Maximum Entropy Deep Inverse Reinforcement Learning

Markus Wulfmeier and **Peter Ondrúška** and **Ingmar Posner**
Mobile Robotics Group
Department of Engineering Science
University of Oxford

Abstract

This paper presents a general framework for employing deep architectures - in particular neural networks - to solve the inverse reinforcement learning (IRL) problem. Specifically, we propose to exploit the representational capacity and favourable computational complexity of deep networks to approximate complex, nonlinear reward functions in scenarios with large state spaces. We show that the Maximum Entropy paradigm for IRL lends itself naturally to the efficient training of deep architectures. At test time, the approach leads to a computational complexity independent of the number of demonstrations. This makes it especially well-suited for applications in life-long learning scenarios commonly encountered in robotics. We demonstrate that our approach achieves performance commensurate to the state-of-the-art on existing benchmarks already with simple, comparatively shallow network architectures while significantly outperforming the state-of-the-art on an alternative benchmark based on more complex, highly varying reward structures representing strong interactions between features. Furthermore, we extend the approach to include convolutional layers in order to eliminate the dependency on precomputed features of current algorithms and to underline the substantial gain in flexibility in framing IRL in the context of deep learning.

1 Introduction

The objective of inverse reinforcement learning (IRL) is to infer the underlying reward structure guiding an agent's behaviour based on observations as well as a model of the environment. This may be done either to learn the reward structure for modelling purposes or to provide a method to allow the agent to imitate a demonstrator's specific behaviour (Ramachandran and Amir 2007).

Much of the prior art in this domain relies on parametrisation of the reward function based on pre-determined features. In addition to better generalisation performance than direct state-to-reward mapping, this approach enables the transfer of learned reward functions between different scenarios with the same feature representation. A number of early works from (Ziebart et al. 2008), (Abbeel and Ng 2004) and (Ratliff, Bagnell, and Zinkevich 2006), express the reward function as a weighted linear combination of hand selected features. To overcome the inherent limitations of linear models, (Choi and Kim 2013) and (Levine, Popovic, and Koltun 2010) extend this approach to a limited set of nonlinear rewards by learning a set of composites of logical conjunctions of atomic features. Non-parametric methods such as Gaussian Processes (GPs) have also been employed to cater for potentially complex, nonlinear reward functions (Levine, Popovic, and Koltun 2011). While in principle this extends the IRL paradigm to the flexibility of nonlinear reward approximation, the use of a kernel machine makes this approach prone to requiring a large number of reward samples in order to approximate highly varying reward functions (Bengio, LeCun, and others 2007). Even sparse GP approximations as used in (Levine, Popovic, and Koltun 2011) lead to a query complexity time in dependency of the size of the active set or the number of experienced state-reward pairs. Situations with increasingly complex reward function leading to higher requirements regarding the number of inducing points can quickly render this nonparametric approach computationally impracticable. In comparison to (Babes et al. 2011), we focus on a singular expert in what finally leads to an an end-to-end learning scenario in section 4 from raw input to reward without compression or preprocessing on the input representation.

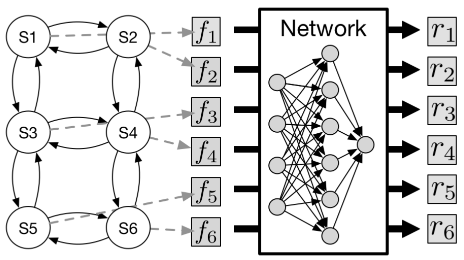


Figure 1: Schema for reward function approximation based on the feature representation of MDP states

