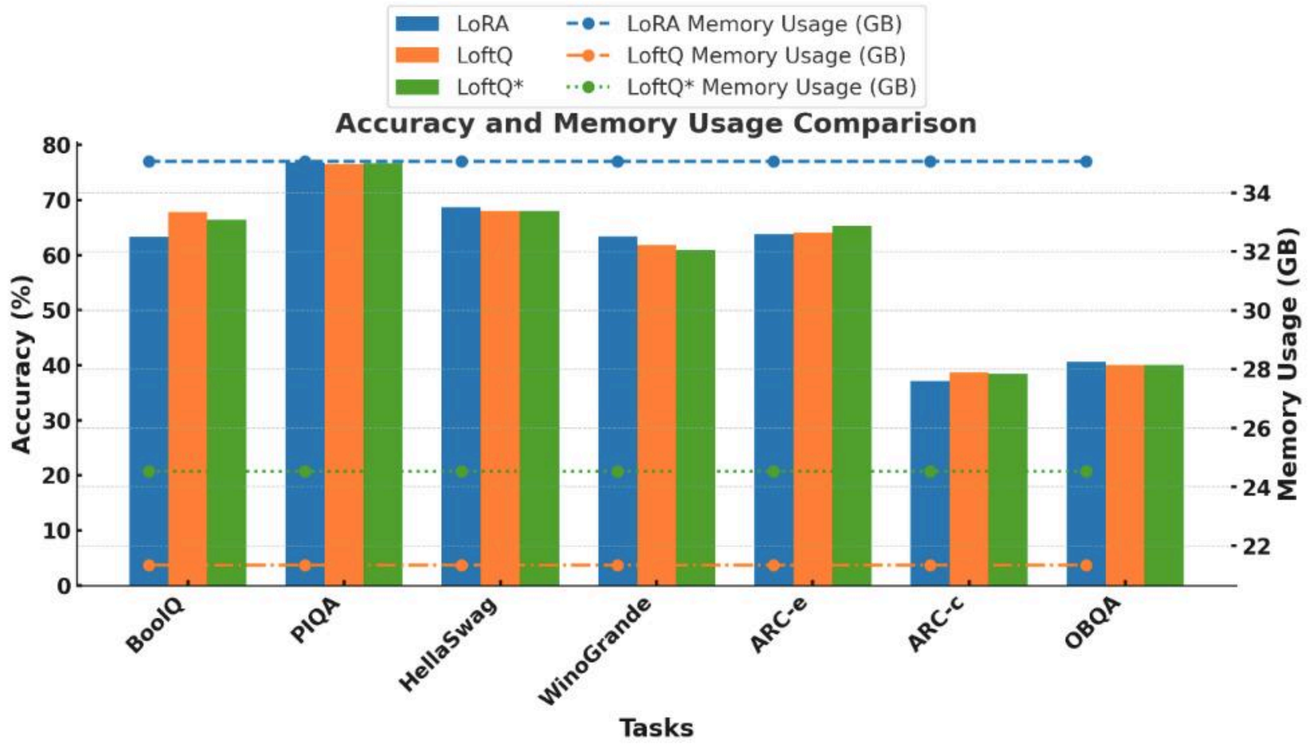
이를 통해 정수 X_{INT}를 고정밀 근사 X_{D}∈R로 사상한다

Background

Simulated Quantization for Matrices. In practice, it is often more efficient to use simulated quantization for matrices rather than directly operating on quantized values (Bai et al., 2020; Shen et al., 2020). In this method, quantized weight matrices are stored as encoded integers and are temporarily dequantized into simulated high-precision matrices during multiplication operations. This process is denoted by $q_N(\cdot): \mathbb{R}^{m\times n} \rightarrow \mathbb{R}_N^{m\times n}$, where $\mathbb{R}_N : \{\mathcal{T}[i] \in \mathbb{R} | 0 \leq i < 2^N\}$.

행렬에 대한 시뮬레이션 양자화: 실제로는 양자값을 직접 연산하기보다, 곱셈 시 일시적으로 복원하는 시뮬레이션 양자화가 효율적인 경우가 많다. 즉, 가중치 행렬은 정수로 저장하되 연산 시 잠깐 고정밀로 복원한다.

Background



Example: LLaMA-7B 에 프루닝율 20% 를 적용해 실험했다.

프루닝은 LLM-Pruner가 찾은 최적 전략을 사용했다. 비교 방법은 (1) 16-bit 균일의 LoRA, (2) 4-bit 균일의 LoftQ, (3) 레이어별 4/8-bit 혼합정밀 LoftQ* 이다.

