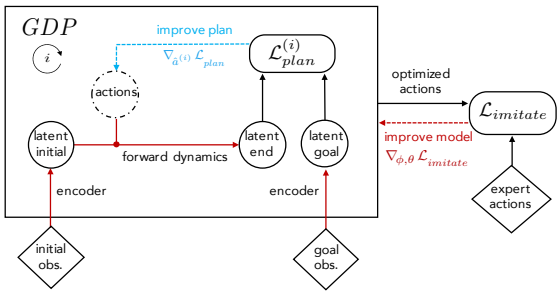
Universal Planning Networks

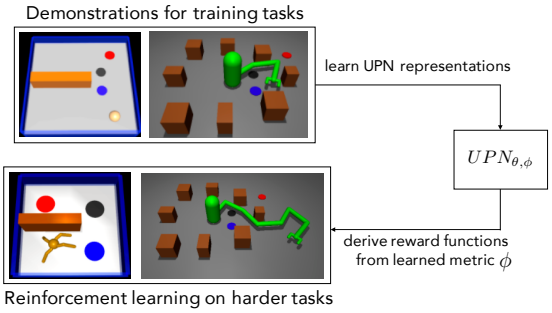
Aravind Srinivas [1] **Allan Jabri** [1] **Pieter Abbeel** [1] **Sergey Levine** [1] **Chelsea Finn** [1]

Abstract

A key challenge in complex visuomotor control is learning abstract representations that are effective for specifying goals, planning, and generalization. To this end, we introduce universal planning networks (UPN). UPNs embed differentiable planning within a goal-directed policy. This planning computation unrolls a forward model in a latent space and infers an optimal action plan through gradient descent trajectory optimization. The plan-by-gradient-descent process and its underlying representations are learned end-to-end to directly optimize a supervised imitation learning objective. We find that the representations learned are not only effective for goal-directed visual imitation via gradient-based trajectory optimization, but can also provide a metric for specifying goals using images. The learned representations can be leveraged to specify distance-based rewards to reach new target states for model-free reinforcement learning, resulting in substantially more effective learning when solving new tasks described via image-based goals. Visit https://sites.google.com/view/upn-public/home for video highlights.



(a) Universal Planning Network (UPN)



(b) Leveraging learned latent representations

Figure 1. An overview of the UPN, which embeds a gradient descent planner (GDP) in the action-selection process. We demonstrate transfer to different, harder control tasks, including morphological (yellow point robot to ant) and topological (3-link to 7-link reacher) variants, as shown above.

1. Introduction

Learning visuomotor policies is a central pursuit in building machines capable of performing complex skills in the variety of unstructured and dynamic environments seen in the real world (Levine et al., 2016; Pinto et al., 2016). A key challenge in learning such policies lies in acquiring representations of the visual environment and its dynamics that are suitable for control. This challenge arises both in the construction of the policy itself and in the specification of the task. Extrinsic and perfect reward signals are typically not available for real world reinforcement learning and users must manually specify tasks via hand-crafted rewards with hand-crafted representations. To automate this process, some prior methods have proposed to specify tasks by providing an image of the goal scene (Deguchi & Takahashi, 1999; Watter et al., 2015; Finn et al., 2016b). However, a reward that measures success based on matching the raw pixels of the goal image is far from ideal: such a reward is both uninformative and overconstrained, since matching all pixels is usually not required for succeeding in tasks. If we can automatically identify the right representation, we can both accelerate the policy learning process and simplify the specification of tasks via goal images. Prior work in visual representation learning for planning and control has relied predominantly on unsupervised or self-supervised objectives (Watter et al., 2015; Finn et al., 2016b), which in principle only provide an indirect connection to the utility

[1]UC Berkeley, Computer Science. Correspondence to: Aravind Srinivas <aravind@cs.berkeley.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

of the representation for the underlying control problem. Effective representation learning for planning and control remains an open problem.

In this work, instead of learning from unsupervised or auxiliary objectives and expecting that useful representations should emerge, we directly optimize for *plannable* representations; that is, representations through which gradient-based planning is successful with respect to the goal-directed task. We propose universal planning networks (UPN), a neural network architecture that can be trained to acquire a plannable representation. By embedding a differentiable planning computation inside the policy, our method enables joint training of the planner and its underlying latent encoder and forward dynamics representations. An outer imitation learning objective ensures that the learned representations are directly optimized for successful gradient-based planning on a set of training demonstrations. In principle, however, the architecture could also be trained with other policy optimization techniques such as those from reinforcement learning. An overview is provided in Figure 1(a).

We demonstrate that the representations learned by UPN not only support gradient-based trajectory optimization for successful visual imitation, but in fact acquire a meaningful encoding of state, which can be used as a metric for task-specific latent distance to a goal. We find that we can reuse this representation to specify latent distance-based rewards to reach new target states via standard model-free reinforcement learning, resulting in substantially more effective learning when using image targets. These properties are naturally induced by the agent's reliance on the minimization of the latent distance between its predicted terminal state and goal state throughout the planning process. By learning *plannable* representations, the UPN learns an optimizable latent distance metric. Our findings are based on a suite of challenging vision-based simulated robot control tasks that involve planning.

