Learning Temporal Strategic Relationships using Generative Adversarial Imitation Learning

Tharindu Fernando Simon Denman Sridha Sridharan Clinton Fookes
Image and Video Research Laboratory, Queensland University of Technology (QUT), Australia
t.warnakulasuriya,s.denman,s.sridharan,c.fookes@qut.edu.au

ABSTRACT

This paper presents a novel framework for automatic learning of complex strategies in human decision making. The task that we are interested in is to better facilitate long term planning for complex, multi-step events. We observe temporal relationships at the subtask level of expert demonstrations, and determine the different strategies employed in order to successfully complete a task. To capture the relationship between the subtasks and the overall goal, we utilise two external memory modules, one for capturing dependencies within a single expert demonstration, such as the sequential relationship among different sub tasks, and a global memory module for modelling task level characteristics such as best practice employed by different humans based on their domain expertise. Furthermore, we demonstrate how the hidden state representation of the memory can be used as a reward signal to smooth the state transitions, eradicating subtle changes. We evaluate the effectiveness of the proposed model for an autonomous highway driving application, where we demonstrate its capability to learn different expert policies and outperform state-of-the-art methods. The scope in industrial applications extends to any robotics and automation application which requires learning from complex demonstrations containing series of subtasks.

KEYWORDS

Generative Adversarial Imitation Learning; Autonomous Driving; Long term Planing with Autonomous Agents

ACM Reference Format:
Tharindu Fernando Simon Denman Sridha Sridharan Clinton Fookes. 2018. Learning Temporal Strategic Relationships using Generative Adversarial Imitation Learning. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

The requirement of a predefined reward function limits the practical application of Reinforcement Learning (RL) methodologies to complex problems. Hand engineering a reward function in a fully observable environments such as arcade style computer games [6, 26] is possible, yet becomes far more difficult for complex tasks such as autonomous driving, where we have to consider multiple factors (i.e safety, fuel economy, travel time, etc) that affect the reward signal.

To counter this problem researchers widely utilise imitation learning procedures [36, 40, 42], where the model learns the policy following expert demonstrations, instead of directly learning the policy through the reward signal. Most recently a Generative Adversarial Imitation Learning (GAIL) approach was proposed in [23] and further augmented in [30] to automatically discover the latent factors influencing human decision making. These approaches offer greater flexibility as they result in a model-free imitation learning platform; however, we observe that existing GAIL approaches generate short term responses instead of attaining long term planning.

This work propose a novel data driven model for executing complex tasks that require long term planning. Instead of deriving a policy while only observing the current state [23, 27], we model the state temporally with the aid of external memory modules. This results in the proposed approach having both the ability to discriminate between the sub level tasks, and determine their sequential relationships to successfully complete an overall task; as the proposed method automatically determines the optimal action to perform in the current temporal context.

In a typical real world environment, a task such as driving involves a series of sub tasks including, turn, overtake, follow a lane, merge into a free way, . . . , etc. Hence a standard demonstration from a human expert would include a mixture of such sub tasks and each would have different initial states. Even though [23, 27, 30] are highly effective at discriminating between sub-tasks, we observe that these methods fail to identify the temporal context in which the task was performed due to inheritant architectural deficiencies. The neural network architectures proposed in [23, 27] simply map the current state to an action without considering the temporal context (i.e. what has happened previously), which may impact the interpretation of the current state. Understanding temporal context is essential to interpreting how the different sub tasks are sequentially linked to each other. As such at simulation time those methods [23, 27, 30] cannot decide on the optimal action to perform in the current environmental setting in order to successfully compete the overall task, as they do not have the capacity to oversee the task in it's entirety, owing to the problem being presented a single state embedding at a time.

Furthermore, depending on the skill of the expert demonstrator, social values and individual likes and dislikes, different experts may vary their behaviour when presented similar contexts. For instance in the same autonomous driving setting, a certain expert may decide to give way to the merging traffic where as another driver may not and show aggressive behaviour. We show that these different strategies of different experts for similar context can be disentangled by attending over historical states of the expert demonstration, and learning different ways to behave under similar conditions.

We incorporate two separate memories, namely global and local memories, in order to capture these two tiers of the context. The local memory is used to model the expert behaviour during each

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

demonstration, and is reset at the end of the demonstration. In contrast, the global memory maps expert preferences during a batch of demonstrations. Local memory allows us to attend over different sub tasks within a particular trajectory and learn how these sub level tasks are sequentially linked to each other. We attend over the global memory to understand how different experts vary their behaviour under similar context and interpret how these different choices render different policies.

Our experimental evaluations with applications to autonomous driving demonstrate the ability of the proposed model to learn the semantic correspondences in human decision making using complex expert demonstrations. The novel contributions of this paper are summarised as follows:

- We introduce a novel architecture for GAIL which captures the sequential relationships in human decision making encoded within complex expert demonstrations.
- We incorporate neural memory networks into the imitation learning problem where it learns to automatically store and retrieve important facts for decision making without any supervision.
- We utilise two memory modules, one for capturing sub task level dependencies and one for modelling social and behavioural factors encoded within diverse expert demonstrations.
- We demonstrate how the hidden state representation of the memory can be utilised as a powerful information queue to augment the reward signal and generate smooth state transitions between the sub tasks.
- We provide extensive evaluations of the proposed method with applications to autonomous driving using the TORCS [45] driving simulator, where the proposed method is capable of learning different expert policies and outperforms state-of-the-art methods.

