Contextual Online Imitation Learning (COIL): Using Guide Policies in Reinforcement Learning

Alexander Hill, Marc Groefsema, Matthia Sabatelli, Raffaella Carloni[*] and Marco Grzegorczyk[*]
Faculty of Science and Engineering, Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, The Netherlands

Keywords: Machine Learning.

Abstract: This paper proposes a novel method of utilising guide policies in Reinforcement Learning problems; Contextual Online Imitation Learning (COIL). This paper demonstrates that COIL can offer improved performance over both offline Imitation Learning methods such as Behavioral Cloning, and also Reinforcement Learning algorithms such as Proximal Policy Optimisation which do not take advantage of existing guide policies. An important characteristic of COIL is that it can effectively utilise guide policies that exhibit expert behavior in only a strict subset of the state space, making it more flexible than classical methods of Imitation Learning. This paper demonstrates that through using COIL, guide policies that achieve good performance in sub-tasks can also be used to help Reinforcement Learning agents looking to solve more complex tasks. This is a significant improvement in flexibility over traditional Imitation Learning methods. After introducing the theory and motivation behind COIL, this paper tests the effectiveness of COIL on the task of mobile-robot navigation in both a simulation and real-life lab experiments. In both settings, COIL gives stronger results than offline Imitation Learning, Reinforcement Learning, and also the guide policy itself.

1 INTRODUCTION

Imitation Learning (IL) is a technique of solving Reinforcement Learning (RL) problems where instead of directly training an agent using the rewards collected through exploration of an environment, an expert guide policy provides the learning agent with a set of demonstrations of the form $(s,\, \pi_{\text{guide}}(s))$ where $\pi_{\text{guide}}(s)$ is the action the guide policy takes in state s (Halbert, 1984). The guide policy π_{guide} is defined as any mapping from states in the environment to actions that exhibit the desired behavior. The agent then tries to learn the optimal policy by imitating the expert's decisions. In recent years, research in the field of Imitation Learning has become very topical, with many new methods being developed such as Confidence-Aware Imitation Learning (Zhang et al., 2021) and Coarse-to-Fine Imitation Learning (Edward, 2021), and Jump-Start Reinforcement Learning (Uchendu et al., 2022), and Adversarially Robust Imitation Learning (Wang et al., 2022). Furthermore, applications of Imitation Learning are very broad, ranging from self-driving car software (Chen et al., 2019) and industrial robotics (Fang et al., 2019) , to financial applications such as stock market trading (Liu et al., 2020). In general, Imitation Learning is a powerful technique when one has access to an expert guide policy able to show the desired behaviour in the environment. Influential examples of this methodology are Behavioral Cloning (Ross et al., 2011), DAgger (Pomerleau, 1998), and Generative Adversarial Imitation Learning (Ho and Ermon, 2016).

Imitation Learning methods can utilise expert guide policies in many different ways, such as estimating the underlying reward function of a task directly from guide policy demonstrations such as in Inverse Reinforcement Learning, through minimizing a carefully crafted loss function representing the relative distance between the learning policy and the guide policy, or even through using the guide policy for a more sophisticated sampling of states for the learning agent (Uchendu et al., 2022). Contextual Online Imitation Learning (COIL) utilises the guide policy in a different way. Instead, the agent's observations of the environment are modified to include the guide policies 'suggestions' for what action to take next. From here, the agent is able to learn complex relationships between the guide policies actions and the

[*]These authors contributed equally to this work and share last authorship

Hill, A., Groefsema, M., Sabatelli, M., Carloni, R. and Grzegorczyk, M.
Contextual Online Imitation Learning (COIL): Using Guide Policies in Reinforcement Learning.
DOI: 10.5220/0012312700003636
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 3*, pages 178-185
ISBN: 978-989-758-680-4; ISSN: 2184-433X
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

underlying state of the environment. This allows for the agent to extract value from the guide policy on a *state by state basis*, which is fundamentally different to most current methods of Imitation Learning (Osa et al., 2018). In COIL, the agent can learn for itself exactly *how* and *where* in the environment is advantageous to mimic the guide policy. This potentially allows researchers and engineers to use guide policies that only offer high rewards in a strict subset of states in the environment. Furthermore, COIL can be easily generalised to using multiple guide policies.

