AUTHOR(S): HUSSEIN, A., GABER, M.M. and ELYAN, E.

TITLE: Deep active learning for autonomous navigation.

YEAR: 2016

Publisher citation: HUSSEIN, A., GABER, M.M. and ELYAN, E. 2016. Deep active learning for autonomous navigation. In Jayne, C. and Iliadis, L. (eds.) Communications in computer and information science, 629. Engineering applications of neural networks: proceedings of the 17th International conference on engineering applications of neural networks (EANN 2016), 2 - 5 September 2016, Aberdeen, UK. Cham: Springer [online], pages 3-17. Available from: https://doi.org/10.1007/978-3-319-44188-7_1

OpenAIR citation: HUSSEIN, A., GABER, M.M. and ELYAN, E. 2016. Deep active learning for autonomous navigation. In Jayne, C. and Iliadis, L. (eds.) Communications in computer and information science, 629. Engineering applications of neural networks: proceedings of the 17th International conference on engineering applications of neural networks (EANN 2016), 2 - 5 September 2016, Aberdeen, UK. Cham: Springer, pages 3-17. Held on OpenAIR [online]. Available from: https://openair.rgu.ac.uk

Deep Active Learning for Autonomous Navigation

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan

School of Computing, Robert Gordon University, Garthdee Road Aberdeen, UK AB10 7QB

Abstract. Imitation learning refers to an agent's ability to mimic a desired behavior by learning from observations. A major challenge facing learning from demonstrations is to represent the demonstrations in a manner that is adequate for learning and efficient for real time decisions. Creating feature representations is especially challenging when extracted from high dimensional visual data. In this paper, we present a method for imitation learning from raw visual data. The proposed method is applied to a popular imitation learning domain that is relevant to a variety of real life applications; namely navigation. To create a training set, a teacher uses an optimal policy to perform a navigation task, and the actions taken are recorded along with visual footage from the first person perspective. Features are automatically extracted and used to learn a policy that mimics the teacher via a deep convolutional neural network. A trained agent can then predict an action to perform based on the scene it finds itself in. This method is generic, and the network is trained without knowledge of the task, targets or environment in which it is acting. Another common challenge in imitation learning is generalizing a policy over unseen situation in training data. To address this challenge, the learned policy is subsequently improved by employing active learning. While the agent is executing a task, it can query the teacher for the correct action to take in situations where it has low confidence. The active samples are added to the training set and used to update the initial policy. The proposed approach is demonstrated on 4 different tasks in a 3D simulated environment. The experiments show that an agent can effectively perform imitation learning from raw visual data for navigation tasks and that active learning can significantly improve the initial policy using a small number of samples. The simulated testbed facilitates reproduction of these results and comparison with other approaches.

1 Introduction

One of the important aspects of artificial intelligence is the ability of autonomous agents to behave effectively and realistically in a given task. There is a rising demand for applications in which agents can act and make decisions similar to human behavior in order to achieve a goal. Imitation learning is a paradigm in which an agent learns how to behave by observing demonstrations of correct behavior provided by a teacher. In contrast to explicit programming, learning

from demonstrations does not require knowledge of the task to be integrated in the learning process. It favors a generic learning process where the task is learned completely from observing the demonstrations. Thus, an intelligent agent can be trained to perform a new task simply by providing examples. Since an agent is able to learn complex tasks by mimicking a teacher's behavior, imitation learning is relevant to many robotic applications [2][30][6][25][4][12][14][37] and is considered an integral part in the future of intelligent robots [32].

One of the biggest challenges in imitation learning is finding adequate representations for the state of the agent in its environment. The agent should be able to extract meaningful information from sensing of its surroundings, and utilize this information to perform actions in real time. Deep learning methods have recently been applied in a wide array of applications and are especially successful in handling raw data. One of the most popular deep learning techniques is *Convolutional Neural Networks* (CNNs). CNNs are particularly popular in vision applications due to their ability to extract features from high dimensional visual data. The ability of deep networks to automatically discover patterns provides a generic alternative to engineered features which have to be designed for each specific task. For instance traditional planning approaches that use computer-vision methods of object recognition and localization need to tailor the methods for every individual target and task. CNNs achieve results competitive with the state of the art in many image classification tasks [9][18] and have been recently used to learn Atari 2600 games from raw visual input [21][22]. These and other recent attempts have shown that deep learning can be successful in teaching an agent to perform a task from visual data. However, most studies focus on 2D environments with stationary views; which does not reflect real world applications. Moreover, direct imitation is performed without considering refining the policy based on the agent's performance. To the best of our knowledge, training an agent from raw visual input using deep networks and active learning in a 3D environment has not been done.