In contrast to prior art, we explore the use of deep architectures - in particular neural networks - to approximate the reward function. Deep Neural Networks (DNNs) already achieve state-of-the-art performance across a variety of domains such as computer vision, natural language processing, speech recognition (Bengio, Courville, and Vincent 2012) and reinforcement learning (Mnih et al. 2013). Their application in IRL is attractive due to their compact representation of highly nonlinear functions through the composition and reuse of the results of many nonlinearities in the layered structure (Bengio, LeCun, and others 2007). In addition, DNNs provide favourable computational complexity ($\mathcal{O}(1)$) at query time with respect to observed demonstrations, which provides for scaling to problems with large state spaces and complex reward structures – circumstances which might render the application of existing prior methods intractable or ineffective. With the approach represented in Figure 1, a state's reward can be determined either solely based on its own feature representation or – in using convolutional layers – analysed in combination with its spatial context. To our knowledge the only other work considering a roughly similar approach is given by (Levine et al. 2015), who focus on directly approximating policies with neural networks but shortly refer to the possibility of extension for cost function learning.

Our principal contribution is a framework for *Maximum Entropy Deep Inverse Reinforcement Learning* (DeepIRL) based on the Maximum Entropy paradigm for IRL (Ziebart et al. 2008), which lends itself naturally for training deep architectures by leading to an objective that is - without approximations - fully differentiable with respect to the network weights. Furthermore, we demonstrate performance commensurate to state-of-the-art methods on a publicly available benchmark, while outperforming the state-of-the-art on a new benchmark where the true underlying reward has complex interacting structure over the feature representation. In addition, we emphasise the flexibility of the approach and eliminate the requirement of preprocessing and precomputed features by applying convolutional layers to learn spatial features of relevance to the IRL task.

We argue that these properties are important for practical large-scale applications of IRL as can be seen in life-long learning approaches with often complex reward functions and increasing scale of demonstrations requiring high capacity models and fast computational speeds.

2 Inverse Reinforcement Learning

This section presents a brief overview of IRL. Let a Markov Decision Process (MDP) be defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the set of possible actions, $\mathcal{T}$ denotes the transition model and r denotes the reward structure. Given an MDP, an optimal policy π^* is one which, when adhered to, maximizes the expected cumulative reward. In some cases an additional factor $\gamma \in [0, 1]$ may be considered in order to discount future rewards.

IRL considers the case where a MDP specification is available but the reward structure is unknown. Instead, a set of expert demonstrations $\mathcal{D} = \{\varsigma_1, \varsigma_2, ..., \varsigma_N\}$ are provided which are drawn from a user policy π, i.e. provided by a demonstrator. Each demonstration consists of a set of state-action pairs such that $\varsigma_i = \{(s_0, a_0), (s_1, a_1), ..., (s_K, a_K)\}$. The goal of IRL is to uncover the hidden reward r from the demonstrations.

A number of approaches have been proposed to tackle the IRL problem (see, for example, (Abbeel and Ng 2004), (Neu and Szepesvári 2012), (Ratliff, Bagnell, and Zinkevich 2006), (Syed and Schapire 2007)). An increasingly popular formulation is Maximum Entropy IRL (Ziebart et al. 2008), which was used to effectively model large-scale user driving behaviour. In this formulation the probability of user preference for any given trajectory between specified start and goal states is proportional to the exponential of the reward along the path

$$P(\varsigma|r) \propto \exp\{\sum_{s,a\in\varsigma} r_{s,a}\}. \qquad (1)$$

As shown in Ziebart's work, principal benefits of the Maximum Entropy paradigm include the ability to handle expert suboptimality as well as stochasticity by operating on the distribution over possible trajectories. Moreover, the Maximum Entropy based objective function given in Equation 8 enables backpropagation of the objective gradients to the network's weights. The training procedure is then straightforwardly framed as an optimisation task computable e.g. via conjugate gradient or stochastic gradient descent.

Approximating the Reward Structure

Due to the dimensionality and magnitude of the state space in many real world applications, the reward structure can not be observed explicitly for every state. In these cases state rewards are not modelled directly per state, but the reward structure is restricted by imposing that states with similar features, x, should have similar rewards. To this end, function approximation is used in order to regress the feature representation onto a real valued reward using a mapping $g : \mathbb{R}^N \to \mathbb{R}$, with N being the dimensionality of the feature space such that

$$r = g(f, \theta). \qquad (2)$$

This feature representation, f, is usually hand-crafted, but can be learned based on the proposed framework - as shown in section 4.

