Continuous Reinforcement Learning & Catastrophic Forgetting with Pac-Man

Victor Goncalves

Abstract

The ability to learn multiple tasks sequentially within reinforcement learning could serve to create more powerful and versatile agents. However, neural networks fail in this continuous learning case due to the problem of *catastrophic forgetting*. To alleviate this forgetting, we look at the levels of similarity between these multiple tasks to see which combination of tasks minimizes forgetting. Using a Pac-Man reinforcement learning model, we investigate the levels of catastrophic forgetting using a variety of different tasks–such as modifying ghost movement and the map the agent must navigate through. We demonstrate how we were able to minimize forgetting in Pac-Man by training on multiple similar tasks.

Keywords
continuous reinforcement learning, catastrophic forgetting, machine learning

1 Introduction

Continuous RL & Catastrophic Learning

There has been tremendous success with using reinforcement learning within video games to maximize the performance of an agent on a particular task, such as winning a game or receiving high scores (Kessler et al., 2022). However, many of these results deal with training an agent on a *single* task. In this project, I will tackle the continuous reinforcement learning problem, where agents are trained on multiple tasks sequentially while seeking to maintain solid performance on previously mastered tasks (Kessler et al., 2022). Specifically, in continuous reinforcement learning, a key issue of *catastrophic forgetting* arises: an agent overwrites network parameters important to previous tasks when trying to train for new tasks (Kessler et al., 2022).

Continuous reinforcement learning could be very influential to the field of RL and artificial intelligence. It can take a long time to train an agent to complete a task, so if an agent can retain performance on previously mastered tasks while mastering new tasks, it could cut significant amounts of training time and could create a more robust agent. Moreover, using an agent to tackle multiple tasks is more representative of real-world problems, and could serve to be very useful if agents could master multiple tasks. In this project, I will train an agent to play Pac-man with multiple tasks sequentially and test its performance against previously mastered tasks.

Hypothesis

Using a full python emulator of the famous game Pac-man, by Tycho van der Ouderaa and the University of Amsterdam, I will train a Pac-man agent to dodge, scare, eat ghosts, and attempt to eat all the food on the map against a variety of different task environments (van der Ouderaa et

al., 2016). We can use directional ghost movement, where the ghost always goes directly to Pac-man, as an initial training task for the control. Afterwards, we can additionally train this agent on other tasks to see if it retains performance in the control environment to test the effects of continuous reinforcement learning.

When testing these continuously trained agents' performance on the previously mastered control task, we will expect them to perform worse in terms of score and winning ratio when compared to its original performance. For example, we can expect win-ratio results to decrease from 70% with the control to 40% after being trained continuously with a new task. The decrease in performance is expected due to the catastrophic forgetting issue as a result of continuous reinforcement learning and rewriting neural network weights when training on multiple tasks sequentially (Kessler et al., 2022). Lastly, I argue that the difference in agent performance will vary depending on the similarity between the original and new tasks. For example, if the new task is similar to the original task in terms of the ghosts moving directly towards Pac-man, it will perform with decent results, such as 60% game win-rate compared to the 70% control win-rate.

2 Literature Review

Continuous reinforcement learning and catastrophic forgetting has been a popular topic within today's literature due to the popularity of reinforcement learning and its variety of uses such as: disease detection, self-driving automobiles, and creating superhuman-performing Go agents such as AlphaGo Zero (Dai et al., 2021; Salleb et al., 2017; Silver et al., 2017). Recent literature on the topic of catastrophic forgetting includes: papers that define continuous reinforcement learning and catastrophic forgetting using the MNIST dataset and video game experiments (Kessler et al. 2022), papers that look into measuring levels of catastrophic forgetting (Kemker et al., 2017), and papers that look into overcoming catastrophic forgetting using elastic weight consolidation models (Kirkpatrick et al, 2017). The deep learning Pac-man DQL model by Tycho van der Ouderaa and the University of Amsterdam made it possible to test the effects of catastrophic forgetting in Pac-Man along with its parameters and documentation (van der Ouderaa et al., 2016), and the Stanford paper was helpful to see feasible tests, parameters, and evaluations based on van der Ouderaa Pac-man model (Gnanasekaran et al., 2017).

