Mid-Level Visual Representations Improve Generalization and Sample Efficiency for Learning Active Tasks

Alexander Sax\textsuperscript{1,2} Bradley Emi\textsuperscript{3} Amir R. Zamir\textsuperscript{1,3} Leonidas Guibas\textsuperscript{2,3} Silvio Savarese\textsuperscript{3} Jitendra Malik\textsuperscript{1,2}

\textsuperscript{1} University of California, Berkeley  \textsuperscript{2} Facebook AI Research  \textsuperscript{3} Stanford University

http://perceptual.actor/

Abstract

One of the ultimate promises of computer vision is to help robotic agents perform active tasks, like delivering packages or doing household chores. However, the conventional approach to solving “vision” is to define a set of off-line recognition problems (e.g. object detection) and solve those first. This approach faces a challenge from the recent rise of Deep Reinforcement Learning frameworks that learn active tasks from scratch using images as input. This poses a set of fundamental questions: what is the role of computer vision if everything can be learned from scratch? Could intermediate vision tasks actually be useful for performing arbitrary downstream active tasks?

We show that proper use of mid-level perception confers significant advantages over training from scratch. We implement a perception module as a set of mid-level visual representations and demonstrate that learning active tasks with mid-level features is significantly more sample-efficient than scratch and able to generalize in situations where the from-scratch approach fails. However, we show that realizing these gains requires careful selection of the particular mid-level features for each downstream task. Finally, we put forth a simple and efficient perception module based on the results of our study, which can be adopted as a rather generic perception module for active frameworks.

1. Introduction

The renaissance of deep Reinforcement Learning (RL) started with the Atari DQN paper [28] in which it was shown a wide set of video games could be learned to play directly from pixels using RL. Robotics quickly adopted deep RL for learning to control, using frames from an on-board camera, and commonly referred to as pixel-to-torque. This interdisciplinary success has led to remarkable recent progress in RL research and has implications for various other fields, in particular, perception.

The basic premise of this approach, pertinent to perception, is: performing an active task can be effectively learned from scratch directly from images. The premise poses an existential question for computer vision: what is the use of computer vision, if all one needs from images can be learned from scratch using RL? In this paper, we focus on this question and try to identify what the consequences of this paradigm are, if and when vision can be learned from scratch using RL, and how computer vision could actually help with learning active tasks.

To take account of the situation, it is important to note that there are two common themes when learning from scratch using RL: I. the policies require a massive number of data points to learn as they are sample inefficient (e.g. DQN requires tens of millions of frames to learn [28]). Even state of the art Q-learning methods like Rainbow [18] still require millions of samples. II. the policies are of-
ten tested in the same environment as training, as they exhibit difficulties generalizing across environments with even modest differences. This is while generalization and sample efficiency are essential requirements for any practically useful system operating in the real-world. As an example, biological organisms are known to be capable of learning active tasks from few samples and effortlessly generalize them to places other than where learning occurred. We show that indeed the lack of an appropriate perception module is one of the primary causes of these two phenomena which can be consequently alleviated by adopting a perception solution. We recognize that conventional computer vision consists of defining a set of offline recognition problems, e.g. object classification, and tackling them on their own without a clear path towards effective integration in active frameworks. We will demonstrate that such standard vision tasks can be turned into mid-level vision skills and then integrated into RL frameworks for learning arbitrary active tasks.

To be more specific: our goal is to learn an arbitrary downstream active task like visual navigation. We assume a set of standard imperfect visual estimators (e.g. depth, orientation, objects, etc.) are available; we refer to them as mid-level vision tasks. We then study if and how mid-level vision can provide benefits towards learning the downstream active task, compared to not adopting a perception. Our metrics are how quickly the active task is learned and how well the policies generalize to unseen test spaces. We do not care about the task-specific performance of mid-level visual estimators or their vision-based metrics, as our sole goal is the downstream active task and mid-level vision is only in service to that.

