In proceedings of the AAMAS 2016 workshop on Adaptive Learning Agents
May 2016

Deep Imitation Learning for Parameterized Action Spaces

Matthew Hausknecht
Department of Computer Science
University of Texas at Austin
mhauskn@cs.utexas.edu

Yilun Chen
Department of Automation
Tsinghua University
cyl12@tsinghua.edu.cn

Peter Stone
Department of Computer Science
University of Texas at Austin
pstone@cs.utexas.edu

ABSTRACT

Recent results have demonstrated the ability of deep neural networks to serve as effective controllers (or function approximators of the value function) for complex sequential decision-making tasks, including those with raw visual inputs. However, to the best of our knowledge, such demonstrations have been limited to tasks either fully discrete or fully continuous actions. This paper introduces an imitation learning method to train a deep neural network to mimic a stochastic policy in a parameterized action space. The network uses a novel dual classification/regression loss mechanism to decide which discrete action to select as well as the continuous parameters to accompany that action. This method is fully implemented and tested in a subtask of simulated RoboCup soccer. To the best of our knowledge, the resulting networks represent the first demonstration of successful imitation learning in a task with parameterized continuous actions.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Learning

General Terms

Connectionism and neural nets

Keywords

Half Field Offense, RoboCup, Imitation Learning

1. INTRODUCTION

Sequential decision making in continuous action spaces has historically proven to be a challenge. One existing way to circumvent the problem of learning in high dimensional spaces is to mimic the actions of a teacher. Known as learning from demonstration, imitation learning, or apprenticeship learning, this family of methods is typically used on challenging robotic domains in which learning tabula rasa is not feasible [4, 5].

Meanwhile, recent results have demonstrated the ability of deep neural networks to serve as effective controllers (or function approximators of the value function) for complex sequential decision-making tasks [18, 15], including those with raw visual inputs. However, to the best of our knowledge, such demonstrations have been limited to tasks with either fully discrete or fully continuous actions.

This paper synthesizes these two lines of research by introducing an imitation learning method to train a deep neural network to mimic a stochastic policy in a domain with high-dimensional input and a parameterized continuous action space. The network uses a novel dual classification/regression loss mechanism to decide which discrete action to select as well as the continuous parameters to accompany that action.

This deep imitation learning method is fully implemented and tested in a subtask of RoboCup simulated soccer, which features a parameterized action space in which the agent must first select the type of action to perform from a discrete list of high level actions and then specify the continuous parameters to accompany that action. To the best of our knowledge, no past research has successfully leveraged teacher policies using imitation learning in a domain with such high dimensional inputs and continuous actions.

Successful imitation learning requires a good choice of policy representation for the learner. We choose deep neural networks to represent learned policies because of their power as general function approximators, their ability to generalize beyond the states and actions observed during training, and the ability to easily increase the complexity of the network by adding nodes or layers. Indeed, after learning in our testbed domain, the networks prove capable of selecting effective sequences of actions required to locate the ball, dribble, and score.

The main contributions of this paper are: 1) it demonstrates for the first time the possibility of learning a task with parameterized continuous actions through imitation learning of a stochastic policy; 2) it contributes a mimic network topology and training methodology that enables learning of such a task; 3) and it reports on a detailed case study showing the success of this network on a complex task and analyzing the critical factors that enable this success.

The remainder of this paper is organized as follows: the next two sections present related work and introduce the Half Field Offense domain. Next the architecture of the deep neural network and training procedure used for mimicking is discussed. Experiments and results are then presented, followed by discussion and conclusions.

2. RELATED WORK

There are three areas of closely related work: parameterized action space learning, imitation learning, and RoboCup soccer learning.

Masson and Konidaris [17] present a parameterized-action MDP formulation and approaches for model-free reinforcement learning in such environments. Applied to a simplified abstraction of simulated RoboCup soccer, the resulting agents operate over a parameterized action space and can

score on a fixed-policy goalie. There are three main differences from our work: first, Masson and Konidaris start each episode by co-locating the agent and ball. In our paper, trials start by randomly positioning both the agent and the ball. Thus our agent's policy must be able to locate and approach the ball, as in a real game of soccer. Second, Masson and Konidaris use a higher-level action space consisting only of parameterized kick, shoot-left-of-goalie, and shoot-right-of-goalie actions. Their agent automatically moves towards the ball and only needs to learn where to kick. In contrast, our agent must learn to follow the ball while dribbling and must decide how and where to shoot on goal without the benefit of actions to shoot left or right of the goalie. Finally, we use a higher-dimensional state space consisting of 58 continuous features as opposed to the 14 used by Masson and Konidaris.