In COIL, at each time step t and state s during training of the agent, the action of the guide policy $\pi_{\text{guide}}(s)$ is provided as an additional observation to the agent, along with any other useful observations that are given by the environment. This proposed methodology is shown in Fig. 1.

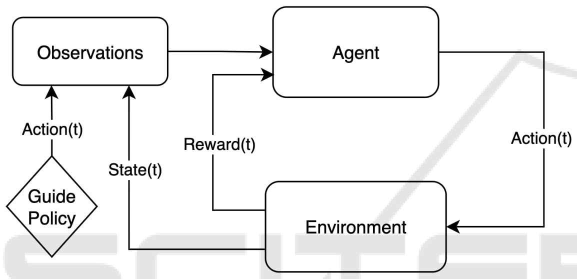

Figure 1: Illustration of the Contextual Online Imitation Learning (COIL) methodology.

Additionally, if we have a set of guide policies $\pi_{\text{guide}_1}, \ldots, \pi_{\text{guide}_n}$, then we can generalise this framework to include the actions of all guide policies at time step t as observations to the learning agent.

Now that the agent is able to see what the guide policy would do given the current context of the environment, it is able to learn through exploration how best to use this information in order to maximise its expected future reward. In other words, the agent is able to learn for itself in which states within the environment it should 'listen' to the guide policy's suggestion, and in which states within the environment it should 'ignore' the guide policy's suggestion. This methodology is therefore a form of contextual Imitation Learning, as the agent is *not* directly encouraged to follow the actions of the guide policy in all possible states. Furthermore, this method is *online* as it is trained using continuous interaction with the environment. Therefore, we named this method Contextual Online Imitation Learning.

It is important to note that COIL makes no explicit assumption about the type of Reinforcement Learning algorithm equipped with this new observation to the agent. COIL can be used equally well with Value-based methods as with Policy-gradient methods.

A major advantage of COIL in comparison to other methods of utilising guide policies is that there is no requirement for the guide policy to be an expert at all states within the learning environment. Even if the guide policy is only able to exhibit success in some strict subset of the total states within the environment, the agent is able to learn this fact through enough exploration of the environment, and then appropriately use this information to its advantage. Additionally, if the guide policy exhibits only moderate success across all states in the environment, the agent could also learn to *fine-tune* the actions of the guide policy in order to maximise reward.

This method can therefore be seen from two perspectives. The first is that COIL allows us to take a non-optimal hand-crafted guide policy, and use Reinforcement Learning methods to encode additional desired behaviours into the policy via the reward function. The other perspective is that this method allows us to utilise existing guide policies to aid the agent during training, and in the process the agent can learn to use the suggested actions of the guide policy in whatever way maximises the total expected reward. In COIL, there is no explicit penalty in deviating from the guide policy, this gives the Reinforcement Learning agent complete flexibility to use the guide policy in complex and situational ways.

2 CONTEXTUAL ONLINE IMITATION LEARNING

In many tasks in Reinforcement Learning there are multiple competing objectives that determine the overall success of a given policy. Let $\Omega_1, \ldots, \Omega_n$ be the set of *objectives* that we want the agent to learn during training, where n is the total number of objectives.

Then let Ω_{I} be the set of objectives that the guide policy is capable of achieving near-optimal performance. This guide policy could be hand-crafted or extracted from data via Imitation Learning. We have that $\text{I} \subset \{1, \ldots, n\}$.

