Enhancing Deep Reinforcement Learning: Addressing Task Interference through Multitask Policy Distillation

Andrei Lixandru

Email address: andrei.lixandru.ilc@gmail.com

Student number: 2051995

Tilburg University

Department of Cognitive Science & Artificial intelligence

Supervised by dr. Giacomo Spigler

19 May 2023

Preface

The human mind is the most fascinating aspect of our universe and the only way to truly understand something is to build it. This idea fuels me throughout the journey of building artificial general intelligence. At the time of writing, the best bet we have to achieve this is through deep reinforcement learning (DRL). From the time I found out about DRL at a guest lecture offered by dr. Giacomo Spigler, I remained in a perpetual wonder at the incredible feets this learning framework can achieve. During the guest lecture, I was struck by the video demo of OpenAI's hide-and-seek agents displaying sophisticated strategies, learned with no explicit prior knowledge, just by interaction with their environment. At that moment, my mind ran wild with ideas about scaling up these methods to language learning, robots learning in the real world, and optimizing agents' bodies through thousands of years of simulated evolution.

This simple concept of an agent learning only through interaction with its environment, with no micromaniging, no carefully orchestrated modules responsible for very specific tasks. All that is needed to create an intelligent system is a singular, end-to-end system, taking your hands off and letting it learn from interaction with its environment.

This thesis project is my best attempt to channel my enthusiasm for DRL into laying a brick on the structure of existing research in the field that is most likely to give rise to AGI.

Abstract

Deep Reinforcement Learning (DRL) is renowned for its susceptibility to task interference, which can significantly prolong the training time required for an agent to learn successful policies. This thesis project addresses the issue of task interference in DRL by examining the performance of models trained using conventional methods and techniques specifically designed to mitigate the impact of interference through the utilization of trained agent experiences. In this research, the focus is on exploring the application of policy distillation in a multitask learning setting. The primary research questions revolve around comparing the training time and performance of a standard multitask agent with a multitask agent trained using Multitask Policy Distillation (MTPD) techniques. MTPD stands out as an innovative approach that capitalizes on knowledge distillation, enabling learning from multiple teacher models, each specializing in a distinct task. The investigation takes place in the Ant-v4 environment within the Mujoco robotics simulator. Specifically, the experiment involves training four teacher models to navigate in different directions (left, right, up, down). By employing MTPD, the thesis aims to determine the effectiveness of training a single agent to competently perform multiple tasks.

Keywords: deep reinforcement learning, multitask learning, lifelong learning, task interference

Enhancing Deep Reinforcement Learning: Addressing Task Interference through Multitask Policy Distillation

Introduction

Within the field of Deep Reinforcement Learning (DRL), this research not only aims to enhance performance and training efficiency but also addresses the challenge of task interference, while contributing to the broader goal of lifelong learning in machine learning. Task interference occurs when training an agent to perform multiple tasks simultaneously, leading to reduced performance on individual tasks. This phenomenon hinders the scalability and effectiveness of DRL models, particularly in complex robotic control tasks.

To overcome task interference, this study explores the potential of Multitask Policy Distillation (MTPD) as an effective method for training a single agent to perform multiple tasks proficiently. By distilling knowledge from multiple teacher models, each specialized in a different task, into a student model, MTPD leverages the expertise of individual models. This approach not only addresses the challenges of training efficiency and scalability but also promotes effective knowledge transfer and robust performance across multiple tasks.

Moreover, the application of MTPD aligns with the concept of lifelong learning in machine learning. Lifelong learning entails the continuous adaptation of a model to new tasks while retaining previously acquired knowledge. Traditional methods of expanding a model's task repertoire often involve retraining the entire model, resulting in the loss of prior knowledge and a waste of computational resources (Silver et al., 2013)2. In contrast, MTPD enables the incremental addition of new tasks by retraining only the student model and leveraging the

preserved knowledge of the existing teacher models. This approach minimizes the computational burden and supports the principle of lifelong learning by allowing the system to continuously learn and adapt to new tasks without discarding previously acquired knowledge.

By exploring the potential of MTPD in addressing task interference, promoting lifelong learning, and optimizing computational resources, this research contributes to the advancement of DRL methodologies and their practical applications in robotics and other domains. The findings from the comparative analysis between the multitask agent trained with MTPD and the multitask agent trained through standard deep reinforcement learning methods will shed light on the effectiveness and advantages of MTPD in achieving these objectives.

