Quantized Reinforcement Learning (QuaRL)

Srivatsan Krishnan[1]

Sharad Chitlangia[2]

Maximilian Lam[1]

Zishen Wan[1]

Alexsandra Faust[3]

Vijay Janapa Reddi[1]

1. Harvard University, USA
2. BITS-Pilani Goa, India
3. Robotics at Google, USA

Outline

- Motivation
- QuaRL Framework
- Experimental Results
- Cases Studies
- Summary

Motivation

- **Deep reinforcement learning**
  ✔️ Deep reinforcement learning has promise in many applications.
  ❌ Computational expensive demands → Training RL models is challenging
  ❌ Resource-constraints of embedded system → Deploying RL models is challenging
  ❌ The total infrastructure cost in the tens of millions of $$$.
- **Quantization**
  ❓ Supervised learning → Reinforcement learning
  ❓ How quantization affects the long-term decision making capability of RL policies
  ❓ Whether quantization would be similarly effective across various RL algorithms
  ❓ How quantization affects performance in tasks of different complexity

How quantization affects deep reinforcement learning?

QuaRL (**Qua**ntized **R**einforcement **L**earning)

- **First comprehensive empirical study** and a **software framework** to benchmark and quantify the effects of quantization on:

1) Various RL algorithms

- PPO
- A2C
- DDPG
- DQN

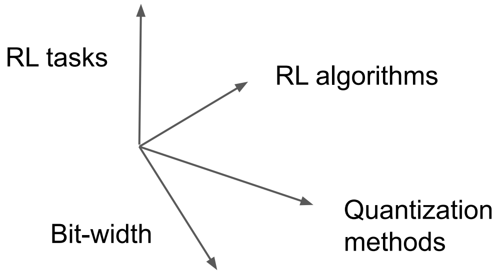
2) Various RL tasks

- OpenAI Gym
- Atari
- Pybullet

3) Various quantization methods

- Post Training Quantization (PTQ)
- Quantization Aware Training (QAT)

4) Various quantized bit-widths



Outline

- Motivation
- QuaRL Framework
- Experimental Results
- Cases Studies
- Summary

Summary of algorithms, environments, quantization scheme in the QuaRL framework

Algorithm	OpenAI Gym		Atari							PyBullet		
	Cartpole	**MountainCar**	**BeamRider**	**Breakout**	**MsPacman**	**Pong**	**Qbert**	**Seaquest**	**SpaceInvaders**	**BipedalWalker**	**HalfCheetah**	**Walker2D**
DQN	PTQ	n/a	PTQ	PTQ	PTQ	PTQ	PTQ	PTQ	PTQ	n/a	n/a	n/a
A2C	PTQ QAT BW		PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW			
PPO	PTQ QAT BW		PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW			
DDPG	n/a	PTQ	n/a	n/a	n/a	n/a	n/a	n/a	n/a	PTQ QAT BW	PTQ QAT BW	PTQ QAT BW

PTQ: Post-Training Quantization

QAT: Quantization-Aware Training

BW: evaluating policy from 8-bits to 2-bits

n/a: cannot evaluate due to algorithm-environment incompatibility

Post-Training Quantization

Algorithm 1: Post-Training Quantization for Reinforcement Learning

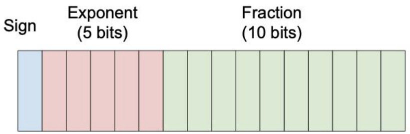
Input: T : task or environment
Input: L : reinforcement learning algorithm
Input: A : model architecture
Input: n : quantize bits (8 or 16)
Output: Reward

M = Train(T, L, A)

$$Q = \begin{cases} Q_{int8} & n = 8 \\ Q_{fp16} & n = 16 \end{cases}$$

return Eval($Q(M)$)

Fp16 Quantization



$$V_{fp16} = (-1)^S \times (1 + \frac{F}{2^{10}}) \times 2^{E-15}$$

$$Q_{fp16}(W) = round_{fp16}(W)$$

Uniform Affine Quantization

$$Q_n(W) = \left\lfloor \frac{W}{\delta} \right\rceil + z$$

$$\text{where } \delta = \frac{|min(W,0)| + |max(W,0)|}{2^n},\ z = \left\lfloor \frac{-min(W,0)}{\delta} \right\rceil$$

$$D(W_q, \delta, z) = \delta(W_q - z)$$

Quantization-Aware Training

Algorithm 2: Quantization Aware Training for Reinforcement Learning

Output: Reward
Input: T : task or environment
Input: L : reinforcement learning algorithm
Input: n : quantize bits
Input: A : model architecture
Input: Qd : quantization delay