Now, we shall denote the part of the reward function associated with the objective Ω_i at time step t as $r_{\Omega_i}(s_t, a_t)$ where s_t is the state at time t and a_t is the action at time t. Then, the overall reward function taking into account each objective is defined as:

$$r(s_t, a_t) = \sum_{i=1}^{n} C_i \cdot r_{\Omega_i}(s_t, a_t) \tag{1}$$

and additionally the reward function that the guide policy achieves near-optimal performance is given by

$$r_{\Omega_I}(s_t, a_t) = \sum_{i \in I} C_i \cdot r_{\Omega_i}(s_t, a_t) \tag{2}$$

where C_i is the coefficient which determines the priority of objective Ω_i. Some objectives are more important than others, therefore these coefficients are very important for encoding the desired behavior for the agent to learn.

So far, the policy of the agent π has been a function of the current state s and action a. In practice, the state is represented by a finite dimensional vector of data points $s = \{s_1, \ldots, s_m\}$ which gives the agent as much information about the context of the environment as possible. Therefore, we have

$$\pi(s,a) = \pi(s_1, \ldots, s_m, a) \tag{3}$$

and in COIL we also provide the action taken by the guide policy π_{guide} in that same state:

$$\pi(s,a) = \pi(s_1, \ldots, s_m, \pi_{\text{guide}}(s), a). \tag{4}$$

Many modern policy-based Reinforcement Learning algorithms use neural networks to model the policy of the agent. A result of the universal approximation theorem is that neural networks are capable of approximating any continuous function given enough units within the hidden layers (Hornik et al., 1989). It is consequently possible that a neural network with sufficient training can learn the simple rule of 'copying' a single input parameter:

$$\pi(s,a) = \pi(s_1, \ldots, s_m, \pi_{\text{guide}}(s), a) \approx \pi_{\text{guide}}(s) \tag{5}$$

where this approximation can become arbitrarily accurate (Hornik et al., 1989). In this case the policy of the agent trained via COIL π_{COIL} is capable of achieving at least the same performance as the guide policy π_{guide}. However, this result is not guaranteed as in many cases training data can be sparse or access to an adequate training environment can be limited. If Q-learning is used, then it is guaranteed that if the guide policy is not the optimal policy then the COIL agent will surpass the guide policy in expected reward given sufficient exploration of the environment. This is a result of Watkins' proof of Q-learning convergence in 1992 (Watkins and Dayan, 1992).

Following this, it is interesting to investigate whether the policy learned via COIL can be shown to achieve better results than the policy learned via Reinforcement Learning methods that do not utilise the guide policy. The success of COIL is likely dependent on the environment, the task, and the quality of the guide policy utilised. Later in this paper we will apply COIL to various applications and clearly demonstrate the capabilities of the new method.

3 EXPERIMENTAL SET UP

3.1 Simulation

A mobile robot simulation was designed in `Python3` using the package `Pygame` in order to test the effectiveness of COIL. In the simulation, the car is able to drive around on a 2D plane where its movement is governed by a dynamic model referred to as a bicycle model. Within the simulation, left cones and right cones specify the sides of a track for the car to drive through. An example of such a track can be seen in Fig. 2. In order to turn this simulation into a

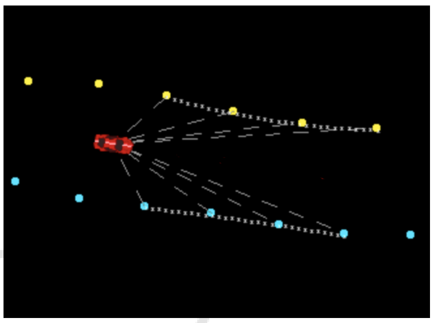

Figure 2: Image of the mobile robot simulation programmed using `Python`.

Reinforcement Learning problem, an action set $\mathcal{A}$, an observation set O, and a reward function $\mathcal{R}$ must be decided.

To construct the observation, a cubic spline was first applied to both the left side and right side of the car's track. This cubic spline acts as a *boundary estimation* for each side of the track. This cubic spline boundary estimation can be seen in Fig. 2 as grey dotted lines between detected cones. For more information on cubic splines the reader is directed to Durrleman and Simons paper on the topic (Durrleman and Simon, 1989).