At a high-level, our approach is a goal-conditioned policy architecture that leverages a gradient-based planning computation in its action-selection process. While the architecture is agnostic to the objective function in the outer loop, we will focus on the imitation learning setting. From the perspective of representation learning, our method provides a way to learn more effective representations suitable for specifying perceptual reward functions, which can then be used, for example, with a model-free reinforcement learner. In terms of meta-learning, our architecture can be seen as learning a planning computation by learning representations that are in some sense *traversible* by gradient descent trajectory optimization for satisfying the outer meta-objective.

In extensive experiments, we show that (1) UPNs learn effective visual goal-directed policies more efficiently (that is, with less data) than traditional imitation learners; (2) the latent representations induced by optimizing for successful planning can be leveraged to transfer task-related semantics to other agents for more challenging tasks through goal-conditioned reward functions, which to our knowledge has previously not been demonstrated; and (3) the learned planning computation improves when allowed more updates at test-time, even in scenarios of less data, providing encouraging evidence of successful meta-learning for planning.

2. Universal Planning Networks

Model-based approaches leverage forward models to search for, or *plan*, sequences of actions to achieve goal states such that a planning objective is minimized. Forward modeling supports simulation of future state and hence, in principle, should allow for planning over extended horizons. In the absence of known environment dynamics, a forward model must be learned. Differentiable forward models allow for end-to-end training of model-based planners, as well as planning by back-propagating gradients with respect to input actions (Schmidhuber, 1990; Henaff et al., 2017).

Nevertheless, learned forward models may: (1) suffer from function approximation modeling error, especially in complex, high-dimensional environments, (2) capture irrelevant details under the incentive to reduce model-bias, as is often the case when learning directly from pixels, and (3) not necessarily align with the task and planning problem at hand, such that the inferred plans are sub-optimal even if the planning objective is optimized.

These issues motivate a central idea of the proposed method: instead of learning from surrogate unsupervised or auxiliary objectives, we directly optimize for what we care about, which is, representations with which gradient-based trajectory optimization leads to the desired actions. We study a model-based architecture that performs a differentiable planning computation in a latent space jointly learned with forward dynamics, trained end-to-end to encode what is necessary for solving tasks by gradient-based planning.

2.1. Learning to Plan

The UPN computation graph forms a goal-directed policy supported by an iterative planning algorithm. Given initial and goal observations (o_t and o_g) as input images, the model produces an optimal plan $\hat{a}_{t:t+T}$ to arrive at o_g, where $\hat{a}_t$ denotes the predicted action at time t. The computation graph consists of a pair of tied encoders that encode both o_t and o_g, and their features are fed into a gradient descent planner (GDP), which produces the action a_t as output. The GDP uses a neural network encoder and forward dynamics model to simulate transitions in a learned latent space and is thus fully differentiable. An overview of the method is presented in Figure 2.

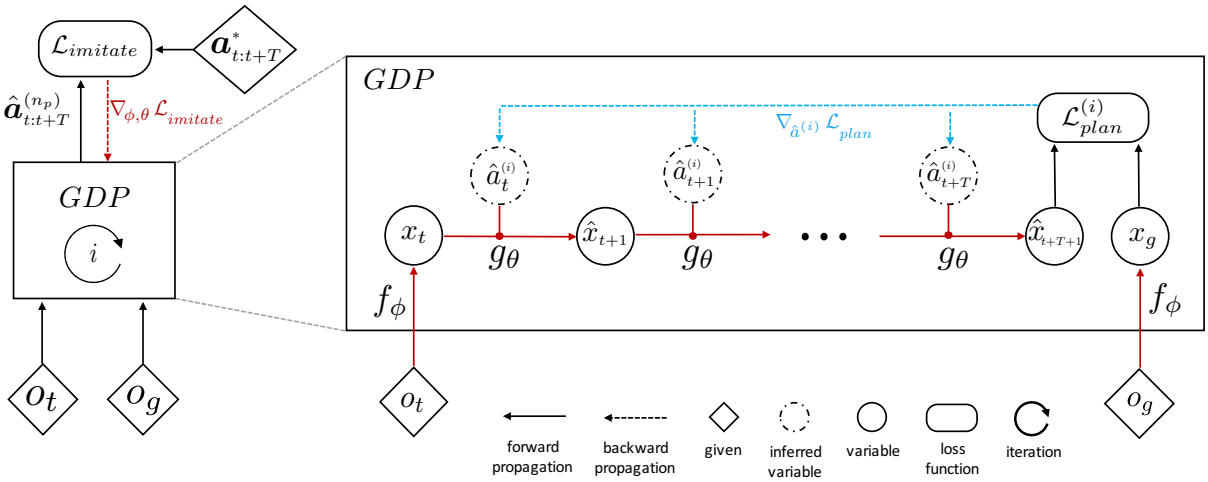


Figure 2. An overview of the proposed method. Given an initial o_t and a goal o_g, the GDP (*gradient descent planner*) uses gradient descent to optimize a plan to reach the goal observation with a sequence of actions in a latent space represented by f_ϕ. This planning process forms one large computation graph, chaining together the sub-graphs of each iteration of planning. The learning signal is derived from the (outer) imitation loss and the gradient is back-propagated through the entire planning computation graph. The blue lines represent the flow of gradients for planning, while the red lines depict the meta-optimization learning signal and the components of the architecture affected by it. Note that the GDP iteratively plans across n_p updates, as indicated by the i^{th} loop.