In this paper we present a novel method that utilizes deep learning and active learning to train agents in a 3D setting. The method is demonstrated on several navigation tasks in a 3D simulated environment. Navigation is one of the most explored domains in imitation learning due to its relevance to many robotic applications, such as flying [30][1][24] and ground vehicles [33][28] [7][27]. Navigation is also an essential base task in high degree of freedom robots (e.g. humanoid robots) [8][31]. We propose a generic method for learning navigation tasks from demonstrations that does not require any prior knowledge of the task's goals, environment or possible actions. A training set is gathered by having a teacher control the agent to successfully perform the task. The controlled agent's view of the 3D environment is captured along with the actions performed in each frame. A deep convolution network is used to learn visual representation from the captured video footage and learn a policy to mimic the teacher's behavior. We also employ active learning to improve the agents policy by emphasizing situations in which it is not confident. We show that active learning can significantly improve the policy with a limited number of queried instances.

Once trained, the agent is able to extract features from the scene and predict actions in real time. We conduct our experiments on benchmark testbed that makes it seamless to replicate our results and compare with other approaches.

Benchmark environments are useful tools for evaluating intelligent agents. A few benchmarks are available for 2D tasks such as [3] [26] [16] and are being increasingly employed in the literature. 3D environments however have not been as widely explored, although they provide a closer simulation to real robotic applications. We use mash-simulator [20] as our testbed to facilitate the evaluation and comparison of learning methods. It is also convenient for extending the experiments to different navigation tasks within the same framework.

In the next section we review related work. Section 3 describes the proposed methods. Section 4 details our experiments and results. Finally we present our conclusions and discuss future steps in Section 5.

2 Related Work

2.1 Navigation

Navigation tasks have been of interest in AI in general and imitation learning specifically from an early stage. Sammut et al [30] provides an early example of an aircraft learning autonomous flight from demonstrations provided via remote control. Later research tackle more elaborate navigation problems including obstacles and objects of interest. Chernova et al [7] use Gaussian mixture models to teach a robot to navigate through a maze. The robot is fitted with an IR sensor to provide information about the proximity of obstacles. This data coupled with input from a teacher controlling the robot is used to learn a policy. The robot is then able to make a decision to execute one of 4 motion primitives(unit actions) based on its sensory readings. In [11] the robot uses a laser sensor to detect and recognize objects of interest. A policy is learned to predict subgoals associated with the detected objects rather than directly predicting the motion primitives. Such sensing methods provide an abstract view of the environment, but can't convey visual details that might be needed for intelligent agents to mimic human behavior. [23] use neural networks to learn a policy for driving a car in racing game using features extracted from the game engine (such as position of the car relative to the track). Driving is a complex task compared to other navigation problems due to the complexity of the possible actions. The outputs of the neural network in [23] are high DOF low level actions. However, the features extracted from the game engine to train the policy would be difficult to extract in the real world. Advances in computational resources have prompted the use of visual data over simpler sensory data. Visual sensors provide detailed information about the agents surrounding and are suitable to use in real world applications. In [29] a policy for a racing game is learned from visual data. Demonstrations are provided by capturing the games video stream and the controller input. The raw frames (downsampled) without extracting engineered features are used as input to train a neural network.

