AutoMP-RL: Automated Mixed-Precision Quantization using Reinforcement Learning

120250347 함효리
120240558 전우리

INDEX

01 Background & Objectives

02 Methodology & Settings

03 Design of State, Action, Reward

04 RL Algorithm & Hyperparameter

05 Experiment setup

06 Results & Discussion

Project Subject

Activation Statistics–Driven Layer-wise Mixed Precision Quantization & Error Correction Adapter for LLM Optimization

- 대규모 LLM 경량화를 위한 새로운 Mixed Precision 전략
- Activation 통계 기반 레이어 민감도 분석
- Adapter 기반 양자화 오차 보정

Project Background

LLM 경량화 필요성

- 수십억 파라미터에 따른 높은 메모리·연산 비용
- 추론 지연 증가, GPU 비용 상승

기존 양자화의 한계

- W4A16은 메모리 절감은 크지만 정확도 급락 문제
- 모든 레이어 동일한 bit 적용 → layer sensitivity 반영 불가
- 단순 clipping/rounding → 양자화 오차 복원 불가

Adapter 사용 시 문제

- 정확도 복구 가능하지만, 추가 파라미터 증가, 학습/메모리 비용 증가

Project Objectives

1. Activation 기반 Layer Sensitivity 분석

- 각 Linear layer activation 통계 수집 (mean, std)
- 민감도가 큰 레이어 식별

2. Error Correction Adapter 설계

- 양자화 오차 E = W_{fp16} - W_q 근사 및 보정
- 경량 구조로 정확도 복원

3. RL 기반 Mixed Precision 자동 탐색

- 레이어별 최적 비트 선택 정책 학습
- Accuracy–Memory trade-off 최적화

Methodology

1.Activation Statistics 기반 Layer Sensitivity 분석

- state.py + forward hook 활용
  - 모든 Linear Layer의 activation abs mean 수집

- Activation 크기: 레이어의 양자화 민감도(sensitivity) 반영
  - 큰 activation → 양자화에 민감 → 고비트 필요
  - 작은 activation → 정보 손실 적음 → 저비트 할당 가능

- Mixed Precision 비트 배정의 기준(state)으로 활용됨

Methodology

2. 양자화 오차 모델링 & Error Correction Adapter

- 양자화 오차 정의 $E = W_{FP16} - W_{Quantized}$
- 양자화 후 weight 손실로 인한 출력 오차: E · X
- Adapter 설계
  - 입력 X를 기반으로 E·X를 근사하는 경량 Adapter 모듈 추가
- 구조 특징
  - 학습 가능한 low-parameter module
  - 추론 시 양자화된 weight 출력에 보정값을 더하는 방식
- 역할
  - W4A16(lower precision)으로 줄어든 weight로 인해 발생한 성능 저하를 보정
  - FP16에 가까운 결과 복원

Methodology

3. Mixed Precision + Adapter 통합

- static quantization의 한계
  - 모든 레이어 동일 bit 적용 → 레이어 민감도 반영 불가
  - clipping/rounding 기반 단순 오차 처리 → 정확도 손실 누적

- 제안 방식 장점
  - Activation 기반 민감도 반영 → 레이어별 최적 bit 선택
  - Adapter 보정 → 양자화 손실 최소화

- 추가 최적화 전략
  - Adapter로 인한 추가 파라미터/연산 비용 증가 문제
  - Low-rank Adapter / Layer-wise parameter sharing / Sparse structure 적용
  - 전체 모델 사이즈 증가 없이 성능 복원 효과 유지 가능

Dataset Configuration

Calibration + Evaluation Datasets

- C4 dataset (primary calibration & evaluation)
- RedPajama (alternative calibration)

Sample Sizes

- Calibration samples: 512
- Evaluation samples: 100
- Max sequence length: 512

State Design

State = Layer-wise Activation Statistics

- Forward Hook 기반 activation 추출
- Activation absolute mean을 scalar로 변환
- Policy 네트워크 입력으로 사용
- Layer sensitivity를 가장 compact하게 표현
  → 고활성 레이어 = 양자화 민감도 높음
  → 저활성 레이어 = bit 절감 여지 큼