2 PRELIMINARIES

An infinite horizon, discounted Markov decision process (MDP) can be represented as a tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho 0, \Upsilon)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ represents the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ denotes the transition probability distribution, $r : \mathcal{S} \rightarrow \mathbb{R}$ denotes the reward function, $\rho 0 : \mathcal{S} \rightarrow \mathbb{R}$ is the distribution of the initial state, and $\Upsilon \in (0, 1)$ is the discount factor. We denote a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, and π_E denotes the expert policy. An expert demonstration τ_E of length ζ is generated following π_E and is composed of state (s_t) and action (a_t) pairs,

$$\tau_E = [(s_1, a_1), (s_2, a_2), \dots, (s_\zeta, a_\zeta)]. \quad (1)$$

Then, the traditional GAIL [23] objective can be denoted as,

$$\min_\theta \max_w V(\theta, w) = \mathbb{E}_{\pi_\theta}[\log D_w(s, a)] + \mathbb{E}_{\pi_E}[\log D_w(s, a)], \quad (2)$$

where policy π_θ is a neural network parameterised by θ which generates the policy imitating π_E, and D_w is the discriminator network parameterised by w which tries to distinguish state-action paris from π_θ and π_E. $\mathbb{E}_\pi[f(s, a)]$ denotes the expectation of f over state action pairs generated by policy π.

In [30] the authors emphasise the advantage of using Wasserstein GANs (WGAN) [4] over the traditional GAIL objective as this method is less prone to vanishing gradient and mode collapse problems. The WGAN objective transferred to GAIL can be written as,

$$\min_\theta \max_w V^*(\theta, w) = \mathbb{E}_{\pi_\theta}[D_w(s, a)] + \mathbb{E}_{\pi_E}[D_w(s, a)]. \quad (3)$$

In this approach the discriminator assigns scores to to its inputs, trying to maximise the score values for the expert policy π_E while minimising the score for generated policy π_θ, in contrast to Eq. 2 which tries to classify the two policies.

3 RELATED WORK

Imitation learning focuses on building autonomous agents that can acquire skills and knowledge from the observed demonstrations of experts [3, 7, 40].

There exists two major lines of work for imitation learning: Behavioural Cloning (BC) [35, 38] which performs direct supervised learning to match observations to actions; and Inverse Reinforcement Learning (IRL) [1, 18, 32] which tries to recover the reward function followed by the experts assuming that the experts follow an optimal policy with respect to the reward function.

Even with the recent advances in deep learning techniques, BC approaches tend to suffer from the problem of cascading errors [37] during simulations [27, 31]. For instance, BC based autonomous driving methods [2, 8, 13, 21] rely heavily on larger training databases in order to generalise for different states, yet fail in practise due to small inaccuracies compounding sequentially [27].

In contrast IRL methods generalise more effectively [17] and have been applied multiple times for modelling human driving behaviour [20, 39].

Yet learning with IRL is computationally expensive as one should first recover the reward function and then apply RL techniques with that reward function in order to find the policy. Hence a third line of work has emerged, namely GAIL, which attempts to imitate the expert behaviour through direct policy optimisation rather than learning the reward function first.

Some recent studies [27, 30] have considered the problem of modelling human highway driving behaviour at a simpler sub task level, but haven't considered learning to perform complex tasks which are composed of a mixture of sequentially linked sub tasks.

We draw our inspiration from [11, 12, 15, 34] where an external memory module is used to capture the current context of the autonomous agent. However these memory architectures are problem specific. For instance in [34] the authors design a memory module to store a map of explored areas for a 3D navigation task where as [11] uses a memory module to store coordinates of the blocks in robot block stacking experiments. Consequently, the structure and operations of the memory modules are specifically tailored for those tasks.

Distinctively, we propose a generalised framework for GAIL which uses memory modules to store local and global salient information and automatically learn the temporal relationships.

Similar to [11, 34] we also rely on a soft attention mechanism [5] to compare the current state representation with the content of the memories, which is widely a used mechanism to learn such dependencies in a variety of problem domains, including human navigation [14, 16], image captioning [46], neural machine translation [28].

4 ARCHITECTURE

This section describes our approach to disentangle the mixture of expert policies at the task level and learn the different strategies employed by different experts in order to complete various sub goals. The high level architecture of the proposed Memory Augmented Generative Adversarial Imitation Learning (MA-GAIL) network is shown in Fig. 1. In contrast to traditional GAIL [23, 27] systems, which utilise only a discriminator (D_w) and a policy generator π_θ, we additionally use two memory modules, named local memory M^L and global memory M^G.

Memory stores important facts from the expert trajectories and attends to them systematically to retrieve facts relevant to the current state. For instance in autonomous driving, visual scene information provides powerful cues and heavily contributes to decision making. Hence if our memory is composed of scene embeddings, by attending over the relevant memory embeddings with similar scene dynamics to the current state, one can retrieve important clues to aid decision making.

Furthermore, as outlined in previous sections, expert decision making in similar situations may vary significantly due to numerous social and behavioural factors. This inspires the need for a global memory to capture, compare and contrast the different expert behaviours under similar contexts and learn different ways to behave.

In the following subsection we describe the neural memory architecture that we use for global and local memory modules and later in Sec. 4.2 we explain how we incorporate those memory modules with the GAIL objective.

4.1 Neural Memory Module

As shown in Fig. 1 each memory module is composed of 1) a memory stack to store the relevant data; 2) a read controller to query content from the memory stack; 3) a write controller to compose and update the memory; and 4) an output controller to send through the output of the read operation.

Formally, let $M \in \mathbb{R}^{k \times l}$ be the memory stack with k memory slots, and l be the embedding dimension of the state representation s. The representation of the memory at time instance $t-1$ is given by M_{t-1}. First the read controller uses a read function to generate a query vector q_t such that,

$$q_t = f_r^{LSTM}(s_t), \tag{4}$$

where f_r^{LSTM} is a read function which uses Long Short Term Memory (LSTM) [24] cells and s_t is the current state of the episode.