Multitask Policy Distillation (MTPD) offers a promising approach to mitigate task interference in Deep Reinforcement Learning (DRL), where agents are trained to perform multiple tasks simultaneously. By distilling knowledge from multiple specialized teacher models into a single student model, MTPD aims to minimize the negative effects of interference. This study aims to empirically test the effectiveness of MTPD in addressing task interference by comparing the performance and training efficiency of a multitask agent trained with MTPD against a multitask agent trained using standard deep reinforcement learning methods.

Related Work

Deep Reinforcement learning

This research is situated within the context of Deep Reinforcement Learning (DRL), a powerful paradigm that combines reinforcement learning and deep learning techniques to enable agents to learn complex behaviors in high-dimensional environments (Mnih et al., 2015)1. DRL has garnered significant attention and has emerged as a promising field with applications in various domains, including video games, real-time strategy games, robotics, and energy management in data centers.

Deep reinforcement learning addresses the challenge of training agents to make sequential decisions in dynamic environments by leveraging the strengths of both reinforcement learning and deep learning. Reinforcement learning provides the framework for learning through interaction with the environment, while deep learning offers powerful function approximators capable of handling high-dimensional input spaces (Mnih et al., 2013)2. By utilizing deep neural networks as function approximators, DRL can effectively learn complex mappings from sensory inputs to actions, enabling agents to exhibit intelligent and adaptive behavior.

The relevance of deep reinforcement learning in the context of this research lies in its ability to tackle complex robotic control problems, such as the task of training a single agent to perform multiple tasks proficiently. Traditional approaches to training models for each specific task suffer from scalability and efficiency issues. Deep reinforcement learning, with its capacity to learn directly from raw sensory inputs, offers a promising alternative for developing end-to-end algorithms that can learn through interaction with the real world. By leveraging the

expressive power of deep neural networks, DRL models can learn representations that capture task-specific knowledge and generalize across different tasks (Levine et al., 2016)[^3^].

In this research, the focus is on exploring the potential of multitask policy distillation within the DRL framework. By utilizing MTPD, the aim is to train a single agent to perform multiple tasks simultaneously, thereby addressing the limitations of training individual models for each specific task. The integration of multitask learning and deep reinforcement learning holds the promise of more scalable, efficient, and adaptable solutions for training agents capable of handling diverse tasks. By leveraging shared representations and distillation techniques, MTPD can enable the student model to effectively leverage the knowledge of multiple teacher models, each specializing in a different task, leading to improved performance and generalization capabilities (Caruana, 1997).

In summary, deep reinforcement learning offers a powerful approach to training agents in complex environments, and within this framework, multitask policy distillation provides a novel method for training a single agent to perform multiple tasks. By leveraging the strengths of both deep learning and reinforcement learning, this research aims to improve the performance and efficiency of DRL models, particularly in the context of multitask learning in robotics.

Multitask learning

Multitask learning was first introduced by Caruana (1997)1. Caruana demonstrated that a single model could learn multiple tasks and even outperform task-specific models due to the shared representations. This work established the foundation for subsequent research into multitask learning.

Over the years, significant strides have been made in the field of multitask learning, particularly with the advent of deep learning. Bengio et al. (2013)[^6^] demonstrated the strength of representation learning in deep architectures, highlighting the benefit of shared representations for multiple tasks. The integration of multitask learning into DRL came later with the works of Parisotto et al. (2015). Parisotto et al. proposed an Actor-Mimic approach where a single network is trained to mimic multiple DRL agents, effectively performing multiple tasks.

Task interference

In the realm of DRL, multitask agents face the issue of task interference, wherein they may exhibit disparate performance levels across multiple tasks. This phenomenon has been studied extensively in the field (Li et al., 2019; Taylor et al., 2020). It is plausible for an agent to excel in a subset of tasks while neglecting or underperforming in others due to variations in task complexity, differences in task rewards, or uneven availability of training data for each task.

Task interference arises as a result of the agent's resource allocation and learning capacity distribution among tasks (Ruder et al., 2019). Owing to the inherent complexity of multitask learning, agents often exhibit a tendency to focus on tasks that offer higher rewards or are comparatively easier to learn. Consequently, tasks with less rewarding or more challenging dynamics may receive less attention during the learning process. [+ problem that if we continue training after convergence to try to raise performance for the neglected task, most often the performance for the rest of the tasks starts to decrease[ask gpt for citation], thus being no straightforward solution to task interference]

Various strategies have been proposed to alleviate task interference and promote balanced learning across multiple tasks. These include reward shaping techniques that adjust the task

rewards to balance their influence (Ruder et al., 2019), curriculum learning approaches that gradually expose the agent to increasingly complex tasks (Bengio et al., 2009), and task prioritization methods that explicitly assign different weights or priorities to individual tasks (Chen et al., 2018). This study investigates how multitask policy distillation can help to alleviate task interference.

