QUARL: QUANTIZED REINFORCEMENT LEARNING

Srivatsan Krishnan [*1] **Sharad Chitlangia** [*2] **Maximilian Lam** [*1] **Zishen Wan** [1] **Aleksandra Faust** [3]
Vijay Japana Reddi [1]

ABSTRACT

Quantization techniques continue to play a significant role in serving deep learning applications on resource-constrained devices. However, whether these prior techniques, applied traditionally to image-based models, work with the same efficacy to the sequential decision-making process in reinforcement learning remains an unanswered question. To address this void, we conduct the first comprehensive empirical study that quantifies the effects of quantization on various deep reinforcement learning policies with the intent to reduce their computational resource demands. We apply techniques such as post-training quantization and quantization aware training to a spectrum of reinforcement learning tasks (such as Pong, Breakout, BeamRider, and more) and training algorithms (such as PPO, A2C, DDPG, and DQN). Across this spectrum of tasks and learning algorithms, we show that policies can be quantized to 6-8 bits of precision without loss of accuracy. We also show that certain tasks and reinforcement learning algorithms yield policies that are more difficult to quantize due to their effect of widening the models' distribution of weights and that quantization aware training consistently improves results over post-training quantization and oftentimes even over the full precision baseline. Finally, we demonstrate real-world applications of quantization for reinforcement learning. We use mixed/half-precision training to train a Pong model 50% faster, and deploy a quantized reinforcement learning-based robot navigation policy onto an embedded system, achieving an 18× speedup and a 4× reduction in memory usage over an unquantized policy.

1 INTRODUCTION

Deep reinforcement learning has promise in many applications, ranging from game playing (Silver et al., 2016; 2017; Kempka et al., 2016) to robotics (Lillicrap et al., 2015; Zhang et al., 2015) to locomotion and transportation (Arulkumaran et al., 2017; Kendall et al., 2018). However, the training and deployment of reinforcement learning models remain challenging. Training is expensive because of their computationally expensive demands of repeatedly performing the forward and backward propagation in neural network training. Achieving state of the art results in the game DOTA2 required around 128,000 CPUs cores and 256 P100 GPUs—the total infrastructure cost in the tens of millions of US dollars (OpenAI, 2018). Deploying deep reinforcement learning (DRL) models is prohibitively expensive, if not even impossible, due to the resource constraints on embedded computing systems typically used for applications, such as robotics and drone navigation. For example, a Ras-Pi3b board has only ∼900 MB of RAM and exceeding this limit may cause accesses to swap memory that may slow down an application by an order of magnitude.

Quantization may substantially reduce the memory, compute, and energy usage of deep learning models without significantly harming their quality (Han et al., 2015; Zhou et al., 2016; Han et al., 2016). However, it is unknown whether the same techniques carry over to reinforcement learning. Unlike models in supervised learning, the quality of a reinforcement learning policy depends on how effective it is in sequential decision making. Specifically, an agent's current input and decision heavily affect its future state and future actions; it is unclear how quantization affects the long-term decision making capability of reinforcement learning policies. Also, there are many different algorithms to train a reinforcement learning policy. Algorithms like actor-critic methods (A2C), deep-q networks (DQN), proximal policy optimization (PPO) and deep deterministic policy gradients (DDPG) are significantly different in their optimization goals and implementation details, and it is unclear whether quantization would be similarly effective across these algorithms. Finally, reinforcement learning policies are trained and applied to a wide range of environments, and it is unclear how quantization affects performance in tasks of differing complexity.

Here, we aim to understand quantization effects on deep reinforcement learning policies with the goal of reducing

[*]Equal contribution [1]Harvard University [2]BITS-Pilani Goa [3]Robotics at Google. Correspondence to: Srivatsan Krishnan <srivatsan@seas.harvard.edu>.

Proceedings of the 3rd SysML Conference, Austin, TX, USA, 2020.

memory and compute to enable faster and cheaper training/deployment. Hence, we comprehensively benchmark the effects of quantization on policies trained by various reinforcement learning algorithms on different tasks, conducting in excess of 350 experiments to present a representative and conclusive analysis. We perform experiments over 3 major axes: **(1)** environments (Arcade Learning Environment, PyBullet, OpenAI Gym), **(2)** reinforcement learning training algorithms (Deep-Q Networks, Advantage Actor-Critic, Deep Deterministic Policy Gradients, Proximal Policy Optimization) and **(3)** quantization methods (post-training quantization, quantization aware training).