Then we generate a score vector a_t over each memory slot representing the similarity between the current memory state M_{t-1} and the generated query vector q_t by attending over the memory slots such that,

$$a_{(t,j)} = q_t M_{(t-1,j)}, \tag{5}$$

where $M_{(t-1,j)}$ denotes the content of the j^{th} memory slot of the memory at $t-1$ where $j = [1, \dots, k]$. Then the score values are normalised using soft attention [5], generating a probability distribution over each memory slot as follows,

$$\alpha_{(t,j)} = \frac{\mathbb{E}(a_{(t,j)})}{\sum_{j=1}^{k} \mathbb{E}(a_{(t,j)})}. \tag{6}$$

Now the output controller can retrieve the memory output for the current state by,

$$m_t = \sum_{j=1}^{k} \alpha_{(t,j)} M_{(t-1,j)}. \tag{7}$$

Then we generate a vector $\grave{m}_t$ for the memory update by passing the output of the memory through a write function f_w^{LSTM} composed of LSTM memory cells,

$$\grave{m}_t = f_w^{LSTM}(m_t), \tag{8}$$

and update the memory using,

$$M_t = M_{t-1}(I - \alpha_t \otimes e_k)^T + (\grave{m}_t \otimes e_l)(\alpha_t \otimes e_k)^T, \tag{9}$$

where I is a matrix of ones, $e_l \in \mathbb{R}^l$ and $e_k \in \mathbb{R}^k$ are vectors of ones and $\otimes$ denotes the outer product which duplicates its left vector l or k times to form a matrix.

The functions f_r^{LSTM} and f_w^{LSTM} use the gated operations defined in [24] and shown below in Equations 10 to 15. Instead of simply generating a query to read, or an update vector to hard write the current memory content, LSTM based read and write operations allow us to retain long term histories of those operations and decide upon how much the query or memory update vector should differ based on the historical read/ write operations. We define the operations of f_r^{LSTM} and f_r^{LSTM} as,

$$f_t = \sigma(w_f[h_{t-1}, x_t]), \tag{10}$$

$$i_t = \sigma(w_i[h_{t-1}, x_t]), \tag{11}$$

$$\grave{c}_t = \tanh(w_c[h_{t-1}, x_t]), \tag{12}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \grave{c}_t, \tag{13}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t]), \tag{14}$$

$$h_t = o_t \cdot \tanh(c_t), \tag{15}$$

where $\sigma(\cdot)$ is the sigmoid activation function and w_f, w_i, w_c, w_o are the weight vectors for forget, input, cell state and output gates of the LSTM cell. h_{t-1} denotes the hidden state of the LSTM cell at time instance $t-1$ where as c_{t-1} denotes the cell state of the LSTM cell at the same time instance. The input to the cell at time t is given as x_t. When applied these equations to the proposed approach, for Eq. 4 the LSTM input is the current state s_t, and for Eq. 9 it is the memory output vector m_t. The LSTM cell output is denoted o_t in Eq. 14 and is replaced with q_t and $\grave{m}_t$ for Eq. 4 and Eq. 9 respectively.

4.2 Memory Augmented Generative Adversarial Imitation Learning (MA-GAIL)

We deploy two instances of the neural memory module introduced in Sec. 4.1, functioning as local and global memories denoted as M_t^L and M_t^G respectively in Fig. 1. Note that via passing the state embeddings to the memory, our autonomous agent can question the current temporal context that it is in.

For the local memory, as the state embeddings for similar sub tasks would have similar features, they would generate stronger activations allowing the model to estimate the sub task that it is currently in.

In contrast, global memory contains a batch of expert trajectories. Hence the memory output m_t^G of the global memory would have

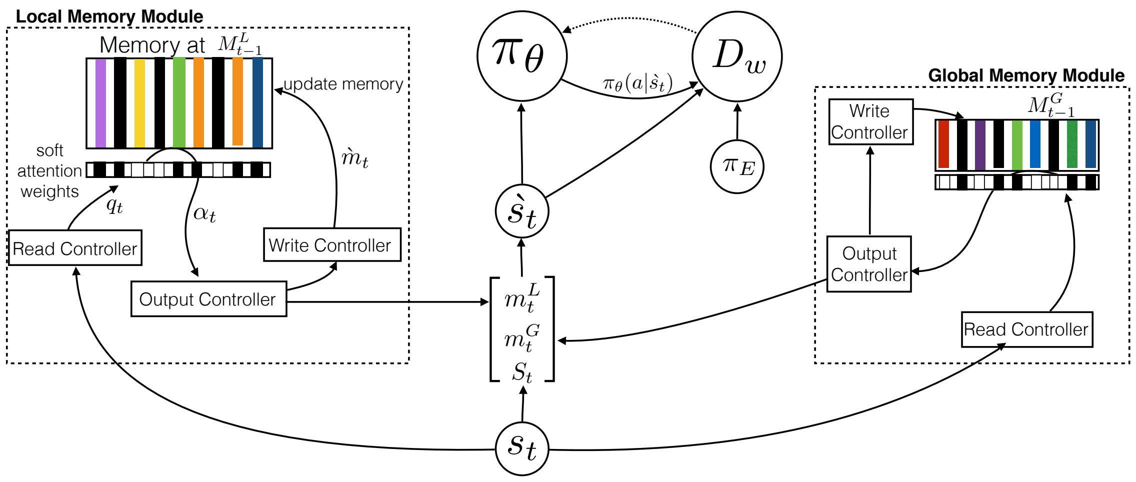


Figure 1: The proposed MA-GAIL model: The model is composed of a policy generator π_θ, a discriminator D_w and two memory modules for capturing local M^L and global M^G contexts. The input state embedding s_t triggers a read operation in both memories which generates a query q_t to question the temporal relationships between the current state and the content of the memories, M^L_{t-1} and M^G_{t-1}. Then using soft attention, we generate a weighted distribution α_t quantifying the similarity between each memory slot and the query vector. This is used by the output controller to generate a memory output and produces m^L_t and m^G_t from the respective memory modules. The write control module of each memory updates the respective memories accordingly. We directly concatenate the m^L_t and m^G_t vector representations together with state embedding s_t to generate an augmented state representation $\grave{s_t}$. The policy generator accepts this as the input and outputs an action for that particular state. The discriminator also uses the augmented state in order to provide feedback to guide the policy generator using the expert policy pi_E as the guideline.

distributions for different ways of how an expert would react in the current state, capturing their different strategies.