Policy distillation

Policy distillation is a technique that allows for the transfer of knowledge from a complex model to a simpler one. It involves training a smaller model, referred to as the student model, to imitate the behavior of a larger and more accurate model, known as the teacher model (Hinton, 2015). The process of policy distillation has gained attention in the field of reinforcement learning as a way to reduce the computational requirements for deploying complex models.

The idea behind policy distillation is to distill the knowledge embedded in the teacher model by training the student model to produce similar output given the same input. The training process typically involves providing the student model with input samples and using the teacher model's predictions as target values for the student model. The student model learns to mimic the teacher model's behavior by adjusting its internal parameters to minimize the difference between its predictions and the teacher's predictions(Rusu, 2016).

Policy distillation has been applied in various domains, such as robotics, game playing, and autonomous vehicles, to name a few. It offers several advantages, including faster inference times and reduced memory requirements. By distilling the knowledge from a large teacher model into a smaller student model, it becomes possible to deploy the student model on resource-constrained devices or in real-time applications (Hadsel, 2016).

Our approach is inspired by these previous works but distinguishes itself by making a specific comparative analysis between multitask learning with standard methods and multitask learning with policy distillation. We hypothesize that this approach can lead to a model capable of performing multiple tasks as effectively as individual models trained through standard DRL, potentially offering a more efficient and robust learning process than standard techniques.

Lifelong learning

In the broader context of machine learning, multitask policy distillation plays an essential role in facilitating lifelong learning. Lifelong learning, or continual learning, is an approach where the model continually learns from a stream of data, adapting to new tasks while retaining knowledge from previous tasks (Chen & Liu, 2018). The traditional approach to expanding a model's repertoire of tasks has been to retrain the entire model with the old and new tasks combined. This method, however, often leads to catastrophic forgetting, where the model unlearns previously acquired knowledge while learning new tasks (McCloskey & Cohen, 1989).

Multitask policy distillation provides a potential solution to this problem. Rather than retraining the entire model, only the student model needs to be retrained when a new task is introduced. The teacher models, which hold the expertise in individual tasks, remain intact. A new expert can be trained for the new task and added to the pool of teacher models. Then, the student model is retrained to distill the knowledge from all the teacher models, old and new. This process retains the knowledge of the original tasks, avoiding catastrophic forgetting (Silver et al., 2013). This approach not only preserves the knowledge of the teacher models but also ensures that the student model continually adapts to new tasks. It enables the system to expand its repertoire of tasks without needing to be trained from scratch each time a new task is introduced.

Thus, multitask policy distillation supports the principle of lifelong learning by allowing the system to learn new tasks while maintaining proficiency in previous tasks (Kirkpatrick et al., 2017).

Methods

Software

The experiments from this study were performed using multiple Reinforcement Learning (RL) toolkits and libraries. The functionality of the Stable Baselines (Hill, 2018) library was used for developing the model architecture and training the models. Stable Baselines is a Python library that provides a collection of high-quality, reliable, and well-documented implementations of various RL algorithms.

The environments used to train the agent were developed using the OpenAI Gym toolkit (Brockman, 2016). OpenAI Gym provides a standardized set of environments and interfaces for developing and evaluating RL algorithms. More specifically, we used OpenAI Gym's implementation of the Ant-v4 Mujoco environment. To run the simulations for the Ant-v4 robotics environment, we used the Mujoco physics engine (Todorov, 2012). Mujoco is a dedicated physics engine primarily focused on simulating articulated systems for robotics and biomechanics research.

The Ant-v4 environment

The environment used in this study was mainly based on the OpenAI gym implementation of Ant-v4. The Ant-v4 environment simulates a four-legged ant-like robot with a complex musculoskeletal system. Agents must master the control of its multiple joints and

actuators to achieve stable locomotion in a 3D world. This environment provides a platform to develop and evaluate reinforcement learning algorithms that focus on the challenges of coordinating the ant's body composition and movement. The observation space and reward function of the original implementation were modified for the purposes of the study.