Finally, to turn the boundary estimates into a fixed-size observation for the agent, we can take n equidistant sample points along each side of the track. For this practical implementation $n = 5$ was an appropriate choice given the computational constraints. Therefore, we end up with 10 sample points in total, each of which is comprised of two pieces of information; the distance to the car r, and the relative angle of the sample point with respect to the car axis, θ. These 20 values are what we shall use as our observation to the agent, thus we have

$$O = \{r_1, \theta_1, \ldots, r_{10}, \theta_{10}\}$$

where the first five (r, θ) pairs represent the boundary sample points on the left side of the track, and the last five (r, θ) pairs represent the boundary sample points on the right side of the track.

For the set of actions $\mathcal{A}$ we simply allow the agent to decide the steering angle (in degrees) of the car in the interval $[-80, 80]$. Thus

$$\mathcal{A} = [-80, 80].$$

The mobile robot in our simulation has three main objectives: Ω_1: Survival, Ω_2: Speed, and Ω_3: Smoothness. Four properties were chosen in order to encode these objectives within the reward function:

i) A positive reward for successfully progressing along the track (Ω_1)
ii) A negative reward for crashing (Ω_1)
iii) A positive reward for driving fast (Ω_2)
iv) A negative reward for unstable driving (Ω_3)

Using these four motivations, our reward function to impose these objectives can be:

- $r_{\Omega_1} = 70 \cdot \mathbb{1}\{\text{Car finishes track}\} - 100 \cdot \mathbb{1}\{\text{Car crashes}\} + 2.5 \cdot \mathbb{1}\{\text{New cone detected}\}$
- $r_{\Omega_2} = -0.8T \cdot \mathbb{1}\{\text{Car finishes track}\}$
- $r_{\Omega_3} = - \parallel \theta - \tilde{\theta} \parallel$

where $\mathbb{1}$ is the indicator function, T is the time taken for the car to finish the track, θ is the direction the car is facing, and $\tilde{\theta}$ is a exponentially weighted moving average of previous car directions. The coefficients superseding the indicator functions in the reward function (70, -100, 2.5, -0.8) were decided in order to incentivise the agent to learn the desired behaviour with respect to the three competing objectives. Our final reward function for the entire task is then defined as

$$r = \sum_{i=1}^{3} r_{\Omega_i} \ .$$

Following this, in order to use COIL with this simulation, we needed a guide policy π_{guide}. Consequently, it was necessary to design a suitable guide policy. Thus a guide policy was created that aims to keep the car as 'safe' as possible at all times. The guide policy first calculates an estimate for the centre path that goes through the middle of the track, and then takes the steering angle best suited to follow this path. We can call this guide policy the 'safety policy' as it encodes our safety objective Ω_1.

This guide policy is a great example of a non-expert guide policy as it achieves optimal performance on only certain subsets of the state space. If the track is straight, driving directly down the middle optimises all three objectives, but on corners it exhibits sub-optimal performance for the speed objective Ω_2 as 'corner-cutting' can reduce the lap time.

Proximal Policy Optimisation (PPO), designed in 2017 by OpenAI (Schulman et al., 2020), is a powerful Reinforcement Learning technique which has seen significant success in solving Reinforcement Learning tasks with greater stability and reliability than previous state of the art algorithms (Wang et al., 2020; Yu et al., 2021). For this reason PPO (with MLP network architecture for both actor/critic networks, 2 layers of 64 each, Adam optimizer, learning rate of 10^{-5}, and tanh activation function) was chosen as the Reinforcement Learning algorithm for our agent, and also when applying the COIL methodology.

To train the RL and COIL agents, 10 tracks were created in the simulation. For training, 7 of the tracks were provided to the agent randomly during each episode of the simulation, this was to ensure the agent would not just learn memorise a single track. An episode of our simulation is defined by 3 full laps around the randomly selected training track. The remaining 3 tracks were reserved for testing generalisation ability. An example of one of the simulated tracks can be seen in Fig 3.