The choice of model used for function approximation has a dramatic impact on the ability of the algorithm to capture relationship between the state feature vector f and user preference. Commonly, the mapping from state to reward is simply a weighted linear combination of feature values

$$g(f, \theta) = \theta^\top f. \qquad (3)$$

This choice, while appropriate in some scenarios, is suboptimal if the true reward can not be accurately approximated by a linear model. In order to alleviate this limitation (Choi and Kim 2013) extend the linear model by introducing a mapping $\Phi : \mathbb{R}^N \to \{0, 1\}^N$ such that

$$g(f, \theta, \Phi) = \theta^\top \Phi(f). \qquad (4)$$

Here Φ denotes a set of composite features which are jointly learned as part of the objective function. These composites are assumed to be the logical conjunctions of the predefined, atomic features f. Due to the nature of the features used the representational power of this approach is limited to the family of piecewise constant functions.

In contrast, (Levine, Popovic, and Koltun 2011) employ a Gaussian Processes (GP) framework to capture the potentially unbounded complexity of any underlying reward structure. The set of expert demonstrations $\mathcal{D}$ is used in this context to identify an active set of GP support points, X_u, and associated rewards u. The mean function is then used to represent the individual reward at a state described by f

$$g(f, \theta, \mathcal{X}_u, u) = K_{f,u}^{\top} K_{u,u}^{-1} u. \tag{5}$$

Here $K_{f,u}$ denotes the covariance of the reward at f with the active set reward values u located at X_u and $K_{u,u}$ denotes the covariance matrix of the rewards in the active set computed via a covariance function $k_\theta(f_i, f_j)$ with hyperparameters θ.

Nevertheless, a significant drawback of the GPIRL approach is a computational complexity proportional to the number of demonstrations and the size of the active set of inducing points, which in turn depends on the reward complexity. While the modelling of complex, nonlinear reward structures in problems with large state spaces is theoretically feasible for the GPIRL approach, the cardinality of the active set will quickly become unwieldy, putting GPIRL at a significant computational disadvantage or, worse, rendering it entirely intractable. These shortcomings are remedied when using deep parametric architectures for reward function approximation, as outlined in the next section.

3 Reward Function Approximation with Deep Architectures

We argue that IRL algorithms scalable to MDPs with large feature spaces require models, which are able to efficiently represent complex, nonlinear reward structures. In this context, deep architectures are a natural choice as they explicitly exploit the depth-breadth trade-off (Bengio, LeCun, and others 2007) and increase representational capacity by reusing the computations of earlier nodes in the following layers.

For the remainder of the paper, we consider a DNN architecture which accepts as input state features x, maps these to state reward r and is governed by the network parameters $\theta_{1,2,..n}$. In the context of Section 2, the state reward is therefore obtained as

$$\begin{aligned} r &\approx g(f, \theta_1, \theta_2, ..., \theta_n) && (6)\\ &= g_1(g_2(...(g_n(f, \theta_n), ...), \theta_2), \theta_1). && (7) \end{aligned}$$

While many choices exist for the individual building blocks of a deep architecture, it has been shown that a sufficiently large DNN with as little as two layers and sigmoid activation functions can represent any binary function (Hassoun 1995) or any piecewise-linear function (Hornik, Stinchcombe, and White 1989) and can therefore be regarded as a *universal approximator*. While this holds true in theory, it can be far more computationally practicable to extend the depth of the network structure and reduce the number of required computations in doing so (Bengio 2009).

Importantly, in applying backpropagation, DNNs also lend themselves naturally to training in the maximum entropy IRL framework and the network structure can be adapted to suit individual tasks without invalidating the main IRL learning mechanism. In the DeepIRL framework proposed here the full range of architecture choices thus becomes available. Different problem domains can utilise different network architectures as e.g. convolutional layers can remove the dependency on handcrafted spatial features. Furthermore, it is straightforward to show that the linear maximum entropy IRL approach proposed in (Ziebart et al. 2008) can be seen as a simplification of the more general deep approach and can be created by applying the rules of backpropagation to a network with a single linear output connected to all inputs with zero bias term.