We test three core hypotheses: I. if mid-level vision provides an advantage in terms of sample efficiency of learning an active task (answer: yes) II. if mid-level vision provides an advantage towards generalization to unseen spaces (answer: yes) III. if a fixed mid-level vision feature could suffice or a set of features would be essential to support arbitrary active tasks (answer: a set is essential). We use statistical tests where appropriate. Finally, we put forth a simple and practical perception module based on the findings of our study which can be adopted in lieu of raw pixels to gain the advantages of mid-level vision.

To perform our study, we needed to adopt an end-to-end framework for learning arbitrary active tasks and we chose deep RL—however any of the common alternatives, such as imitation learning or classic control, would be viable choices as well. In our experiments, we use neural networks from existing vision techniques [54, 58, 6, 52] trained on real images for each mid-level task and use their internal representations as the observed state provided to the RL policy. We do not use synthetic data to train the visual estimators and do not assume they are perfect. The code and trained models are available on our website.

2. Related Work

Our study has connections to a broad set of topics, including lifelong learning, un/self supervised learning, transfer learning, Reinforcement and Imitation Learning, control theory, active vision and several others. We overview the most relevant ones within constraints of space.

**Conventional (Offline) Computer Vision** encompasses a wide set of approaches—e.g. fully supervised learning [23], self-supervised learning [32, 35, 7, 56, 55, 33], unsupervised learning [9, 4, 7, 56, 8, 41]—adopted to solve various standard perception tasks, e.g. depth estimation [24], object detection [23] and segmentation [43], pose estimation [57], etc. The common characteristic shared across these methods is that they are offline (i.e. trained and tested on pre-recorded datasets) and evaluated in terms of their immediate tasks. In this paper we study how such methods can be plugged into a bigger framework targeted towards solving downstream active tasks. In recent years the computer vision community has become increasingly interested in robotic tasks.

**Reinforcement Learning**, [46] and its variants like Meta-RL [10, 14, 30, 12, 21] or its sister fields such as Imitation Learning [2], commonly focus on the last part of the end-to-end active task pipeline: how to choose an action given a state from the world. These methods can be viewed as users of our study as we essentially update their input state from pixel (or single fixed features) to a set of generic mid-level vision features.

**Feature Learning** literature shares its goal with our study: how to encode images in a way that provides benefits over just using raw pixels. There are a number of successful works in this area. Compression techniques like autoencoders [19] squeeze images into a lower-dimensional representation. Extensions [49, 22] to autoencoders may enforce desirable properties on the latent space. Another family, Generative Adversarial Networks [47, 13], formulates the problem as a game where the empirical data distribution is an optimal solution. Another method that often reduces the sample complexity compared to using raw pixels is “self-supervised” learning, which uses a novel loss function designed to encourage learning a single useful feature [56, 33, 32, 45, 50]. This can be considered a learned variant of hand-designed features like [25, 3]. Self-supervised approaches in particular have been used to reduce sample complexity for active tasks [16, 34, 29, 15]. Here we show that the appropriate choice of feature for active tasks actually depends on the desired downstream task, hence no single feature was able to sufficiently support arbitrary tasks. Consequently, a set of mid-level features is necessary. This is consistent with recent works in computer vision showing that no single vision feature could be the perfect transfer source for all other vision tasks, echoing the need for a set [54].
Transfer Learning reuses pre-learned knowledge to somehow benefit learning a new task outside of the original training distribution. [48, 36, 44, 11, 27, 31, 26, 37, 53, 38]. Our study is a case of transfer learning with the important characteristic that we are interested in transferring from mid-level static recognition tasks to downstream sequential active tasks.

3. Methodology

3.1. Transferring from Mid-Level Vision to Downstream Active Tasks

How might we use mid-level features to support a downstream task? We choose to use a typical transfer learning setup in which we use a pretrained neural network as a feature extractor. For example, we might take a network trained for image reshading estimation, pass in the observed image and use the intermediate “reshaded” representations as the sole input to another neural network. Only the second neural network is updated during training. This setup is shown graphically in figure 2. Other network configurations are possible and our observed performance gain is therefore a lower bound and any better method would increase the significance of our conclusions. Freezing the features has the advantage that they can be reused without degrading the performance on already-learned tasks, in order to learn multiple tasks over the lifetime of the agent.