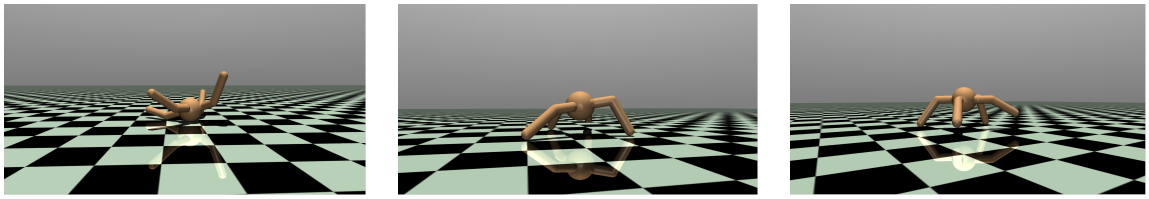

Figure 1: The Mujoco Ant-v4 environment. Here, it can be observed an ant stuck on its back, an ant struggling to get up and an ant walking.

Observation space

The original observation space of the Ant-v4 environment consists of positional values and velocities of different body parts of the ant. The observations are provided as a one-dimensional array with 111 elements. The elements of the observation space include the following:

Elements 1 to 5: the positional and orientational values of the torso (center) of the ant. Elements 6 to 13: the angles between the torso and the various leg parts. Elements 14 to 19: the velocities of the torso in the x, y, and z coordinates, as well as angular velocities. Elements 20 to 27: the angular velocities of the leg parts. The remaining 84 elements (elements 28 to 111) represent contact forces applied to the center of mass of each of the links. These forces include external forces (force x, y, z) and torques (torque x, y, z) and are associated with the ground link, torso link, and the three links of each leg.

On top of the 111 long vector of observations of the original implementation, a one-dimensional vector with 2 units was added to encode the task id - the first position of the vector represents the desired velocity of the agent on the x axis and the second position represents the velocity on the y axis. For example, a task vector of [1,0] means the task is to walk to the right, while a task vector of [0,-1] means the task is to walk backward.

Action space

The action space of the Ant-v4 environment is a one-dimensional array with 8 elements. Each element of the action array corresponds to the torque applied at a specific hinge joint in the ant's body. The range of each action element is between -1 and 1. The action space allows controlling the coordination of the ant's four legs by applying torques on the eight hinges connecting the leg links and the torso.

Reward function

The reward function of the agent has 4 components. Forward reward: a reward for moving in the direction encoded by the task vector. Healthy reward: incentivises the agent to not lay on the floor by being rewarded when it keeps its torso between 0.2 and 1.0 on the z dimension. Control cost: a negative reward for penalizing the ant if it takes actions that are too large. Contact cost: a negative reward for penalizing the ant if the external contact force is too large.

Proposed approach

Two approaches to multitask learning will be compared in this study. The first approach uses standard DRL methods to train a multitask agent. More specifically, the model optimizes its

parameters with Proximal Policy Optimization (PPO), based on episodes performed in a collection of environments sampled from the set of desired environments pool.

The second approach uses multitask policy distillation. In this framework, a collection of expert models is trained in a singletask setting through standard DRL methods as previously described. Then, a multitask student network is trained on episodes collected from the teachers' performance using the policy distillation algorithm Proximal Policy Distillation (PPD).