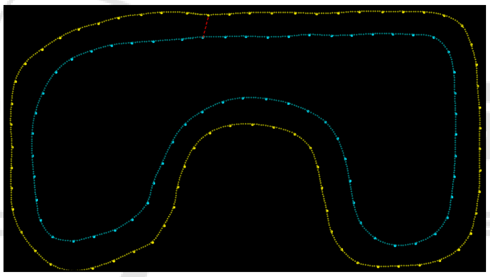
Figure 3: Example of one of the seven tracks used to train the Reinforcement Learning agent to drive autonomously.

As a final mobile robot navigation method to compare with COIL, an Offline Imitation Learning (OIL) method is applied to mimic the behavior of the guide policy discussed previously, in particular, Behavioral Cloning is applied. As the behavior of the guide policy is simple, well-defined, consistent across the entire state space, and by design a non-expert policy, we found that it is sufficient in this case to use Behavioral Cloning over a more complex method. However, in future research more advanced Offline Imitation Learning algorithms could also be compared to COIL.

At each time step t, the observation (boundary estimate samples) $(r_1, \theta_1, \ldots, r_{10}, \theta_{10})$ is fed into a Machine Learning model, with the objective of predicting the action of the guide policy $\pi_{\text{guide}}(s)$. In order to train the model, a dataset of demonstrations was collected by the guide policy driving around each of the training tracks for 3 laps, and at each time step t the sample

$$\left\{ (r_1(t), \theta_1(t), \dots, r_{10}(t), \theta_{10}(t)) \ , \ \pi_{\text{guide}}(s_t) \right\}$$

was stored. In total, this generated a dataset with 16510 data points. The Machine Learning model that we used was a Random Forest Regression model. This was chosen as it offered a very low Mean Square Error (0.01489) for this task compared to other methods we tested. For details on Random Forests the reader is directed to the original 2001 paper by Breiman (Breiman, 2001).

The `Python` code for this mobile robot simulation can be found on Github at: https://github.com/alex21347/Self_Driving_Car. Furthermore, the simulation was ran using a 2.3 GHz Intel Core i5 CPU, an Intel Iris Plus Graphics card, and 8GB of RAM.

3.2 Mobile Robot

COIL was also tested on a real life mobile robot in order to answer the following research questions:

i) *Does COIL still work effectively with a mobile robot with real sensors?*

ii) *How robust are the algorithms with regards to transfer learning?*

Transfer learning is the method of using a pre-trained model on a different task than the one that was used for training the model (Pan and Yang, 2009). Transfer learning has become an important research topic in recent years (Ruder et al., 2019; Wan et al., 2021; Aslan et al., 2021), and in this paper COIL will be also tested in its robustness to Transfer learning.

The robot can be seen in Fig. 4 below. On top of the robot there is a single Lidar detector (RPLidar A1M8) which is the only sensor available for autonomous driving.

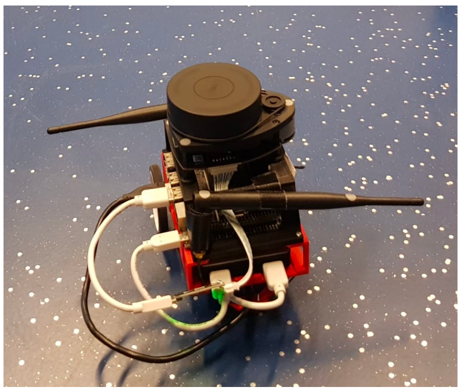

Figure 4: Photograph of the robot car used for comparing autonomous driving algorithms.

Using only this robot and the Lidar sensor, the self-driving algorithms trained in the simulation can now be compared using the task of navigating a real-life track made out of cones. This test track can be seen in Fig. 5.

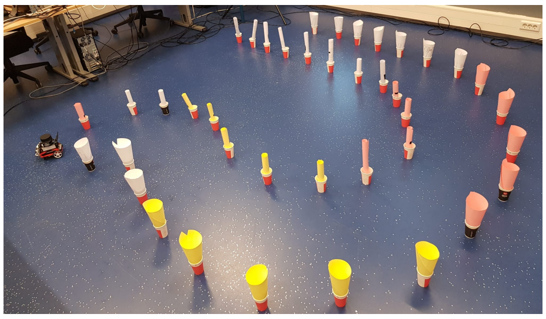

Figure 5: Photograph of the test track used for comparing autonomous driving algorithms.