We show that deep reinforcement learning models can be quantized to 6-8 bits of precision without loss in quality. Furthermore, we analyze how each axis affects the final performance of the quantized model to develop insights into how to achieve better model quantization. Our results show that some tasks and training algorithms yield models that are more difficult to apply post-training quantization as they widen the spread of the models' weight distribution, yielding higher quantization error. This motivates the use of quantization aware training, which we show demonstrates improved performance over post-training quantization and oftentimes even over the full precision baseline.

To demonstrate the usefulness of quantization for deep reinforcement learning, we train and deploy a quantized reinforcement learning based navigation policy onto an embedded system (proxy for compute system on aerial robots) and achieve an 18× speedup and a 4× reduction in memory usage over an unquantized policy. We also demonstrate that quantization during training can result in upto 1.6× speedup. This can be particularly useful during reinforcement learning training on memory constrained GPUs.[1]

2 Related Work

Reducing neural network resource requirements is an active research topic. Techniques include quantization (Han et al., 2015; 2016; Zhu et al., 2016; Jacob et al., 2018; Lin et al., 2019; Polino et al., 2018; Sakr & Shanbhag, 2018), deep compression (Han et al., 2016), knowledge distillation (Hinton et al., 2015; Chen et al., 2017), sparsification (Han et al., 2016; Alford et al., 2018; Park et al., 2016; Louizos et al., 2018b; Bellec et al., 2017) and pruning (Alford et al., 2018; Molchanov et al., 2016; Li et al., 2016). These methods are employed because they compress to reduce storage and memory requirements as well as enable fast and efficient inference and training with specialized operations. We provide background for these motivations and describe the specific techniques that fall under these categories and motivate why quantization for reinforcement learning needs study.

[1]GPU memory can vary from anywhere from 3 GB to 12GB.

Compression for Memory and Storage: Techniques such as quantization, pruning, sparsification, and distillation reduce the amount of storage and memory required by deep neural networks. These techniques are motivated by the need to train and deploy neural networks on memory-constrained environments (e.g., IoT or mobile). Broadly, quantization reduces the precision of network weights (Han et al., 2015; 2016; Zhu et al., 2016), pruning removes various layers and filters of a network (Alford et al., 2018; Molchanov et al., 2016), sparsification zeros out selective network values (Molchanov et al., 2016; Alford et al., 2018) and distillation compresses an ensemble of networks into one (Hinton et al., 2015; Chen et al., 2017). Various algorithms combining these core techniques have been proposed. For example, Deep Compression (Han et al., 2015) demonstrated that a combination of weight-sharing, pruning, and quantization might reduce storage requirements by 35-49x.

Quantization for Reinforcement Learning: Prior work in quantization focuses mostly on quantizing image / supervised models. However, there are several key differences between these models and reinforcement learning policies: an agent's current input and decision affects its future state and actions, there are many complex algorithms (e.g: DQN, PPO, A2C, DDPG) for training, and there are many diverse tasks. To the best of our knowledge, this is the first work to ask the question as to how quantization affects deep reinforcement learning. To this end, we apply and analyze the performance of quantization across a broad of reinforcement learning tasks and training algorithms.

3 Quantized Reinforcement Learning (QuaRL)

We develop *QuaRL*, an open-source software framework that allows us to systematically apply traditional quantization methods to a broad spectrum of deep reinforcement learning models.[2] We use the *QuaRL* framework to **1)** evaluate how effective quantization is at compressing reinforcement learning policies, **2)** analyze how quantization affects/is affected by the various environments and training algorithms in reinforcement learning and **3)** establish a standard on the performance of quantization techniques across various training algorithms and environments.