Standard multitask and teacher learning framework

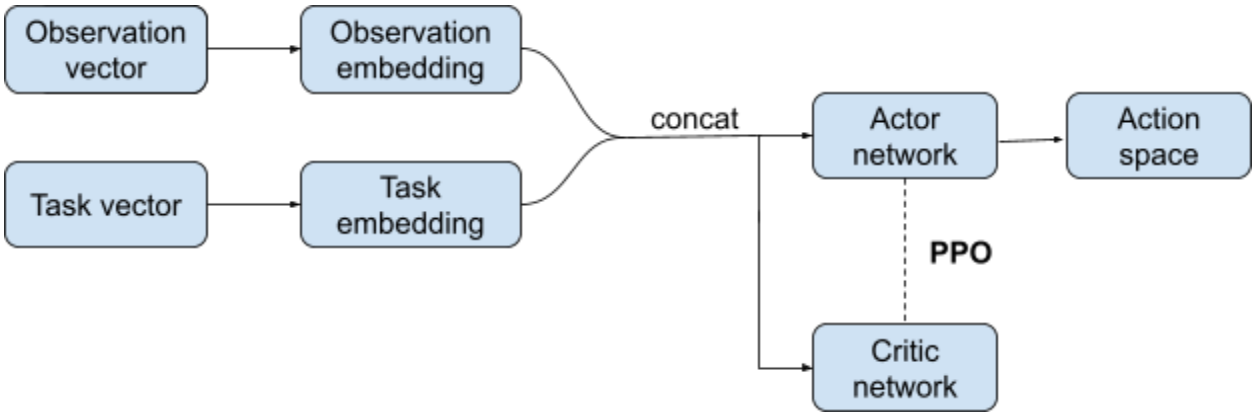

Figure 2: The learning system for standard multitask and teacher learning

The model's input is a vector with the agent's observations in the environment and the task vector. These two vectors are passed through multi-layer perceptrons that play the role of encoders. We increase the dimensionality of the task embedding from a 2-dimensional vector to a 32-dimensional vector to address the significant difference in dimensionality between the environment observation vector (111 elements) and the task vector (2 elements). This transformation of the state and task embedding into a higher-dimensional vector serves as a precautionary measure to prevent the policy network from potentially disregarding the information contained within the task vector due to its relatively small presence in the input

layer. The two embeddings returned by the encoders (observation embedding with 128 elements and task embedding with 32 elements) are concatenated and passed as input to the actor and critic networks that optimize their parameters with the Proximal policy optimization learning algorithm.

Neural network architectures

The same network architecture has been used in standard multitask learning and multitask learning with policy distillation. That is, an observation embedder with 2 hidden layers of size 256 and 128, a task embedder with 2 hidden layers of size 16 and 32, an actor network and critic network with 1 hidden layer of 128 units. The choice of architecture was based on the minimum number of layers and units required for the standard multitask model to learn a successful policy. Multiple architectures have been tested on the teacher models and the one that performed the best was kept. The teachers differ by having a smaller observation embedder with 2 hidden layers of size 128 and 128 and the actor and critic networks containing 2 layers, 64 units each.

Proximal Policy Optimization

Throughout this study, we employ the Proximal Policy Optimization (PPO) algorithm (Schulman, 2017) as a key component of our reinforcement learning framework. PPO is a state-of-the-art algorithm widely used for training agents in sequential decision-making tasks. It belongs to the family of policy optimization methods and is specifically designed to address the challenge of achieving stable and efficient training in reinforcement learning. PPO leverages a trust region approach to ensure controlled updates of the policy, preventing large policy changes that may lead to unstable training. By iteratively updating the policy through multiple epochs,

PPO strikes a balance between exploration and exploitation, enabling the agent to learn optimal policies in complex environments. This algorithm has demonstrated remarkable performance on a wide range of tasks, making it a reliable choice for our research endeavors.[PPO paper citation]

It is important to note that the specific DRL algorithm itself is not the focus of investigation. Rather, it serves as a constant to facilitate a fair and comparative evaluation of the different systems and methodologies explored.

Policy distillation algorithm

The policy distillation algorithm is a variant of the Proximal Policy Distillation algorithm (Spigler, 2023), adapted for a multitask learning setting. The policy distillation algorithm operates in the following manner. A student agent undergoes training similar to standard Proximal Policy Optimization (PPO) methodology. The student agent engages in an iterative process where it collects rollout batches of trajectories by interacting with the task environment using its current policy, denoted as pi_theta_k, and subsequently performs policy updates. During the PPO iterations, the updates are computed through minibatch gradient descent, utilizing samples from the rollout buffer for a fixed number of epochs. The update process can be represented as:

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a\sim\pi_{\theta_k}} \left[L_{\text{PPO}}(\theta) + \frac{1}{2} L_{\text{value}}(\theta) - \lambda \text{KL}(\pi_\theta \| \pi_{\text{teacher}}) \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon \right) \right]$$

In the equation above, L_{PPO} represents the standard PPO-clip loss, while L_{value} denotes the mean-squared error for the value function. The policy π_{θ_k} corresponds to the policy employed in the previous PPO iteration, which was utilized for collecting the rollout buffer. Notably, unlike the original implementation of Proximal Policy Distillation (PPD), we compute the KL divergence of π_θ with respect to π_{teacher} rather than the opposite direction. This modification

is motivated by the desire to promote more-seeking behavior rather than mean-seeking behavior, which is particularly advantageous in a multitask learning scenario.