Action Design

Action = Layer Mixed Precision bit choice

- option: 3bit / 4bit / 8bit
- Policy가 각 레이어별로 bit allocation
- Sampling mode: exploration
- Deterministic mode: 최종 config 생성

Reward Design

Memory–Accuracy Trade-off 최적화

$$Reward = -0.1(PPL - baseline) + 2.0 \cdot memory\_saving$$

Reward 요소

- PPL penalty → 정확도 저하 시 큰 페널티 부여
- Memory saving → bit 합 기준 절감률로 계산
- RL이 탐색해야 하는 목표 → 정확도 최소 손실 + bit 최소 사용

RL Algorithm

REINFORCE Policy Gradient

- Action space가 discrete (bit 선택)
- Layer-wise 선택 구조와 잘 맞음
- End-to-end sampling 기반 학습 가능
- DQN

- 특징
  - Reward standardization → variance 감소
  - Entropy bonus → 탐색 안정성 증가
  - Episode 단위 bit config 학습

Policy Network 구조

- 입력(Input)
  - Layer activation mean (scalar)

- 구조: 3-layer MLP
  - Linear → ReLU
  - Linear → ReLU
  - Linear → logits(3)

- 출력(Output)
  - bit 옵션 3개에 대한 확률/로짓값
  - Sampling or Argmax로 action 결정

Hyperparameter

Hyperparameter	Value
Learning rate	1e-3
Episodes	50-300
Gamma	0.99
Available bits	[3, 4, 8]
α (PPL penalty)	0.1
β (Memory reward)	2.0

Experiment Setting

Component	Setting
GPU	NVIDIA A6000
PyTorch	≥ 2.1 (tested on 2.9.1)
Transformers	≥ 4.40
OS	Ubuntu 20.04/22.04, WSL2
Model	Meta-Llama-3-8B-Instruct