Environments: We evaluate quantized models on three different types of environments: OpenAI gym (Brockman et al., 2016), Arcade Learning Environment (Bellemare et al., 2012), and PyBullet (which is an open-source implementation of the MuJoCo). These environments consist of a variety of tasks, including CartPole, MountainCar, LunarLandar, Atari Games, Humanoid, etc. The complete list

[2]Source code for QuaRL can be found here: https://github.com/harvard-edge/quarl

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

Table 1. Summary of algorithms, environments, and quantization scheme in the *QuaRL* framework. PTQ means post-training quantization, QAT refers to Quantization-Aware Training, BW corresponds to evaluating the policy from 8-bits to 2-bits. The Atari games are the no frameskip ver-sions with 4 frames stacked as input to the models. n/a means we cannot evaluate the combination due to algorithm-environment incompatibility. All put together, including the individual bitwidth experiments, we conduct over 350 experiments to present a deep understanding of how quantization affects deep reinforcement learning. This is the first such (comprehensive) study.

of environments used in the *QuaRL* framework is listed in Table 1. Evaluations across this spectrum of different tasks provide a robust benchmark on the performance of quantization applied to different reinforcement learning tasks.

Training Algorithms: We study quantization on four popular reinforcement learning algorithms, namely Advantage Actor-Critic (A2C) (Mnih et al., 2016), Deep Q-Network (DQN) (Mnih et al., 2013), Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2015) and Proximal Policy Optimization (PPO) (Schulman et al., 2017). Evaluating these standard reinforcement learning algorithms that are well established in the community allows us to explore whether quantization is similarly effective across different reinforcement learning algorithms.

Quantization Methods: We apply standard quantization techniques to deep reinforcement learning models. Our main approaches are post-training quantization and quantization aware training. We apply these methods to models trained in different environments by different reinforcement learning algorithms to broadly understand their performance. We describe how these methods are applied in the context of reinforcement learning below.

In context of the *QuaRL* framework, the policy neural network is retrained from scratch after inserting the quantization functions between weights and activations (all else being equal). At evaluation full precision weights are passed through the uniform affine quantizer.

4 RESULTS

In this section, we first show that quantization has regularization effect on reinforcement learning algorithms and can boost exploration. Secondly, We show that reinforcement learning algorithms can be quantized safely without significantly affecting the rewards. To that end, we perform evaluations across the three principal axes of *QuaRL*: environments, training algorithms, and quantization methods.For post-training quantization, we evaluate each policy for 100 episodes and average the rewards. For Quantization Aware Training (QAT), we train atleast three policies and report the mean rewards over hundred evaluations.

Effectiveness of Quantization: To evaluate the overall effectiveness of quantization for deep reinforcement learning, we apply post-training quantization and quantization aware learning to a spectrum of tasks and record their performance. We present the reward results for post-training quantization in Table 2. We also compute the percentage error of the performance of the quantized policy relative to that of their corresponding full precision baselines (E_{fp16} and E_{int8}). Additionally, we report the mean of the errors across tasks for each of the training algorithms.

The absolute mean of 8-bit and 16-bit relative errors ranges between 2% and 5% (with the exception of DQN), which indicates that models may be quantized to 8/16 bit precision without much loss in quality. Interestingly, the overall performance difference between the 8-bit and 16-bit post-training quantization is minimal (with the exception of the DQN algorithm, for reasons we explain in Section 2). We believe this is because the policies weight distribution is narrow enough that 8 bits is able to capture the distribution of weights without much error. In a few cases, post-training quantization yields better scores than the full precision policy. We believe that quantization injected an amount of noise that was small enough to maintain a good policy and large enough to regularize model behavior this supports some of the results seen by (Louizos et al., 2018a; Bishop, 1995; Hirose et al., 2018).

For quantization aware training, we train the policy with fake-quantization operations while maintaining the same model and hyperparameters. Figure 1 shows the results of quantization aware training on multiple environments and training algorithms to compress the policies down from 8-