In the proposed approach we directly concatenate the physical state of the system s_t with the relevant memory outputs, m^L_t and m^G_t, from local and global memories respectively, to produce an augmented state,

$$\grave{s}_t = \begin{bmatrix} s_t \\ m^L_t \\ m^G_t \end{bmatrix}.$$

Training a policy $\pi_\theta(a|\grave{s}_t, c)$ on the augmented state representation generates a dynamic policy which can fully utilise memory states to capture expert dynamics. We would like to note that we do not provide any guidance on what memory content should be extracted or updated. The read/ write controllers on each memory module learn a distribution over the input states and learn salient factors that they should focus on in order to maximise the overall GAIL performance.

Now we modify the GAIL objective in Eq. 3 with the augmented state to obtain the proposed MA-GAIL,

$$\min_\theta \max_w \grave{V} = \min_{\theta,w} \max_w \mathbb{E}_{\pi_\theta}[D_w(\grave{s}, a)] - \mathbb{E}_{\pi_E}[D_w(\grave{s}, a)], \tag{16}$$

with π_θ updated by the TRPO method introduced in [41] and the discriminator D_w updated with RMSprop [43].

4.3 Reward augmentation with memory states

When designing autonomous agents it is desirable to have smooth transitions between the sub tasks. For example in an autonomous driving setting, when exiting the freeway from an exit ramp, the agent should learn to smoothly shift between the sub tasks such as merge, slow down and follow the exit ramp; all the while avoiding sudden braking, accelerations and turns.

A naive way to solve this problem is to penalise sudden action changes, but this limits the capability of the agent. For instance by limiting steering angels between consecutive actions we eradicate the capability of the agent to perform u-turns, or by controlling the acceleration we might reduce the capability of driving at high speed on a freeway. Similar problems exist if we penalise sudden state transitions. For the same autonomous driving example, where the state is composed of visual inputs of the road scene, scene changes such as illumination variations, merging traffic or changes in road conditions result in diverse changes in the state embeddings. Hence considering state embeddings alone leads to erroneous behaviour.

We overcome this problem by observing the changes in local memory states. The local memory generates temporal attention over a series of state embeddings, rectifying sudden fluctuations in state due to the various local factors listed above. Hence changes in local memory attention occur only due to the sub task changing as it captures salient aspects of the trajectory, and shifts in attention denoting the changes in the behaviour.

We monitor the dispersion between local memory outputs in consecutive time steps and penalise sudden changes. Let m^L_{t-1} and

m_t^L be the respective local memory outputs for input states s_{t-1} and s_t. We define a function f^* which extracts memory outputs following Eq. 4 to Eq. 7,

$$m_t^L = f^*(s_t) \tag{17}$$

Now we can define a penalty for memory state change using the cosine distance [33] between the consecutive memory states as,

$$\eta(\pi_\theta) = \mathbb{E}_{s_t, s_{t-1} \sim \pi_\theta}[cos(m_t^L - m_{t-1}^L)], \tag{18}$$

where $cos(m_t^L - m_{t-1}^L)$ is given by,

$$\begin{aligned} cos(m_t^L - m_{t-1}^L) &= 1 - \frac{m_t^L \cdot m_{t-1}^L}{||m_t^L||_2 ||m_{t-1}^L||_2} \\ &= 1 - \frac{\sum_{j=1}^{l} m_{(t,j)}^L m_{(t-1,j)}^L}{\sqrt{\sum_{j=1}^{l} (m_{(t,j)}^L)^2} \sqrt{\sum_{j=1}^{l} (m_{(t-1,j)}^L)^2}}, \end{aligned} \tag{19}$$

where $m_{(t,j)}^L$ and $m_{(t-1,j)}^L$ are the components of the memory vectors m_t^L and m_{t-1}^L and $j = [1, \ldots, l]$ where l is the embedding dimension of the state. Then the combined MA-GAIL objective with a reinforced objective for smoothed state transitions can be denoted as,

$$\min_\theta \max_w \grave{V} + \lambda_0 \eta(\pi_\theta), \tag{20}$$

where $\lambda_0 > 0$ is a hyper parameter which controls the tradeoff between imitation and reinforcement learning objectives.

5 EVALUATIONS AND DISCUSSION

In many real world applications, representing the state, s, in relation to visual inputs such as images or image sequences is highly beneficial. It is often inexpensive to obtain and highly informative [13, 30]. Furthermore, direct learning from visual inputs, rather than hand crafting the state features, enables the policy generator to be directly transferable from one domain to another. Hence we demonstrate the performance of the proposed method with applications to autonomous driving from visual inputs.

5.1 Environment setup

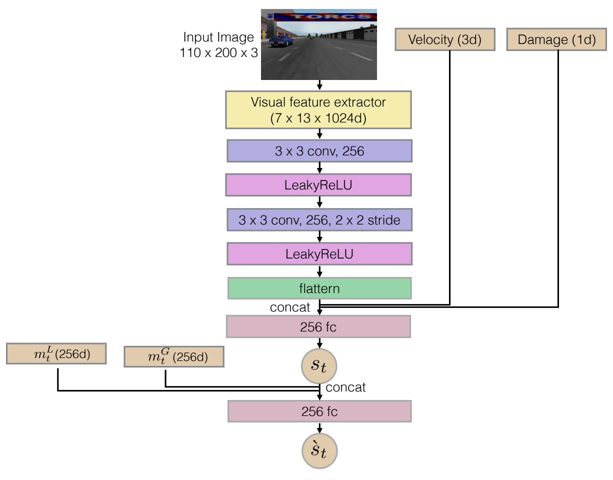
In [30] the authors provide an API for The Open Racing Car Simulator (TORCS) [45] similar to OpenAI Gym [6], which generates a realistic dashboard view and driving related information.