Also in parameterized action space, Hausknecht and Stone [12] applied actor-critic deep reinforcement learning to the problem of learning, from scratch, complete policies for goal scoring. Their work represents a related but different approach towards learning, one which does not rely on a trajectories from a teacher.

Approaches to learning in parameterized action spaces other than robot soccer include: Guestrin et al. [9] factor the parameterized action space using a dynamic Bayesian network before attempting to compute an approximate value function. Sanner and Vianna [23, 25] use symbolic dynamic programming to solve the continuous portions of the parameterized MDPs. In contrast our work harnesses the function approximation power of deep neural networks, which have proven effective for learning control policies in reinforcement learning domains [18]. As our experiments demonstrate, without the depth of modern neural networks and rectified linear activation functions (ReLU), imitation learning would not be possible on this domain.

Imitation learning has been applied effectively in a wide variety of domains [7, 1, 24, 4, 20, 6, 22]. None of the examined domains have parameterized action spaces or use deep neural networks to represent the learned policy.

Recently, Guo et al. used deep neural networks to mimic sequential decision making policies in Atari games [10]. In this case, the policy to mimic comes from a Monte-Carlo Tree Search planner and features a discrete action space which allows the deep network to learn using the standard cross-entropy loss (the typical loss function for 1-of-n classification tasks). Likewise, Lillicrap demonstrates deep neural networks learning in continuous action spaces [15]. In contrast, the HFO task examined in this paper requires an entirely different approach due to its parameterized action space.

Parisotto et al. [19] describe a method for training a Actor-Mimic network: a network that imitates a Deep Q Network using policy regression to emulate the teacher's policy and feature regression to mimick the teacher's features. A single Actor-Mimic network is able to learn from several DQN teacher networks and achieve high scores across a set of different Atari games. Additionally, using Actor-Mimic multitask pretraining is shown to increase learning speed on a target task. Our approach differs by learning in parameterized space from teachers whose policies are not deep networks.

RoboCup 2D soccer has a rich history of learning. In one of the earliest examples, Andre used Genetic Programming to evolve policies for RoboCup 2D Soccer [3]. By using a sequence of reward functions, they first encourage the players to approach the ball, kick the ball, score a goal, and finally to win the game. Similarly, our work features players whose policies are entirely trained and have no hand-coded components. Our work differs by using a gradient-based learning method and learning from demonstration rather than a reward signal.

Competitive RoboCup agents are primarily hand-coded but may feature components that are learned or optimized for better performance. Examples of this include the Brainstormers who used neural reinforcement learning to optimize individual skills such as intercepting and kicking the ball [21]. However, these skills were optimized in the context of a larger, already working policy. Similarly, MacAlpine employed the layered-learning framework to incrementally learn a series of interdependent behaviors [16]. Such learning techniques have been shown to be applicable to physical robots in addition to simulated ones [14, 11, 8]. Instead of optimizing small portions of a larger policy, we take the approach of learning the full policy from a teacher.

In summary, our work is the first to apply imitation learning to a parameterized-action space and demonstrate the complex policies can be learned by deep neural networks.

3. HALF FIELD OFFENSE DOMAIN

Simulated Half Field Offense (HFO) is a soccer task in which two teams of simulated autonomous agents compete to score goals. Each agent receives its own state sensations and must independently select its own actions. HFO is naturally characterized as an episodic multiagent POMDP because of the sequential partial observations and actions on the part of the agents and the well-defined episodes which culminate in either a goal being scored or the ball leaving the play area. The following subsections introduce the low-level state and action space used by agents in this domain.