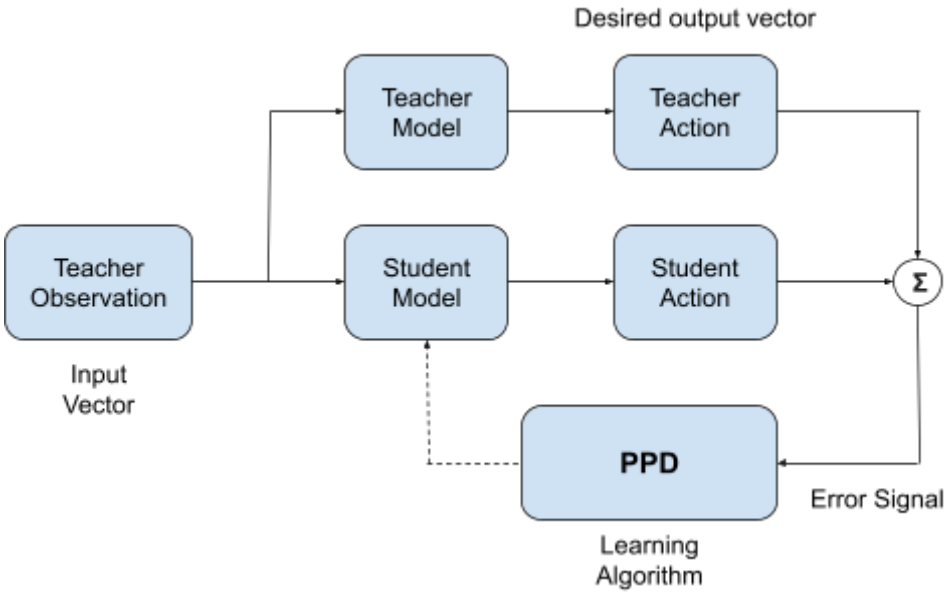

Figure 3: Policy Distillation general framework

Multitask policy distillation framework

Policy distillation operates in a supervised learning setting, where the student network plays the role of a function approximator trying to map a teacher's observation to its action. In the student's learning process, exploration, rewards, and interactions with the environment are not needed. The student receives as input an observation from a teacher and outputs a prediction of the teacher's action distribution. Its parameters get updated based on the difference between

the student and teacher action distributions.

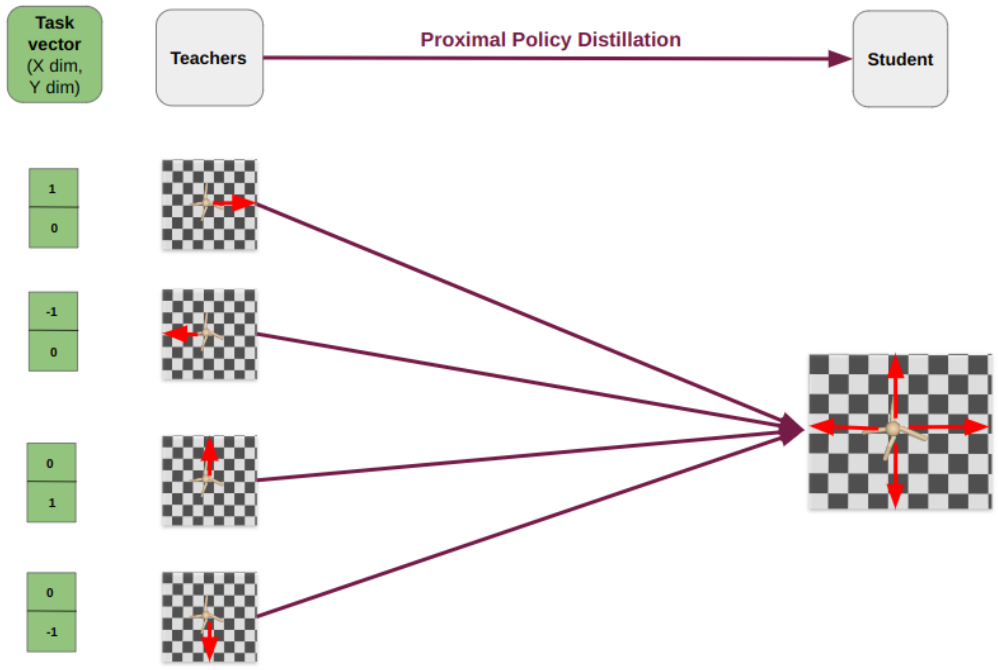

Figure 4: Multitask policy distillation in the Ant-v4 environment.

Experiment

Two base multitask agents were trained in the Ant-v4 environment to check for how many training timesteps the models perform best. The base model for the standard multitask agent was trained for 25 million timesteps and it was found that the optimal number of timesteps for training is around 10 million. The base model for the agent trained with MTPD was trained for 10 million timestepsteps. After finding the optimal number of training timesteps, 6 other agents have been trained for an optimal duration, 3 for each condition. All 6 agents have been trained to walk in 4 directions (forward, backward, right and left).