Algorithm →	A2C					DQN					PPO					DDPG				
Datatype →	fp32	fp16		int8		fp32	fp16		int8		fp32	fp16		int8		fp32	fp16		int8	
Environment ↓	Rwd	Rwd	E_{fp16} (%)	Rwd	E_{int8} (%)	Rwd	Rwd	E_{fp16} (%)	Rwd	E_{int8} (%)	Rwd	Rwd	E_{fp16} (%)	Rwd	E_{int8} (%)	Rwd	Rwd	E_{fp16} (%)	Rwd	E_{int8} (%)
Breakout	379	371	2.11	350	7.65	214	217	-1.40	78	63.55	400	400	0.00	368	8.00					
SpaceInvaders	717	667	6.97	634	11.56	586	625	-6.66	509	13.14	698	662	5.16	684	2.01					
BeamRider	3087	3060	0.87	2793	9.52	925	823	11.03	721	22.05	1655	1820	-9.97	1697	-2.54					
MsPacman	1915	1915	0.00	2045	-6.79	1433	1429	0.28	2024	-41.24	1735	1735	0.00	1845	-6.34					
Qbert	5002	5002	0.00	5611	-12.18	641	641	0.00	616	3.90	15010	15010	0.00	14425	3.90					
Seaquest	782	756	3.32	753	3.71	1709	1885	-10.30	1582	7.43	1782	1784	-0.11	1795	-0.73					
CartPole	500	500	0.00	500	0.00	500	500	0.00	500	0.00	500	500	0.00	500	0.00					
Pong	20	20	0.00	19	5.00	21	21	0.00	21	0.00	20	20	0.00	20	0.00					
Walker2D																1890	1929	-2.06	1866	1.27
HalfCheetah																2553	2551	0.08	2473	3.13
BipedalWalker																98	90	8.16	83	15.31
MountainCar																92	92	0.00	92	0.00
Mean			1.66		2.31			-0.88		8.60			-0.62		0.54			1.54		4.93

Table 2. Post training quantization error for DQN, DDPG, PPO, and A2C algorithm. The "Rwd" column corresponds to the rewards. The negative error percentage means the quantized policy performed better than fp32 policy. We summarize the error in rewards using arithmetic mean.

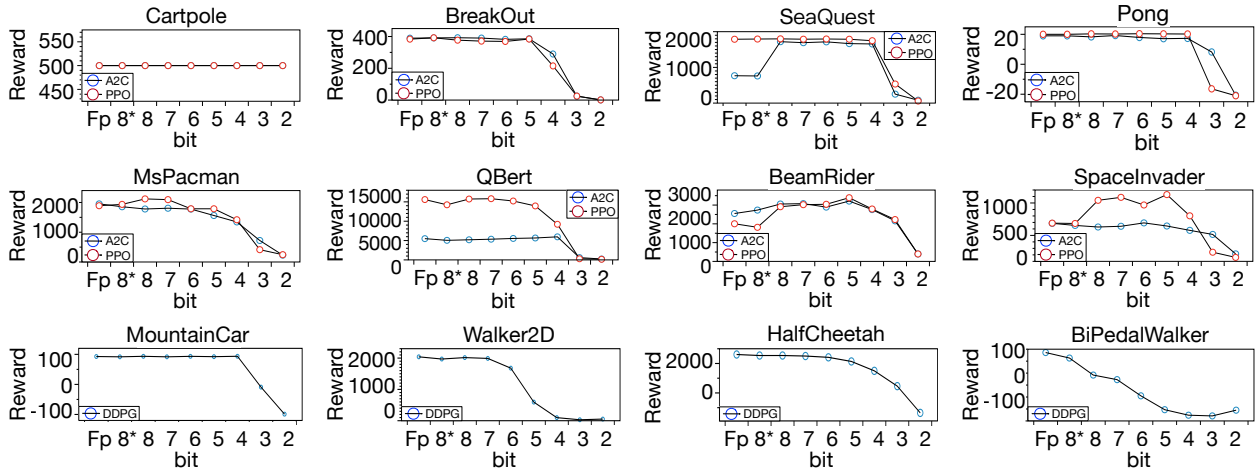

Figure 1. Quantization aware training of PPO, A2C, and DDPG algorithms on OpenAI gym, Atari, and PyBullet. FP is achieved by fp32 and 8* is achieved by 8-bit post-training quantization.

bits to 2-bits. Generally, the performance relative to the full precision baseline is maintained until 5/6-bit quantization, after which there is a drop in performance. Broadly, at 8-bits, we see no degradation in performance. From the data, we see that quantization aware training achieves higher rewards than post-training quantization and also sometimes outperforms the full precision baseline.

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

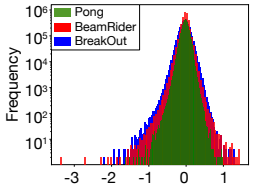

Figure 2. Weight distribution and corresponding 8-bit quantized error for models trained on the `Breakout`, `Beamrider` and `Pong` environments with DQN.

Effect of Environment on Quantization Quality: To analyze the task's effect on quantization quality we plot the distribution of weights of full precision models trained in three environments (`Breakout`, `Beamrider` and `Pong`) and their error after applying 8-bit post-training quantization on them. Each model uses the same network architecture, is trained using the same algorithm (DQN) with the same hyperparameters.