2.2 Deep learning

Deep learning methods are highly effective in problems that don't have established sets of engineered features. CNNs have been used with great success to extract features from images. In recent studies [21][22] CNNs are coupled with reinforcement learning to learn several Atari games. A sequence of raw frames is used as input to the network and trial and error is used to learn a policy. Trial and error methods such as reinforcement learning have been extensively used to learn policies for intelligent agents [17]. However, providing demonstrations of correct behavior can greatly expedite the learning rate. Moreover, learning through trial and error can lead the agent to learn a way of performing the task that doesn't seem natural or intuitive to a human observer. In [13] learning from demonstrations is applied on the same Atari benchmark. A supervised network is used to train a policy using samples from a high performing but non real time agent. This approach is reported to outperform agents that learn from scratch through reinforcement learning. Other examples of using deep learning to play games include learning the game of 'GO' using supervised convolution networks [10] and a combination of supervised and reinforcement learning [34]. These examples all focus on learning 2D games that have a fixed view. However in real applications, visual sensors would capture 3D scenes, and the sensors would most likely be mounted on the agent which means it is unrealistic to have a fixed view of the entire scene at all times.

In [19] a robot is trained to perform a number of object manipulation tasks. First a trajectory is learned using reinforcement learning with the position of the objects and targets known to the robot. These trajectories then serve as demonstrations train a supervised convolutional neural network. In this case no demonstrations are needed to be provided by a teacher. However, this approach requires expert knowledge for the initial setup of the reinforcement learning phase. Compared to related work that employs deep learning to teach an intelligent agent, this is a realistic application implemented with a physical robot. However, the features are extracted from a set scene with small variations. This is different from applications where the agent moves and turns around, and with that completely altering it's view.

2.3 Active learning

In many imitation learning applications direct imitation is not sufficient for robust behavior. One of the common challenges facing direct imitation is that the training set doesn't fully represent the desire task. The collected demonstrations only include optimal actions performed by the teacher. If the agent makes an error it arrives at a state that was not represented in its learned policy [36]. It is therefore necessary in many cases to provide further training to an agent based on its own performance of the task. One of the methods to enhance a trained agent is active learning. Active learning relies on querying a teacher for the correct decision in cases where the trained model performs poorly. The teacher's answers are used to improve the model in its weakest areas. In [8]

active learning is used to teach a robot navigation tasks. The agent estimates a confidence measure for its prediction and queries a teacher for the correct action when the confidence is low. Erroneous behavior may also be identified by the teacher. In [5] the robot is allowed to perform the task while a human teacher physically adjusts its actions, which in turn provides corrected demonstrations. Some imitation learning tasks involve actions that are performed continuously over a period of time (i.e an action is comprised of a series of motions performed in sequence). In such cases a correction can be provided by the teacher at any point in the action trajectory [29][15]. This way the agent is able to adapt to errors in the trajectory.

3 Proposed Method

In this section we detail our proposed method for learning navigation tasks from demonstrations. The source code for this work can be accessed at:
https://github.com/ahmedsalaheldin/ImitationMASH.git

3.1 Collecting Demonstrations

In imitation learning it is assumed that a human teacher is following an unknown optimal policy. It is therefore possible to use an optimal policy if it exists to collect demonstrations. To collect a training set we use a deterministic automated teacher that has access to information hidden from a human or intelligent playing agent such as position of targets and obstacles in a 3D space. Each training instance consists of a raw 120×90 image of the rendered 3D scene and the action performed by the teacher. We only use the current frame (not a sequence of previous frames) in an instance because for the navigation tasks investigated here adhere to the Markov property. That is, that current state is sufficient to make a decision. And any previous actions and states need not be included in the representation of the current state. In that case training an imitation learning policy is reduced to a supervised image classification problem; where the current view of the agent is the image and the action chosen by the teacher is the label. Subsequently the trained agent will be able to predict a decision (as it would be taken by the teacher) given its current view. More formally, the agent learns a policy π from a set of demonstrations $D = (x_i, y_i)$ such that $u = \pi(x, \alpha)$. Where x_i is a 120×90 image, y is the action performed by the teacher at frame i, u is the action predicted by policy π for input x and α is the set of policy parameters that are changed through learning.