4 RESULTS

The COIL and RL agents were trained in the simulation over 6 runs for 2 million time steps each, and the maximum average episode reward[1] achieved during training was recorded for each run. The average maximum reward for COIL and RL can be seen in Table 1.

Table 1: Table of training results for Reinforcement Learning (RL) and Contextual Online Imitation Learning (COIL).

Method	Mean Max Reward
COIL	**477.6**
RL	414.2

It can be seen in Table 1 that COIL achieves much more advantageous policies during training, giving a 15.3% increase in the average max reward. A one-sided t-test on the mean maximum reward between COIL and RL gave $p = 0.034$, indicating statistically significant improvement over regular RL during training.

An additional trait that was noticed during training was that the COIL agent consistently trained ‘quicker’ than the Reinforcement Learning agent, especially in the earlier stages of training. This can be seen in Fig. 6.

Therefore, there is evidence that by providing the actions of the guide policy to the agent during training in COIL, we achieve both quicker training and with a higher maximum reward. In the next step of our analysis, the different algorithms are tested on 3 unseen

[1]average over all training episodes in the given iteration.

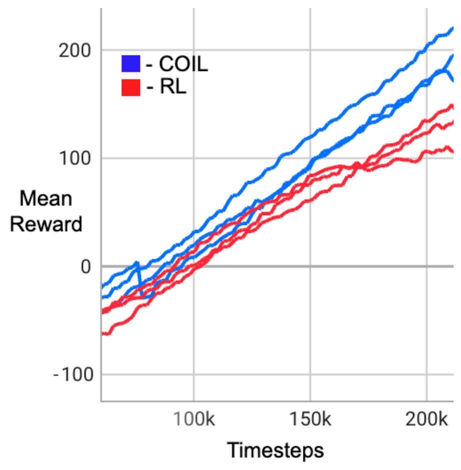

Figure 6: The first 200k timesteps of agents trained via PPO with and without COIL. COIL can be seen to offer faster increases in average reward in these early stages of training. For clarity, only 3 runs of each algorithm were selected for this figure.

test tracks in order to investigate generalisation capabilities.

Table 2 contains the results for the different policies on the unseen test tracks (10 trials with 3 laps each). It should be noted that for COIL and RL, the policies that were chosen for testing were the policies that achieved the highest average episode reward during training. Table 2 shows that COIL achieved the highest mean reward of all four methods when tested on the test tracks. This provides evidence that COIL has strong generalisation ability, as its performance is highly competitive both on the training tracks and the test tracks. A one-sided t-test on the mean reward between COIL and RL gave $p = 4.79 \cdot 10^{-8}$, indicating statistically significant improvement over regular RL on the unseen test tracks. COIL is also the fastest of the policies, achieving the lowest average lap time of the four methods. However, COIL is also the least smooth of the policies. This might be because the agent has learned to prioritise speed over smoothness in order to achieve the higher rewards.

The smoothness is calculated by taking the average magnitude of all of the r_{Ω_3} values generated during the tests. To recap, we have that

$$r_{\Omega_3} = - \parallel \theta - \tilde{\theta} \parallel$$

where θ is the direction the car is facing, and $\tilde{\theta}$ is a exponentially weighted moving average of previous car directions. Therefore, the larger the average magnitude of r_{Ω_3}, the less smooth the journey is.

Lastly, we will analyse how the four methods hold up in a more complex situation and a significant change in incoming sensor data using the robot results.

Table 2: Table of results for four mobile robot driving algorithms tested in a `Python` simulation on 3 unseen test tracks. For each track, 30 laps were completed by each algorithm.