The GDP uses gradient descent to optimize for a sequence of actions $\hat{a}_{t:t+T}$ to reach the encoded goal observation o_g from an initial o_t. Since the model is differentiable, back-propagation through time allows for computing the gradient with respect to each planned action in order to end up closer to the desired goal state. Each iteration of the GDP thus involves unrolling the trajectory of latent state encodings using the current planned actions, and taking a step along the gradient to improve the planning objective. The cumulative planning process forms a large, differentiable computation graph, chaining together each iteration of planning.

The actual learning signal is derived from an outer loss function, which supervises the entire computation graph (including the GDP) to output the correct action sequence. The outer loss can in principle take any form, but in this work we use an imitation learning loss derived from demonstrations. The outer loss provides task-specific grounding to optimize for representations that support effective iterative planning for the task and environment at hand, as the gradient is back-propagated through the entire iterative planning computation graph.

Training thus involves nested objectives. One can view the learning process as first deriving a plan to achieve the goal and then updating the model parameters to make the planning procedure more effective for the outer objective. In other words, we seek to learn the planning computation through its underlying representations for latent state encoding and latent forward dynamics.

Parameters: The model is composed of a forward dynamics model g_θ and an encoder f_ϕ, where θ and ϕ are neural network parameters that are learned end-to-end:

$$x_t = f_\phi(o_t) \qquad \hat{x}_{t+1} = g_\theta(x_t, a_t)$$

Specifically, f_ϕ is a convolutional network and g_θ is a fully connected network. Further architectural details can be found in the supplementary. [1]

Planning by Gradient Descent: The planner starts with an element-wise randomly initialized plan $\hat{a}^{(0)}_{t:t+T} \sim \mathcal{U}(-1, 1)$ and aims to minimize the distance between the predicted terminal latent state and the encoded goal observation. T denotes the horizon over which the agent plans, which can depend on the task and hence may be treated as a hyper-parameter, while n_p is the number of planning updates performed. Algorithm 1 describes the iterative optimization procedure that is implemented by the GDP.

Huber Loss: In practice, for $\mathcal{L}^{(i)}_{plan}$, we use a Huber Loss centered around x_g for well-behaved inner loop gradients instead of a direct quadratic $||\hat{x}^{(i)}_{t+T+1} - x_g||_2^{\,2}$. This usage is inspired from the Deep Q Networks paper of Mnih et al. (2015) and similar metrics have also been used by Levine et al. (2016) and Sermanet et al. (2017).

Action selection at test-time: At test-time, Algorithm 1

[1] We note that one could also use an action encoder $h_\alpha(a_t) = u_t$, with g_θ operating on x_t and u_t. A temporal encoder h would allow for abstract sequences of actions (options), for an option conditioned latent forward model g_θ. We work with flat sequences of actions, leaving hierarchical extensions for future work.

Algorithm 1 $GDP(o_t, o_g, \alpha) \rightarrow \hat{a}_{t:t+T}$

Require: α: hyperparameter for step size
  Randomize an initial guess for the optimal plan $\hat{a}^{(0)}_{t:t+T}$
  for i from 0 to $n_p - 1$ **do**
    Compute $x_t = f_\phi(o_t)$, $x_g = f_\phi(o_g)$
    for j from 0 to T **do**
      $\hat{x}^{(i)}_{t+j+1} = g_\theta(\hat{x}^{(i)}_{t+j}, \hat{a}^{(i)}_{t+j})$
    end for
    Compute $\mathcal{L}^{(i)}_{plan} = ||\hat{x}^{(i)}_{t+T+1} - x_g||_2^2$
    Update plan: $\hat{a}^{(i+1)}_{t:t+T} = \hat{a}^{(i)}_{t:t+T} - \alpha \nabla_{\hat{a}^{(i)}_{t:t+T}} \mathcal{L}^{(i)}_{plan}$
  end for
  Return $\hat{a}^{(n_p)}_{t:t+T}$

can be used to produce a sequence of actions. A more sophisticated approach is to use Algorithm 1 to re-plan at each timestep. The agent first plans a trajectory suitable to reach o_g from o_t, but only executes the first action, before replanning. This allows the agent to achieve goals requiring longer planning horizons at test-time even if the GDP was trained with a shorter horizon. This amounts to using model-predictive control (MPC) over our learned planner.

2.2. Imitation as the Outer Objective

An idea central to our approach is to directly optimize the planning computation for the task at hand, through the outer objective. Though in this work we study the use of an imitation loss as the outer objective, the policy can in principle be trained through any gradient-based policy search method including policy gradients (Schulman, 2016) and value functions (Sutton & Barto, 1998).

To learn parameters ϕ and θ, we do not directly optimize the planning error under $\mathcal{L}_{plan}$, but instead learn the planner insofar as it can imitate an expert agent by iteratively applying $\mathcal{L}_{plan}$ (Algorithm 2). The model is therefore trained to plan in such a way as to produce actions that match the expert demonstrations.

Note that the subroutine $GDP(o_t, o_g, \alpha)$ is an accumulated computation graph composed of several iterations of planning, each of which includes encoding observations and unrolling of latent forward dynamics through time. Learning end-to-end thus requires that we back-propagate the behavior cloning loss under the produced plan through the GDP subroutine as depicted in Figure 2. We note that the gradients obtained on the network parameters θ and ϕ from the outer objective are composed of first-order derivatives of these parameters. Therefore, even though the computation graph of UPN may seem long and complicated, it is not prohibitively expensive to compute.