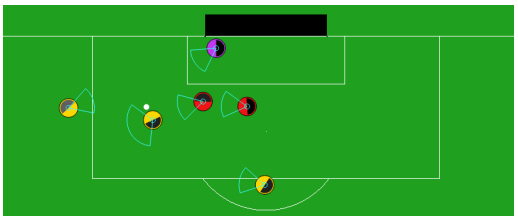


Figure 1: 3v3 Half Field Offense: Yellow offense agents search for an opening in the defensive formation. Red defenders and purple keeper strive to intercept the ball or force it out of bounds. HFO is better understood by video than picture: 1v1 `https://vid.me/sNev`, 2v2 `https://vid.me/JQTw`, 3v3 `https://vid.me/1b5D`

State Space: The agent uses a low-level, egocentric viewpoint encoded using 58 continuously-valued features. These features are derived through Helio-Agent2D's [2] world model and provide angles and distances to various on-field objects of importance such as the ball, the goal, and the other players. Figure 2 depicts the perceptions of the agent. The most relevant features include: Agent's position, velocity, and orientation, and stamina; Indicator if the agent is able to kick; Angles and distances to the following objects: Ball, Goal, Field-Corners, Penalty-Box-Corners, Teammates, and Op-

ponents. A full list of state features may be found at `https://github.com/LARG/HFO/blob/master/doc/manual.pdf`.

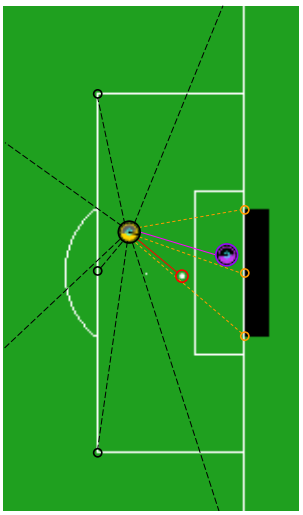


Figure 2: **RoboCup-2D State Representation** uses a low-level, egocentric viewpoint providing features such as distances and angles to objects of interest like the ball, goal posts, corners of the field, and opponents.

Action Space: HFO features a low-level, parameterized action space. There are four mutually-exclusive discrete actions: Dash, Turn, Tackle, and Kick. At each timestep the agent must select one of these four to execute. Each action has 1-2 continuously-valued parameters which must also be specified. An agent must select both the discrete action it wishes to execute as well as the continuously valued parameters required by that action. The full set of parameterized actions is:

-*Dash*(power, direction): Moves in the indicated direction with a scalar power in $[0, 100]$. Movement is faster forward than sideways or backwards.

-*Turn*(direction): Turns to indicated direction.

-*Tackle*(direction): Contests the ball by moving in the indicated direction. This action is only useful when playing against an opponent.

-*Kick*(power, direction): Kicks the ball in the indicated direction with a scalar power in $[0, 100]$.

Instead of tackling the full team-based HFO problem, we focus on a single agent that is first tasked with scoring on an empty goal and later with scoring on a goalie. To begin each episode, the agent and ball are positioned randomly on the offensive half of the field. The agent must first locate and approach the ball, then dribble towards the goal, and kick on target to score. Since there is no dribble action, the agent learns its own sequence of dashes and short kicks to move the ball in a desired direction without losing possession.

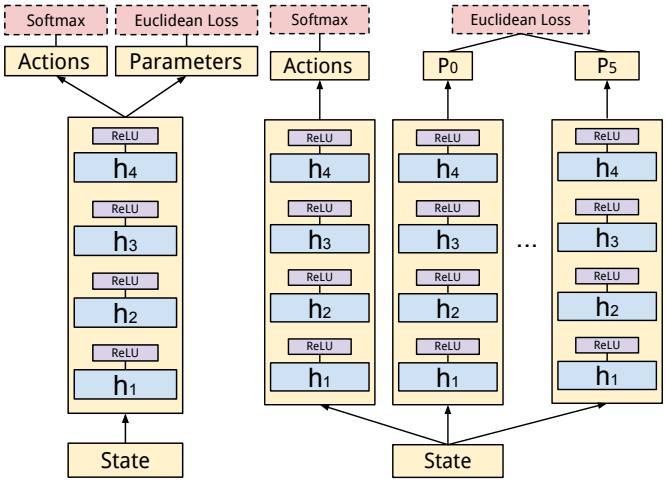
Having introduced the HFO domain, we now focus on the networks and training methods which make imitation learning possible.

4. MIMIC NETWORK

The success of imitation learning critically depends on the choice of policy representation for the learner. Too simple a representation limits the complexity of policies that may be learned; too complex a representation runs the risk of overfitting a limited supply of training data. We choose to use a single deep neural network to represent the policy for selecting both discrete actions and continuous parameters. This network is referred to as the mimic. The mimic takes as input a vector of continuous state features and returns a parameterized action which can be executed in the game.