Specifically the paper by Kessler et al. seems most useful to addressing catastrophic forgetting within this Pac-man model. The paper lays out different experiments with video games that I can use to create the various experiments to test Pac-man, and also gives insight on different techniques used to minimize forgetting, such as *regularization* where new learning produces weights which are similar to previous tasks.

Researchers have addressed the issue of catastrophic forgetting by identifying and defining the issue within various experiments and while trying to mitigate the effects of this forgetting. However, levels of *forgetting* between similar tasks has not been extensively studied. With a perfect environment like Pac-man that makes it easy to test various different tasks within a set space, I can test how similar training tasks affect performance drops and forgetting. If training on multiple similar tasks results in minimal performance drops with *forgetting*, it might show the promise of training specialized agents that can tackle multiple similar tasks within a space.

3 Method

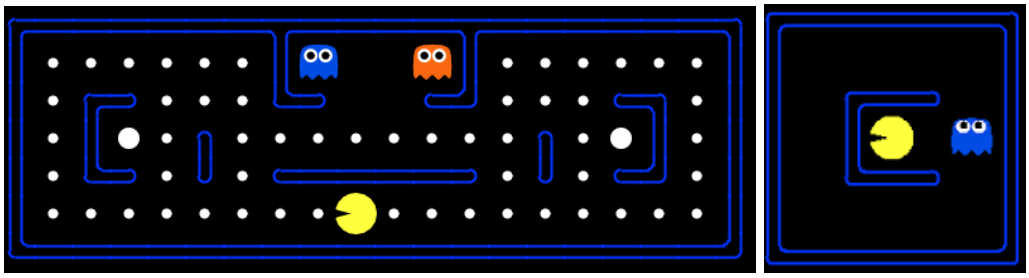
The dataset for this project will be obtained by running the Pac-man simulator by Tycho van der Ouderaa, and training its DQL agent model. The input will be the environment of the game Pac-Man including: the agent Pac-Man and its state, the ghosts, the coins scattered around the map, and the map itself (van der Ouderaa et al., 2016). The output will be the agent who has the ability to play Pac-Man with its moveset: up, left, right, and down.

I will train an agent(s) on the 2 following Pac-man maps in van der Ouderaa's codebase: *smallGrid*, *smallClassic* (van der Ouderaa et al., 2016). As shown in Gnanasekaran et al., training an agent on any map larger is computationally unreasonable (Gnanasekaran et al., 2017). Furthermore, for each map, I will train the agents on 100,000 iterations with the ghosts constantly moving directly towards Pac-man (labeled as the transition function 'DirectionalGhost'). This will be task 1, and the performance results on this task will be the control for all the other experiments. Next, I will train these agents on an additional 30,000 iterations on a variety of different experiments as labeled below (such as avoiding random ghost movement, etc..). Call this task 2 in the continual reinforcement learning assignment (Kessler et al. 2022). I will test these new agents on multiple 1000 game sessions with the control Task 1, average the results, and compare its results with the control to measure its *catastrophic forgetting*. Furthermore, I will compare the level of forgetting, through the difference in win ratios and average scores, between each of the experiments to analyze in the final report.

In terms of changing the codebase, I modified the function to run the Pac-Man games in order to automatically run through all of my experiments. In Github, I saved my models for my tests and labeled their directories here to load them automatically when creating the Pac-Man agents. Furthermore, I created a system to average win-rate and score over multiple 1000 game sessions in order to produce more accurate results. Also, I created a function *plot_games()* in pacman.py that uses Matplotlib to plot these different results.

4 Maps & Training

Maps



(b): smallClassic *(a): smallGrid*

Figure 1

Training Episodes

Experiments w/ Training Iterations	*smallGrid*	*smallClassic*
DirectionalGhost (control)	100,000 (iterations)	100,000
Random ghost movement	30,000	30,000
Less pebbles on the board	30,000	30,000
More ghosts on the board	30,000	30,000
Total Trained Agents	**8**	
Total Training Iterations	**380,000**	

Figure 2

In Figure 1, the total number of training iterations are listed. The computation was expensive, particularly with *smallClassic* map training. The total training time for these 8 agents was roughly around 2 weeks of straight computation.