In learning to plan via imitation, the agent jointly optimizes for latent state and dynamics representations that capture

Algorithm 2 Learning the Planner via Imitation

Require: $GDP(o_t, o_g, \alpha)$, expert $a^*_{t:t+T}$, step sizes α, β
  for n from 1 to N **do**
    Sample a batch of demonstrations $o_t, o_g, a^*_{t:t+T}$
    Compute $\hat{a}_{t:t+T} = GDP(o_t, o_g, \alpha)$
    Compute $\mathcal{L}_{imitate} = ||\hat{a}_{t:t+T} - a^*_{t:t+T}||_2^2$
    Update $\theta := \theta - \beta \nabla_\theta \mathcal{L}_{imitate}$
    Update $\phi := \phi - \beta \nabla_\phi \mathcal{L}_{imitate}$
  end for

notions of state comparison useful for the imitation task and that are traversible by gradient descent for trajectory optimization. This is naturally induced by the agent's reliance on the minimization of the latent distance between its predicted terminal state and goal state throughout the planning process. Thus, in necessitating *plannable* representations, the encoder learns an optimizable latent distance metric. This is key to the viability of using the learned latent space as a metric from which to derive reward functions for reinforcement learning.

2.3. RL with Rewards from a pre-trained UPN

A potential solution to the reward specification problem for image targets is to use distance to the target image in an abstract representation. Previous work such as Watter et al. (2015) and Finn et al. (2016b) propose unsupervised learning with autoencoders, while others attempt to fine-tune representations from Imagenet using auxiliary losses tailor-made for robotic manipulation Sermanet et al. (2017).

With the UPN having been trained to acquire *plannable* representations, we might naturally expect the learned latent space encoded by f_ϕ to serve well as an abstract representation through which rewards can be specified for RL visuomotor tasks with image targets from scratch. We can leverage the learned f_ϕ from pre-trained UPN to provide reward functions of the form $r(o_t, o_g) = -||f_\phi(o_t) - f_\phi(o_g)||_2^2$. The RL procedure is portrayed in Figure 3. As we will see in subsection 4.4, this enables transferring general task semantics learned by the UPN such that other RL agents of different form can be trained with RL *from scratch*.

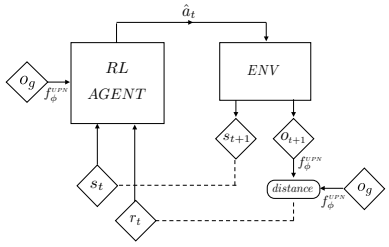


Figure 3. UPN-RL Agent: The rewards are based on the difference between o_t and o_g in the UPN latent space, while the policy is conditioned on joint angles and velocities specific to the agent, s_t; and the feature vector of the goal, $f_\phi^{UPN}(o_g)$. The agent has to reason about the goals and how to achieve them.

3. Related Work

Our work is primarily concerned with learning representations that can support planning for tasks described through an image target. Watter et al. (2015) and Finn et al. (2016b) take an unsupervised learning approach to learning such representations, which they use for planning with respect to target images using iLQR (Tassa et al., 2012). However, reconstructing all the pixels in the scene could lead to the encoding of state variables not necessarily useful in the context of planning (Higgins et al., 2017) and discard state variables that are not visually prominent (Goodfellow et al. (2016), Chapter 15). Our approach avoids this problem by explicitly optimizing a representation for *plannability through gradient descent* as the only criterion. Self-supervised methods that avoid pixel reconstruction by using other intermediate forms of supervision that can be obtained automatically from the data have also been used to learn representations for visuomotor control (Sermanet et al., 2016; 2017). We again differ by optimizing directly for what we need: plannable representations, instead of intermediate objectives.

There has been work in learning state representations usable for model-free RL when provided rewards (Lange et al., 2012; Jonschkowski & Brock, 2015; Jonschkowski et al., 2017; Higgins et al., 2017; de Bruin et al., 2018). The key difference in our work is that we focus on learning representations that can be used for defining metric-based rewards for new tasks, as opposed to just learning state representations for RL from external environment rewards.

Learning representations capable of providing metric based rewards naturally relates to inverse reinforcement learning (IRL) (Ng & Russell, 2000; Abbeel & Ng, 2004; Finn et al., 2016a; Ho & Ermon, 2016; Baram et al., 2017) and reward shaping (Ng et al., 1999). However, IRL from raw pixels is challenging due to the lack of sufficient constraints in the problem definition Our work can be viewed as connecting IRL and reward shaping: learning representations *amenable* to gradient-based trajectory optimization is one way to extract a perceptual reward function. However, we differ significantly from conventional IRL in that our derived reward functions are effective even for new tasks.