We collected human expert demonstrations by manually driving the vehicle along the racing track. We could not use the demonstrations provided in [30] as each of those episodes contains only a single sub level task such as overtaking and turning. We believe it is an over simplification of the real world driving task where a typical demonstration would contain a series of such sub level tasks.

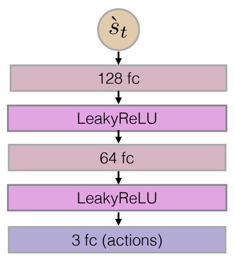
Therefore we collected 150 expert trajectories from 3 experts with each demonstration lasting 500 frames. Within each expert trajectory a series of driving tasks such as lane change, turns, staying within a lane, and overtaking are included.

5.2 Network structure

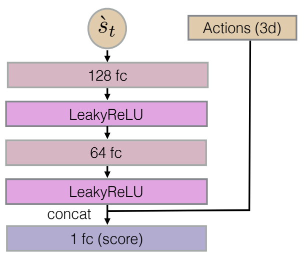
The TORCS environment outputs a dashboard camera view, the current speed in x, y and z dimensions of the car, and a real value



(a) State embedding



(b) Policy generator π_θ



(c) Discriminator D_w

Figure 2: Network architecture of the proposed MA-GAIL model. *conv* denotes a convolution layer, and *fc* denotes a fully connected layer. (a) State embedding: The input image is passed through a pre-trained visual feature extractor and later combined with velocity and current damage of the vehicle to generate a current state representation. We combine this with memory outputs m_t^L and m_t^G of local and global memories to generate the augmented state embedding. (b) Policy generator: accepts the augmented state embedding as the input and generates a 3 dimensional action to take at that particular state. (c) Discriminator: Uses the augmented state embedding and the action to output a score value to discriminate the policy that is generated by the input state-action pair. It gives a higher score value to expert policies π_E and a lower score to π_θ.

representing the damage of the car. Fig 2 illustrates our approach to combine these diverse information sources to a single state representation.

We use the Keras [9] implementation of the Deep Residual Network (DRN) [22] pre-trained on the ImageNet classification task [10], as our pre-trained feature extractor to extract salient information from the visual input. We pass each input frame through the DRN and extract the activations from the 40th activation layer. Then we apply a series of convolution operations to compress our spatial feature representation and construct a 256 dimension representation of the current state s_t.

We combine this representation with local m_t^L and global m_t^G memory outputs and generate an augmented state $\grave{s}_t$ by passing those embeddings through a series of fully connected layers.

Note that each memory stack M^L and M^G, is of dimension $l \times k$ where $l = 256$ is the embedding dimension of the state s_t, and k is the number of memory slots. For local memory we set $k = 500$ because each expert trajectory is composed of 500 frames. Global memory should retain information for a batch of expert demonstrations, hence $k = 500 \times 50$ where 50 is the batch size.

Our policy generator Fig 2 (b) uses this augmented state representation as its input and outputs a three dimensional action representing $[steering, acceleration, breaking]$.

The discriminator D_w, shown in Fig. 2 (c) also accepts the augmented state embedding ($\hat{s}_t$) along with the current action a_t and outputs a score value for the policy that generated input state action pair. As per [30] we first train the model with behavioural cloning using random weights, and the learned weights initialise the network for imitation learning

5.3 Evaluation of driving behaviour

5.3.1 Baseline models. We compare our model against 5 state-of-the-art baselines. The first baseline model we consider is a Static Gaussian (SG) model which uses an static gaussian distribution $\pi(G|s) = N(a|\mu, \sigma)$ which assumes an unchanged policy through out the simulation, and is fitted using maximum likelihood [44]. The second baseline is a Mixture Regression (MR) [29] model which is a gaussian mixture over the joint space of the actions and state features and is trained using expectation maximisation [19]. The third baseline we compare our model against is the GAIL-GRU model proposed in [27] which optimises a recurrent policy. The first 3 baselines only accept a hand crafted feature representation of the state. Hence, following a similar approach to [27] we extract the features listed in Tab. 1 and used the implementations provided by the authors of [27] and available online [1]. Our next baseline is the Info-GAIL model presented in [30] and uses a visual feature representation of the scene along with the auxiliary information (i.e velocity at time t, previous actions at time $t-1$ and $t-2$, damage of the car) to represent the current state. In order to emphasise the importance of the imitation learning strategy rather than simply cloning the behaviour of the expert using behavioural cloning, we also provide evaluations against the Behavioural Cloning (BC) method given in [30]. For the Info-GAIL model and BC model we use the implementation released by the authors [2].

5.3.2 Validation. To evaluate the relative performance of each model we simulate 500 frame length simulations 20 times each in identical environments. We also asked a human expert to drive the car in the same environment to provide a comparative upper bound on the evaluated metrics.

It is desirable to have driver models that match real world human behaviour. Therefore we measure the dispersion between the human expert and modelled distributions over emergent quantities using Kullback-Leibler Divergence (KL) [25]. Similar to [27], for each model we compute empirical distributions over speed, acceleration, turn-rate, jerk, and inverse time-to-collision (iTTC) over simulated trajectories.

[1]https://github.com/sisl/gail-driver
[2]https://github.com/YunzhuLi/InfoGAIL

Table 2: KL Divergence between different prediction models and human expert behaviour.