Furthermore, these are the different experiments/tasks run on the different Pac-Man agents. In terms of modifying ghost movement as tasks, 'DirectionalGhost' had ghosts constantly move directly toward Pac-Man, 'RandomGhost' had random ghost movement, and 'HalfAndHalf' had both direct and random ghost movement, but alternated between them about fifty percent of the time. The other two tasks included having less ghosts on the board and more pebbles on the board; however, due to *smallClassic* already including the maximum number of pebbles and *smallGrid* only having the environment capacity to include one ghost, these tasks were map specific.

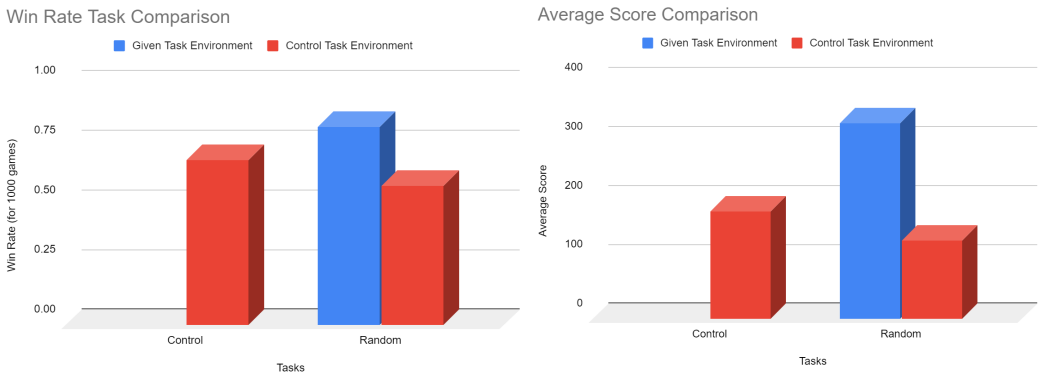
5 Results

Preliminary Results

For the preliminary results, I downloaded the repository, and trained and tested a Pac-man DQL agent on the smallGrid map for a total of 60,000 episodes: 40,000 with the directional ghost movement, and 20,000 episodes following with the random ghost movement. These agents were tested on 1,000 games, and their results are as follows:

	Directional Ghost (Control–Task 1) 40,000 iterations	**Random Ghost (Task 2)** 20,000 iterations	**Retesting the multi tasked agent on Task 1**
Win-rate after 1000 games	685/1000 (0.69)	832/1000 (0.83)	581/1000 (0.58)
Average score after 1000 games	182.098	331.565	131.837

(a)

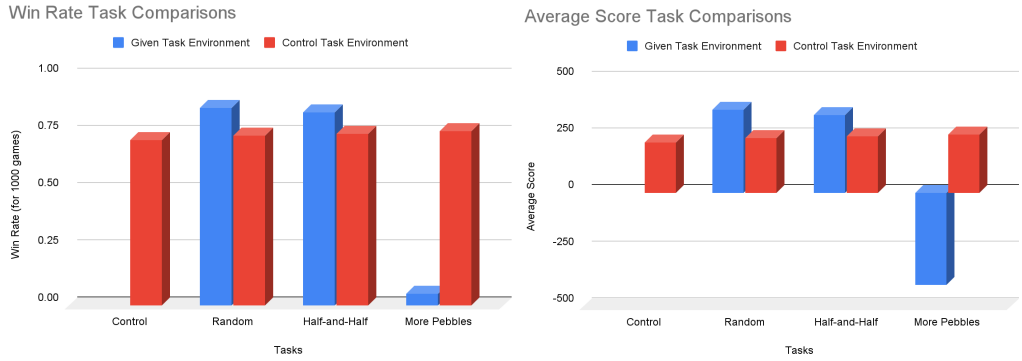


(b) *(c)*

Figure 3: Preliminary Results

According to the results of *catastrophic forgetting* discussed in Kessler et al., the results from Figure 2 are exactly what was expected to occur. Training the original agent on random ghost movement and testing it on a previously mastered task resulted in worse performance compared to the control–the agent seemed to overwrite the original weights in the neural network that dealt with the control task.