In practice, several factors influence the appropriate structure of the network. While the complexity of the true underlying reward is often unobservable the amount of training data available is one such factor. As the number of training data increases, adding hidden layers and increasing the number of nodes will allow the network to fit more complex feature dependencies and thus model more complex rewards. A common design guideline in this context is to increase network capacity until overfitting occurs and subsequently regularise.

Training Procedure

The task of solving the IRL problem can be framed in the context of Bayesian inference as MAP estimation, maximizing the joint posterior distribution of observing expert demonstrations, $\mathcal{D}$, under a given reward structure and of the model parameters θ.

$$\mathcal{L}(\theta) = \log P(\mathcal{D}, \theta | r) = \underbrace{\log P(\mathcal{D}|r)}_{\mathcal{L}_\mathcal{D}} + \underbrace{\log P(\theta)}_{\mathcal{L}_\theta}. \tag{8}$$

This joint log likelihood is differentiable with respect to the network parameters θ, which allows the application of gradient descent methods (Snyman 2005). The Maximum Entropy based objective function given in the data term $\mathcal{L}_\mathcal{D}$ of Equation 8 is differentiable with respect to the rewards r and therefore enables backpropagation of the objective gradients to the network's weights.

The final gradient is given by the sum of the gradients with respect to θ of the data term $\mathcal{L}_\mathcal{D}$ and the model term $\mathcal{L}_\theta$

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}_\mathcal{D}}{\partial \theta} + \frac{\partial \mathcal{L}_\theta}{\partial \theta}. \tag{9}$$

The gradient of data term can be expressed in terms of the derivative of the expert demonstration with respect to rewards as well as the derivative of these rewards with respect to network weights θ, such that

$$\begin{aligned} \frac{\partial \mathcal{L}_\mathcal{D}}{\partial \theta} &= \frac{\partial \mathcal{L}_\mathcal{D}}{\partial r} \cdot \frac{\partial r}{\partial \theta} && (10)\\ &= (\mu_\mathcal{D} - \mathbb{E}[\mu]) \cdot \frac{\partial}{\partial \theta} g(f, \theta), && (11) \end{aligned}$$

where $r = g(f, \theta))$. As shown in (Ziebart et al. 2008), the gradient of the expert demonstration term $\mathcal{L}_\mathcal{D}$ with respect to state rewards r is equal to the difference in the state visitation counts exhibited by the expert demonstrations and the expected visitation counts for the learned systems trajectory distribution in 1, which depends on the reward approximation given the corresponding optimal policy.

$$\mathbb{E}[\mu] = \sum_{\varsigma : \{s,a\} \in \varsigma} P(\varsigma | r) \tag{12}$$

Computation of $\mathbb{E}[\mu]$ usually involves summation over exponentially many possible trajectories. In (Ziebart et al. 2008) an effective algorithm based on dynamic programming was proposed computing this quantity in polynomial-time. The effective computation of the gradient $\frac{\partial \mathcal{L}_D}{\partial \theta}$ thus involves first computing the difference in visitation counts using this algorithm and then passing this as an error signal through the network using back-propagation. The complete proposed method is described by Algorithm 1, with the loss and gradient derivation in lines 6 and 7 given by the linear Maximum Entropy formulation. The expert's state action frequencies μ_D^a, which are needed for the calculation of the loss are simply summed over the actions to compute the expert state frequencies $\mu_D = \sum_{a=1}^{A} \mu_D^a$.

The derivative of the model term $\mathcal{L}_\theta$ with respect to the network parameters is expressed as a regularisor. There exists a variety of choices to prevent over-fitting in deep neural networks. L_1 and L_2 regularisation are used as well as dropout, which can be seen as a sub-model averaging approach, or methods corrupting the original training data, for example, by adding noise to train for invariance.