그림과 같이, LoftQ(4-bit 균일) 는 LoRA(16-bit) 와 비슷한 성능을 보이면서 메모리는 크게 적게 썼다(21.33GB vs 35.06GB).

일부 과제에서 약간의 성능 하락이 있었지만, 혼합정밀 LoftQ* 는 메모리 효율을 유지하면서 성능을 추가로 향상할 가능성을 보였다.

QPruner

- 구조적 프루닝은 모델 크기를 줄이는 데 효과적이지만, 레이어 중요도의 균형을 깨 성능 저하를 유발할 수 있다. 따라서 이 불균형을 완화하고 성능을 복구하기 위해서는 파라미터 조정이 필요하다.

- 그러나 파라미터 업데이트는 큰 메모리를 요구하므로, 우리는 메모리 소비를 줄이기 위해 양자화를 함께 사용한다.
- 동기 예시에서 보았듯, 프루닝과 양자화를 단순 결합하는 것은 최선이 아니다. 프루닝된 모델에서는 레이어 중요도가 크게 달라지므로, 레이어별 비트폭을 더 미세하게 제어해야 하고, 이는 비트폭 할당 문제를 야기한다.
- 이를 해결하기 위해, 이러한 절충을 효과적으로 균형 잡는 2-단계 할당 전략을 설계했다.

- 이 통찰을 바탕으로, 자원 효율형/저자원 NLP 를 겨냥한 통합 프레임워크 QPruner를 제안한다.
- QPruner는 구조적 프루닝, 혼합정밀 양자화, 효율적 미세조정을 결합해 메모리 효율과 성능 사이의 균형 문제를 해결한다.

QPruner

3.1 Structured Pruning

Our framework does not impose specific requirements on the pruning method; as new technologies evolve, the pruning method can be replaced. The only requirement for this step is to produce a smaller model. Although some methods can achieve good performance without fine-tuning (An et al., 2024), most real-time systems require dynamic adaptation, which means that the pruned model must be fine-tuned to improve performance.

프레임워크는 프루닝 방법에 대해 특정 요구사항을 강제하지 않는다. 새로운 기술이 등장하면 프루닝 방법은 교체될 수 있다.

이 단계의 유일한 요구사항은 더 작은 모델을 산출하는 것이다.

일부 방법은 미세조정 없이도 좋은 성능을 달성할 수 있지만, 대부분의 실시간 시스템은 동적 적응을 요구하므로, 프루닝된 모델의 성능을 높이기 위해서는 미세조정이 필요하다.