Method	**Mean Lap Time**	**Mean Smoothness**	**Mean Reward**
Guide Policy	48.81s	0.121	609.4
OIL	48.82s	**0.061**	620.8
RL	48.58s	0.557	614.5
COIL	**48.48s**	0.606	**632.9**

The reward function designed for the simulation cannot be directly applied in real life because we lack the necessary information to calculate the reward at each time step. Instead, the lap time of the robot on the test track can be used and by recording the steering angle of the car over time we can also determine the smoothness of the cars journey. The smoothness of the cars journey was calculated by the same method as in the simulation, by taking the difference between the current steering angle and the exponentially weighted moving average of previous steering angles.

For each mobile robot algorithm, three attempts at driving around the track were made and the lap times and angular velocities were recorded. The results can be seen in Table 3. Table 3 shows that COIL offers the fastest and smoothest journey of all four mobile robot algorithms. This provides evidence that COIL is capable of effectively solving more complex and difficult challenges. Despite the change of environment, agent, and observations, COIL still achieves strong results as a mobile robot algorithm.

Table 3: Table of results for four mobile robot driving algorithms tested by a robot. The algorithms are judged on speed, smoothness, and safety.

Method	**Mean Smoothness**	**Mean Lap Time**
Guide Policy	0.265	43.67s
OIL	0.104	43.32s
RL	0.102	40.14s
COIL	**0.082**	**39.53s**

It must be noted that the success of the real-life mobile-robot navigation algorithm is highly dependent on how realistic the simulation is during training. Aligning the dynamics of a simulation to reality is very important when testing algorithms. The accuracy and complexity of the simulation was limited by the resources at hand and scope of the project, and thus the results for the real-life robot have additional components to consider. By switching from simulation to reality, we are introducing a significant shift between the training data (from the simulation) and the test data (from the Lidar sensor), therefore these results allow us to examine the affect of transfer learning on our algorithms. Given this additional difficultly in the task, the results in this section are indicative that each algorithm exhibits robustness to transfer learning. Given more resources and time, the alignment between reality and simulation can be further improved, and the results will be more suitable for direct comparison.

These results show that COIL offers a method of effectively utilising guide policies even when the guide policy is both non-expert, and when faced with transfer learning. In this case, not only is the task much more difficult because of the transfer learning, but COIL is faced with the additional difficulty that the actions from the guide-policy are *also* effected by the transfer learning. Despite the degradation of the guide policy, COIL remains the strongest algorithm at the task, demonstrating that COIL is capable of utilising non-expert guide policies even when withstanding the effect of transfer learning. Consequently, there is evidence that COIL is a robust and flexible method of effectively incorporating guide policies into Reinforcement Learning problems.

5 DISCUSSION

The computational cost of running the COIL agent in the environment is greater than Reinforcement Learning methods that do not utilise the guide policy because it is necessary to compute the action of the guide policy at each time step (on top of the other computations in the Reinforcement Learning algorithm). In practice, the computational cost of computing the guide policies action is lightweight and thus using COIL is only marginally more computationally expensive than Reinforcement Learning methods that do not use the guide policy. However, in rare cases where calculation of the guide policies action is computationally heavy, the available computing power should be taken into account when using COIL. In this case, it might be more feasible for researchers to apply Imitation Learning techniques in order to achieve a more lightweight approximation of the guide policy.

Additionally, a potential shortcoming of COIL is the requirement of a *present* guide policy. If the guide policy cannot be queried during the exploration of the environment then it is not possible to use it in COIL. However, there is a suitable alternative given a guide policy that is not present. In this case researchers can add a preliminary step of applying Imitation Learning techniques to generate a corresponding present guide policy as an approximation of the original non-present guide policy. In theory, any Offline Imitation Learning algorithm could be used for this purpose. Therefore, although the requirement of a present guide policy is certainly a limitation of the proposed COIL method, given the successful implementation of Offline Imitation Learning COIL can still be used. Furthermore, as COIL does not require an expert guide policy, it is still applicable even if this Offline Imitation Learning approximation is limited in capacity. Thus theoretically, even a non-present *and* non-expert guide policy can still be utilised with COIL, which reinforces the notion that COIL is a highly-flexible technique.