3.2 Deep learning

To learn the policy we employ a deep convolutional neural network. The proposed network uses several convolution layers to automatically extract features from the raw visual footage. Then a fully connected layer is used to map the learned features to actions. Each convolution layer is followed by a pooling layer that

down-samples the output of the convolution layer. The convolution layers take advantage of spacial connection between visual features to reduce connections in the network. The pooling layers reduce the dimensionality to further alleviate the computations needed. Our network follows the pattern in [22]. It consists of 3 convolution layers each followed by a pooling layer. The input to the first layer is a frame of 120×90 pixels. We apply a luminance map to the colored images to obtain one value for each pixel instead of 3 channels, resulting in a feature vector of size 10,800. Figure 1 shows the architecture of the network. The filter sizes for the three layers are 7×9, 5×5 and 4×5 respectively; and the number of filters are 20, 50 and 70 respectively. The pooling layers all use maxpool of shape (2,2). Following the last convolution layer is a fully connected hidden layer with rectifier activation function and fully connected output layer with three output nodes representing the 3 possible actions. Table 1 summarizes the architecture of the network.

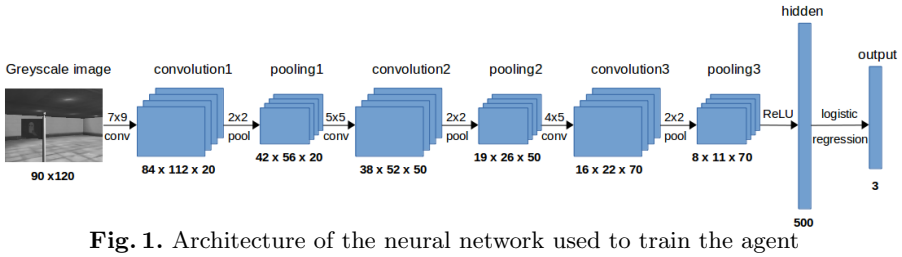

Fig. 1. Architecture of the neural network used to train the agent

Table 1. Neural network architecture

Layer	Size of activation volume
Input	120 * 90
Conv1	7 * 9 * 20
Conv2	5 * 5 * 50
Conv3	4 * 5 * 70
FC	500
Output(FC)	3

3.3 Active learning

Active learning is employed to improve the initial policy learned from demonstrations. This is achieved by acquiring a new data set to train the agent that emphasizes the weaknesses of the initial policy. The agent is allowed to perform the task for a number of rounds. For each prediction the network's confidence

is calculated, and if the confidence is low the optimal policy is queried for the correct action. The action provided by the teacher is performed by the agent and is recorded along with the frame image. The confidence is measured as the entropy of the output of the final layer in the network. The entropy $H(X)$ is calculated as:

$$H(X) = -\sum_i P(x_i) \log_2 P(x_i) \tag{1}$$

Where X is the prediction of the network, $P(x_i)$ is the probability distribution produced by the network for action i.

The active samples are added to the training set and used to update the initial policy. We find that updating a trained network using only the active samples results in forgetting the initial policy in favor of an inadequate one rather than complementing it. Therefore the training set is augmented with the active samples collected from the playing agent. The augmented dataset is used to update the network that was previously trained. We find that it is easier and faster for the network to converge if it is pre-trained with the initial dataset than training from scratch. Algorithm 1 shows the steps followed to perform active learning.

Low confidence predictions are mainly caused by situations that were not covered by the training data. Therefore, for active learning to be effective, it is important that it is performed in the simulation rather than on a collected dataset. Because by performing its current policy in the simulation, the agent arrives at unfamiliar situations where it is not confident in its behavior and thus utilize active learning.

Algorithm 1 Active Learning Algorithm

Given: A policy π trained on a Data set $D = (x_i, y_i)$
    Confidence threshold β
while Active_Learning **do**
    x = current_frame
    $u = \pi(x, \alpha)$
    $H(X) = -\sum_i P(u_i) \log_2 P(u_i)$
    if $H(X) < \beta$ **then**
        $y = Query(x)$
        perform action y
        add (x, y) to D
    else
        perform $max(u)$
Update π using D

4 Experiments

We conduct our experiments in the framework of mash-simulator [20]. Mash-simulator is a tool for benchmarking computer vision techniques for navigation tasks. The simulator includes a number of different tasks and environments. As well as optimal policies for a number of tasks. All the navigation is viewed from the first person perspective. The player has 4 possible actions: 'Go forward', 'Turn left', 'Turn right' and 'Go back'. Although there are 4 possible actions, the action 'Go back' was never used in the demonstrations by the optimal policy. Therefore the network is only presented with 3 classes in the training set and thus has 3 output nodes.