Results

Task	Teacher reward	Standard MT reward run #1	Standard MT reward run #2	Standard MT reward run #3	MTPD reward run #1	MTPD reward run #2	MTPD reward run #3
↑	4924 ±328	2514 ±1179	1608 ±1128	3052 ±1112	3550 ±510	2796 ±166	3679 ±853
↓	4700 ±735	**812** ±663	3459 ±921	3322 ±854	2352 ±921	3329 ±597	3641 ±847
→	5201 ±418	2856 ±1318	**1403** ±892	2607 ±1158	2883 ±401	3613 ±158	2980 ±1357
←	5092 ±509	2545 ±980	2728 ±1097	3431 ±1067	2064 ±339	2845 ±691	3064 ±117
Mean reward	-	2182	2299	3103	2712	3145	3341

Table 1: The performance table including the mean rewards received by an agent for the 4 tasks (walking forward, backward, to the left and to the right) over 30 episodes ± one standard deviation around the mean.

Table 1 contains the comparison in performance between standard multitask learning and multitask learning with policy distillation. In standard MT run #1 and #2, for the task of walking backward and to the right, the effects of task interference can be seen in the performance of the standard multitask agent. The agent learns a successful policy for all the other tasks, but at the cost of the task of walking backward being neglected (the agent gets almost half the reward for walking backwards than it does for any of the other directions.

The most remarkable finding from the performance table is the robust performance of the model trained with policy distillation. The model learns a successful policy for every direction, being seemingly immune to task interference.

Discussion

In this thesis project, we aimed to address the issue of task interference in Deep Reinforcement Learning (DRL) by exploring the application of Multitask Policy Distillation (MTPD) techniques. Task interference occurs when training agents to perform multiple tasks simultaneously, leading to reduced performance on individual tasks. Our investigation focused on comparing the performance and training efficiency of a standard multitask agent trained using conventional methods with a multitask agent trained using MTPD.

The results of our experiments demonstrate the effectiveness of MTPD in mitigating task interference and improving overall performance. The standard multitask agent showed significant performance degradation for the task of walking backward, indicating the presence of task interference. In contrast, the multitask agent trained with MTPD exhibited robust performance across all tasks, with no task being neglected. This finding highlights the ability of MTPD to effectively leverage the expertise of individual teacher models, enabling the student model to learn from multiple sources of knowledge without suffering from detrimental interference effects.

The successful performance of the multitask agent trained with MTPD can be attributed to the knowledge distillation process, where the student model learns from multiple specialized teacher models. By distilling knowledge from these teacher models, MTPD facilitates effective knowledge transfer and promotes robust performance across multiple tasks. Furthermore, the

MTPD approach aligns with the principle of lifelong learning by enabling the incremental addition of new tasks without discarding previously acquired knowledge. This contributes to the scalability and adaptability of the agent, as it can continuously learn and adapt to new tasks while retaining proficiency in previous tasks.

Our findings have important implications for the advancement of DRL methodologies and their practical applications in robotics and other domains. The MTPD framework offers a promising solution to the challenge of task interference, enabling agents to effectively learn and perform multiple tasks. By leveraging shared representations and distillation techniques, MTPD enhances robust training across tasks, promotes lifelong learning, and optimizes computational resources. These benefits make MTPD an attractive approach for developing more scalable, efficient, and adaptable solutions in multitask learning scenarios.

Conclusion

This study contributed to the larger research area of lifelong learning by making a comparative analysis between multitask learning with standard deep reinforcement learning and multitask learning with policy distillation. The results of this investigation were yet another confirmation of the susceptibility of standard multitask learning in DRL to fall prey to task interference. The results also indicate that policy distillation in a multitask DRL setting allows the model to build more robust representations and avoid catastrophic forgetting.

The findings of this study have several implications for the field of DRL and lifelong learning. First, MTPD offers a promising approach to address the challenge of task interference in multitask learning settings. By leveraging the expertise of multiple teacher models, MTPD enables the student model to effectively learn from their collective knowledge, leading to

improved performance and generalization capabilities. This approach has the potential to enhance the scalability and efficiency of DRL models, particularly in complex robotic control tasks. Second, the application of MTPD aligns with the concept of lifelong learning. By incrementally adding new tasks to the multitask agent without retraining the entire model, MTPD enables continuous adaptation to new tasks while retaining previously acquired knowledge. This approach minimizes computational burden and supports the principle of lifelong learning, allowing the system to continuously learn and adapt without discarding valuable knowledge. MTPD provides a more efficient and robust learning process compared to traditional methods of expanding a model's task repertoire. Future research in this area can explore several directions. First, investigating the scalability of MTPD to more complex environments and tasks would provide insights into the applicability of the approach in real-world scenarios. Additionally, exploring different variations of policy distillation algorithms and architectures could further enhance the performance and efficiency of multitask agents.