More specifically, the mimic network uses two output layers – one outputs probabilities over discrete actions, and the other outputs continuous values over parameters. These two output layers may be interpreted by first selecting the discrete action with highest probability, and then reading the continuous parameters associated with that action. Beyond that, many choices exist in how to structure the intermediate layers between the inputs and outputs. After experimenting with several different architectures (see Table 1), we describe two successful networks: a unified and a separated network.

Unified Mimic Network: Figure 3a introduces the architecture of the unified mimic network: the input to the neural net consists of 56 state features. Next are four hidden layers, each followed by a rectifier nonlinearity (ReLU) with negative slope 0.01. The hidden layers $h_1 \dots h_4$ are fully-connected with 1000, 512, 200, and 64 units respectively. The ouput from the final hidden layer h_4 is divided into two separate fully-connected linear layers, one for the four discrete actions and the other the six action-parameters.



(a) Unified Mimic (b) Separated Mimic

Figure 3: **The Mimic Network** features dual classification/regression loss layers and either shares parameters (left) or features separate towers for the discrete actions and each of the parameters (right). Dashed loss layers are only included during training time and not during inference.

Training Architecture: In order to train the mimic network to output both probabilities over discrete actions and continuous parameters, we jointly minimize dual loss functions. The discrete actions are trained using a Multinomial, Cross-Entropy (Softmax) Loss L_A between the mimic's probability of selecting each discrete action $\hat{a}$ and the teacher's choice of action a. Equation 1 shows the form of this loss when applied to a minibatch of N examples. The action-parameters are trained using a Regression Loss (Euclidean Loss) L_P computed over the action-parameters output by the mimic $\hat{p}$ and the teacher p (Equation 2). Since the mimic's output layer contains parameters for all discrete ac-

tions, but the teacher chooses only single discrete action, we do not compute loss (or provide gradients) for the parameters not associated with the teacher's selected action.

In Equation 3, the mimic network, parameterized by θ, is updated using a step of size α in the direction that minimizes both losses. The parameter β trades off between the two losses. In the experiments described, $\beta = 0.5$ meaning that both loss functions contributed equally to the gradients flowing through the common layers of the network. Further exploration of methods for adaptively setting β is left for future work.

$$L_A = -\frac{1}{N}\sum_{n=1}^{N} \log P(a_n|\hat{a}_n) \quad (1)$$

$$L_P = -\frac{1}{2N}\sum_{n=1}^{N} \|p_n, \hat{p}_n\|_2^2 \quad (2)$$

$$\theta_{i+1} = \theta_i + \alpha\big(\beta\nabla_\theta L_A(\theta_i) + (1-\beta)\nabla_\theta L_P(\theta_i)\big) \quad (3)$$

Separated Mimic Network: There are inherent potential drawbacks to the unified mimic network (Figure 3a). Particularly, the parameters of the shared hidden layers may be driven in opposite directions by the dual loss functions being optimized. One way to alleviate this concern is to provide separate paths for the gradient of each loss function to follow. This results in the separated mimic network. Shown in Figure 3b, this network features a separate set of hidden layers for the discrete actions and each of the action-parameters. The number of parameters in this network increases by a factor of seven, resulting in slower training and inference. However, the benefits of the separated mimic become apparent on the complex task of scoring on a goalie.

Having defined the architecture and training procedure for both training and inference, we now introduce the teacher agents whose policies will guide the learning process.

5. TEACHER AGENTS

Imitation learning requires a teacher policy to mimic. Fortunately, RoboCup 2D features a long history of competition and code releases by various teams. These provide a wealth of available teachers. This section introduces two teacher policies: one deterministic and the other stochastic. Both teacher-agents are hand-coded by human experts and are capable of localization, decision making, and scoring. We believe the selected teacher agents are broadly representative of the types of policies found in RoboCup and expect that the imitation learning results presented would generalize to other teacher agents.

Deterministic Agent: The simpler of the two agents, the deterministic agent relies on a fixed strategy to score on an empty goal: it *Turns* towards the ball whenever the angle to the ball is higher than a threshold of 10 degrees. If facing the ball, it *Dashes* forward with full (power 100) speed. Upon reaching the ball, it *Kicks* with full power towards the center of the goal. If the kick does not result in a goal, the agent approaches the ball for another attempt. This teacher represents a baseline: the simplest scoring policy to mimic.