Main Results

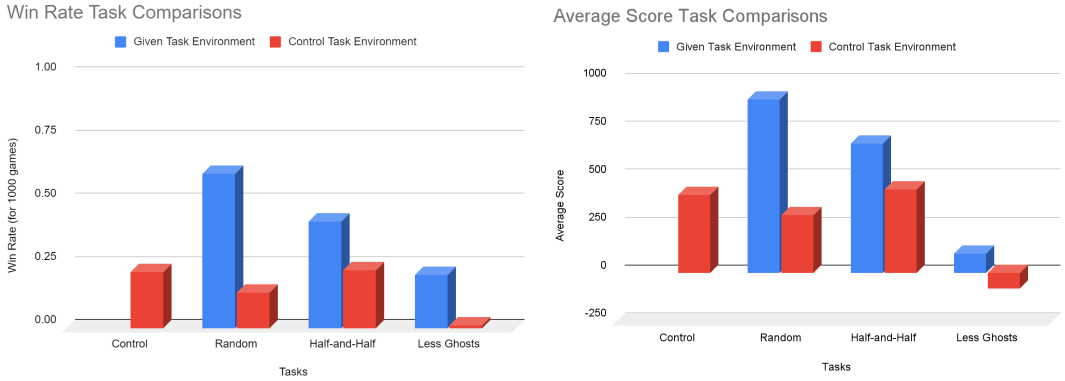


(a) *(b)*

Figure 4: smallGrid results

The results from Figure 3 were trained with the following 130,000 episodes: 100,000 with directional ghosts and 30,000 on various tasks as listed above. Results varied from each task when being tested in its respective environment (for example: Pac-Man being tested in an environment with random ghost movement). Particularly, in the environment with more pebbles, Pac-Man had a much lower win rate and average score when compared to the other tasks. However, the most interesting results lied in the agents' performance against the control task. After being sequentially trained, the agent performed equally to the control in all cases–no catastrophic forgetting occurred. In the preliminary study on *smallGrid*, results on a total of 60,000 training episodes showed catastrophic forgetting, but no catastrophic forgetting was shown when using 130,000 iterations. Perhaps, catastrophic forgetting can be minimized with

more training; however, it raises concerns about a decrease in performance due to overfitting the models.



(a) *(b)*

Figure 5: smallClassic results

Again, the results from Figure 4 were trained with the following 130,000 episodes: 100,000 with directional ghosts and 30,000 on various tasks as listed above. Win rates were much lower on this map than *smallGrid* due to increased difficulty of the map with the larger map size, more ghosts and more food pebbles for the agent to collect. Overall, forgetting was present in almost every task, with the largest decrease in performance in the case with fewer ghosts when compared to the control. Furthermore, the performance of the random ghost movement task was significantly high which is reasonable due to the map's large size (meaning random movement might make it extremely difficult for the ghost to reach Pac-Man). Interestly, in the 'HalfAndHalf' ghost movement task, it produced a slightly higher win-rate and average score when compared to the control task. As this task is similar to the control directional ghost movement task, these results are reasonable. According to Kesser et al., these results show *regularization,* in which we guided and calibrated the model by using similar coefficients needed for testing data. In other words, we guided our Pac-Man agent to perform well on the control by using a *similar* task to the control, 'HalfAndHalf' ghost movement. This was the only case in which catastrophic forgetting was not present, and training on multiple similar tasks might serve to reduce catastrophic forgetting in other trained models.