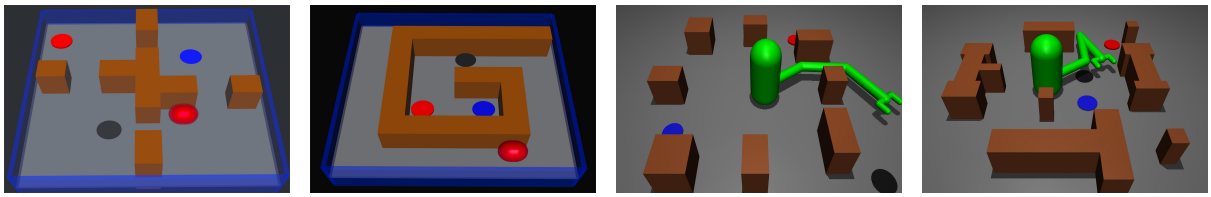
From an architectural standpoint, we embed a differentiable planner within our computation graph. Value iteration networks of Tamar et al. (2016) embed an approximate differentiable value iteration computation, though their architecture only supports discrete planning and is evaluated on tasks with sparse state transition probabilities. We seek a more general planning computation for more complex transition dynamics and continuous actions suitable for motor control from raw pixels. Concurrently, a few recent efforts have been developed to embed differentiable planning procedures in computation graphs (Guez et al., 2018; Farquhar et al., 2017). However, to our knowledge, our paper is the first to connect the use of differentiable planning procedures to learning reusable representations that generalize across complex visuomotor tasks.

The idea of planning by gradient descent has existed for decades (Kelley, 1960). While such work relied on known analytic forms of environment dynamics, later work (Schmidhuber, 1990) explored jointly learning approximate models of dynamics with neural networks. Henaff et al. (2017) adopt gradient-based trajectory optimization for model-based planning in discrete action spaces, but rely on representations learned from unsupervised pretraining. Oh et al. (2017) and Silver et al. (2016) have also explored forward predictions in a latent space that is learned by decoding the value function of a state. Our architecture is related in so far as distance to goal in the learned latent space can be viewed as a value function. However, we also differ significantly by not relying on extrinsic rewards and focusing on continuous control tasks.

4. Experiments

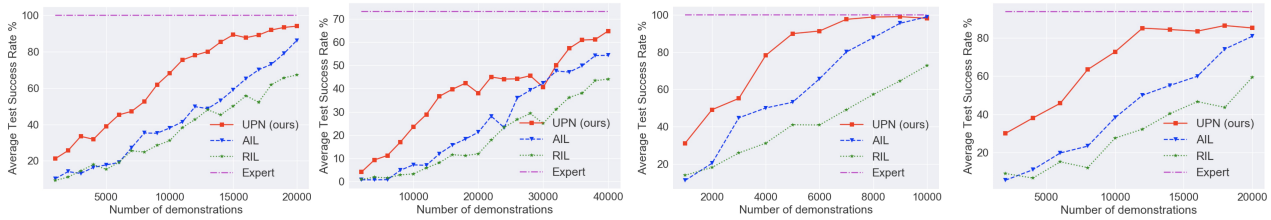
We designed experiments to answer the following questions: (1) does embedding a gradient descent planner help learn a policy that can map from pixels to torque control when provided current and goal observations at test-time ? (2) how does our method compare to reactive and autoregressive behavior cloning agents as the amount of training data varies? (3) what are the properties of the representation learned by UPN? (4) how can the learned representations from UPN be leveraged for transfer to new and more complex tasks, compared to representations from standard imitation and unsupervised methods (e.g. VAE)?

Methods for comparison: We consider two alternative imitation learning approaches for comparison: (1) a reactive imitation learner (RIL), composed of a convolutional feedforward policy that takes as input the current and goal observation; (2) an auto-regressive imitation learner (AIL), composed of a recurrent decoder initially conditioned on convolutionally encoded representations of the current and goal observation, trained to output a sequence of intermediate actions. Both (1) and (2) are methods adopted from Pathak* et al. (2018). These comparisons are important for studying the effects of the inductive bias of gradient descent planning that is embedded within UPN. More specifically, comparing to (1) allows us to understand the need for such an inductive bias, while comparing to (2) is necessary to understand whether the benefits are not purely due to recurrent computations. All methods are trained on the same synthetically-generated expert demonstration datasets. We refer the reader to the supplementary materials for details on the architectures and dataset generation.



(a) Pointmass config. 1 (b) Pointmass config. 2 (c) Reacher config. 1 (d) Reacher config. 2

Figure 4. Examples of the visuomotor tasks considered for the zero shot generalization study. We consider two 2D robot models: a force-controlled point robot and a 3-link torque-controlled reacher robot. We consider two types of generalization: fixing the obstacles while varying the target goals (FOVG) and varying both the obstacle and target goals (VOVG). These tasks require non-trivial generalization combining visual planning with low level motor control.



(a) Pointmass- VOVG (b) Reacher - VOVG (c) Pointmass- FOVG (d) Reacher - FOVG

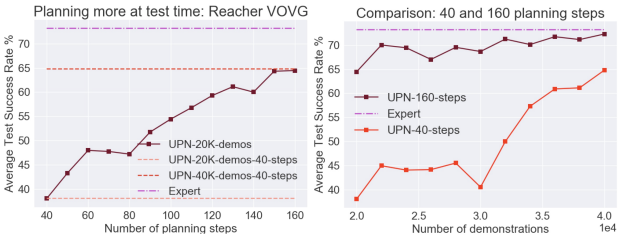
Figure 5. **Notation:** VOVG - Varying Obstacles and Varying Goals, FOVG - Fixed Obstacles and Varying Goals; Success on test tasks as a function of the dataset size. Our approach (UPN) outperforms the RIL and AIL consistently across the four generalization conditioned considered and is more sample efficient. As expected, the AIL improves with more data to eventually almost match the UPN. This illustrates the tradeoff between inductive bias and expressive architectures when given sufficient data.