Stochastic Agent: The stochastic teacher agent uses a policy derived from Helios, the 2012 RoboCup 2D champion team [2]. This policy is designed to coordinate with a full set of teammates and play against a full set of opponents. Thus, it is more sophisticated than the deterministic policy above. Moreover, the precise action selected at each timestep varies according to internally-seeded random number generators, resulting in a stochastic policy which is significantly harder to mimic than the deterministic policy. Figure 4 shows a sample trajectory.

Both teacher policies exhibit near-optimal performance for the empty-goal task with stochastic teacher scoring 96% of the time and the deterministic teacher scoring 99% of the time. Both take 72 steps on average to score. The difference between these two teachers becomes clear when a goalie is added to the task; in this case the stochastic teacher scores 71% of the time while the deterministic scores only 3.5% of the time.

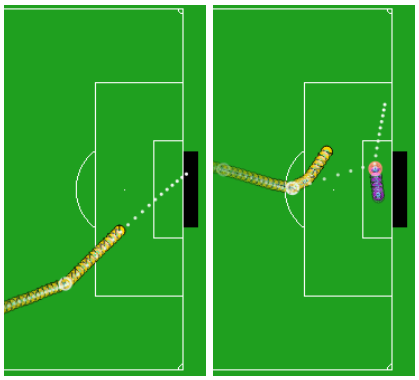


Figure 4: Left: without a goalie, the stochastic teacher (Helios-Agent2D) is free to take a direct path towards scoring. Right: when opposed by a goalie, even this policy can't prevent the goalie from occasionally deflecting the shot.

6. TRAINING PROCEDURE

Throughout this paper the same training procedure is applied; Only the dataset or network architecture is varied. Caffe [13] is used to train the deep mimic networks in conjunction with the AdaDelta [26] adaptive learning rate method. A momentum of 0.95 and base learning rate of 1 are used. Training continues for 30 epochs over a dataset containing 15,000 episodes of play by a teacher, roughly one million experience tuples in all. Each training iteration processes a minibatch of 32 examples in parallel. The learning rate is not reduced as we observe no evidence of overfitting.

7. SCORING ON AN EMPTY GOAL

The first, and simpler, of the tasks examined by this paper is scoring on an empty goal. This task begins by placing the agent and ball at different random locations on the offensive half of the field. The agent must first locate and move to the ball, then dribble it towards the goal, and shoot on target. If the agent kicks the ball out of bounds or fails to gain possession of ball within 100 timesteps, the trial ends in failure. Successful trials typically require a specific sequence of between 60 and 80 actions.

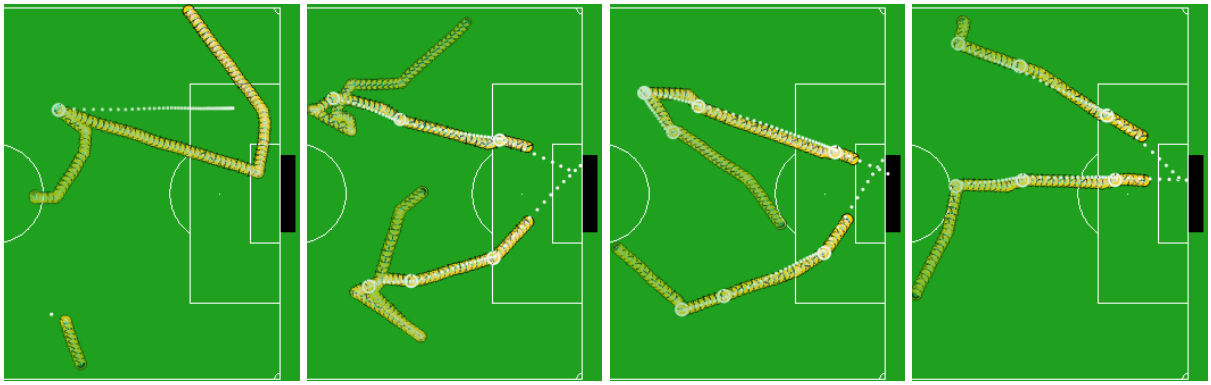
We explore the performance of the unified network mimicking the stochastic teacher as a function of the complexity of the underlying deep neural network. Specifically we examine unified mimic networks using 1-4 hidden layers with a varying number of hidden units in each layer. Results in