While the features can be trained on the same image distribution that the agent will see at test time, our features were trained on standard computer vision datasets. This induces a problem of domain shift where the perception networks are not necessarily well-calibrated in the new environment. However, our network outputs are still informative and we show qualitative examples in figures 3 and the supplementary material.

3.2. Hypothesis Testing

The following section details how we test our three core hypotheses (see Fig. 2). We use nonparametric significance tests where possible since nonparametric approaches avoid unnecessary assumptions on the shape or type of the population distributions. For pairwise tests, we use the Wilcoxon rank-sum test and we correct for multiple comparisons by controlling the False Discovery Rate (FDR) through Benjamini-Hochberg [5].

---

1 All RL evaluation suffers from some additional estimation error stemming from the fact that we estimate the average performance of a particular random seed, and then use that to further estimate the expected perfor-
Hypothesis I: Does mid-level vision provide an advantage in terms of sample efficiency when learning an active task? We want to examine whether using an agent equipped with mid-level vision can learn faster than a comparable agent learning tabula rasa, from images alone. Since there is significant variation in the asymptotic performance between different random seeds, any given agent may never achieve a given level of performance. We could not come up with a satisfying metric of relative sample efficiency that could be robustly evaluated from just a few seeds. We therefore provide the curves and ask the readers to use their judgment, which is currently standard practice. Unless designated otherwise, all of our curves are evaluated in unseen test environments.

Hypothesis II: Can mid-level vision features generalize better to unseen spaces? If our mid-level perception networks provide standard encodings of the visual observations that are less space-specific, then we might expect that agents using these features will also learn policies that are robust to differences between the training and testing environments and consequently generalize better. We test this in H2 by evaluating the performance of scratch and feature-based agents at the end of training, after convergence. Specifically we test which, if any, feature-based agents outperform scratch in unseen test environments, correcting for multiple hypothesis testing.

Hypothesis III: Can a single feature support all arbitrary downstream tasks? Or is a set of features required for that? We show there is no universal visual task that provides the best features regardless of the downstream activity. We show this by demonstrating rank-reversal. Rank reversal is when the features that are best for one task are not ideal for the other (and vice-versa). Concretely, we find that depth estimation features perform the best for exploration while object classification is ideal for target-driven navigation. We demonstrate rank reversal by showing that for exploration $\text{perf}_{\text{exp}}(\hat{\phi}_{\text{depth}}) > \text{perf}_{\text{exp}}(\hat{\phi}_{\text{object detection}})$ while for navigation $\text{perf}_{\text{nav}}(\hat{\phi}_{\text{object detection}}) > \text{perf}_{\text{nav}}(\hat{\phi}_{\text{depth}})$ (performing hypothesis tests for both of these).

3.3. Mid-Level Feature Selection Module

If the three formulated hypotheses are correct, as we will demonstrate in Sec. 4.4, then a proper perceptual support demands integrating one or a few mid-level visual representations from a larger set, conditional on the actual downstream task. Though it is viable and probably advantageous to define a complicated model, we propose an extremely simple one as a jumping-off point. Our feature-selection module simply takes a sparse linear combination of the pretrained features.

Specifically, the module learns a percept $\hat{\phi}(I)$ of an input image $I$. The learned percept is a sparse blend of the outputs from $n$ pretrained feature extractors $\phi_1, \ldots, \phi_n$. Our module chooses blending weights $\mu_i$ for the $n$ mid-level features and we enforce that at most $k$ of these are nonzero. This sparsity reduces noise in the features and allows us to only evaluate a few of the $\phi_i$ at any given time, saving significant computational complexity.

The learned percept is then the weighted average of these weights:

$$\hat{\phi}(I) = \sum_{i=1}^{n} \mu_i \cdot \phi_i(I).$$

There are a myriad of ways to train such a module. For example, one could train the selection and blending weights via policy gradients, with gradient boosting, as a bandit problem (Thompson sampling or upper confidence bounds), or using supervised learning (e.g. noisy gates or the Gumbel-Softmax trick). We choose supervised learning with the noisy gating formulation in [42].