Algorithm 1 Maximum Entropy Deep IRL

```
Input: $\mu_D^a, f$
Output: optimal weights $\theta^*$
1: $\theta^1$ = initialise_weights()
2: for n = 1 : max_iterations do
3:     $r^n$ = nn_forward($f, \theta^n$)    {compute current reward}
4:     $\pi^n$ = solve_mdp($r^n$)    {determine optimal policy}
5:     $\mathbb{E}[\mu^n]$ = propagate_policy($\pi^n$)    {get state frequencies}
6:     $\mathcal{L}_D^n = \log(\pi^n) \times \mu_D^a$
7:     $\frac{\partial \mathcal{L}_D^n}{\partial r^n} = \mu_D - \mathbb{E}[\mu^n]$    {given by (Ziebart et al. 2008)}
8:     $\frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n}$ = nn_backprop($\frac{\partial \mathcal{L}_D^n}{\partial r^n}$)    {propagate gradients through network}
9:     $\theta^{n+1}$ = update_weights($\theta^n, \frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n}$)
10: end for
```

4 Experiments

We assess the performance of DeepIRL on an existing as well as an alternative benchmark task against current state-of-the-art approaches : GPIRL (Levine, Popovic, and Koltun 2011), NPB-FIRL (Choi and Kim 2013) and the original MaxEnt (Ziebart et al. 2008) to illustrate the necessity of non-linear function approximation.

All tests are run multiple times on training and transfer scenarios for the different settings, while learning is performed based on synthetically generated stochastic demonstrations based on the optimal policy to evaluate performance on suboptimal example sets. This is achieved by providing a number of demonstrations sampled from the optimal policy based on the true reward structure, but including 30% of random actions.

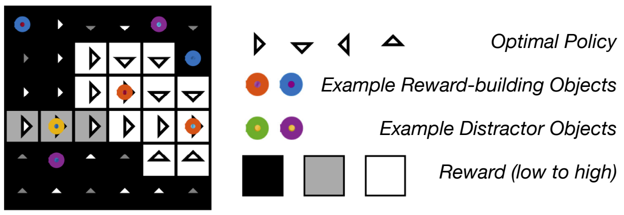


Figure 2: Objectworld benchmark. The true reward is displayed by the brightness of each cell and based on the surrounding object configuration. Only a subset of colors influences the reward, while the others serve as distracting features.

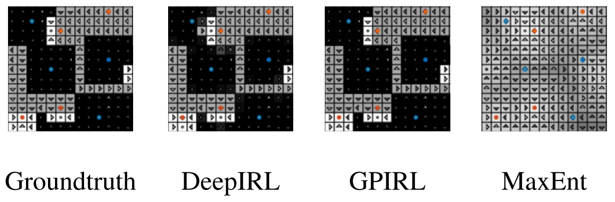


Figure 3: Reward reconstruction sample in Objectworld benchmark provided $N = 64$ examples and $C = 2$ colours with continuous features. White - high reward; black - low reward.

In our experiments, we employ a fully connected feed-forward network with two hidden layers and rectified linear units as activation functions as function approximator between state feature representation and reward. This rather shallow networks structure suffices for the application based on strongly simplified toy benchmarks. However, the whole framework can be utilised for training networks of arbitrary capacity. For these benchmarks, we apply AdaGrad (Duchi, Hazan, and Singer 2011), an approach for stochastic gradient descent with per parameter adaptive learning rates. Significant parts of the neural network implementation are based on MatConvNet (Vedaldi and Lenc 2014).

In line with related works, we use *expected value difference* as principal metric of evaluation. It is a measure of the sub-optimality of the learned policy under the true reward. The score represents the difference between the value function obtained for the optimal policy given the true reward structure and the value function obtained for the optimal policy based on the learned reward model. Additionally to the evaluation on each specific training scenario, the trained models are evaluated on a number of randomly generated test environments. The test on these *transfer* examples serves to analyse each algorithm's ability to generalise to the true reward structure without over-fitting.