Network Structure	Loss: Softmax Loss	Loss: Euclidean Loss	Discrete Actions Acc: Dash Acc	Discrete Actions Acc: Turn Acc	Discrete Actions Acc: Kick Acc	Action Parameter Deviation: Dash Power	Action Parameter Deviation: Dash Angle	Action Parameter Deviation: Turn Angle	Action Parameter Deviation: Kick Power	Action Parameter Deviation: Kick Angle	Real Game: Score Percent	Real Game: Average Trial Time
256	0.1753	62.87	99.13	60.59	97.23	0.4475	0.8364	16.29	8.714	5.026	28.1	147.2
256-32	0.1434	50.06	99.28	61.73	97.47	0.5586	0.6943	13.59	8.680	4.668	37.5	120.3
256-100-32	0.1452	43.50	99.26	65.54	97.56	0.5346	0.5446	10.04	8.280	4.659	81.3	90.09
500-256-100-32	0.1407	40.65	99.19	71.08	97.88	0.5028	0.6710	7.745	8.111	4.308	93.8	78.89
1000-512-200-64	0.1366	42.52	99.39	72.16	98.28	0.6150	0.6702	7.163	8.072	4.339	96.9	76.54

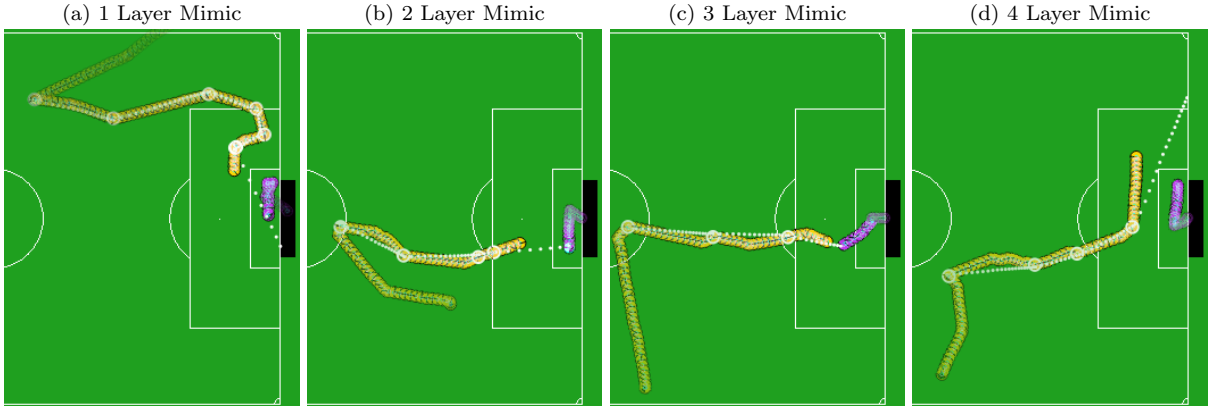
Table 1: **Mimicking stochastic teacher on empty goal task**: From left to right: Softmax Loss L_A over discrete actions; Euclidean Loss L_P over action-parameters; Discrete Action Accuracy shows, for each action, how frequently the mimic selected the same discrete action as the teacher. Action Parameter Deviation depicts the L1-norm between the mimic and teacher's parameters. The last columns show how frequently the mimic successfully scores goals and the average number of steps required.

Arch	Teacher	Goalie	Softmax Loss	Euclidean Loss	Dash Acc	Turn Acc	Kick Acc	Dash Power	Dash Angle	Turn Angle	Kick Power	Kick Angle	Score Percentage	Average Trial Time
UNI	DET	N	0.0331	11.69	99.44	91.50	99.94	0.1282	0.2374	1.645	0.7089	2.4378	97(99.8)	72.78(72.67)
UNI	STO	N	0.1366	42.52	99.39	72.16	98.28	0.6150	0.6702	7.163	8.072	4.339	92.4(96.4)	76.54(72.84)
UNI	STO	Y	0.1993	197.2	98.78	59.64	98.27	1.356	0.6923	14.21	14.30	33.39	4(71.5)	97.78(80.92)
SEP	STO	Y	0.098	175.26	99.37	78.01	99.23	1.237	.771	8.48	13.18	32.45	12(71.5)	84(80.92)

Table 2: **Difficulty of mimicking depends on complexity of teacher policy**: Performance of mimicking deterministic (DET) and stochastic (STO) teachers. It proves harder to mimic the stochastic teacher than the deterministic one, and hardest when the mimicking the stochastic teacher playing against a goalie. Parentheses show baseline performance of the corresponding teacher policy. Performance improves across the board when using the separated mimic (SEP) rather than the unified (UNI).