The module can additionally be conditioned on the input image $I$. The formula is then:

$$\hat{\phi}(I) = \sum_{i=1}^{n} \mu_i(I) \cdot \phi_i(I).$$

Such a setup encourages the agent to learn perception dynamics because it must choose a limited number of percepts to use for a given observation. One can imagine myriad useful improvements such as finetuning the $\hat{\phi}(i)$ to support a set of tasks, or designing the gating function $\mu$ to be especially adaptable (the gating is meta-learned).

4. Experiments

In this section we describe our experimental setup and present the results of our hypothesis tests and of the selection module. With 20 vision features and 4 baselines, our approach leads to training between 3-8 seeds per scenario in order to control the false discovery rate. The total number of policies used in the evaluation is about 600 which took 90,000 GPU hours to train.

4.1. Experimental Setup

Environments: We are ultimately interested in how to reduce sample complexity for agents learning in the real world. There are two options for training—either on a real robot or by training in simulation. Training on a real robot is practically slow and tedious, but more importantly does not provide an easy way to control experiments and reproduce results for a proper statistical hypothesis test. Thus, due to the scale of the study, we opt to train in simulation.
Figure 4. Visualization of training and test buildings from Gibson database [52]. The training space (highlighted in red and zoomed on the left) and the testing spaces (remaining on the right). Actual sample observations from agents virtualized in Gibson framework [52] are shown in the bottom of each box.

The downside of training in simulation is that there is a realism gap between the simulator and the physical world. We attempt to mitigate this issue in two ways. First, we choose a recent simulator (Gibson [52]) that is designed to be perceptually similar to the real world as it operates by virtualizing scans of real buildings. Gibson is also integrated with the PyBullet physics engine which contains a fast collision-handling system used to simulate dynamics.

Second, we also perform universality experiments in a second simulator, VizDoom [20]. VizDoom [20] is based on the 1992 game Doom and is one of the simplest examples of a 3D environment. It allows the agent to move around and contains a rudimentary physics engine that handles momentum and enables some basic interactions. The latter is of interest to us since it unlocks certain tasks that are not currently feasible in Gibson (e.g. opening a door, removing an enemy, etc). VizDoom is visually distinct from Gibson and we include it to show that our findings are rather robust to the idiosyncrasies of the particular environment.

4.1.1 Train/Test split

For each environment we define a clear train/test split. In Gibson we train in one building and test in different and completely unseen buildings used only for evaluation (fig. 4). We also test in 10 additional unseen spaces of comparable size and the results are in section 4.5. In Gibson, the training space for the visual navigation task covers 40.2 square meters and the testing space covers 415.6 square meters. The training space for the local planning and exploration tasks covers 154.9 square meters and the testing space covers 1270.1 square meters. The universality experiments in Doom also use a train/test split of textures which is provided in the supplementary material.

4.1.2 Downstream Active Tasks

We try to choose practically useful tasks in order to test our hypotheses. The tasks are visual target-driven local navigation, visual exploration, and local planning. These are depicted in figure 5 and described below.

Visual Target-Driven Local Navigation: In this scenario the agent must locate a target object as fast as possible with only sparse rewards. Upon touching the target there is a large positive reward and the episode ends. Otherwise there is a small negative reward for living. The target visually remains the same between episodes although the location and orientation of both the agent and target are randomized according to a uniform distribution over a predefined boundary within the floor plan of the space. In Gibson, the target is a box and in Doom, the target is a green torch, but the agent must learn to identify of the target during the course of training. The maximum episode length is 400 timesteps and the shortest path averages around 30 steps.