Objectworld Benchmark

The *Objectworld* scenario (Levine, Popovic, and Koltun 2011) consists of a map of $M \times M$ states for $M = 32$ where possible actions include motions in all four directions as well as staying in place. Two different sets of state features are implemented based on randomly placed colours to evaluate the algorithms. For the continuous set $x \in \mathbb{R}^C$. Each feature dimension describes the minimum distance to an object of one of C colours. Building on the continuous representation the discrete set includes $C \times M$ binary features, where each dimension indicates whether an object of a given colour is closer than a threshold $d \in \{1, ..., M\}$.

The reward is positive for cells which are both within the distance 3 of color 1 and distance 2 of color 2, negative if only within distance 3 of color 1 and zero otherwise. This is illustrated for a small subset of the state space in Figure 2.

In line with common benchmarking procedures, we evaluated the algorithms with a set number of features and increasing demonstrations. Additionally, the learned reward functions are deployed on randomly generated transfer scenarios to uncover any overfitting to the training data.

While the original MaxEnt is unable to capture the nonlinear reward structure well, both DeepIRL and GPIRL provide significantly better approximations as represented in Figure 3. Evaluation of NPB-FIRL on this benchmark was done in (Choi and Kim 2013) where it showed a similar level of performance as GPIRL. GPIRL generates a good model already with few data points whereas DeepIRL achieves commensurate performance when increasing the number of available expert demonstrations. The same behaviour is exhibited when using both continuous and discrete state features (Fig. 4). The requirement for more training data will be rendered unimportant in robot applications based on autonomous data acquisition, while enforcing the lower algorithmic complexity as dominant advantage of the parametric approach.

Additional tests are performed with increased number of distractor features to evaluate each approach's overfitting tendency. The corresponding figures are left out due to limited space. Both DeepIRL and GPIRL show robustness to distractor variables, though DeepIRL shows minimally bigger signs of overfitting as the number of distractor variables is increased. This is due to the DNN's capacity being brought to bear on the increasing noise introduced by the distractors and will be addressed in future work with additional regularisation methods, such as Dropout (Hinton et al. 2012) and ensemble methods.

Binaryworld Benchmark

In order to test the ability of all approaches to successfully approximate more complex reward structures, the *Binaryworld* benchmark is presented. This test scenario is similar to *Objectworld*, but in this problem every state is randomly assigned one of two colours (blue or red). The feature vector for each state consequently consists of a binary vector of length 9, encoding the colour of each cell in its 3x3 neighbourhood. The true reward structure for a particular state is fully determined by the *number* of blue states in its local neighbourhood. It is positive if exactly four out of nine neighbouring states are blue, negative if exactly five are blue and zero otherwise. The main difference compared to the Objectworld scenario is that a single feature value does not carry much weight, but rather that higher-order relationships amongst the features determine the reward.

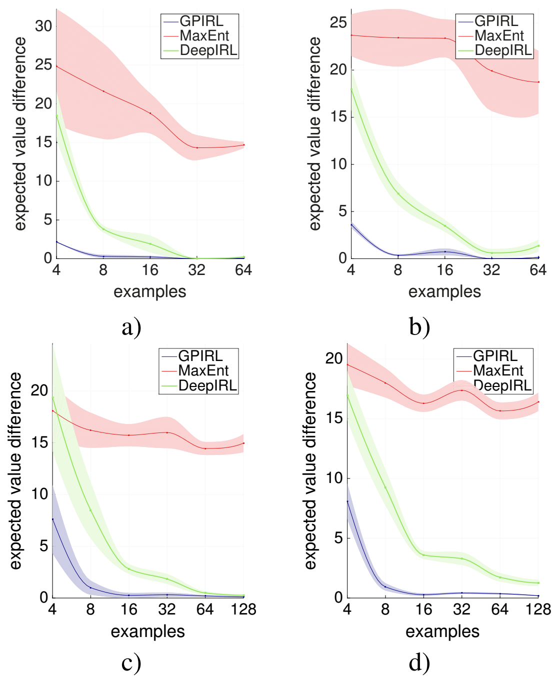
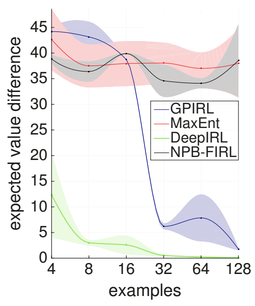


Figure 4: Objectworld benchmark. From top left to bottom right: expected value difference (EVD) with $C = 2$ colours and varying number of demonstrations N for training a) and transfer case b) with continuous and subsequently with discrete features in c) & d) ; As the number of demonstrations grows DeepIRL is able to quickly match performance of GPIRL on the task.

