Choosing neural network compression methods for reinforcement learning

Nidhir Guggilla, Kaushik Kunal Singh, Andrew Kim, Gaurav Bhatnagar
CS 262A Fall 2023

Introduction

Improvements in neural networks have led to increased usage and performance in reinforcement learning, but have often incurred significant storage, memory, and compute costs prohibitive to real-time use. This has led to an interest in developing model compression methods.

While a number of neural network model compression methods have been proposed and are in use today, there has not been an analysis of tradeoffs between compression methods for reinforcement learning.

We analyze compression methods like low-rank approximation (LRA), sparsification (pruning), and quantization for a two tasks to provide richer insights of the tradeoffs between compression methods.

Methods

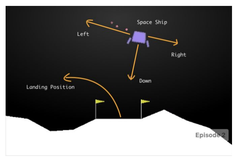


Fig 1: LunarLander

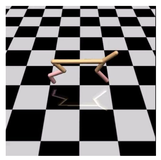


Fig 2: HalfCheetah

We test agents trained with Policy Gradients (PG) and Double Deep Q-Learning (DDQNs) on the HalfCheetah-v2 and LunarLander-v2 tasks of Gymnasium respectively. LunarLander is a discrete control environment, while HalfCheetah is a continuous one.

We train on a single machine with a 1660Ti GPU and an isolated processor for reproducible behavior and availability constraints.

Results

Sparsification (Pruning) and Low Rank Approximation (LRA)

- All compression lowers performance, but pruning has more of an effect than rank reduction (Fig. 3, 6)
- On the other hand, rank reduction has a much more significant effect on working memory (RAM), CPU utilization, and evaluation time (Fig. 5, 7, 9)
- As a byproduct of this, we observed a positive relationship between performance and evaluation time (Fig. 4). The evaluation time here was standardized to 100 runs, so models that took longer to perform a forward pass, i.e. runs that were computationally expensive in decision-making for our lunar lander game, were able to produce better results. Across most pruning levels there was a sharp fall in evaluation time around 30% rank reduction(Fig. 5).
- Fine-tuning causes lower deviation between performance for a given eval time (Fig. 4), but higher deviation for a given RAM usage (Fig. 10)
- Occasionally, we saw performance rise with minor compression amounts (in the range of <20%). This may be due to a reduction in overfitting. Performance had an elastic relationship with evaluation time at low eval time for naive compression (Fig. 4).

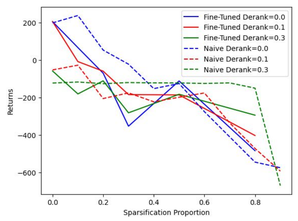


Fig 3: % pruned vs return

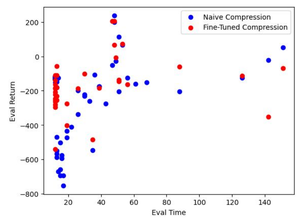


Fig 4: % eval time vs return

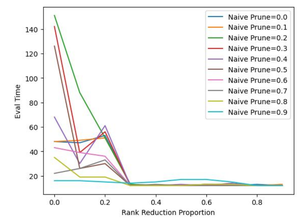


Fig 5: % rank reduced vs eval time

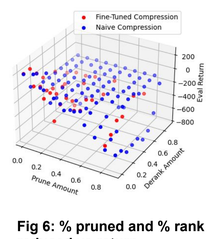


Fig 6: % pruned and % rank reduced vs return

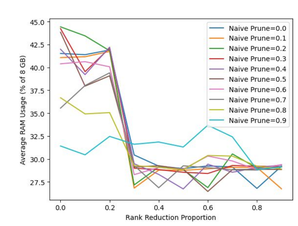


Fig 7: % rank reduced vs %RAM

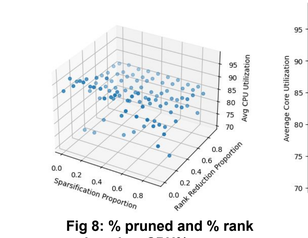


Fig 8: % pruned and % rank reduced vs CPU%

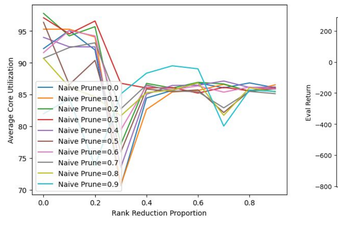


Fig 9: % rank reduced vs CPU%

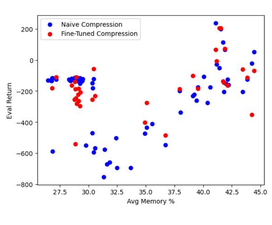
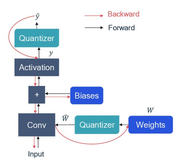


Fig 10: %RAM vs return

Quantization (32 bit → 8 bit)

We did not notice a significant difference in average cpu utilization or RAM utilization after compressing the model with in-training quantization. This is likely due to the addition of quantize and dequantize nodes in the computation graph, as shown below -



However, we did notice benefits in the memory required to store the model weights, from 0.3MB to 0.1MB (compared to the 0.075MB theoretical storage required).

Conclusion

We show the performance achieved for compressed models on reinforcement learning tasks, as well as the CPU, MEM, and time saved by compression. We also show the impacts of fine-tuning using compressed target networks.

Interestingly, fine-tuned compression did not give significant gains as opposed to naive compression, while taking significantly more resources to further train the RL agent. This suggests that practitioners can try “naive” compression methods first for RL models.

Future directions of research involve choosing continuous environments instead of discrete environments, testing methods such as knowledge distillation, and exploring more efficient model storage methods. We hope to improve our fine-tuning methods to achieve better performance than their naive counterparts.

A future improvement in methodology is to mitigate the steeper drop-off in performance from higher levels of rank reduction than pruning - a potential improvement would be to set minimum ranks for input/output network layers.

References

J. O. Neill, “An Overview of Neural Network Compression.” arXiv, Aug. 01, 2020.

G. Brockman et al., “OpenAI Gym.” arXiv, Jun. 05, 2016.