4.1 Tasks

The experiments are conducted on the following 4 navigation tasks:

Reach the flag This task is set in a single rectangular room with a flag placed randomly in the room. The goal is to reach the flag. The task fails if the flag is not reached within a time limit.

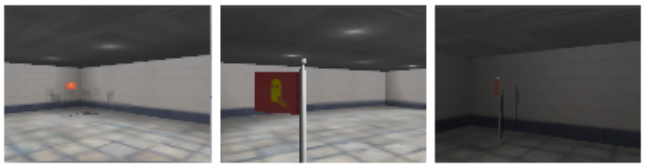

Fig. 2. sample images from "Reach the flag"

Follow the line This task is set in a room with directed lines drawn on the floor. The lines show the direction to follow in order to reach the flag. The target is to follow the line to the flag, and the agent fails if it deviates from the line on the floor.

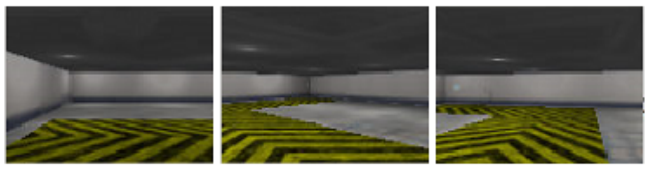

Fig. 3. sample images from "Follow the line"

Reach the correct object In this task two objects are placed on pedestals in random positions in the room. The objective is to reach the pedestal with the trophy on it. The task fails if a time limit is reached or if the player reaches the wrong object. The wrong object has the same material of the trophy and can take different shapes.

Fig. 4. sample images from "Reach the correct object"

Eat all disks This task is set in a large room containing several black disks on the floor. The target is to keep reaching the disks. A disk is 'eaten' once the agent reaches it and dissapears . New disks appear when one is eaten. The goal of this task is to eat as many disks as possible within a time limit.

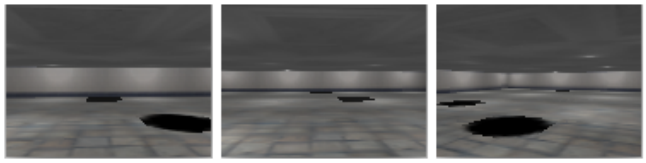

Fig. 5. sample images from "Eat all disks"

Figures 2 - 5 show sample images of the 4 tasks in the 120×90 size used in the experiments.

4.2 Setup

To evaluate the proposed methods, the performance of the agent is measured over 1,000 rounds. A round starts when the task is initialized and ends when the agent reaches the target or a time limit is reached. The number of frames in a round might vary depending on how fast the agent can reach the target. For all tasks, in each round the environment is randomized including room size and shape, lighting and the location of the target and the agent. A time limit is set for each round and the round fails if the limit is reached before the agent reaches the target. The time limit is measured in frames to avoid any issues with different frame rates. The time limit is set as the maximum time needed for the optimal policy to finish the task; which is 500 frames for "Reach the flag" and

"Reach the correct object" and 5000 frames for "Follow the line". In "Eat all disks" the task is continuous, so a time limit was set to match the total number of frames in the other tasks.

4.3 Implementation details

Inter-process communication is used to communicate data across the different components of the testbed. The agent acts as a client and communicates with the simulator via a TCP connection as follows: The agent requests a task from the server, the server initiates a round and sends an image to the client. The client sends an action to the server. The server calculates the simulations and responds with a new image. Figure 6 shows a flowchart of the data collection process.

The network used for prediction is also decoupled from the agent. The network acts as a predicting server where an agent sends frames that it receives from the simulator and in return receives a decision from the network. The entire process of communication with both servers occurs in real time. This implementation facilitates experimentation, as making changes to the network doesn't affect the client or the simulator server. Moreover, it is easier to extend this system to physical robots. A predicting server can be located on the robot or on another machine if the robot's computational capabilities are not sufficient. A predicting server can also serve multiple agents simultaneously. The agent client is implemented in c++ to facilitate interfaceing with the mash-simulator. The predicting server and the training process are implemented in python using the Theano deep learning library [35]. Figure 7 shows a flowchart of the agent performing a task.