(a) 1 Layer Mimic (b) 2 Layer Mimic (c) 3 Layer Mimic (d) 4 Layer Mimic



(e) Teacher (f) Mimic (g) Mimic (h) Mimic

Figure 5: **Visualizing learned policies**: The first row depicts the trained mimic using a neural network featuring 1 to 4 hidden layers. With too few hidden layers, the policy fails to locate and approach the ball. As more hidden layers are added, the approach becomes smoother and the shots better targeted. **The second row** depicts the complexity of the policy required to score on a goalie. In (e), the teacher dribbles towards the goal and doubles back before shooting at edge of the goal. The unified mimic cannot master this task and learned policies (f-h) result in the ball being captured, intercepted, and kicked out of bounds.

Table 1 indicate that both depth and width influence the representational capacity of the network and its ability to successfully mimic the teacher, confirming the general intuition that deeper, more complex networks yield improved performance, up to a point. Additionally, the first row of Figure 5 visualizes the improvement in learned policies as a function of the number of hidden layers in each network.

Examining the dual loss functions in Table 1 shows that as complexity is added to the mimic network, the accuracy of mimicking both the discrete actions of the teacher and the continuous parameters increases. Of the discrete actions, Dash and Kick start off with high accuracy, leaving little room for improvement. The Turn action starts with low accuracy and shows the most improvement. Similarly, of the parameterized actions, Turn-Direction shows the most relative improvement, followed by Kick-Power and Kick-Direction. Thus the Dash and Kick actions are relatively straightforward and the more complex networks use the additional representation power to make better decisions regarding when and how to Turn. Dash-Power gets progressively less accurate in the more complex networks. This does not affect overall scoring performance, which monotonically increases as a function of network complexity.

The deterministic teacher proves much easier to mimic than the stochastic one. Table 2 shows that overall loss for both discrete actions and continuous parameters are approximately three times smaller when mimicking the deterministic teacher compared to the stochastic teacher. This translates into a mimic that can more reliably score in shorter amounts of time. Regardless, on this task, both policies are largely successful, with the worst of the two conserving over 95% of the scoring potential of the teacher.

8. SCORING ON A GOALIE

Far more difficult than the task of scoring on an empty goal is the challenge of scoring on an active goalie. More than just requiring the player to kick around the goalie, this task requires a complex dance of positioning and baiting to draw the keeper out and strike when the goal is open. An example of the sophisticated strategy used to score on a goalie is shown in Figure 5e. The nondeterministic policy used by the keeper is from the winning Helios-Agent2D codebase [2] and strikes an effective balance between repositioning itself near the goal to minimize open shots and moving out to tackle the ball. In this task, agent uses an augmented state representation with eight additional features encoding the goalkeeper's angle, distance, velocity, and orientation.

Table 3 shows the performance of teachers and mimics on this task. The stochastic teacher (Helios-Agent2D) proves quite capable, with a scoring percentage of 70%. The deterministic teacher, whose policy is unaware of the goalie, ends up getting the ball captured by the goalie 96.5% of the time. Mimicking the stochastic policy using the Unified Network proves highly difficult, and in terms of scoring goals, the resulting mimic fares no better than the deterministic teacher (Figures 5f-5h visualize attempts to score by this policy). However, the separated network proves more capable with higher accuracy and lower deviation in nearly all categories (Table 2) as well as three times the amount of goals scored. We hypothesize that the lack of shared layers between action-parameters in the separated network allowed each parameter to achieve a higher degree of specialization than possible in the unified network.

Policy	Goal	CAP	OOB	OOT
Stochastic Teacher	143	34	20	3
Deterministic Teacher	7	193	0	0
Unified Mimic	8	138	49	5
Separated Mimic	24	123	44	9

Table 3: **Scoring on a goalie**: The 200 trials of each policy end with the goalie capturing the ball (CAP), the ball going out of bounds (OOB), or running out of time (OOT). Learning from the stochastic teacher, the separated mimic network performs better on this complex task than the unified, and both outperform the deterministic teacher.

9. DISCUSSION AND CONCLUSIONS

The promise of imitation learning lies only partially in the learned policy. More interesting is the ability to convert between different policy representations. Starting from a human engineered teacher policy represented by thousands of lines of source code, imitation learning allows us to derive a mimic policy in the form of neural network that captures a subset of the behaviors of the teacher. This neural network representation allows the efficient computation of policy-gradients with respect to network parameters. In contrast, estimating these gradients in the original teacher policy would be time-consuming if not impossible and would first require human expertise to determine and expose the tuneable parameters of the code base.