Visual Exploration: For Visual Exploration, the agent is tasked to visit as many new parts of the space as quickly as possible. The environment is partitioned into small occupancy cells and and the cells are “unlocked” upon being seen by the agent. The reward at each timestep is proportional to the number of revealed occupancy cells. The episode ends after 1000 timesteps. The agent is equipped with a myopic range scanner. This scanner reveals the area directly in front of the agent for up to 1.5 meters. Since our agents are memoryless, we provide them with an odometric
In all tasks, the state space contains the RGB image and minimum amount of side information so that the task is solvable. We stack the most recent 4 RGB frames as input and do not share weights between these frames to allow the agent to infer its local dynamics. Unlike the common practice in reinforcement learning, we do not include any proprioception information such as the agent’s joint positions or velocities or any other side information that could be useful, but is not essential to solving the task, such as a map of obstacles or floor layout. For visual navigation, the state space is only the image. For local planning, the agent also receives the vector to the target in its own inertial reference frame as a vector \([r, \cos \theta, \sin \theta]\) where \(r\) is the Euclidean distance to the target and \(\theta\) is the angle relative to the agent’s heading in the ground plane. For visual exploration, the task requires some form of memory for the agent to know where it has already been. Since our neural network architecture is memoryless (aside from the frame stacking), we choose to encode the memory as an occupancy grid which is translated and rotated to align with the agent’s inertial reference frame, whose cell values are 1 if the agent has already observed a given cell and 0 otherwise. The occupancy grid contains no global information about the scene such as walls or obstacles; it is only the previous output of the robot’s laser sensor.

### 4.1.4 Action Space

In all tasks in this section we assume that there is a low-level controller for robot actuation. Therefore the policies have an action space of

\[ A = \{ \text{turn left}, \text{turn right}, \text{move forward} \} \]

move_forward corresponds to a 0.1 m translation in the direction of the robot’s heading in the ground plane, and turn_left and turn_right correspond to in-place rotations of the robot’s heading of \pm 0.14 radians. No frame skipping is used for Gibson. For Doom, actions are selected and repeated for 4 frames. All actions are available at every timestep, with the physics engines responsible for enforcing the physical boundaries of the spaces.

### 4.2. Learning Setup

In all experiments we use the common Proximal Policy Optimization (PPO) [40] algorithm with Generalized Advantage Estimation [39]. PPO is a stable and well-tested algorithm. Due to the computational load of rendering perceptually realistic images in Gibson we are only able to use a single rollout worker and we therefore decorrelate our batches using experience replay and off-policy variant of PPO. The formulation is similar to Actor-Critic with Experience Replay (ACER) [51] in that full trajectories are sampled from the replay buffer and reweighted using the first-order approximation for importance sampling. We include the full formulation in the supplementary material. For the universality experiments in Doom, we use standard

---

2 This problem formulation is equivalent to assuming the initial coordinates to the target are given, and the robot has a perfect localization system (ideal IMU). In a deployment setting, noise could be added to the target vector to simulate real-world conditions.
PPO with 16 rollout environments. The standard PPO objective is

\[
\max_{\theta \in \Theta} \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[ \log \pi_\theta(a_t | s_t) \hat{A}_t \right] 
\]

where \(\hat{A}_t\) is the advantage function at timestep \(t\) (some sufficient statistic for the value of a policy at timestep \(t\), in our experiments we choose the generalized advantage estimator [39]) and \(\tau\) is a trajectory drawn from the current policy.

For each task and each environment we conduct a hyperparameter search optimized for the scratch baseline (see section 4.3). We then fix this setting and reuse it for every feature. This setup should favor scratch and possibly other baselines that use the same architecture.

For our experiments we use a set of 20 different computer vision tasks. This set covers various common modes of computer vision tasks, from texture-based tasks like denoising, to 3D pixel-level tasks like depth estimation, to low-dimensional geometric tasks like room layout estimation, to semantic tasks like object classification.

Our feature networks were trained on a dataset of 4 million static images in of indoor scenes [54]. We use the pretrained networks of [54]. Each network encoder consists of a ResNet-50 [17] without a global average-pooling layer. This preserves spatial information in the image. The feature networks were all trained using identical hyperparameters.

All network architectures and full experimental details, as well as videos of the pretrained networks evaluated in our environments, are included in the supplementary material.

4.3. Baselines

We include several controls to provide a baseline for the visual-feature-based agents and to address possible confounding factors.

**Scratch Learning:** Learning from scratch, or “vanilla” RL for the perception aspect, is among the common practices today. In this condition the agent starts with an appropriate random initialization and receives the raw RGB image as input. This baseline uses the common AtariNet [28] tower.