Experiment setup

Evaluation Metric

1. Perplexity (PPL)

- 측정 대상: 평가 데이터셋(C4/WikiText)
- 양자화 모델 정확도 지표
- Baseline 대비 변화량도 함께 기록

$$\Delta PPL = \frac{PPL_{quant} - PPL_{baseline}}{PPL_{baseline}}$$

2. Memory Reduction (%)

- Layer-wise bit-sum 기반 상대 메모리 절감률

$$MemorySaving = 1 - \frac{\sum bits_{quant}}{\sum 16\,bit}$$

3. Reward

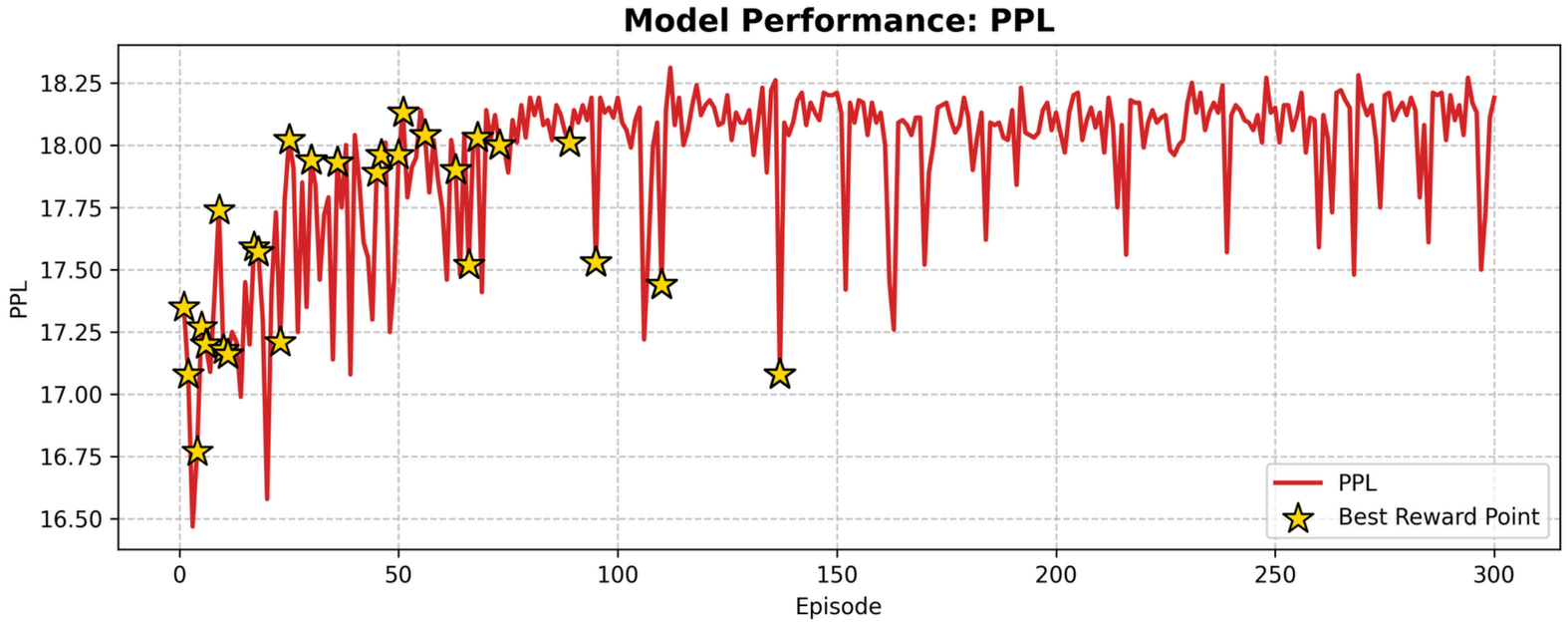
- RL 학습 성능 지표
- PPL penalty + Memory reward 균형을 반영