Method	Speed	acceleration	turn-rate	jerk	iTTC
SG	0.43	0.33	**0.24**	1.95	0.58
MR	0.41	0.45	0.51	1.78	0.49
GAIL-GRU	0.38	1.20	1.90	1.56	0.45
BC	0.38	1.40	1.50	1.24	0.52
Info-GAIL	0.35	1.00	0.97	0.91	0.38
MA-GAIL	**0.31**	**0.31**	0.50	**0.45**	**0.30**

From the results tabulated in Tab. 2 we observe the poorest performance in SG model as it models a static policy. We observe that the poor steering performance of GAIL-GRU, BC and Info-GAIL is caused by oscillations of the steering angle, which can also be seen in Fig. 3 for GAIL-GRU. We observe that when switching between the sub tasks such as lane following, changing lanes, and overtaking the model is unsure on the present context due to the limited history it possesses and will alternate between outputting small positive and negative turn-rates rather than performing the overall task successfully (see Fig. 3). This leads those models having higher dispersions in acceleration, turn-rate and jerk compared to the MA-GAIL model. With the aid of local and global memories the proposed model clearly identifies the required sub level tasks at different temporal contexts and performs them successfully, achieving better performance compared to the baselines.

In Fig. 3 and 4 we visualise the input frames along with the predicted steering wheel angles, shown in blue, and acceleration shown in red, which are quantised between −0.5 and +0.5, for GAIL-GRU and MA-GAIL models respectively. The two models observe the same initial state but it can be seen that the GAIL-GRU generates a noisy predictions where it outputs small positive and negative turn-rates and frequent fluctuations in acceleration. Hence the GAIL-GRU model oscillates within the lane change behaviour Fig. 3 (a)-(d); and maneuvers the car off-road Fig. 3 (h)-(j); instead of successfully completing the overall task.

In contrast, the proposed MA-GAIL model anticipates the required sub tasks and identifies their sequential relationships via long term planing which led the model to successfully complete the required sub tasks including lane following Fig. 4 (a)-(b); lane change to left lane Fig. 4 (c)-(e); overtake Fig. 4 (f); lane change to right lane Fig. 4 (g)-(i) without such oscillations.

We further evaluate the emergent behaviour metric proposed in [27] to measure the quality of the demonstrated policy. The considered metrics are 1) lane change rate, 2) off-road duration, 3) hard break rate and 4) traversed distance.

The lane change rate is the average number of times a vehicle makes a lane change within a generated trajectory. Off-road duration is the average number of time steps per trajectory that a vehicle spends more than 1m outside the closest outer road marker. The collision rate is the number of times where the simulated vehicle intersects with another traffic participant. The hard brake rate captures the frequency at which a model chooses to brake harder

Table 1: Handcrafted feature for the SG, MR and GAIL-GRU baselines

Feature	Range	Description
Angle	[-180 180]	Angle between the car direction and the direction of the track axis
Track	[0, 200] meters	Vector of 19 range finder sensor detections, denoting the distance between the track edge and the car
Track position	$[-\infty, +\infty]$	Distance between the car and the track axis
Speed X	$[-\infty, +\infty]$ Km/h	Speed of the car along the longitudinal axis of the car
Speed Y	$[-\infty, +\infty]$ Km/h	Speed of the car along the transverse axis of the car
Speed Z	$[-\infty, +\infty]$Km/h	Speed of the car along the z axis of the car
Wheel Spin velocity	$[-\infty, +\infty]$ rad/s	Vector of 4 values representing the speed of each wheel of the car
Front Distance	[0 ,200] meters	Distance to the closest car in front
Back Distance	[0 , 50] meters	Distance to the closest cars in back

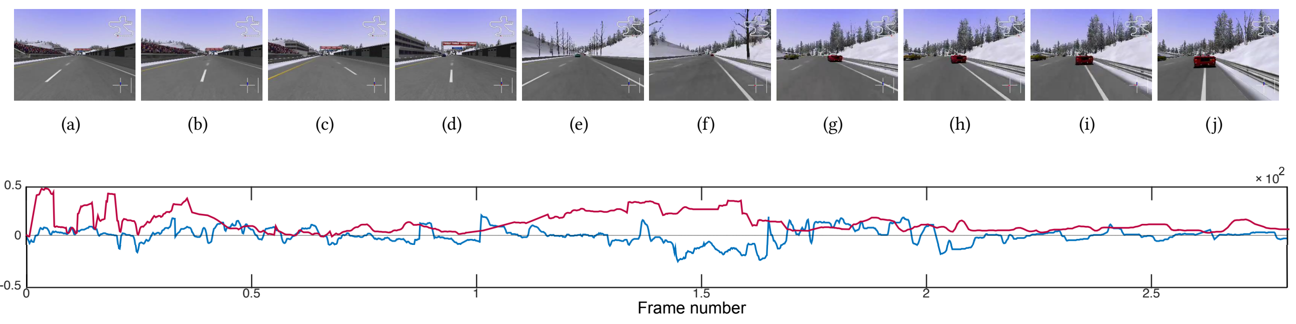


Figure 3: Visual inputs along with the predicted steering wheel angles (in blue) and acceleration (in red) for GAIL-GRU model.

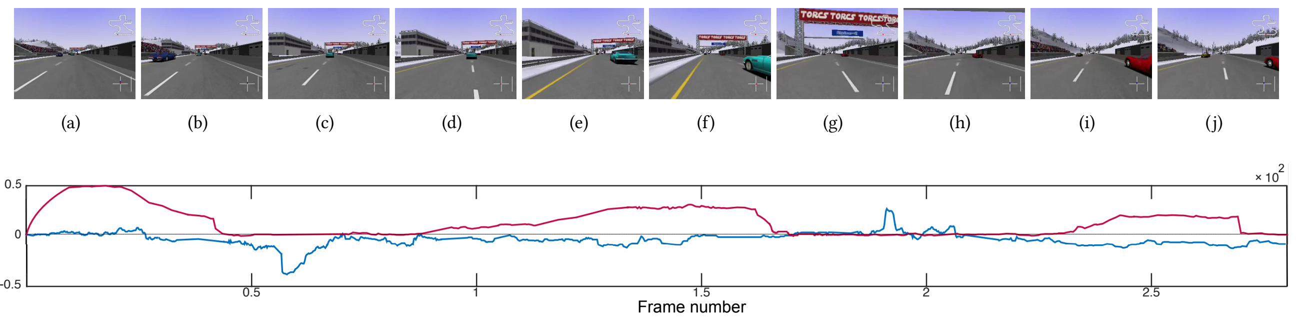


Figure 4: Visual inputs along with the predicted steering wheel angles (in blue) and acceleration (in red) for MA-GAIL model.