**Blind Intelligent Actor:** The Blind Intelligent Actor (blind) baseline is the same as scratch except that the visual input is constant and does not depend on the state of the environment. The blind agent indicates how much performance can be squeezed out of the nonvisual biases, correlations, and overall structure of the environment. If our tasks were essentially nonvisual, e.g. a narrow maze where the layout leads the agent to the target, then that would manifest as a small performance gap between blind and scratch.

**Random Nonlinear Projections:** To rule out the possibility that the perceptual architecture, not the source task, is the primary factor for good representations we include the Random Nonlinear Projection (random) baseline. This condition is the same as the pretrained features condition.
except that this network is randomly initialized and then frozen. As a result, the policy network learns from a random nonlinear projection of the input image. These features contain much of the information in the original image.

**Pixels as State:** This baseline considers the possibility that the small representations size is easier to learn from. Pixels-as-state downsamples the input image to a 16x16x3 image and then stacks two of these and two other copies of the greyscale version to produce a 16x16x8 tensor; that is the same shape as the pretrained activations. This tensor is then passed as the representation.

### 4.4. Experimental results on hypothesis testing I-III

We report our findings for the effect of intermediate representations on sample efficiency and generalization. All the results are evaluated in the test environment with multiple random seeds, unless otherwise explicitly stated.

#### 4.4.1 Hypothesis I: Sample Complexity Results

In this experiment we check whether an agent can learn faster using pretrained visual features than it would be able to learn from scratch. We evaluate 20 different features against the four control groups on each of our tasks: visual target-driven local navigation, visual exploration, and local planning. As shown in Fig. 6, we find that in all cases feature-based the agents learn significantly faster and may achieve a higher final performance than an agent that learns from scratch, even after averaging over many random seeds. We explore when an agent may not achieve higher test performance in section 4.6.

#### 4.4.2 Hypothesis II: Generalization Results

Do policies trained with pretrained features generalize better to unseen test environment? The previous experiment tested how quickly learning saturated, but this experiment tests for superior generalization performance with a given level of data. We find that specific feature-based policies exhibit superior generalization performance compared to scratch when tested in environments unseen at training time.

**Generalization Significance Analysis:** As shown in Fig. 7, we find that for each of the tasks there are some features that generalize significantly better than scratch. We used a nonparametric significance test and adjusted for multiple comparison, using a False Discovery Rate of 20%. If there were no actual difference, the probability of all of these results being spurious is < 0.002 for exploration (fig. 7, center) and negligible for navigation and local planning (fig. 7, left, right). After using the additional seeds from the follow-up experiment in the next section, the p-value for exploration is also negligible.

**Generalization Gap:** We found a large generalization gap between agent performance in the training vs. test

---

Figure 7. **Features vs. scratch.** The plots above show training and test performance of scratch vs. some selected features throughout training. For all tasks there is a significant gap between train/test performance for scratch, and a much smaller one for the best feature. The tables above show significance tests of the performance of feature-based agents vs from scratch in Gibson. P-values come from a Wilcoxon rank sum test, adjusted for multiple hypothesis testing with a FDR of 20%. Significant rows are in white, and these blocks are ordered by average episode reward. The plots show training and testing curves, and there are significant gaps between them. Scratch often fails to generalize (bottom), while feature-based agents generalize better (top). Sometimes, models may appear to learn in the training environment, but they fail at test time—underlining the importance of a good test environment in RL.
environments, shown in the plots in fig. 7. All our policies exhibit some gap, but agents trained from scratch seem to overfit completely—they do not show test improvement during training. We note that Autoencoder learns quickly in the training environment but fails to carry that over to the test set. Since Variational Autoencoders are a commonly-used form of perception in RL, we caution that such methods must be evaluated in terms of performance in a test environment and not solely on the training set.