A_q = InsertAfterWeightsAndActivations(Q_n^{train})
$M, TensorMinMaxes$ = TrainNoQuantMonitorWeightsActivationsRanges(T, L, A_q, Qd)
M = TrainWithQuantization(T, L, M, $TensorMinMaxes$, Q_n^{train})
return Eval(M, Q_n^{train}, $TensorMinMaxes$)

- Fake quantization
- Additional parameter: quantization delay
- Monitor minimum and maximum values during training

$$Q_n^{train}(W, V_{min}, V_{max}) = \left\lfloor \frac{W}{\delta} \right\rceil + z$$

$$\text{where } \delta = \frac{|V_{min}| + |V_{max}|}{2^n},$$

$$z = \left\lfloor \frac{-V_{min}}{\delta} \right\rceil$$

Outline

- Motivation
- QuaRL Framework
- Experimental Results
- Cases Studies
- Summary

Effectiveness of Quantization (PTQ)

- A2C rewards for Fp32, Fp16 and int8:

Environment	fp32	fp16	E_fp16	int8	E_int8
Breakout	379	371	2.11%	350	7.65%
SpaceInvaders	717	667	6.97%	634	11.58%
BeamRider	3087	3060	0.87%	2793	9.52%
MsPacman	1915	1915	0.00%	2045	-6.79%
Qbert	5002	5002	0.00%	5611	-12.18%
Seaquest	782	756	3.32%	753	3.71%
CartPole	500	500	0.00%	500	0.00%
Pong	20	20	0.00%	19	5.00%
Mean			**1.66 %**		**2.31 %**

- PPO rewards for Fp32, Fp16 and int8:

Environment	fp32	fp16	E_fp16	int8	E_int8
Breakout	400	400	0.00%	368	8.00%
SpaceInvaders	698	662	5.16%	684	2.01%
BeamRider	1655	1820	-9.97%	1697	-2.54%
MsPacman	1735	1735	0.00%	1845	-6.34%
Qbert	15010	15010	0.00%	14425	3.90%
Seaquest	1782	1784	-0.11%	1795	-0.73%
CartPole	500	500	0.00%	500	0.00%
Pong	20	20	0.00%	20	0.00%
Mean			**-0.62%**		**0.54%**

- DQN rewards for Fp32, Fp16 and int8:

Environment	fp32	fp16	E_fp16	int8	E_int8
Breakout	214	217	-1.40%	78	63.55%
SpaceInvaders	586	625	-6.66%	509	13.14%
BeamRider	925	823	11.03%	721	22.05%
MsPacman	1433	1429	0.28%	2024	-41.24%
Qbert	641	641	0.00%	616	3.90%
Seaquest	1709	1885	-10.30%	1582	7.43%
CartPole	500	500	0.00%	500	0.00%
Pong	21	21	0.00%	21	0.00%
Mean			**-0.88%**		**8.60%**

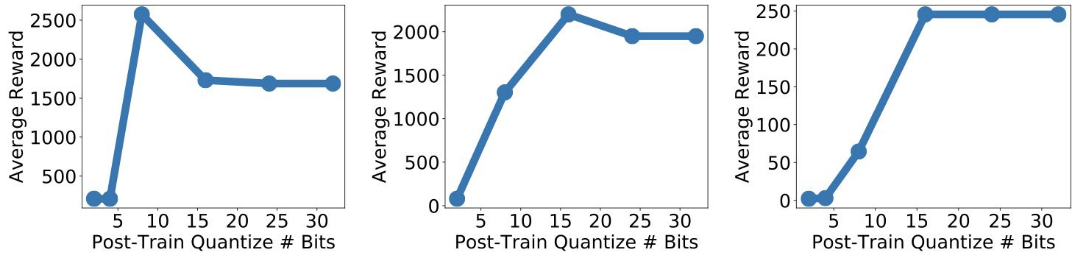
- DDPG rewards for Fp32, Fp16 and int8:

Environment	fp32	fp16	E_fp16	int8	E_int8
Walker2D	1890	1929	-2.06%	1866	1.27%
HalfCheetah	2553	2551	0.08%	2473	3.13%
BipedalWalker	98	90	8.16%	83	15.31%
MountainCarContinuous	92	92	0.00%	92	0.00%
Mean			**1.54%**		**4.93%**

- Models can be quantized to 8/16 bit precision without much loss in quality
- Quantization yields better scores than full precision policy in few cases.