Table 3: Emergent behaviour metric evaluations

Method	Lane change ↓	Off-road ↓	Hard break ↓	Traverse ↑
SG	0.66	1.20	0.31	0.41
MR	0.56	0.43	0.52	0.72
GAIL-GRU	0.58	0.45	0.23	0.83
BC	0.48	0.51	0.27	1.01
Info-GAIL	0.43	0.42	0.21	1.34
MA-GAIL	0.33	**0.31**	**0.19**	**1.40**
Human	**0.31**	0.43	0.20	**1.40**

than -3 m/s^2. Finally traversed distance indicates the average total length traversed in the simulated trajectory in kilometres.

The emergent values tabulated in Tab. 3 shows that the proposed MA-GAIL method is able to outperform all the considered baselines and demonstrates human level control. The SG model performs poorly in all considered methods except hard brake rate because it only drives straight. As a consequence it has a higher collision rate and the smallest traverse distance. We observe an increase of performance from the SG model to MR and the GAIL-GRU model due to the increased capacity of the model to capture dynamic policy of the expert. Still the models perform worse than the BC and Info-GAIL models, largely due to the limited information in the hand-crafted state representation.

The lack of capacity to model long term temporal dependencies at the sub task level led GAIL-GRU, BC and Info-GAIL models to achieve a higher lane changes and lower traverse distances compared to the MA-GAIL model. In contrast, the proposed MA-GAIL model successfully localises the present context using the local memory and identifies the optimal way to behave using global

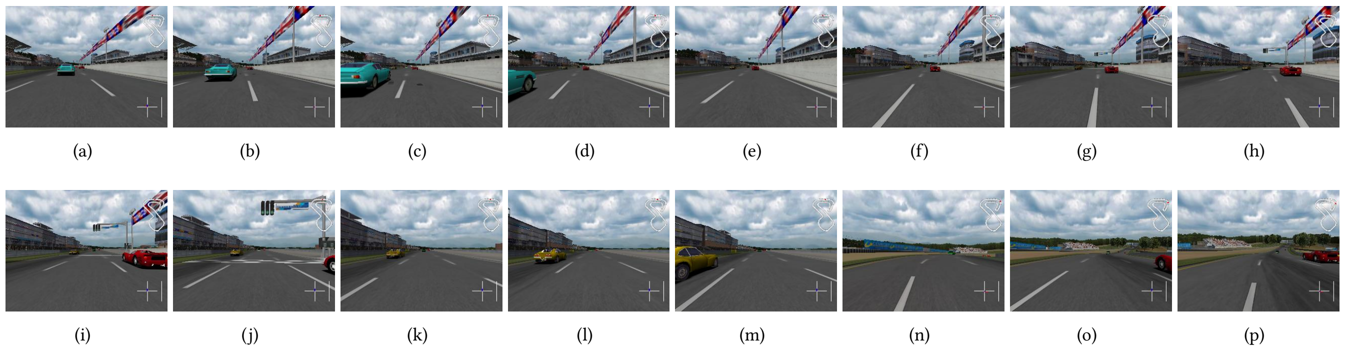


Figure 5: Visual inputs to the MA-GAIL model at different time steps of a single simulation experiment. Different sub level tasks such as lane change, over take, turn , lane following is attempted and successfully completed within a single simulation.

Table 4: Ablation experiment evaluations

Method	Lane change ↓	Off-road ↓	Hard break ↓	Traverse ↑
MA-GAIL / M^L	0.59	0.42	0.23	1.03
MA-GAIL / M^G	0.41	0.42	0.20	1.09
MA-GAIL / RA	0.38	0.38	0.21	1.28
MA-GAIL (proposed)	**0.33**	**0.31**	**0.19**	**1.40**

memory. We would like to further point out that the proposed MA-GAIL model has even outperformed the human expert in the Off-road and Hard break metrics, and matched human level performance in traversed distance metric.

In Fig. 5 we show visual inputs to the proposed MA-GAIL model at different time steps of a particular simulation. The proposed method successfully completes the sub level tasks such as lane change: Fig. 5 (a)-(i); over take: Fig. 5 (c), (i), (m); turn: Fig. 5 (n)-(p); and lane following: Fig. 5 (j)-(m); despite the vast diversity of the visual inputs. It should be noted that it demonstrates the lane change from left lane to right Fig. 5 (a)-(d) and from right lane to left in Fig. 5 (f)-(i). The model possess the capability to understand the current temporal context and has knowledge of different ways it can behave at that particular context. It successfully completes the task at hand and swiftly moves to the next sub task.

5.4 Ablation experiments

To further demonstrate our proposed approach, we conduct a series of ablation experiments identifying the crucial components of the proposed methodology to successfully learn an effective policy. In the same settings as the previous experiment we compare the **MA-GAIL (proposed)** method to a series of counterparts constructed by removing components of the MA-GAIL model as follows,

- **MA-GAIL / RA**: removes the reward augmentation methodology proposed in Sec. 4.3.
- **MA-GAIL / $\mathbf{M^L}$**: removes the local memory and retains only the global memory.
- **MA-GAIL / $\mathbf{M^G}$**: removes the global memory and retains only the local memory.