**Qualitative Generalization Results:** Feature-based policies behave qualitatively differently than those trained from scratch. Different features also exhibit different types of behaviors. Fig. 6 highlights these differences by plotting the trajectories of random rollouts for various policies. We see that the navigation agent trained with semantic features is able to effectively identify and then navigate to the target. However, the semantic agent is not very adept at exploring through hallways and doors, as evidence in the exploration task. This is where the depth-based agent shines: despite not having fine grained knowledge about the objects in the scene, as is clear in the navigation task, the depth-based can cover much more ground in the exploration task by seeking paths that lead to wide open spaces. Both agents perform noticeably better than scratch, which wanders about the test environment seemingly at random.

### 4.4.3 Hypothesis III: Rank Reversal Results

It is well-known that ImageNet-based features transfer well. Indeed, such pretraining is often the default choice. We find that there may not be one or two single features that consistently outperform all the others. Instead, the choice of pretrained features should depend upon the downstream task. This experiment exhibits a case of *rank reversal* where features that work well on one task are not ideal for another, and vice-versa.

**Rank Reversal Significance Analysis:** We compare the top-performing navigation feature against the top-performing exploration feature. It happens that the best feature for Gibson navigation was indeed an Object Classification network based on ImageNet, but the best feature for Gibson exploration in was Distance Estimation. For navigation, Object Classification was better than Distance Estimation at the $\alpha = 0.0005$ level. The order is reversed for exploration, also at the $\alpha = 0.0005$ level.

**Rank Reversal Among Related Visual Tasks:** The trend of rank reversal appears to be a widespread phenomenon. Fig. 9 shows that semantic features seem to be useful for navigation while geometric features are useful for exploration. In Gibson, the graph is nearly complete bipartite (indicating that this distinction is quite useful). In the universality experiments (partially shown in figure 9), a similar trend holds despite the fact that Doom is visually quite distinct from Gibson.

Similarly, we see in figure 8 that the trend is not just among the top few features, but holds for families of computer vision tasks found in [54].

### 4.5. Robustness in Additional Environments

We repeated our testing in 9 other buildings to account for the possibility that our main test building is anomalous in some way. We found that the average reward in our main test building and in the 9 other buildings was extremely strongly correlated with a Spearman’s rho of 0.93 for navigation and 0.85 for exploration. Full experimental setup and results are included in the supplementary material.

### 4.6. Universality Experiments in Doom

We also implemented navigation and exploration in Doom to evaluate whether one can expect to see similar effects hold across environments. We found (shown in fig. 8) that features which perform well in Gibson also tend to perform well in Doom, and that similar tradeoffs exist in between tasks regardless of environment. Specifically, we again find the geometric/semantic distinction from Gibson appearing in Doom, and the results are highly statistically
significant (figure 9). We also find that there is no universal feature in either Doom or Gibson, and that maximizing the combined score (see figure 8) requires choosing the third- or fourth-best feature for any given task.

We also found that features were more robust to changes in texture than learning from scratch. While scratch achieves the highest final performance when the agent learns in a video game environment where there are many train textures that emulate the test textures (fig. 6), scratch fails to generalize when there is little or no variation in texture during training. On the other hand, feature-based agents are able to generalize even without texture randomization, as shown in fig. 10.

4.7. Evaluation of Representation Selection Module

Since the choice of feature seems to have a significant impact on the agent’s test performance, we show that the feature selection can be stably learned using our perception module. We evaluated the stability of feature selection in Doom since we could not fit all 20 perception networks as well as the Gibson environment onto a V100. However, this is not a fundamental limitation of the method and the feature rankings in Doom were shown in section 4.6 to be similar to those in Gibson. To fit the model into memory in Doom, we needed to use a single rollout worker. Furthermore, we reduced the learning rate by 10× to give the policy network time to adapt to changing perception.

In figure 11 we examined which feature the module eventually selects between 20 options. The module selects the same final feature even when presented with other subsets of perception features (11, 6, and 3 options). We note that using only one rollout worker (the setup for this experiment), scratch was wholly unable to learn either task, while the module-based agent meaningfully improved during training.