The performance of DeepIRL compared to GPIRL, linear MaxEnt and NPB-FIRL is depicted in Fig. 5. In this increasingly more complex scenario, DeepIRL is able to learn the higher-order dependencies between features, whereas GPIRL struggles as the inherent kernel measure can not correctly relate the reward of different examples with similarity in their state features. GPIRL needs a larger number of demonstrations to achieve good performance and to determine an accurate estimate on the reward for all 2^9 possible feature combinations.

Perhaps surprising is the comparatively low performance of the NPB-FIRL algorithm. This can be explained by the limitations of this framework. The true reward in this scenario can not be efficiently described by the logical conjunctions used. In fact, it would require 2^9 different logical conjunctions, each capturing all possible combinations of features, to accurately model the reward in this framework.

Fig. 6 shows the reconstruction of the reward structures estimated by DeepIRL, MaxEnt and GPIRL. While GPIRL was able to reconstruct the correct reward for some of the states having features it has encountered before it provides inaccurate rewards for states which were never encountered. It produces an overall too smooth reward function due to assumptions and priors in the GP approximation. On the other hand, DeepIRL is able to reconstruct it with high accuracy demonstrating the ability to effectively learn the highly-



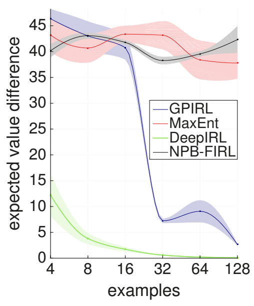


Figure 5: Value differences observed in the Binaryworld benchmark for GPIRL, MaxEnt and DeepIRL for the training scenario (left) and the transfer task (right).

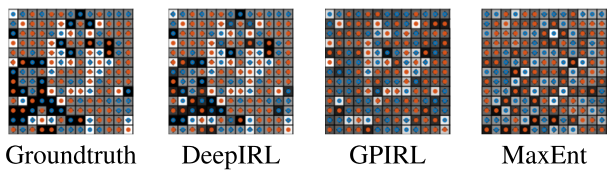


Figure 6: Reward reconstruction sample for the Binaryworld benchmark provided $N = 128$ demonstrations. White - high reward; black - low reward.

varying structure of the underlying function.

Convolutional Feature Learning

While the earlier benchmarks visualise performance compared to current algorithms in the context of precomputed features, the approach can be extended via the use of convolutional neural networks (CNNs) to eliminate the requirement of preprocessing or manual design of features. Figure 7 represents the results for both earlier benchmarks, but instead of using the earlier described feature representations, the CNN builds the reward based on the raw input representation, which for each state only includes the availability of each specific object at that specific state. All spatial information is derived based on the convolutional filters. Based on the simplicity of the benchmarks, we employed a five layer approach with 3x3 convolutional kernels in the first two layers. By increasing the depth of the network and include convolutional filters, we add enough capacity to enable the learning of features as well as their combination into the reward function in the same architecture and process.

Due to the increasing number of parameters, the approach requires additional training data to perform at equal accuracy but with increasing number of expert samples converges towards the performance with predefined features. Since the given features in these simplified toy problems are optimal and the true reward is directly calculated on their basis, automatically learned features cannot exceed the performance. However, in real-world scenarios, the compression of raw data - such as images - to feature representations leads to information loss and the learning of task-relevant features gains even more importance.