We believe that the mimicked policies described in this paper will form a solid foundation for future work on policy improvement, perhaps using the policy gradients provided by a critic network. To this end, source code and full information about the feature and action spaces for the Half Field Offense domain is available at `https://github.com/LARG/HFO` and for mimic-learning at `https://github.com/mhauskn/dqn-hfo/tree/mimic`.

In summary, this paper demonstrates, for the first time, successful imitation learning of goal scoring policies in a parameterized action space. Neural networks trained with a dual classification/regression loss prove capable of high-fidelity imitation learning from goal-scoring stochastic and deterministic teacher policies. This work represents a step in the direction of learning rather than programming complex policies and confirms that deep neural networks show much promise as function approximators for these policies. Our ongoing research seeks to incorporate learning to further improve the mimic policies.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-1330072, CNS-1305287), ONR (21C184-01), AFRL (FA8750-14-1-0070), AFOSR (FA9550-14-1-0087), and Yujin Robot. Additional support from the Texas Advanced Computing Center, and Nvidia Corporation.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International*

Conference on Machine Learning, ICML '04, pages 1–, New York, NY, USA, 2004. ACM.

[2] Hidehisa Akiyama. Agent2d base code, 2010.

[3] David Andre and Astro Teller. Evolving Team Darwin United. *Lecture Notes in Computer Science*, 1604:346–353, 1999.

[4] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[5] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *Proc. 14th International Conference on Machine Learning*, pages 12–20. Morgan Kaufmann, 1997.

[6] J Andrew Bagnell and Stéphane Ross. Efficient Reductions for Imitation Learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010*, volume 9, pages 661–668, 2010.

[7] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, 2004.

[8] Bruno Castro da Silva, Gianluca Baldassarre, George Konidaris, and Andrew G. Barto. Learning parameterized motor skills on a humanoid robot. In *ICRA*, pages 5239–5244. IEEE, 2014.

[9] Carlos Guestrin, Milos Hauskrecht, and Branislav Kveton. Solving factored mdps with continuous and discrete variables. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 235–242, Arlington, Virginia, United States, 2004. AUAI Press.

[10] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3338–3346. Curran Associates, Inc., 2014.

[11] Matthew Hausknecht and Peter Stone. Learning powerful kicks on the aibo ers-7: The quest for a striker. In *Proceedings of the RoboCup International Symposium 2010*. Springer Verlag, 2010.

[12] Matthew J. Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *CoRR*, abs/1511.04143, 2015.

[13] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[14] Nate Kohl and Peter Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conference on Artificial Intelligence*, pages 611–616, July 2004.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *ArXiv e-prints*, September 2015.

[16] Patrick MacAlpine, Mike Depinet, and Peter Stone. UT Austin Villa 2014: RoboCup 3D simulation league champion via overlapping layered learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, January 2015.

[17] Warwick Masson and George Konidaris. Reinforcement learning with parameterized actions. *CoRR*, abs/1509.01644, 2015.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[19] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR*, abs/1511.06342, 2015.

[20] Nathan Ratliff, David Bradley , J. Andrew (Drew) Bagnell, and Joel Chestnutt. Boosting structured prediction for imitation learning. In B. Scholkopf, J.C. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007. MIT Press.

[21] Martin A. Riedmiller and Thomas Gabel. On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup. In *CIG*, pages 17–23. IEEE, 2007.

[22] StÃl'phane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 627–635. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[23] Scott Sanner, Karina Valdivia Delgado, and Leliane Nunes de Barros. Symbolic dynamic programming for discrete and continuous state MDPs. *CoRR*, abs/1202.3762, 2012.

[24] David Silver, James A. Bagnell, and Anthony Stentz. High performance outdoor navigation from overhead data using imitation learning. In Oliver Brock, Jeff Trinkle, and Fabio Ramos, editors, *Robotics: Science and Systems*. The MIT Press, 2008.

[25] Luis Gustavo Vianna, Scott Sanner, and Leliane Nunes de Barros. Bounded approximate symbolic dynamic programming for hybrid MDPs. *CoRR*, abs/1309.6871, 2013.

[26] Matthew D. Zeiler. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.