QPruner

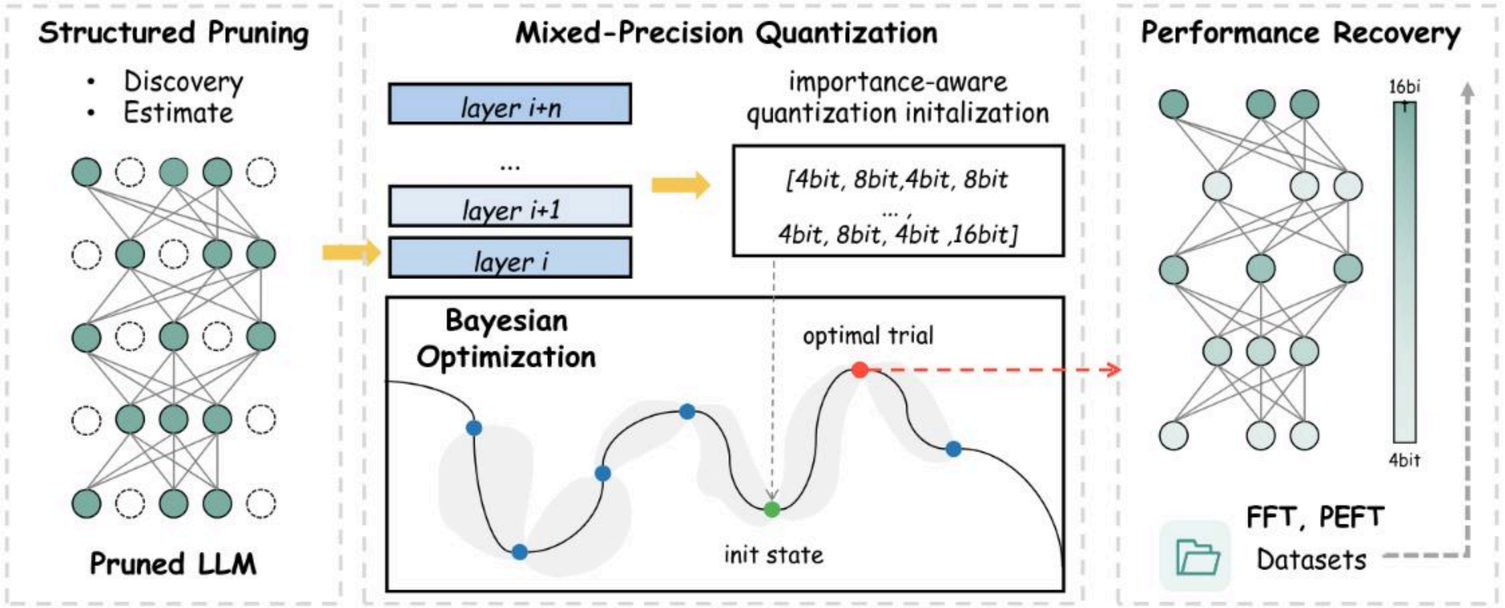


Figure 2: Overview of the QPruner framework.

(프루닝된 LLM → 구조적 프루닝(탐색·추정) ← 데이터셋 / 혼합정밀 양자화(중요도 인지 양자화 초기화, 예: ...[4bit, 8bit, 4bit, 8bit, ..., 4bit, 8bit, 4bit, 16bit]) ← 베이지안 최적화(초기 상태→최적 시도) / 성능 회복(FFT, PEFT; 4bit/16bit))

QPruner

$$I_{\mathbf{W}_i} = |\mathcal{L}_{\mathbf{W}_i}(\mathcal{D}) - \mathcal{L}_{\mathbf{W}_i=0}(\mathcal{D})|, \quad (4)$$

where $\mathcal{L}$ represents the prediction loss.

Using a second-order Taylor expansion, the importance can be approximated as:

$$\left| \frac{\partial \mathcal{L}(\mathcal{D})}{\partial \mathbf{W}_i} \mathbf{W}_i - \frac{1}{2} \mathbf{W}_i^\top \mathbf{H} \mathbf{W}_i \right|, \quad (5)$$

where $\mathbf{H}$ is the Hessian matrix of the loss function.

$$\left| \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_k^i} W_k^i - \frac{1}{2} (W_k^i)^2 H_{kk} \right|,$$

다음으로, 이렇게 결합된 구조들을 그룹화하고 그 중요도를 추정해 효과적으로 프루닝한다. 각 항목의 중요도는 좌측 식으로 표현된다.

여기서 LLL은 예측 손실이다. 2차 테일러 전개를 사용하면 중요도를 다음과 같이 근사할 수 있다.

여기서 HHH는 손실 함수의 헤시안이다. 개별 파라미터 W^{k}_{i}에 대해서는 좌측 최하단 식으로 표현된다.

QPruner

3.2 Mixed-Precision Quantization

After pruning, we apply mixed-precision quantization to further reduce memory usage while maintaining model performance. Instead of assigning a uniform bit-width across all layers, different bit-widths are allocated based on each layer's contribution to the final model output. The contribution of each layer is quantified using mutual information between the layer's output and the model's prediction.

To compute mutual information, we first run representative data samples through the pruned model. For each layer, we record its output X and the final prediction Y. The mutual information $I(X;Y)$ between the output of layer X and prediction Y is computed as:

프루닝 이후, 우리는 혼합정밀 양자화를 적용해 메모리 사용을 추가로 줄이면서 성능을 유지한다. 모든 레이어에 균일한 비트폭을 할당하는 대신, 각 레이어가 최종 출력에 기여하는 정도에 따라 서로 다른 비트폭을 배정한다.

레이어 기여도는 레이어 출력과 모델 예측 간 상호정보량(MI)으로 정량화한다.

상호정보량 계산을 위해, 대표 데이터 샘플을 프루닝된 모델에 통과시킨다. 각 레이어에 대해 출력 X와 최종 예측 Y를 기록한다. 레이어 출력 X와 예측 Y의 상호정보량 I(X;Y)는 아래 식으로 정리된다.

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}, \quad (7)$$

QPruner

3.3 Performance Recovery

After the steps of structured pruning and mixed-precision quantization, significant memory savings are achieved. However, model performance typically needs to be restored through fine-tuning. Full-parameter fine-tuning is often impractical due to the large memory footprint it requires, but our compression technique makes full model fine-tuning feasible by reducing both memory and computational costs.