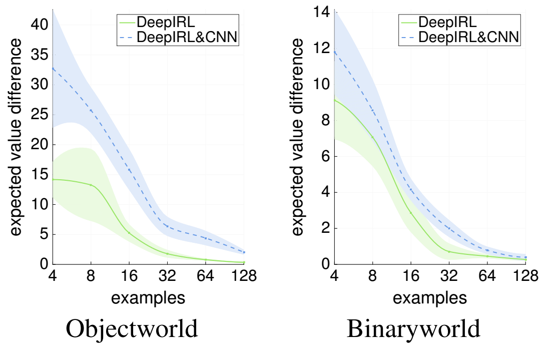


Figure 7: Application of convolutional layers for spatial feature learning. Convolutional feature learning quickly converges to performance with optimally designed features.

5 Conclusion and Future Work

This paper presents Maximum Entropy Deep IRL, a framework exploiting DNNs for reward structure approximation in Inverse Reinforcement Learning. DNNs lend themselves naturally to this task as they combine representational power with computational efficiency compared to state-of-the-art methods. Unlike prior art in this domain DeepIRL can therefore be applied in cases where complex reward structures need to be modelled for large state spaces. Moreover, DNN training can be achieved effectively and efficiently within the popular Maximum Entropy IRL framework. A further advantage of DeepIRL lies in its versatility. Custom network architectures and types can be developed for any given task while exploiting the same cost function in training.

Our experiments show that DeepIRL's performance is commensurate to the state-of-the-art on a common benchmark. While exhibiting slightly increased requirements regarding training data in this benchmark, a principal strength of the approach lies in its algorithmic complexity independent of the number of demonstrations samples. Therefore, it is particularly well-suited for life-long learning scenarios in the context of robotics, which inherently provide sufficient amounts of training data. We also provide an alternative evaluation on a new benchmark with a significantly more complex reward structure, where DeepIRL significantly outperforms the current state-of-the-art and proves its strong capability in modeling the interaction between features. Furthermore, we extend the approach to include convolutional layers in order to eliminate the dependency on precomputed features and to emphasise the adaptability of framing IRL in the context of deep learning.

In future work we will explore the benefits of autoencoder-style pretraining to reduce the increased demand of expert demonstrations when employing convolutional neural networks (CNNs). Especially when based on high dimensional inputs such as raw image data, the easily available unsupervised training data will help to learn features which then only need to be refined during the supervised IRL-based training phase.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st international conference on Machine learning*, 1. ACM.

Babes, M.; Marivate, V.; Subramanian, K.; and Littman, M. L. 2011. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 897–904.

Bengio, Y.; Courville, A. C.; and Vincent, P. 2012. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR* abs/1206.5538.

Bengio, Y.; LeCun, Y.; et al. 2007. Scaling learning algorithms towards AI. *Large-scale kernel machines* 34(5).

Bengio, Y. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning* 2(1):1–127.

Choi, J., and Kim, K.-E. 2013. Bayesian nonparametric feature construction for inverse reinforcement learning. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 1287–1293. AAAI Press.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

Hassoun, M. H. 1995. *Fundamentals of artificial neural networks*. MIT press.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5):359–366.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2015. Learning deep vision-based costs and policies. *Robotics: Science and Systems. WS: Learning from Demonstration*.

Levine, S.; Popovic, Z.; and Koltun, V. 2010. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, 1342–1350.

Levine, S.; Popovic, Z.; and Koltun, V. 2011. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, 19–27.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *CoRR* abs/1312.5602.

Neu, G., and Szepesvári, C. 2012. Apprenticeship learning using inverse reinforcement learning and gradient methods. *CoRR* abs/1206.5264.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. *Urbana* 51:61801.

Ratliff, N. D.; Bagnell, J. A.; and Zinkevich, M. A. 2006. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, 729–736. ACM.

Snyman, J. 2005. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 97. Springer Science & Business Media.

Syed, U., and Schapire, R. E. 2007. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, 1449–1456.

Vedaldi, A., and Lenc, K. 2014. Matconvnet – convolutional neural networks for matlab. *CoRR* abs/1412.4564.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*, 1433–1438.