4.1. UPNs Learn Effective Imitation Policies

Here, we study the suitability of the UPN for learning visual imitation policies that generalize to new goal-directed tasks. We focus on two tasks: (1) navigating a 2D point robot around obstacles to desired goal locations amidst distractors (Figures 4(a) and 4(b)), wherein the color of the goal is randomized; (2) a harder task of controlling a 3-DoF planar arm to reach goals amidst scattered distractors and obstacles, as shown in Figures 4(c) and 4(d).

For these tasks, we consider two types of generalization: (1) generalizing to new goals for a fixed configuration of obstacles having trained on the same configuration; (2) generalizing to new goals in new obstacle configurations having trained across varying obstacle configurations. Figures 4(c) and 4(d) show two different obstacle configurations for the reaching task, while the differently colored locations in Figure 4 represent varying goal locations.

We employ the action selection process described in subsection 2.1 with a chosen maximum episode length. Results shown in Figure 5 compare performance over a varying number of training demonstrations. As expected, the inductive bias of embedding trajectory optimization via gradient descent in UPN supports generalization from fewer demonstrations. With more demonstrations, however, the expressive AIL is able to almost match the performance of the UPN. This is consistent with the conclusions of Tamar et al. (2016), who observed that the benefit of the value iteration inductive bias shrinks in regimes in which demonstrations are plentiful. Note that generalization across obstacle configurations in the reacher case (Figure 4(c)) is a hard task; expert performance is only 73.12%. We encourage the reader to refer to the supplementary for further details about the experiment.



(a) (b)

Figure 6. **(a)** The effect of additional planning steps at test-time. UPN learns an effective gradient descent planner whose convergence improves with more planning steps at test-time. **(b)** A comparison of the success rate of UPN between 40 and 160 planning steps at test time with varying number of demonstrations on Reacher VOVG. Using 160 planning steps is consistently better than using 40 steps (though the relative benefit shrinks with more demonstrations) and allows the UPN to match the expert level.

4.2. Analysis of the Gradient Descent Planner

The UPN can be viewed in the context of meta-learning as learning a planning algorithm and its underlying representations. We take inspiration from Finn & Levine (2018), which studied a gradient-based model-agnostic meta-learning algorithm and showed that a classifier trained for few-shot image classification improves in accuracy at test-time with additional gradient updates. In our case, the inner loop is the GDP, which may not necessarily converge due to the fixed number of planning updates. Hence, it is worth studying whether additional test-time GDP updates yield more accurate plans and therefore better success rates.

Planning more helps: Figure 6(a) shows that with more planning steps at test-time, a UPN trained with fewer demonstrations (20000) can improve on task success rate beginning from 38.1% with 40 planning steps to 64.44% with 160 planning steps. As a reference, the average test success rate of the expert on these tasks is 73.12% while the best UPN model with 40 planning steps (trained on twice the number of demos (40000)) achieves 64.78%. Thus, with more planning steps, we see that UPN can improve to match the performance of a UPN with fewer planning steps but trained on twice the number of demonstrations. We also find that 160 steps is consistently better than using 40 steps (though the relative benefit shrinks with more demonstrations) and that the UPN is able to match expert performance (Figure 6(b)). This finding suggests that the learned planning objective is well defined, and can likely be reused for related control problems, as we explore in Sections 4.4 and 4.5.

4.3. Latent Space Visualization

We offer a qualitative analysis for studying the acquired latent space for an instance of the reacher with obstacles task. Given the selected initial pose, we compute the distance in the learned f_ϕ space for 150 random final poses and illustrate these distances qualitatively on the environment arena by color mapping each end-effector position accordingly. The result is shown in Figure 7; lighter blue corresponds to larger distances in the feature space. We see that the learned distance metric is *obstacle-aware* and task-specific: regions below the initial position in Figure 7 are less desirable even though they are near, while farther regions above are comparatively favorable.

4.4. Transfer to Harder Scenarios

We have seen in subsections 4.1 and 4.2 that UPNs can learn effective imitation policies that can perform close to the expert level on visuomotor planning tasks. In principle, deriving reward functions from a pre-trained UPN as explained in subsection 2.3 should allow us to extend beyond the capabilities of the expert on harder scenarios where the expert fails. We study this idea in the reaching scenario with obstacle configuration as presented in Figure 4(d). The difference between fixed and varying goals is that for varying goals, we feed in a feature vector of the goal image as an additional input to the RL policy. We use PPO (Schulman et al., 2017) for model-free policy optimization of the rewards derived from the feature space(s).

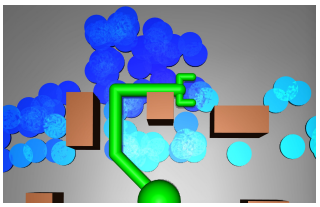


Figure 7. Visualization of the learned metric in the UPN latent space on the reacher with obstacles task. Lighter color → larger latent distance. The learned distance metric is obstacle-aware and supports obstacle avoidance.