Post-Training Quantization (PTQ) Sweet-Spot

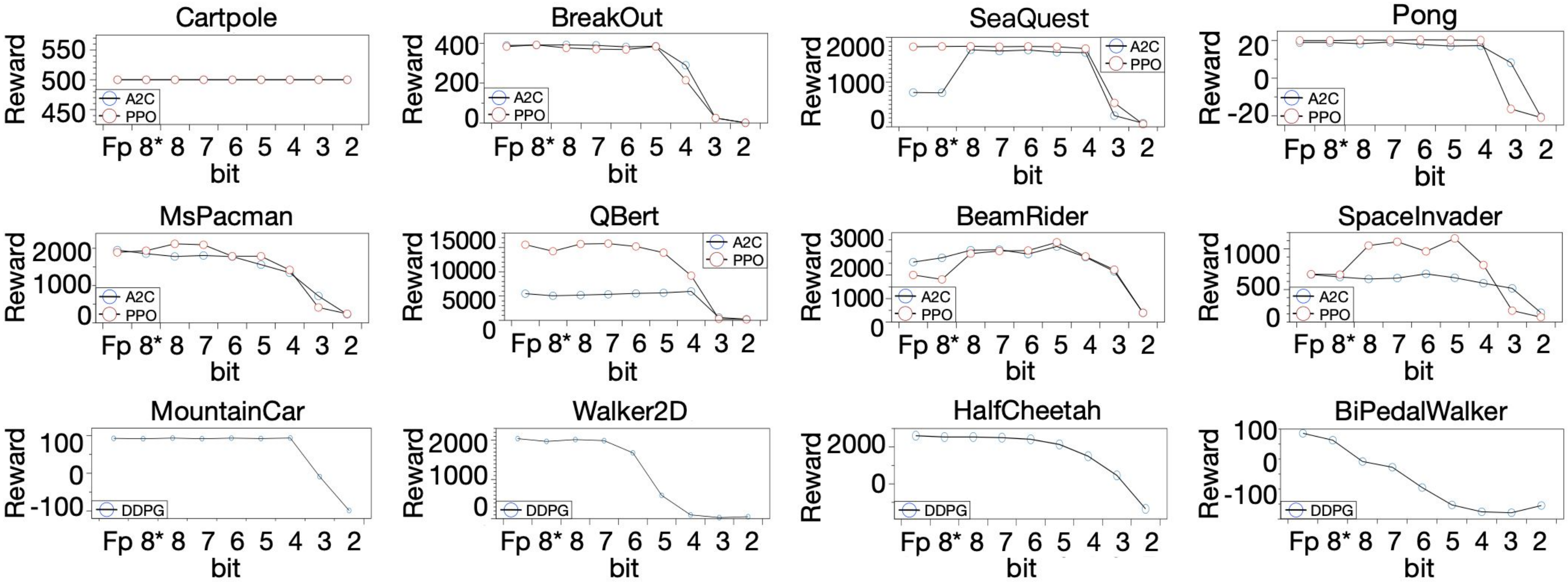
PTQ Sweet-Spot for DQN MsPacman, DQN SeaQuest, DQN Breakout:



- Sometimes quantizing to fewer bits outperforms full precision.
- PTQ sweet-spot depends on the specific task.

Effectiveness of Quantization (QAT)

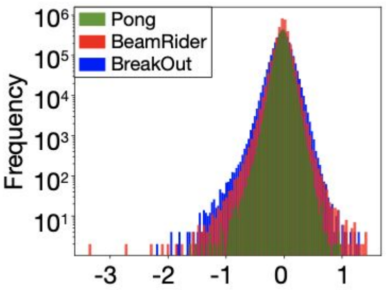
Fp: Fp32   8*: 8-bit PTQ



- Policies can be quantized to 5/6 bits of precision without loss of accuracy
- QAT achieves higher rewards tha PTQ, and sometimes outperforms the full precision baseline

Effect of Environment on Quantization Quality

Environment	E_{Int8}
Breakout	63.55%
BeamRider	22.05%
Pong	0%



Same: Algorithm: DQN; Quantization Scheme: PTQ

Sweep: different environment

Breakout: highest error ⟺ widest weight distribution

BeamRider: second-highest error ⟺ narrower weight distribution

Pong: lowest error ⟺ narrowest weight distribution

Conclusion:

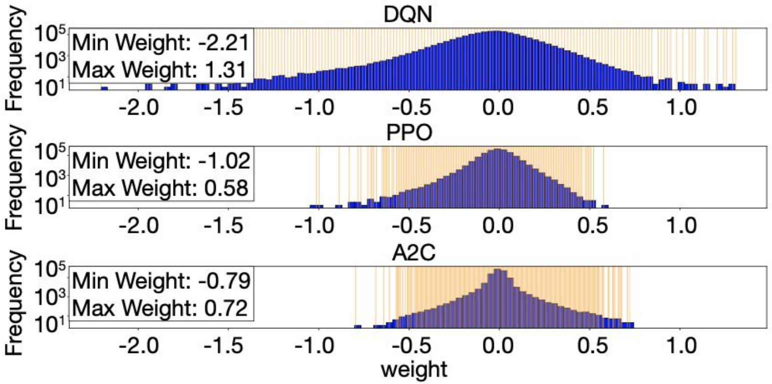
Environment affects models' weight distribution spread ⟹ affects quantization performance.