5. Conclusion and Limitations

We investigated the role of mid-level perception for learning active tasks using an RL platform. Our study suggested a notable benefit associated with adopting proper perceptual features, in contrast to using pixel as the state of the world or learning perception entirely from scratch via RL. The benefits were particularly stark in terms of sample complexity and generalization to unseen spaces. We consequently put forth a simple and efficient module for mid-level feature selection based on the findings of our study.

In retrospect, the finding that mid-level vision improves learning speed and generalization to new places is somewhat expected. Mid-level features encapsulate a readable and easy to understand state of the world (e.g. for 3D features, remove shadows and texture to convey the true underlying geometry) and are presumably designed to provide generic abstract information about the world that is not specific to a certain place. We found that by kickstarting the perception of an active system with such representations instead of bewildering the entire system with raw unprocessed sensory data that is full of information but hard to parse, the system will develop rewarding behavior faster.

It is worth noting a number of limitations of our framework. Our selection of active tasks was primarily oriented around locomotion. Though locomotion is a significant enough problem, our study does not necessarily convey conclusions about other important active tasks, such as manipulation. Also, given that RL was our experimental platform, our findings are enveloped by the limitations of existing RL methods, e.g. difficulties in long-range exploration or credit assignment for sparse reward functions. We also use a fixed basis of mid-level features during learning.

What exactly these mid-level tasks should be, how to learn their estimators efficiently, and how to incrementally expand the dictionary or improve each element are important research questions. Answering them would have benefits towards adapting the agents for the perceptual characteristics of new spaces, reducing computational cost, and would bring the problem closer to true life-long learning.

Acknowledgements We gratefully acknowledge the support of ONR MURI (N00014-14-1-0671), NVIDIA NGC beta, and TRI. Toyota Research Institute (“TRI”) provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.
References

[1] Incorporation of clustering effects for the wilcoxon rank sum test: A large-sample approach. *Biometrics*, 59(4):1089–1098, 2003. 4

[2] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML ’04, pages 1–, New York, NY, USA, 2004. ACM. 2

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. 2

[4] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 2

[5] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995. 3

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2

[7] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 2

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014. 2

[9] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010. 2

[10] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. 2

[11] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *Robotics and Automation (ICRA)*, 2016 *IEEE International Conference on*, pages 512–519. IEEE, 2016. 3

[12] C. Finn, K. Xu, and S. Levine. Probabilistic Model-Agnostic Meta-Learning. *ArXiv e-prints*, June 2018. 2

[13] J. Fu, J. D. Co-Reyes, and S. Levine. EX2: exploration with exemplar models for deep reinforcement learning. *CoRR*, abs/1703.01260, 2017. 2

[14] E. Grant, C. Finn, S. Levine, T. Darrell, and T. L. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *CoRR*, abs/1801.08930, 2018. 2

[15] S. Gupta, J. Davidson, S. Levine, R. Suktankar, and J. Malik. Cognitive mapping and planning for visual navigation. *CoRR*, abs/1702.03920, 2017. 2

[16] D. Ha and J. Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 7

[18] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298, 2017. 1

[19] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[20] M. Kempka, M. Widydmuch, G. Runc, J. Toczek, and W. Jaskowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. *CoRR*, abs/1605.02097, 2016. 5

[21] T. Kim, J. Yoon, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian Model-Agnostic Meta-Learning. *ArXiv e-prints*, June 2018. 2

[22] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems* 25, pages 1097–1105. Curran Associates, Inc., 2012. 2

[24] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016. 2

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 2

[26] Z. Luo, Y. Zou, J. Hoffman, and L. F. Fei-Fei. Label efficient learning of transferable representations across domains and tasks. In *Advances in Neural Information Processing Systems*, pages 164–176, 2017. 3

[27] L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI*, volume 7, pages 608–614, 2007. 3

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015. 1, 7

[29] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson. Visual representations for semantic target driven navigation. *CoRR*, abs/1805.06066, 2018. 2

[30] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. 2

[31] A. Niculescu-Mizil and R. Caruana. Inductive transfer for regression. In *Sixth International Conference on Artificial Intelligence and Statistics*, pages 339–346, 2007. 3

[32] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 2

[33] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. *arXiv preprint arXiv:1708.06734*, 2017. 2