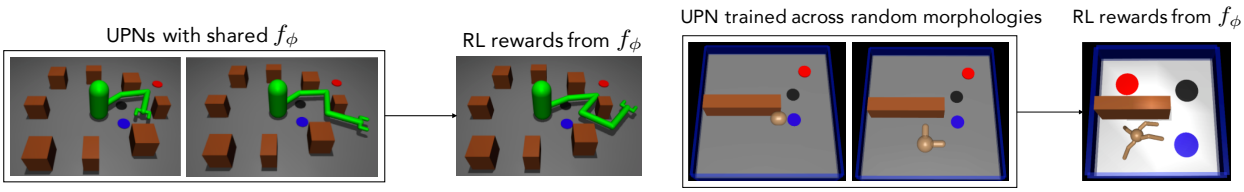
Table 1. Average Success Rate % in solving the task described in Figure 4(d) for fixed and varying goals

FEATURE SPACE	FIXED	VARYING
RIL-RL	0%	0.01%
AIL-RL	0%	4.72%
VAE-RL	20.23%	24.67%
UPN-160 IMITATION	45.82%	47.99%
EXPERT	46.77%	51.1 %
UPN-RL	**69.84%**	**71.12%**

Though the subsection 2.3 explains an RL procedure based on using the f_ϕ of a UPN, one could use a trained encoder f_ϕ from other methods such as our supervised learning comparisons RIL, AIL. In addition to RIL and AIL, a feature space we compare to is an encoder obtained from training a variational auto-encoder (VAE) (Kingma & Welling, 2013) on the images of the demonstrations. This comparison is necessary to judge how useful the feature space of a UPN is for downstream reinforcement learning when compared to pixel reconstruction methods such as VAEs. In Table 1, we see that reinforcement learning on the feature space of RIL and AIL clearly fail, while RL on the UPN feature space is better compared to that of a VAE. We also see that UPN-RL is able to outperform the expert and the imitating UPN-160.

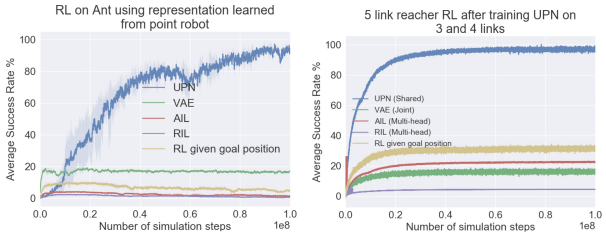
4.5. Transfer Across Robots

Having seen the success of reinforcement learning using rewards derived from UPN representations in subsection 4.4, we pose a harder problem in this subsection: can we leverage UPN representations trained on some source task(s) to provide rewards for target task(s) with significantly different dynamics and action spaces? We propose to do this by training and testing with different robots (morphological variations) on the same desired functionality (reaching / locomotion, around obstacles). This study will highlight



(a) Transfer to more complex topology (Reacher) (b) Transfer to new morphology (Point to Ant)

Figure 8. Transfer between robots as described in subsection 4.5.



(a) Point robot to Ant transfer (b) Reacher transfer

Figure 9. RL with rewards from the UPN representation is significantly more successful compared to other feature spaces (VAE, AIL, RIL, shaped rewards), suggesting that UPNs learn transferrable, generalizable latent spaces.

the extrapolative generalizability of UPN representations. The idea of trajectory optimization with a learned metric is a fundamental prior that can hold across a large class of visuomotor control problems. Having trained UPN to learn such a prior, it is natural to expect the underlying representation to be amenable to providing suitable metric based rewards for similar but unseen tasks. We craft two challenging experimental scenarios to verify this hypothesis.

Reacher with new morphology: Having trained a UPN with a shared f_ϕ and different g_θ for a 3-link and 4-link reacher (on the obstacles task), can we leverage the learned f_ϕ to specify rewards for reinforcement-learning a 5-link reacher to reach different goals around the same obstacles? Figure 8(a) visually depicts this experiment. To the best of our knowledge, such a transfer scenario has not previously been studied in the past for visuomotor control. The dynamics of a 5-link reacher are more complex (compared to 3 and 4 link reachers), thereby posing a harder control problem to solve at test time. However, a good path-planning reward function learned from 3 and 4-link reachers is likely to help for a 5-link reacher due to morphological similarities. We train the UPN on both the 3 and 4 link reachers to avoid overfitting the learned metric to a specific dynamical system. As comparison methods, we train RIL and AIL (with a multi-task (head) architecture), and a VAE (jointly on images from both the tasks).

Point to Ant: High level visual planning for navigation should be common across different robots, from a 2D point robot controlled through simple forces to a robot as complex as an 8-joint quadruped ant. We empirically confirm this via an experiment illustrated in Figure 8(b). Here, we learn representations with a UPN on demonstrations collected from a 2D point robot trained to traverse obstacles to reach varying goals. We randomize the robot's morphological appearance across demonstrations (Figure 8(b)), inspired by Sadeghi & Levine (2016); Tobin et al. (2017). This allows the UPN to learn an encoder f_ϕ that is robust to the creature appearance. We then replace the point robot with a 3D-torque-controlled ant to study how well the learned representation transfers to a harder task.