⇓

Regularizing the training process may yield better quantization performance

Effect of Training Algorithm on Quantization Quality

Algorithm	Environment	fp32 Reward	E_{int8}	E_{fp16}
DQN	Breakout	214	63.55%	-1.40%
PPO	Breakout	400	8.00%	0.00%
A2C	Breakout	379	7.65%	2.11%



Same Environment: Breakout

Same Quantization Scheme: PTQ

Different Algorithm: DQN, PPO, A2C

DQN: highest error ⟺ widest weight distribution

PPO: second-highest error ⟺ narrower weight distribution

A2C: lowest error ⟺ narrowest weight distribution

Outline

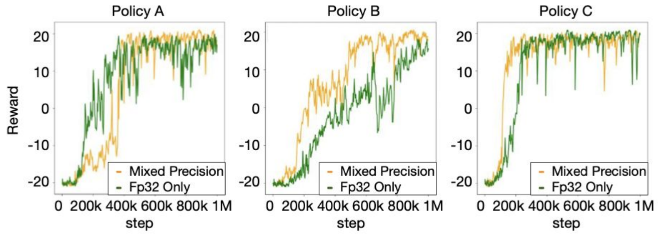
- Motivation
- QuaRL Framework
- Experimental Results
- Cases Studies
- Summary

Case 1: Mixed/Half-Precision Training

- Train a pong model using three policies (increase model complexity)

Algorithm	Policy Architecture
Policy A	3 Layer Conv (128 Filters) + FC (128)
Policy B	3 Layer Conv (512 Filters) + FC (512)
Policy C	3 Layer Conv (1024 Filters) + FC (2048)

- In mixed precision training, the weights, activations and gradients are represented in Fp16. A master copy of the weights are stored in Fp32 and are updated during backward pass.



Mixed precision v/s Fp32 training rewards

Algorithm	Network Parameter	fp32 Runtime (min)	MP Runtime (min)	Speedup
DQN-Pong	Policy A	127	156	0.87×
	Policy B	179	172	1.04×
	Policy C	391	242	1.61×

1.6x speed up

Case 2: Quantized Policy for Deployment

Methodology:

- Train three point-to-point navigation models for aerial robots using AirLearning platform.
- Deploy policies onto RasPi-3b, a proxy for the compute platform on the aerial robot

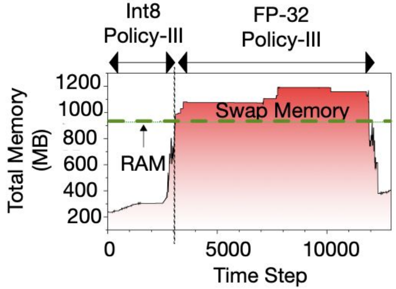
Embedded System	Ras-Pi 3b
CPU Cores	4 Cores (ARM A53)
CPU Frequency	1.2 GHz
GPU	None
Power	<1W
Cost	$35

Results:

- Inference speed on RasPi-3b and success rate

Policy Name	Network Parameters	fp32 Time (ms)	fp32 Success Rate (%)	int8 Time (ms)	int8 Success Rate (%)	Speed up
Policy I	3L, MLP, 64 Nodes	0.147	60%	0.124	45%	1.18 ×
Policy II	3L, MLP, 256 Nodes	133.49	74%	9.53	60%	14 ×
Policy III	3L, MLP (4096, 512, 1024)	208.115	86%	11.036	75%	18.85 ×

- Memory Consumption for Policy Ⅲ



18x speed up
4x memory reduction

Outline

- Motivation
- QuaRL Framework
- Experimental Results
- Cases Studies
- Summary

Summary

- Perform the first study of quantization effects on deep reinforcement learning using QuaRL, a software framework to benchmark and analyze the effects of quantization on RL.
- Applied different quantization techniques to a spectrum of RL tasks and algorithms.
- Demonstrated policies can be quantized to 6-8 bits without loss of accuracy.
- Demonstrated certain RL tasks and algorithms are difficult to quantize, and analyzed from weight distribution aspects. Demonstrated QAT consistently overperforms PTQ.
- Demonstrated real-world applications of quantization for RL. 1.5x speed up in training Pong model, 18x speedup and 4x memory reduction in real navigation task.