구조적 프루닝과 혼합정밀 양자화를 거치면 상당한 메모리 절감을 달성할 수 있다. 그러나 모델 성능은 보통 미세조정으로 복원해야 한다. 전 파라미터 미세조정은 큰 메모리 기록 때문에 종종 비현실적이지만, 여기서의 압축 기법은 메모리와 계산 비용을 줄여 전량 미세조정도 가능하게 만든다.

QPruner

$$\mathbf{Y} = \mathbf{W}_0\mathbf{X} + \Delta\mathbf{W}\mathbf{X} = \mathbf{W}_0\mathbf{X} + \mathbf{A}\mathbf{B}\mathbf{X}, \quad (9)$$

전통적인 전량 미세조정 외에도, LoRA(Low-Rank Adaptation)(Hu et al., 2021) 같은 효율적 미세조정(PEFT) 이 특히 데이터가 제한된 상황에서 효과적임이 입증되었다.

LoRA는 원래의 가중치 행렬 W0를 고정하고,
저랭크 근사 ΔW = A B (A ∈ R^{d×r}, B ∈ R^{r×d}, r ≪ d)만 학습하여
학습 파라미터 수를 크게 줄인다.

QPruner

		Method	BoolQ	PIQA	HellS	WinoG	ARC-e	ARC-c	OBQA	Memory (GB)
LLaMA-7B	Rate = 0%	w/o tuning	73.09	78.35	72.98	67.09	67.42	41.38	42.40	-
	Rate = 20%	LLM-Pruner	63.30	76.82	68.68	**63.38**	63.76	37.11	40.60	35.06
		QPruner^1	67.77	76.55	68.03	61.80	64.06	38.65	40.00	21.78
		QPruner^2	68.60	76.79	68.43	62.78	65.50	38.74	40.40	23.05
		QPruner^3	**69.11**	**77.23**	**68.80**	63.17	**66.16**	**39.20**	**41.00**	23.32
	Rate = 30%	LLM-Pruner	62.45	74.37	**63.14**	**61.96**	**59.22**	33.70	**39.60**	31.38
		QPruner^1	58.96	71.22	58.10	58.88	52.19	32.34	38.40	20.12
		QPruner^2	62.20	72.88	60.64	60.50	55.61	33.56	38.40	22.87
		QPruner^3	**66.50**	**74.43**	61.14	61.40	58.12	**34.47**	39.20	22.15
	Rate = 50%	LLM-Pruner	43.76	68.88	44.85	50.99	45.20	28.75	34.60	23.89
		QPruner^1	45.14	68.34	44.39	52.96	43.86	29.01	35.80	15.47
		QPruner^2	47.08	68.85	45.53	53.65	44.31	29.36	36.20	16.85
		QPruner^3	**48.37**	**69.20**	**45.19**	**54.45**	**45.28**	**29.70**	**36.40**	16.65
Vicuna-7B	Rate = 0%	w/o tuning	75.69	77.75	71.06	67.80	69.07	40.78	42.20	-
	Rate = 20%	LLM-Pruner	57.77	77.56	67.16	63.14	67.30	37.71	40.40	35.25
		QPruner^1	57.95	76.82	66.42	62.51	66.62	37.37	40.60	21.65
		QPruner^2	59.70	77.20	66.31	62.66	67.12	37.48	40.80	22.95
		QPruner^3	**59.85**	**77.59**	**67.31**	**63.20**	**67.84**	**37.85**	**41.20**	23.10
	Rate = 30%	LLM-Pruner	**58.81**	74.37	60.70	**60.62**	59.01	33.79	38.80	31.83
		QPruner^1	53.85	74.76	60.65	60.06	59.72	34.30	38.20	19.95
		QPruner^2	55.64	75.07	61.65	60.31	59.54	34.47	38.60	21.65
		QPruner^3	57.23	**75.90**	**62.00**	60.37	**60.81**	**34.79**	**39.40**	21.80
	Rate = 50%	LLM-Pruner	59.51	66.87	43.18	52.01	48.40	26.45	34.00	24.55
		QPruner^1	59.51	67.90	43.30	50.83	48.82	27.49	34.60	14.50
		QPruner^2	61.31	68.56	44.54	53.02	49.50	28.13	35.40	15.90
		QPruner^3	**61.56**	**68.80**	**43.72**	**53.39**	**49.66**	**27.98**	**35.80**	15.35