Efficient Deep Learning for Multi Agent Pathfinding

Natalie Abreu

University of Southern California
3710 McClintock Ave
Los Angeles, CA 90089
nrabreu@usc.edu

Abstract

Multi Agent Path Finding (MAPF) is widely needed to coordinate real-world robotic systems. New approaches turn to deep learning to solve MAPF instances, primarily using reinforcement learning, which has high computational costs. We propose a supervised learning approach to solve MAPF instances using a smaller, less costly model.

Introduction

Multi Agent Path Finding (MAPF) is the problem of finding non-conflicting paths for multiple agents from their start locations to their goal locations in discrete environments. To successfully solve an MAPF instance, the agents must take paths to their goal locations such that the agents do not collide with one another and avoid preventing other agents from reaching their goal locations. MAPF is NP-hard, and recent work has turned to deep learning techniques, often with computationally expensive training processes. For example, the PRIMAL approach developed in (Sartoretti et al. 2019) is based on reinforcement learning and takes upwards of three weeks to train. Recently, a very large dataset of optimal solutions to MAPF instances was collected during research on MAPFAST (Ren et al. 2021), a deep learning-based MAPF algorithm selector. Using this dataset, we propose a supervised learning approach to cut down on computational cost in training MAPF models. We then compare the training times, accuracy, and performance in MAPF instances achieved by models of varying complexity, and analyze how the model complexity affects its performance.

Related Work

Recently, work in MAPF has turned to deep neural networks and reinforcement learning. The first successful approach was the PRIMAL method developed by Sartoretti et al. (Sartoretti et al. 2019), which used a convolutional neural network architecture along with a mixture of imitation and reinforcement learning. Following Primal, PRIMAL_2 improved the performance of PRIMAL in certain highly constrained instances by improving the coordination between agents (Damani et al. 2021). However, PRIMAL as well as other MAPF learning-based approaches can suffer from very high training times. PRIMAL, for instance, was trained for three weeks before convergence. Changing the model architecture and training process may allow us to speed up the training process and use less computational resources, which is important given the monetary and environmental cost of training large neural networks (Strubell, Ganesh, and McCallum 2019).

Approach

Optimal-MAPF Dataset

We used a dataset of optimal MAPF solutions gathered during the development of MAPFAST, an algorithm selector model that selects which algorithm should be fastest to run for any given MAPF instance (Ren et al. 2021). The MAPFAST dataset consists of solution paths for close to 25,000 MAPF instances. It is possible for each instance to have up to four solutions for each of the four solvers used in MAPFAST. The MAPFAST dataset was generated on a variety of maps with different start and goal distributions for a large range of agents. Each step along every path in the dataset can be viewed as a decision point for every agent, to move up, down, left, or right. We converted the MAPFAST dataset into a dataset where every datapoint is the decision for a single timestep for a single agent. For our work, we formatted each point in our dataset so that it contained the same local range of vision format that PRIMAL used (Sartoretti et al. 2019). The features of each datapoint included obstacles, other nearby agents, other agents' goals, and a unit vector to the given agent's goal position. Each label consists of the relative next step that the agent took during that timestep. All together, our dataset consists of 453,506 datapoints.

Model Architectures

We studied convolutional neural networks (CNNs), which consist of a series of convolutional layers followed by a series of linear layers. The PRIMAL architecture is specifically based on the popular VGG-Net architecture (Simonyan and Zisserman 2015). Although this architecture worked well in PRIMAL, it consists of over 100 million network parameters and therefore is expensive to train. We sought to explore how much, if any, performance loss we would encounter with smaller network architectures. We experimented with the number of convolutional and linear layers

Model #	# Conv Layers	# Linear Layers	Params	Accuracy	Training Time	Success Rate for Real Instances
1	2	1	27,902	.788	30m 5s	.648
2	2	2	59,896	.806	33m 27s	.600
3	4	2	185,096	.810	1h 13m 49s	.634
4	3	2	219,944	**.812**	1h 5m 2s	.621
5	4	2	551,976	**.812**	2h 16m 0s	**.669**

Table 1: Results for a variety of model architectures. Models are ordered by size in terms of number of parameters. We found the largest model to achieve both highest accuracy and best performance in solving real MAPF instances.

as well as with the shape of each layer while also considering the added cost of a greater model size.

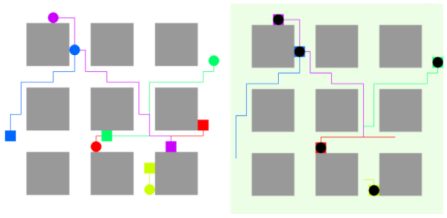

Figure 1: Example of MAPF instance used to evaluate model performance. The lines represent the current A* paths of each agent to their goal location. The image on the left depicts the starting state of the instance, and the image on the right depicts the ending state in which each agent has reached its destination. The remaining paths depict the A* paths that the agents took once they reached positions such that the A* paths had no collisions.

Results

To measure the performance of the different architectures we primarily use two metrics, prediction accuracy and performance on real MAPF instances. Accuracy is measured in terms of how often the model correctly predicts the step direction the agent took in the optimal path. To evaluate each model's performance on real MAPF instances, we ran each model on a sequence of 100 instances on the warehouse environment depicted in Fig. 1, averaging the model's performance over 20 iterations of each instance. Success rate is the fraction of instances successfully solved by the model. The sequences were built using randomized agent start and goal locations, and we kept the sequence of instances constant for each model to compare performance directly. We considered an instance "completed" if the model was able to move agents into positions such that the A* paths of agents to their respective goal locations have no collisions in order to add a "shortcut" to the path finding process. This was done because the main purpose of using machine learning in MAPF is that it can be difficult to coordinate many agents when they are close together; if all agents can reach their goal following simple paths, the hard part is over.

Our results are presented in Table 1. The two largest models achieved an accuracy of .812, the highest among tested models. Based on our model experimentation, we found that model size did correlate with training accuracy, with larger models having higher accuracy. Accordingly, we found the model most successful in solving real MAPF instances to be the largest model. However, the smallest model had performance close to the largest model. These results suggest that a simpler model may be appropriate in solving MAPF problems, and the higher training time that comes with more complex models may not lead to any substantive difference in paths generated by the model.

Conclusion and Future Work

Our work signifies the potential of supervised learning in solving MAPF instances, especially in its ability to reduce computational costs. With low training costs, we achieved an accuracy above .8 in training and above .66 in performance. In performance, the smallest model worked almost as well as the largest model we trained.

There are multiple avenues for future work in this project. First, we are interested in incorporating RL training after supervised training, to investigate the possible cost saved by using our preliminary ML model even when using an RL approach. Additionally, we are interested in using our smaller models in the PRIMAL training method and comparing results to better understand the trade offs of model size/cost and model performance.

Acknowledgements

Thanks to Eric Ewing and Nora Ayanian for their contributions to the project.

References

Damani, M.; Luo, Z.; Wenzel, E.; and Sartoretti, G. 2021. PRIMAL_2: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning-Lifelong. *IEEE RAL*, 6(2): 2666–2673.

Ren, J.; Sathiyanarayanan, V.; Ewing, E.; Senbaslar, B.; and Ayanian, N. 2021. MAPFAST: A Deep Algorithm Selector for Multi Agent Path Finding using Shortest Path Embeddings. In *AAMAS*, 1055–1063.

Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T. S.; Koenig, S.; and Choset, H. 2019. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE RAL*, 4(3): 2378–2385.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Strubell, E.; Ganesh, A.; and McCallum, A. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *ACL*, 3645–3650.