6 Challenges

The main challenge for these experiments included the training time for each agent. The Pac-man simulation is computationally expensive, especially on the larger maps, and can take many hours to finish training. In total, my 380,000 training iterations took over 2 weeks of straight computation. Furthermore, according to Tycho van der Ouderaa's paper on the Pac-Man codebase, there might be a couple of problems with end-game after a large amount of training iterations that can lead to skewed results of my experiments. For example, the model may predict that "moving into a ghost and lose (suicide) results in a higher cumulative reward than finding the last dot in the map" (van der Ouderaa et al., 2016). Therefore my current win-rate evaluation metrics might not give accurate results to this issue, so it might be ideal to find the optimal training iterations for my agents and consider different evaluation methods (such as pebbles eaten or average score).

7 Discussion

We present the results of a variety of different experiments with Pac-Man continuous reinforcement learning exploring catastrophic forgetting. Continuous reinforcement learning deals with training on multiple tasks sequentially which can lead to the issue of catastrophic forgetting–weights important to previous tasks are rewritten during training resulting in a decrease in performance when testing on a previously mastered task. Using the two maps *smallGrid* and *smallClassic* from van der Ouderaa's codebase, we explored forgetting after training an initial 100,000 episodes with directional ghost movement, 30,000 iterations on a variety of different tasks, and retesting the Pac-man agent on the directional ghost environment.

Our results showed cases where Pac-Man "forgot" how to dodge directional ghosts in many experiments, particularly during the preliminary *smallGrid* results and in almost all the *smallClassic* experiments. However, the most interesting cases are where we avoided catastrophic forgetting. In all the *smallGrid* experiments in which we trained on 130,000 episodes, Pac-Man avoided forgetting entirely. This differs from the 60,000 training episodes from the preliminary results, and it might show how more training might be a way to counter catastrophic forgetting. However, I have not conducted exhaustive research to fully test this.

Furthermore, we countered catastrophic forgetting in the *smallClassic* map using a similar task to the control directional ghost movement–'HalfAndHalf' ghost movement where the ghost goes directly to Pac-man half of the time and randomly the other half. It is reasonable that this task produced similar results to the control as 'HalfAndHalf' includes both directional and random ghost movement. From these results, we showed that training on similar tasks compared to its testing task can guide the weights of the neural network in order to increase performance. Continuously training on similar tasks can therefore create more versatile agents that can perform well in multiple similar tasks.

References

Dai, L., Wu, L., Li, H., Cai, C., Wu, Q., Kong, H., Liu, R., Wang, X., Hou, X., Liu, Y., Long, X., Wen, Y., Lu, L., Shen, Y., Chen, Y., Shen, D., Yang, X., Zou, H., Sheng, B., Jia, W. (2021, May 28). A deep learning system for detecting diabetic retinopathy across the disease spectrum. Nature News. Retrieved April 30, 2022, from https://www.nature.com/articles/s41467-021-23458-5

Gnanasekaran, A., Feliu-Faba, J., & An, J. (2017). Reinforcement learning in Pacman - cs229.stanford.edu. Reinforcement Learning in Pacman. Retrieved May 1, 2022, from https://cs229.stanford.edu/proj2017/final-reports/5241109.pdf

Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C. (2017, November 9). Measuring catastrophic forgetting in neural networks. arXiv.org. Retrieved April 30, 2022, from https://arxiv.org/abs/1708.02072

Kessler, S., Parker-Holder, J., Ball, P., Zohren, S., Roberts, S. J. (2022, March 15). Same state, different task: Continual reinforcement learning without interference. arXiv.org. Retrieved April 30, 2022, from https://arxiv.org/abs/2106.02940

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R. (2017, January 25). Overcoming catastrophic forgetting in Neural Networks. arXiv.org. Retrieved April 30, 2022, from https://arxiv.org/abs/1612.00796

Sallab, A. E., Abdou, M., Perot, E., Yogamani, S. (2017, April 8). Deep Reinforcement Learning Framework for autonomous driving. arXiv.org. Retrieved April 30, 2022, from https://arxiv.org/abs/1704.02532

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017, October 19). *Mastering the game of go without human knowledge*. Nature News. Retrieved April 30, 2022, from https://www.nature.com/articles/nature24270/

van der Ouderaa, T., Gavves, E., Reisser, M. (2016, June 10). Deep Reinforcement Learning in Pac-man. Retrieved from http://ai.berkeley.edu/projectoverview.html