Figure 2 shows that the task with the highest error (`Breakout`) has the widest weight distribution, the task with the second-highest error (`BeamRider`) has a narrower weight distribution, and the task with the lowest error (`Pong`) has the narrowest distribution. With an affine quantizer, quantizing a narrower distribution yields less error because the distribution can be captured at a fine granularity; conversely, a wider distribution requires larger gaps between representable numbers and thus increases quantization error. The trends indicate the environment affects

models' weight distribution spread which affects quantization performance: specifically, environments that yield a wider distribution of model weights are more difficult to apply quantization to. This observation suggests that regularizing the training process may yield better quantization performance.

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%

Table 3. Rewards for DQN, PPO, and A2C.

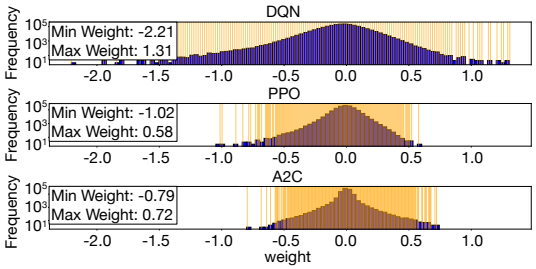

Figure 3. Weight distributions for the policies trained using DQN, PPO and A2C. DQN policy weights are more spread out and more difficult to cover effectively by 8-bit quantization (yellow lines). This explains the higher quantization error for DQN in Table 3. A negative error indicates that the quantized model outperformed the full precision baseline.

Effect of Training Algorithm on Quantization Quality: To determine the effects of the reinforcement learning training algorithm on the performance of quantized models, we compare the performance of post-training quantized models trained by various algorithms. Table 3 shows the error of different reinforcement learning algorithms and their corresponding 8-bit post-training quantization error for the Atari `Breakout` game. Results indicate that the A2C training algorithm is most conducive to int8 post-training quantization, followed by PPO2 and DQN. Interestingly, we see a sharp performance drop compared to the corresponding full precision baseline when applying 8-bit post-training quantization to models trained by DQN. At 8 bits, models trained by PPO2 and A2C have relative errors of 8% and 7.65%, whereas the model trained by DQN has an error of ∼64%. To understand this phenomenon, we plot the distribution of model weights trained by each algorithm, shown in Figure 2. The plot shows that the weight distribution of the model trained by DQN is significantly wider than those trained by PPO2 and A2C. A wider distribution of weights indicates a higher quantization error, which explains the large error of the 8-bit quantized DQN model. This also explains why using more bits (fp16) is more effective for the model trained by DQN (which reduces error relative to the full precision baseline from ∼64% down to ∼-1.4%).

Effect of Training Algorithm on Quantization Quality: To determine the effects of the reinforcement learning training algorithm on the performance of quantized models, we compare the performance of post-training quantized models trained by various algorithms. Table 3 shows the error of different reinforcement learning algorithms and their corresponding 8-bit post-training quantization error for the Atari `Breakout` game. Results indicate that the A2C training algorithm is most conducive to int8 post-training quantization, followed by PPO2 and DQN. Interestingly, we see a sharp performance drop compared to the corresponding full precision baseline when applying 8-bit post-training quantization to models trained by DQN. At 8 bits, models trained by PPO2 and A2C have relative errors of 8% and 7.65%, whereas the model trained by DQN has an error of ∼64%. To understand this phenomenon, we plot the distribution of model weights trained by each algorithm, shown in Figure 2. The plot shows that the weight distribution of the model trained by DQN is significantly wider than those trained by PPO2 and A2C. A wider distribution of weights indicates a higher quantization error, which explains the large error of the 8-bit quantized DQN model. This also explains why using more bits (fp16) is more effective for the model trained by DQN (which reduces error relative to the full precision baseline from ∼64% down to ∼-1.4%).

5 Resource Constrained Case Studies

To show the usefulness of our results, we use quantization to optimize the deployment of reinforcement learning policies on resource constrained aerial robot. To that end, we train and deploy a quantized robot navigation model onto a resource constrained embedded system (RasPi-3b), demonstrating 4× reduction in memory and an 18× speedup in inference. We also train a pong model 1.6x faster by using mixed precision training. Faster training time means running more experiments for the same time on a given system. Achieving speedup on resource-constrained devices enables deployment of the policies on real robots.