The results of our ablation experiment are presented Tab. 4. Model MA-GAIL / M^L performs poorly due to it's inability to capture the temporal context of the trajectory and results in frequent lane changes and hard break rates. With a local memory module (i.e MA-GAIL / M^G) the oscillations are reduced compared to MA-GAIL/M^L as the model can clearly identify the transition between sub tasks. The comparison between models MA-GAIL / RA and MA-GAIL (proposed) clearly emphasises the importance of reward augmentation. The transition between sub level tasks are even smoother in the MA-GAIL (proposed) method (i.e lower Hard break and Lane change rates) as the method clearly identifies the series of sub tasks at hand and achieves them optimally using the experiences stored in M^L and M^G, and tries to minimise diverse state transitions as much as possible.

We would like to further compare evaluation results in of Tab. 4 to those in Tab. 3 where we observe lower hard break rates and off-road distances compared to all the baseline models considered. This is due to the fact that the MA-GAIL model still has the ability to capture the basic dynamics in driving, even with only a single memory module.

6 CONCLUSIONS

In this paper we propose a method to imitate complex human strategies, properly analysing their temporal accordance at a sub task level and identifying strategic differences and similarities among expert demonstrations. We extend the standard GAIL framework with the ability to oversee the history of a long term task, localise the current subtask being completed, and perform long term planning to achieve the overall task. As the process is data driven, it doesn't require any supervision beyond expert demonstrations and could be directly transferred to different tasks without any architectural alterations. Additionally, we introduced a reward augmentation procedure using memory hidden states for smoothing the state transitions, eradicating sudden undesirable manoeuvers in the generated policy. Our quantitative and qualitative evaluations in the TORCS simulation platform clearly emphasise the capacity of the proposed MA-GAIL method to learn complex real world policies and even out performs the human demonstrators.

Acknowledgement

This research was supported by an Australian Research Council's Linkage grant (LP140100221). The authors also thank QUT High Performance Computing (HPC) for providing the computational resources for this research.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 1.

[2] Mohamed Aly. 2008. Real time detection of lane markers in urban streets. In *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 7–12.

[3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR* (2015).

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI gym. *arXiv preprint arXiv:1606.01540* (2016).

[7] Sylvain Calinon. 2009. *Robot programming by demonstration*. EPFL Press.

[8] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. 2015. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*. 2722–2730.

[9] François Chollet. 2017. Keras. *URL http://keras. io, 2017* (2017).

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[11] Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. One-Shot Imitation Learning. *arXiv preprint arXiv:1703.07326* (2017).

[12] Tharindu Fernando, Simon Denman, Aaron McFadyen, Sridha Sridharan, and Clinton Fookes. 2017. Tree Memory Networks for Modelling Long-term Temporal Dependencies. *arXiv preprint arXiv:1703.04706* (2017).

[13] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. 2017. Going Deeper: Autonomous Steering With Neural Memory Networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

[14] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. 2017. Soft+ Hardwired Attention: An LSTM Framework for Human Trajectory Prediction and Abnormal Event Detection. *arXiv preprint arXiv:1702.05552* (2017).

[15] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. 2018. Task Specific Visual Saliency Prediction with Memory Augmented Conditional Generative Adversarial Networks. *Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on* (2018).

[16] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. 2018. Tracking by Prediction: A Deep Generative Model for Multi-Person localisation and Tracking. *Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on* (2018).

[17] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *NIPS Workshop on Adversarial Training* (2016).

[18] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*. 49–58.

[19] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York.

[20] David Sierra González, Jilles Steeve Dibangoye, and Christian Laugier. 2016. High-speed highway scene prediction based on driver models learned from demonstrations. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 149–155.

[21] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. 2009. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics* 26, 2 (2009), 120–144.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[23] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 4565–4573.

[24] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[25] James M Joyce. 2011. Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*. Springer, 720–722.

[26] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 1–8.

[27] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. 2017. Imitating driver behavior with generative adversarial networks. *arXiv preprint arXiv:1701.06699* (2017).

[28] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. 1378–1387.

[29] Stéphanie Lefèvre, Chao Sun, Ruzena Bajcsy, and Christian Laugier. 2014. Comparison of parametric and non-parametric approaches for vehicle speed prediction. In *American Control Conference (ACC), 2014*. IEEE, 3494–3499.

[30] Yunzhu Li, Jiaming Song, and Stefano Ermon. 2017. Inferring The Latent Structure of Human Decision-Making from Raw Visual Inputs. *arXiv preprint arXiv:1703.08840* (2017).

[31] Jeremy Morton and Mykel J Kochenderfer. 2017. Simultaneous Policy Learning and Latent State Inference for Imitating Driver Behavior. *arXiv preprint arXiv:1704.05566* (2017).

[32] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *Icml*. 663–670.

[33] Hieu V Nguyen and Li Bai. 2010. Cosine similarity metric learning for face verification. In *Asian Conference on Computer Vision*. Springer, 709–720.

[34] Emilio Parisotto and Ruslan Salakhutdinov. 2017. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360* (2017).

[35] Dean A Pomerleau. 1989. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*. 305–313.

[36] Nathan D Ratliff, David Silver, and J Andrew Bagnell. 2009. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots* 27, 1 (2009), 25–53.

[37] Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 661–668.

[38] Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*. 627–635.

[39] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. 2016. Planning for Autonomous Cars that Leverage Effects on Human Actions.. In *Robotics: Science and Systems*.

[40] Stefan Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences* 3, 6 (1999), 233–242.

[41] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[42] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. 2017. Third-Person Imitation Learning. *ICLR* (2017).

[43] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[44] Monica C Vroman. 2014. *Maximum likelihood inverse reinforcement learning*. Rutgers The State University of New Jersey-New Brunswick.

[45] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. 2000. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net* (2000).

[46] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. 2048–2057.