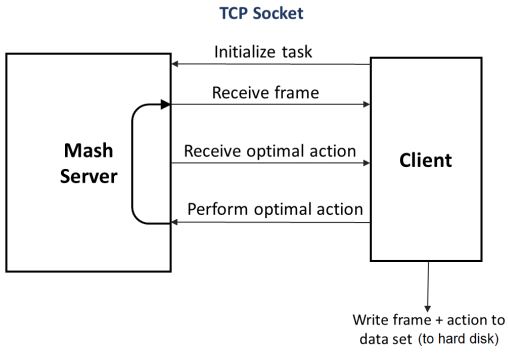

Fig. 6. Dataset Collection Flowchart

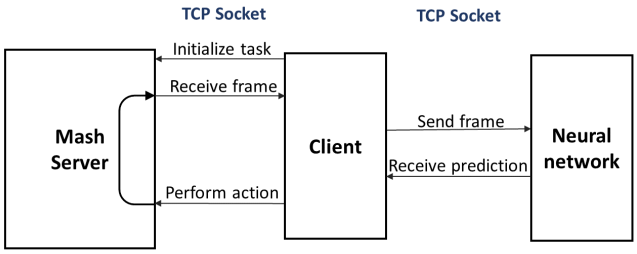

Fig. 7. Imitation Agent Playing Flowchart

4.4 Results

In this section we present the results of the proposed method. The same network and parameters are used to learn all tasks. For each task 20,000 images are used for training. Testing is conducted by allowing an agent to attempt the tasks in the mash-simulator and recording the number of successful attempts. An agent's performance for the first 3 tasks is evaluated as the percentage of times it reaches the target in 1,000 rounds. For "Eat all disks", the performance is measured as the number of disks eaten in 1,000 rounds. We also report the classification error on an unseen test set of 20,000 images collected from the teacher's demonstrations.

Table 2 shows the results for the first 3 tasks. The success measure is the percentage of rounds (out of 1000) in which the agent reached the target. While error is the classification error on the test set collected from the teacher's demonstrations. The agent performs well on "Reach the flag" and is significantly less successful in the other two tasks. "Follow the line" is considerably less fault tolerant than "Reach the flag". As a small error can result in the agent deviating from the line and subsequently failing the round. Whereas in "Reach the flag" the agent can continue to search for the target after a wrong prediction. In "Reach the correct object" the agent is not able to effectively distinguish between the two objects. This could be attributed to insufficient visual details in the training set, as the teacher avoids the wrong object from a distance. Qualitative analysis of "Reach the flag" shows that the agent aims towards corners as they resemble the erect flag from a distance. Upon approaching the corner, as the details of the image become clearer, the agent stops recognizing it as the target and continues its search. While this did not pose a big problem in the agent's ability to execute the task it is interesting to examine the ability of CNNs to distinguish small details in such environments. It is also worth noting that the teacher's policy for "Reach correct object" does not avoid the wrong object if it is in the way of the target and achieves 80.2% success rate

Table 3 shows results for the 4^{th} task "Eat all disks". The table shows the score of the agent compared to the score achieved using the optimal policy. The agent is shown to achieve 97.9% of the score performed by the optimal policy.

To improve the agent's ability to adapt to wrong predictions and unseen situations, active learning is used to train the agent on "Follow the line". In the other tasks where the agent searches for the target, the optimal policy remembers

Table 2. Direct Imitation results

Task	reach the flag	reach object	follow the line
success	96.20 %	53.10%	40.70%
error	2.48%	4.06%	0.86%

Table 3. "Eat all disks" results

Task	Agent	Optimal policy
score	1051	1073
error	1.70%	-

the location of the target even if it goes out of view due to agent error. Therefore active learning samples include information that is not represented in the visual data available to the agent and thus degrade the performance. This can be rectified by devising a teaching policy that does not use historical information, or by incorporating past experience in the learned model.

Figure 8 shows the results of active learning on the "Follow the line" task. Active learning is demonstrated to significantly improve the performance of the agent using a relatively small number of samples. Comparing the classification error with success rate emphasizes the point that the errors come from situations that are not represented in the teacher's demonstrations.