In Figure 9, we see that for both the transfer scenarios, RL with rewards from the UPN representation is more successful compared to other feature spaces (VAE, AIL, and RIL). Additionally, we also compare the UPN-RL setup to a naïve RL agent optimizing a spatial distance to goal in the coordinate space as the reward; this procedure assumes that the spatial position of the goal is known, unlike UPN, RIL, AIL, and VAE where the feature vector of the goal image is provided as input. We note that this method performs poorly, as expected, because the spatial distance in the coordinate space is not obstacle-aware. These results show that optimizing for the rewards derived from UPN correlates with task success, supporting our claim that UPNs learn generalizable and transferrable latent spaces.

5. Discussion

We studied the problem of learning generalizable representations for visuomotor control and introduced universal planning networks, a goal-directed policy architecture with an embedded differentiable planner, that can be trained end-to-end. Our extensive experiments demonstrated that (1) UPNs learn effective visual goal-directed policies efficiently; (2) UPN latent representations can be leveraged to transfer task-related semantics to more complex agents and more challenging tasks through goal-conditioned reward functions; and (3) the learned planner improves with more updates at test-time, providing encouraging evidence of meta-learning for planning. Our transfer learning results demonstrate that UPNs learn generic representations with task-specific structure of agency and goals useful for planning. Future work should investigate self-supervised ways to train UPN-like representations, borrowing ideas from Weber et al. (2017).

Acknowledgements

We thank Aviv Tamar, John Schulman, Rocky Duan, Alyosha Efros and colleagues from Robot Learning Lab at Berkeley for helpful feedback. This research was supported by an ONR PECASE N000141612723AS grant, NVIDIA, AWS, Berkeley EECS and Berkeley AI fellowships, and an NSF GRFP award.

References

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

Baram, N., Anschel, O., Caspi, I., and Mannor, S. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pp. 390–399, 2017.

de Bruin, T., Kober, J., Tuyls, K., and Babuska, R. Integrating state representation learning into deep reinforcement learning. *IEEE Robotics and Automation Letters*, PP(99): 1–1, 2018. doi: 10.1109/LRA.2018.2800101.

Deguchi, K. and Takahashi, I. Image-based simultaneous control of robot and target object motions by direct-image-interpretation method. In *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 1, pp. 375–380. IEEE, 1999.

Farquhar, G., Rocktäschel, T., Igl, M., and Whiteson, S. Treeqn and atreec: Differentiable tree planning for deep reinforcement learning. *arXiv preprint arXiv:1710.11417*, 2017.

Finn, C. and Levine, S. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *International Conference on Learning Representations*, 2018.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016a.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. Deep spatial autoencoders for visuomotor learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 512–519. IEEE, 2016b.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*, volume 1. 2016.

Guez, A., Weber, T., Antonoglou, I., Simonyan, K., Vinyals, O., Wierstra, D., Munos, R., and Silver, D. Learning to search with MCTSnets, 2018. URL https://openreview.net/forum?id=r1TA9ZbA-.

Henaff, M., Whitney, W. F., and LeCun, Y. Model-based planning in discrete action spaces. *arXiv preprint arXiv:1705.07177*, 2017.

Higgins, I., Pal, A., Rusu, A. A., Matthey, L., Burgess, C. P., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

Jonschkowski, R. and Brock, O. Learning state representations with robotic priors. *Autonomous Robots*, 39(3): 407–428, 2015.

Jonschkowski, R., Hafner, R., Scholz, J., and Riedmiller, M. Pves: Position-velocity encoders for unsupervised learning of structured state representations. *arXiv preprint arXiv:1705.09805*, 2017.

Kelley, H. J. Gradient theory of optimal flight paths. *ARS Journal*, 30(10):947–954, 1960.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lange, S., Riedmiller, M., and Voigtlander, A. Autonomous reinforcement learning on raw visual input data in a real world application. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–8. IEEE, 2012.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Ng, A. Y. and Russell, S. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*. Citeseer, 2000.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Oh, J., Singh, S., and Lee, H. Value prediction network. In *Advances in Neural Information Processing Systems*, pp. 6120–6130, 2017.

Pathak*, D., Mahmoudieh*, P., Luo*, M., Agrawal*, P., Chen, D., Shentu, F., Shelhamer, E., Malik, J., Efros, A. A., and Darrell, T. Zero-shot visual imitation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkisuzWRW.

Pinto, L., Gandhi, D., Han, Y., Park, Y.-L., and Gupta, A. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pp. 3–18. Springer, 2016.

Sadeghi, F. and Levine, S. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Schmidhuber, J. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *In Proc. IEEE/INNS International Joint Conference on Neural Networks*, pp. 253–258. IEEE Press, 1990.

Schulman, J. *Optimizing Expectations: From Deep Reinforcement Learning to Stochastic Computation Graphs*. PhD thesis, EECS Department, University of California, Berkeley, Dec 2016. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-217.html.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sermanet, P., Xu, K., and Levine, S. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

Sermanet, P., Lynch, C., Hsu, J., and Levine, S. Time-contrastive networks: Self-supervised learning from multi-view observation. *arXiv preprint arXiv:1704.06888*, 2017.

Silver, D., van Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., et al. The predictron: End-to-end learning and planning. *arXiv preprint arXiv:1612.08810*, 2016.

Sutton, R. S. and Barto, A. G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Tassa, Y., Erez, T., and Todorov, E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4906–4913. IEEE, 2012.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 23–30. IEEE, 2017.

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pp. 2746–2754, 2015.

Weber, T., Racanière, S., Reichert, D. P., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.