In conclusion, this study has demonstrated the effectiveness of Multitask Policy Distillation in mitigating task interference and enabling robust multitask learning. By distilling knowledge from multiple teacher models, MTPD promotes efficient knowledge transfer and enhances the performance of multitask agents. The findings contribute to the advancement of DRL methodologies and their practical applications in robotics and other domains. The concept of lifelong learning, supported by MTPD, holds promise for developing more adaptable and efficient learning systems.

Acknowledgements

First, I owe every half decent thing I've done during this bachelor program to my family who encouraged me to follow my intuition and go on this wild adventure during which I encountered things greater than I could ever imagined.

Second, I want to thank my scientific mentors. Dr. Travis Wiltshire - the first supervisor I had in my scientific career, he believed that I could weaponize my insanity and showed his confidence in my abilities by giving me open research problems to work on. I am proud of my work. It may sound grandiose, but it was a magical experience to be the first person in human history to figure this particular puzzle out.

Dr. Marijn van Wingerden - he took a big risk on me by trusting me with an incredibly hard problem to solve and giving me a lot of independence in figuring it out. During my time working with him, I became convinced that a life in science is for me.

Dr. Giacomo Spigler - if I am to become a university professor, I want to be like dr. Spigler. His enthusiasm and drive in unapologetically working towards AGI is something singular that always had a revitalizing effect while working with him. He has been a great supervisor and our best work together is yet to come.

This thesis is also a product of my awesome CSAI crew that has been involved during the brainstorming process and provided emotional support during the endless hours of debugging. This crew includes Udesh "The Tortoise" Habaraduwa - the bigger brother I never had; the dynamic duo composed Giulio Tosato and Zan Persic; Dennis Lindberg - among the finest gym buddies encountered; Cesare Maria Dalbagno who sent me periodical italian recipes that kept me nourished throughout working on the project; Francesco, a.k.a "Fuma", a.k.a. "the italian giant" Fumagalli and Adiel, a.k.a "the campus rabi", Goldin that always had a wise word to share.

References

Lastname, C. (2008). Title of the source without caps except Proper Nouns or: First word after colon. *The Journal or Publication Italicized and Capped*, Vol#(Issue#), Page numbers.

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., & Hassabis, D. (2013). Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2013). Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In Proceedings of the 26th annual international conference on machine learning (ICML) (pp. 41-48).

Chen, T., Liu, S., Xiao, H., & Wang, T. (2018). LSTNet: Learning long-term dependencies in streaming time series. In Proceedings of the 31st AAAI conference on artificial intelligence (pp. 6770-6777).

Li, Y., Zhang, T., Chen, Y., & Liu, D. (2019). Task-agnostic meta-learning for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 11711-11720).

Ruder, S., Peters, J., & Schaal, S. (2019). Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research, 20(72), 1-81.

Taylor, J., Singh, S. P., & Levine, S. (2020). Variance-based regularization for learning-to-learn. In Proceedings of the 34th AAAI conference on artificial intelligence (pp. 11604-11611).

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., & Mannor, S. (2016). A deep hierarchical approach to lifelong learning in minecraft. In Proceedings of the 30th AAAI conference on artificial intelligence (pp. 1814-1820).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., ... & Hadsell, R. (2016). Policy distillation. arXiv preprint arXiv:1511.06295.

Chen, Z., & Liu, B. (2018). Lifelong Machine Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-207.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. Psychology of Learning and Motivation, 24, 109-165.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., 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., &

Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.

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). Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13), 3521-3526.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. ArXiv Preprint ArXiv:1606.01540.

E. Todorov, T. Erez and Y. Tassa, "MuJoCo: A physics engine for model-based control," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012, pp. 5026-5033, doi: 10.1109/IROS.2012.6386109.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017). Proximal Policy Optimization Algorithms.. CoRR, abs/1707.06347.

Giacomo Spigler (2023). Private communication.

Appendix

All the code used during the project is freely available on this github page:

https://github.com/AndreiLix/WORKING_folder_thesis