We train a reinfocement learning based point-to-point navigation models (Policy I, II, and III) for aerial robots using Air Learning (Krishnan et al., 2019) and deploy them onto a RasPi-3b, a cost effective, general-purpose embedded processor. RasPi-3b is used as proxy for the resource constrained compute platform for the aerial robot. The compute platforms used on aerial robots have similar characteristics. For each of these policies, we report the accuracies and inference speedups attained by the int8 and fp32 policies.

Table 4 shows the accuracies and inference speedups attained for each corresponding quantized policy. We see that quantizing smaller policies (Policy I) yield moderate inference speedups (1.18× for Policy I), while quantizing larger models (Policies II, III) can speed up inference by

up to 18×. This speed up in policy III execution times results in speeding-up the generation of the hardware actuation commands from 5 Hz (201.115 ms for fp32) to 90 Hz (11.036 ms for int8). Note that in this experiment we quantize both weights and activations to 8-bit integers; quantized models exhibit a larger loss in accuracy as activations are more difficult to quantize without some form of calibration to determine the range to quantize activation values to (Choi et al., 2018).

A deeper investigation shows that Policies II and III take more memory than the total RAM capacity of the RasPi-3b, causing numerous accesses to swap memory during inference (which is extremely slow). Hence, quantizing these policies allow them to fit into the RasPi's RAM, eliminating accesses to swap and boosting performance by an order of magnitude. Figure 4 shows the memory usage while executing the quantized and unquantized version of Policy III, and shows how without quantization memory usage skyrockets above the total RAM capacity of the board.

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 ×

Table 4. Inference speeds in millisecond (ms) on Ras-Pi3b+ and success rate (%) for three policies.

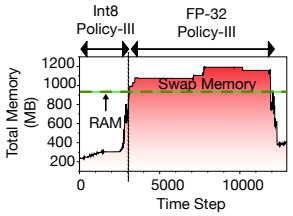

Figure 4. Memory consumption for Policy III's fp-32 and int8 policies.

In context of real-world deployment of an aerial (or any other type of) robot, a speedup in policy execution potentially translates to faster actuation commands to the aerial robot – which in turn implies faster and better responsiveness in a highly dynamic environment (Falanga et al., 2019). Our case study demonstrates how quantization can facilitate the deployment of a accurate policies trained using reinforcement learning onto a resource constrained platform.

Mixed/Half-Precision Training: Motivated by that reinforcement learning training is robust to quantization error, we train three policies of increasing model complexity (`Policy A`, `Policy B`, and `Policy C`) using mixed precision training and compare its performance to that of full precision training (see Appendix for details). In mixed precision training, the policy weights, activations, and gradients are represented in fp16. A master copy of the weights are stored in full precision (fp32) and updates are made to it during backward pass (Micikevicius et al., 2017). We measure the runtime and convergence rate of both full precision and mixed precision training.

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×

Table 5. Mixed precision training for RL.

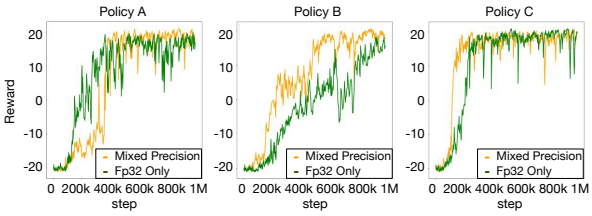

Figure 5. Mixed precision v/s fp32 training rewards.

Figure 5 shows that all three policies converge under full precision and mixed precision training. Interestingly, for `Policy B`, training with mixed precision yields faster convergence; we believe that some amount of quantization error speeds up the training process. Table 5 shows the computational speedup to the training loop by using mixed precision training. While using mixed precision training on smaller networks (`Policy A`) may slow down training iterations (as overhead of doing fp32 to fp16 conversions outweigh the speedup of low precision ops), larger networks (`Policy C`) show up to a 60% speedup. Generally, our results show that mixed precision may speed up the training process by up to 1.6× without harming convergence.