On the basis of the discussed findings, first results in the research of COIL seem very promising, however, a limitation in our present analysis of COIL is that the scope of results in this initial paper are focused on the task of mobile-robot navigation. In future research, COIL must be also tested on other Reinforcement Learning tasks in order to get a deeper understanding of its capabilities.

6 CONCLUSIONS

Contextual Online Imitation Learning (COIL) offers researchers a promising new method of incorporating guide policies into the learning process of Reinforcement Learning algorithms in a practical way. In this paper, COIL has been demonstrated to provide significant improvement over other methods in the context of mobile-robot navigation. In both the simulation and real-life experiments this was the case. Additionally, the results from the lab experiment demonstrated that COIL also has the ability to withstand the effect of transfer learning. Furthermore, COIL demonstrated consistently faster policy training than the corresponding agent trained without COIL.

This paper has introduced the COIL methodology and two applications in which it offers improved results over competing methods. However, future research must still be done to more thoroughly investigate this novel method. Future research on COIL could focus on four interesting topics. Firstly, research could be conducted to analyse the effect of using multiple guide policies with COIL. In particular, it would be interesting to analyse the deviation between the agents policy and the guide policies during exploration of the environment to see which of the guide policies are being ‘listened to’ and in which states. Secondly, additional research could focus on comparing the COIL methodology for different Reinforcement Learning algorithms such as Deep Q-Networks

(Mnih et al., 2016b) or A2C (Mnih et al., 2016a), to see if similar success is achieved. Thirdly, the effect of treating the parameters of the guide policy as additional trainable parameters within the Reinforcement Learning algorithm in order to fine-tune the actions of the guide policy might also be an interesting avenue for future research. Lastly, COIL could be further tested and compared to other Imitation Learning algorithms in various tasks in order to get a broader understanding of how it compares to other existing methods.

REFERENCES

Aslan, M. F., Unlersen, M. F., Sabanci, K., and Durdu, A. (2021). Cnn-based transfer learning–bilstm network: A novel approach for covid-19 infection detection. *Applied Soft Computing*, 98:106912.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chen, J., Yuan, B., and Tomizuka, M. (2019). Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2884–2890.

Durrleman, S. and Simon, R. (1989). Flexible regression models with cubic splines. *Statistics in medicine*, 8(5):551–561.

Edward, J. (2021). Coarse-to-fine imitation learning : Robot manipulation from a single demonstration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4613–4619.

Fang, B., Jia, S., Guo, D., Xu, M., Wen, S., and Sun, F. (2019). Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3(4):362–369.

Halbert, D. (1984). *Programming by example*. PhD thesis, University of California, Berkeley.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems 29*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Elsevier*.

Liu, Y., Liu, Q., Zhao, H., Pan, Z., and Liu, C. (2020). Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2128–2135.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016a). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2016b). Playing atari with deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J. (2018). An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pomerleau, D. (1998). An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*.

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

Ruder, S., Peters, M. E., Swayamdipta, S., and Wolf, T. (2019). Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2020). Proximal policy optimization algorithms. In *Uncertainty in Artificial Intelligence*, pages 113–122.

Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., and Bennice, M. (2022). Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*.

Wan, Z., Yang, R., Huang, M., Zeng, N., and Liu, X. (2021). A review on transfer learning in eeg signal analysis. *Neurocomputing*, 421:1–14.

Wang, J., Zhuang, Z., Wang, Y., and Zhao, H. (2022). Adversarially robust imitation learning. In *Conference on Robot Learning*, pages 320–331.

Wang, Y., He, H., and Tan, X. (2020). Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence*, pages 113–122.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., and Wu, Y. (2021). The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*.

Zhang, S., Cao, Z., Sadigh, D., and Sui, Y. (2021). Confidence-aware imitation learning from demonstrations with varying optimality. In *Advances in Neural Information Processing Systems 34*, pages 12340–12350.