$$Reward = -\alpha(PPL - PPL_{baseline}) + \beta(MemorySaving)$$

4. Stability / Robustness

- 다중 시드(n=10) 실험에서 PPL variance, reward variance 분석
- Boxplot으로 시각화

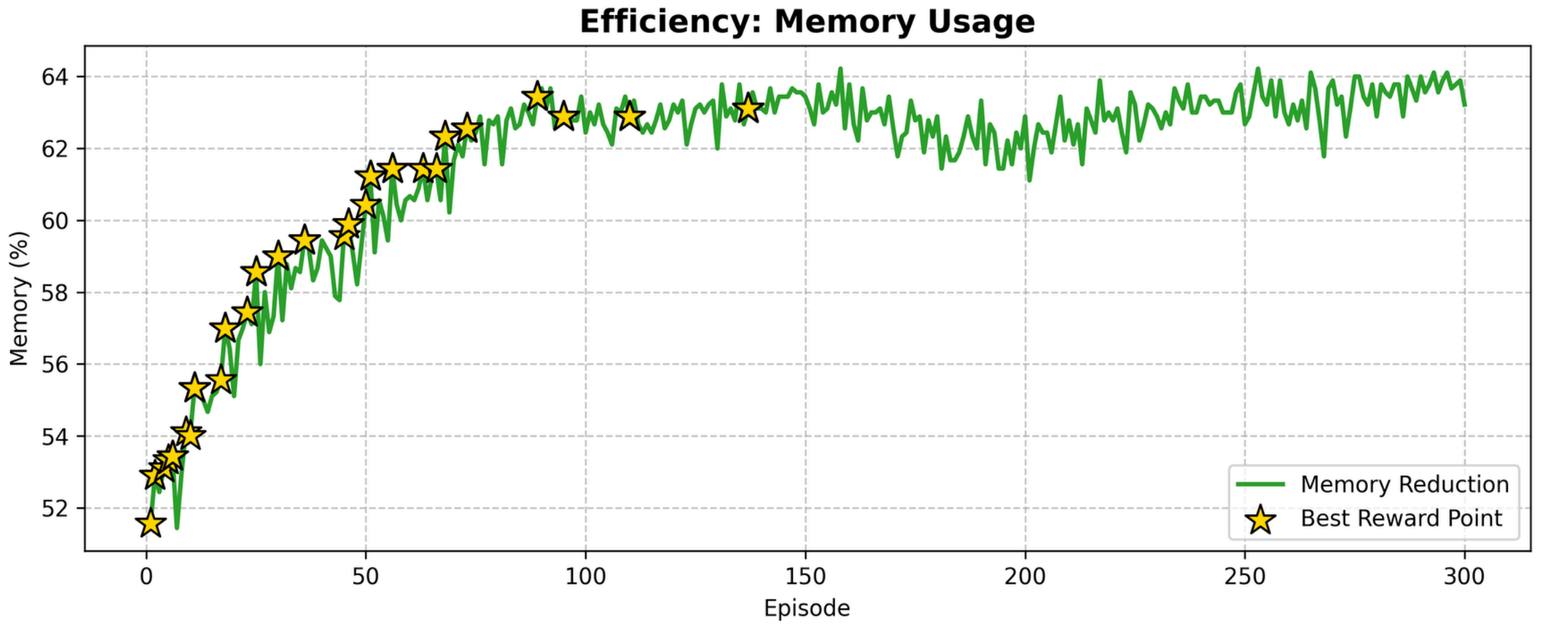
Results



1. Perplexity Comparison

- 제안된 방법이 lowest PPL
- Layer-wise mixed precision이 W4A16 대비 정확도 유지력 우수
- 일부 설정에서는 FP16보다도 더 낮은 PPL의 결과도 관찰됨

Results



2. Memory Usage Comparison

- Proposed Method: ~62% VRAM 절감
- AWQ(W4A16) 대비 동일 수준 또는 더 높은 절감 효과
- Mixed bit 사용에도 불구하고 Memory Saving은 거의 W4A16 수준 유지

Results

3. Training Dynamics

- (a) PPL vs Episode
  - RL 초기엔 8bit 비중 ↑ → PPL 안정적
  - 이후 4bit 탐색 확대 → PPL 소폭 증가 → 최적점 근처에서 수렴

- (b) Memory Reduction vs Episode
  - 8bit 중심 → 메모리 절감 낮음
  - 점차 4bit 비중 증가 → 메모리 절감 증가
  - RL 에이전트가 자동으로 trade-off 학습함

Results

- (c) Reward Progression
  - Memory saving으로 reward 상승
  - PPL degradation 증가 시 reward 하락
  - 균형점을 찾으며 reward 안정적 수렴

- (d) Bit Distribution per Layer
  - Attention/MLP 모듈별 민감도 차이가 나타남
  - Early layers는 상대적으로 robust → 4bit 많이 선택
  - Deeper layers는 sensitivity 높아서 8bit 유지 비율 높음

Results & Discussion

Key Observations

1. RL이 실제로 Mixed Precision Trade-off를 학습

- 초기: 정확도 유지 위해 8bit 선택
- 중반: memory reward를 위해 4bit 탐색
- 후반: PPL degradation과 memory saving의 균형점 학습

2. Layer Sensitivity가 중요하게 반영

- Activation mean이 큰 레이어 = 8bit 유지
- 작은 레이어 = 4bit 전환하여 메모리 절감
- 비정적(static) 양자화보다 더 세밀한 적용 가능

3. AWQ 대비 장점

- AWQ는 uniform W4A16 → flexibility 부족
- 본 연구:
  - layer별 mixed precision
  - RL이 자동 최적화
  - 유사한 PPL + 메모리 절감

4. Robustness

- Seeds 10개 실험 결과
  - PPL variance 매우 낮음
  - 정책의 안정성 확인

Discussion

1. Remaining Limitations

- weight의 최소오차를 줄일 수 있는 Quantization
- Quantized model inference는 빠르지만 training expense가 증가
- Activation statistics 외 더 많은 정보 활용 가능

2. Future Work

- Adapter 기반 오차 보정 완전 적용
- LoRA-style low-rank correction 적용
- PPO/Soft-Actor-Critic 등 고급 RL 기법 도입
- Attention/MLP 등 layer type-aware 정책 연구
- variance bit allocation

감사합니다.