6 Conclusion

We 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 various reinforcement learning tasks and algorithms. We broadly demonstrate that reinforcement learning models may be quantized down to 8/16 bits without loss of performance. To show the usefulness of quantization for reinforcement learning, we apply our results to optimize the training and inference of reinforcement learning models, demonstrating a 50% training speedup for Pong using mixed precision optimization and up to a 18x inference speedup on a RasPi by quantizing a navigation policy. In summary, our findings indicate that there is much potential for the future of quantization of deep reinforcement learning policies to enable deployment on resource constrained systems.

References

Alford, S., Robinett, R., Milechin, L., and Kepner, J. Pruned

and Structurally Sparse Neural Networks. *arXiv e-prints*, art. arXiv:1810.00299, Sep 2018.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. A Brief Survey of Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1708.05866, Aug 2017.

Bellec, G., Kappel, D., Maass, W., and Legenstein, R. A. Deep rewiring: Training very sparse deep networks. *International Conference on Learning Representations (ICLR)*, 2017.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012. URL http://arxiv.org/abs/1207.4708.

Bishop, C. M. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, Jan 1995. doi: 10.1162/neco.1995.7.1.108.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Chen, G., Wongun, C., Yu, X., Han, T., and Chandraker, M. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, 2017.

Falanga, D., Kim, S., and Scaramuzza, D. How fast is too fast? the role of perception latency in high-speed sense and avoid. *IEEE Robotics and Automation Letters*, 4(2): 1884–1891, 2019.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. Eie: efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 243–254. IEEE, 2016.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. Eie: Efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 243–254, June 2016. doi: 10.1109/ISCA.2016.30.

Hinton, G., Vinyals, O., and Dean, J. Distilling the Knowledge in a Neural Network. *arXiv e-prints*, art. arXiv:1503.02531, Mar 2015.

Hirose, K., Uematsu, R., Ando, K., Ueyoshi, K., Ikebe, M., Asai, T., Motomura, M., and Takamaeda-Yamazaki, S. Quantization error-based regularization for hardware-aware neural network training. *Nonlinear Theory and Its Applications, IEICE*, 9(4):453–465, 2018. doi: 10.1587/nolta.9.453.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8. IEEE, 2016.

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J., Lam, V., Bewley, A., and Shah, A. Learning to drive in a day. *CoRR*, abs/1807.00412, 2018. URL http://arxiv.org/abs/1807.00412.

Krishnan, S., Boroujerdian, B., Fu, W., Faust, A., and Reddi, V. J. Air learning: An AI research platform for algorithm-hardware benchmarking of autonomous aerial robots. *CoRR*, abs/1906.00421, 2019. URL http://arxiv.org/abs/1906.00421.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *International Conference on Learning Representations (ICLR)*, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lillicrap, T. P., Hunt, J. J., Pritzel, A. e., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, Sep 2015.

Lin, J., Gan, C., and Han, S. Defensive quantization: When efficiency meets robustness. *International Conference on Learning Representations (ICLR)*, 2019.

Louizos, C., Reisser, M., Blankevoort, T., Gavves, E., and Welling, M. Relaxed quantization for discretized neural networks. *International Conference on Learning Representations (ICLR)*, 2018a.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l0 regularization. *International Conference on Learning Representations (ICLR)*, 2018b.

Micikevicius, P., Narang, S., Alben, J., Diamos, G. F., Elsen, E., García, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. Mixed precision training. *CoRR*, abs/1710.03740, 2017. URL http://arxiv.org/abs/1710.03740.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning Convolutional Neural Networks for Resource Efficient Inference. *International Conference on Learning Representations (ICLR)*, Nov 2016.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Park, J., Li, S. R., Wen, W., Li, H., Chen, Y., and Dubey, P. Holistic sparsecnn: Forging the trident of accuracy, speed, and size. *International Conference on Learning Representations (ICLR)*, 2016.

Polino, A., Pascanu, R., and Alistarh, D. Model compression via distillation and quantization. *International Conference on Learning Representations (ICLR)*, 2018.

Sakr, C. and Shanbhag, N. R. Per-tensor fixed-point quantization of the back-propagation algorithm. *International Conference on Learning Representations (ICLR)*, 2018.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Zhang, F., Leitner, J., Milford, M., Upcroft, B., and Corke, P. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. In *Australasian Conference on Robotics and Automation (ACRA)*, Nov 2015.

Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016. URL http://arxiv.org/abs/1606.06160.

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.