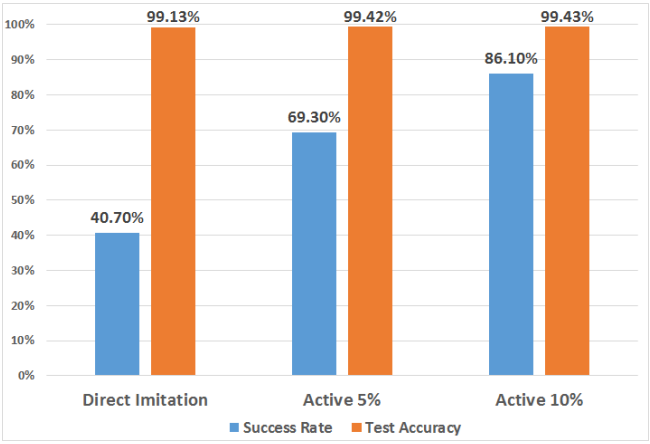

Fig. 8. Results for active learning on "follow the line" task

The task in which the time limit affected the performance was "Reach the flag". As the agent continues to follow its policy in search of the flag even after performing wrong predictions. The effect of the time limit is evaluated in Figure 9 which presents the success rate of "reach the flag" task with different time limits. The horizontal axis represents the time limit as a percentage of the maximum time needed by the teacher. The graph shows that the longer the agent is allowed to look for the target the higher the success rate.

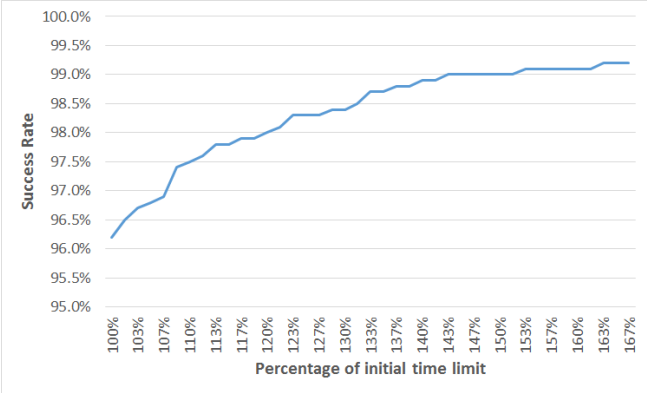

Fig. 9. Results for "reach the flag" task with increasing time limits

Overall the results show good performance on 3 out of the 4 tasks. They demonstrate the effectiveness of active learning to significantly improve a weak policy with a limited number of samples. Even without active learning the agent can learn a robust policy for simple navigation tasks.

5 Conclusion and future directions

In this paper, we propose a framework for learning autonomous policies for navigation tasks from demonstrations. A generic learning process is employed to learn from raw visual data without integrating any knowledge of the task. The experiments are conducted on a testbed that facilitates reproduction, comparison and extension of this work. The results show that CNNs can learn meaningful features from raw images of 3D environments and learn a policy from demonstrations. They also show that active learning can significantly improve a learned policy with a limited number of samples.

Our next step is to conduct an investigation of the proposed approach in more visually cluttered environments to further evaluate the ability of convolution networks to create adequate representations from (relatively) low resolution 3D scenes. As well as extend active learning experiments to more tasks. We also aim to integrate reinforcement learning with learning from demonstrations to improve the learned policies through trial and error. This allows the agent to generalize its policy to unseen situations and adapt to changes in the task without requiring to query the teacher.

References

1. Abbeel, P., Coates, A., Quigley, M., Ng, A.Y.: An application of reinforcement learning to aerobatic helicopter flight. Advances in neural information processing systems 19, 1 (2007)
2. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems 57(5), 469–483 (2009)

3. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. arXiv preprint arXiv:1207.4708 (2012)
4. Bemelmans, R., Gelderblom, G.J., Jonker, P., De Witte, L.: Socially assistive robots in elderly care: A systematic review into effects and effectiveness. Journal of the American Medical Directors Association 13(2), 114–120 (2012)
5. Calinon, S., Billard, A.G.: What is the teachers role in robot programming by demonstration?: Toward benchmarks for improved learning. Interaction Studies 8(3), 441–464 (2007)
6. Cardamone, L., Loiacono, D., Lanzi, P.L.: Learning drivers for torcs through imitation using supervised methods. In: Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on. pp. 148–155. IEEE (2009)
7. Chernova, S., Veloso, M.: Confidence-based policy learning from demonstration using gaussian mixture models. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. p. 233. ACM (2007)
8. Chernova, S., Veloso, M.: Confidence-based policy learning from demonstration using gaussian mixture models. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. p. 233. ACM (2007)
9. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3642–3649. IEEE (2012)
10. Clark, C., Storkey, A.: Training deep convolutional neural networks to play go. In: Proceedings of the 32nd International Conference on Machine Learning (ICML-15). pp. 1766–1774 (2015)
11. Dixon, K.R., Khosla, P.K.: Learning by observation with mobile robots: A computational approach. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. vol. 1, pp. 102–107. IEEE (2004)
12. Gorman, B.: Imitation learning through games: theory, implementation and evaluation. Ph.D. thesis, Dublin City University (2009)
13. Guo, X., Singh, S., Lee, H., Lewis, R.L., Wang, X.: Deep learning for real-time atari game play using offline monte-carlo tree search planning. In: Advances in Neural Information Processing Systems. pp. 3338–3346 (2014)
14. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning rhythmic movements by demonstration using nonlinear oscillators. In: Proceedings of the ieee/rsj int. conference on intelligent robots and systems (iros2002). pp. 958–963. No. BIOROB-CONF-2002-003 (2002)
15. Judah, K., Fern, A., Dietterich, T.G.: Active imitation learning via reduction to iid active learning. arXiv preprint arXiv:1210.4876 (2012)
16. Karakovskiy, S., Togelius, J.: The mario ai benchmark and competitions. Computational Intelligence and AI in Games, IEEE Transactions on 4(1), 55–67 (2012)
17. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research p. 0278364913495721 (2013)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
19. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. arXiv preprint arXiv:1504.00702 (2015)
20. Mash-simulator. https://github.com/idiap/mash-simulator (2014)
21. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)

22. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015)
23. Munoz, J., Gutierrez, G., Sanchis, A.: Controller for torcs created by imitation. In: Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on. pp. 271–278. IEEE (2009)
24. Ng, A.Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., Liang, E.: Autonomous inverted helicopter flight via reinforcement learning. In: Experimental Robotics IX, pp. 363–372. Springer (2006)
25. Nicolescu, M.N., Mataric, M.J.: Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems. pp. 241–248. ACM (2003)
26. Noda, I., Matsubara, H., Hiraki, K., Frank, I.: Soccer server: A tool for research on multiagent systems. Applied Artificial Intelligence 12(2-3), 233–250 (1998)
27. Ollis, M., Huang, W.H., Happold, M.: A bayesian approach to imitation learning for robot navigation. In: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. pp. 709–714. IEEE (2007)
28. Ratliff, N., Bradley, D., Bagnell, J.A., Chestnutt, J.: Boosting structured prediction for imitation learning. Robotics Institute p. 54 (2007)
29. Ross, S., Bagnell, D.: Efficient reductions for imitation learning. In: International Conference on Artificial Intelligence and Statistics. pp. 661–668 (2010)
30. Sammut, C., Hurst, S., Kedzier, D., Michie, D., et al.: Learning to fly. In: Proceedings of the ninth international workshop on Machine learning. pp. 385–393 (1992)
31. Saunders, J., Nehaniv, C.L., Dautenhahn, K.: Teaching robots by moulding behavior and scaffolding the environment. In: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction. pp. 118–125. ACM (2006)
32. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in cognitive sciences 3(6), 233–242 (1999)
33. Silver, D., Bagnell, J., Stentz, A.: High performance outdoor navigation from overhead data using imitation learning. Robotics: Science and Systems IV, Zurich, Switzerland (2008)
34. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature 529(7587), 484–489 (2016)
35. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688 (May 2016), http://arxiv.org/abs/1605.02688
36. Togelius, J., De Nardi, R., Lucas, S.M.: Towards automatic personalised content creation for racing games. In: Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on. pp. 252–259. IEEE (2007)
37. Vogt, D., Amor, H.B., Berger, E., Jung, B.: Learning two-person interaction models for responsive synthetic humanoids. Journal of Virtual Reality and Broadcasting 11(1) (2014)