Foundation Models for Embodied AI

by

Sumedh Anand Sontakke

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

May 2024

Acknowledgements

This thesis would not have been possible without the unwavering support and inspiration from my advisors Professor Laurent Itti and Professor Erdem Bıyık. Laurent accepted me into his lab as I was switching to machine learning from an undergrad in physics-based electrical engineering (e.g., theory of EM fluxes, power systems, and AC machines). He allowed me to learn about my own research interests and gave me the freedom to select problems that I wished to pursue. He has been kind and patient, especially during the pandemic. I am ever grateful for his enthusiastic *"That's great, do it!"* whenever I come to him with a new idea.

Erdem and I started working together as I was starting my final year and I am so very thankful for his tremendous eye for detail, commitment to his students, and kindness. Our relationship more closely resembles a senior grad student mentoring a junior grad student — which has been extremely helpful as I navigated the closing stages of the PhD. His feedback (always provided kindly) improves whatever I do — whether that is making figures for my papers or polishing my job talk. The highest compliment I can pay to both Laurent and Erdem is that I want to be like you when I grow up. Leaving USC will be bittersweet because I will miss working with you both daily.

Many thanks to the seniors in my program — Dr. Karl Pertsch and Dr. Sébastien M. R. Arnold for teaching me how to do research as a PhD student. I learned the iterative process of grad student descent from you. Thank you for your patience and kindness.

I would like to thank my closest collaborator and great friend Jesse Zhang. I learn so much from him every day — whether that's in the gym or at the lab. Jesse lifts my weights for me when they are too heavy for me. Projects feel achievable when he joins me on them. I am so proud of what you have achieved so far Jesse, and I can't wait to work with you again.

I would also like to thank my lab mates at the Lira Lab: Anthony, Pavel, Yigit, Yutai, and Ayush — my brothers-in-arms. I would also like to thank my labmates at the iLab: Gozde, Andy, Kiran, Adam and Shixian for all the memories.

My parents have sacrificed so much for my brother and me and I am so very thankful to them for it. This PhD would not have been possible if it weren't for your calls every day, giving me strength and inspiration. My brother is growing into a kind, intelligent, and hardworking young man. I am excited to see what you achieve and I hope great things for you. I hope you are all proud and I love you very very much.

Finally, I would like to thank my fiancée, Priyamvada. I have spent this PhD living away from you — no more. I can recall the countless times when I have called her in tears because my experiments wouldn't converge. Thank you for your patience and love. Thanks for letting me do this PhD and agreeing to live away from me for 5 years. I love you very much.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction: Building Generalist Algorithms

We live in interesting times! During the process of writing this introduction, I read through my old qualifying examination document and was surprised by how much the ML world has changed during this relatively short time. That old quals document outlined desiderata like sample efficiency, disentanglement, etc. Of course, these things still matter, but with the advent of truly scalable foundation models, the opportunity to build generalist agents has taken priority. My excitement is of course, tempered when I think of what Tom Cargill from Bell Labs said:

> *The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time.*

When I began the PhD, modern Reinforcement Learning (RL) systems [163, 83, 106, 99] had shown impressive results on a host of benchmarks from Atari Games [107], Locomotion [99] to even autonomous control of Stratospheric Balloons [9]. They were still, however, notoriously sample inefficient [131, 110], often requiring millions of interactions with their environments. They also were known to require carefully constructed reward functions and were prone to failure in sparse reward settings [173, 135] (this is still the case). Finally, they were specialists — they generally solved one task at a time. I call this the *pre-foundation model era.*

In contrast of course, humans learn from relatively few demonstrations, requiring no external reward functions and do so without millions of steps of exploration [61, 25]. I began the PhD, with this anthropomorphization as an inspiration.

In the first part of this dissertation, during this *pre-foundation model era* I worked on how knowledge can be collected by RL systems during autonomous exploration. To this end, I invented a new form of intrinsic curiosity, dubbed **Causal Curiosity, published as a spotlight paper at ICML 2021** [158], which encourages agents to be curious about the differences in the properties (unknown to the agent) of a set of environments. This form of curiosity allows the agent to learn behaviors akin to experimental behaviors performed by human scientists. For example, given environments with different values of masses of objects, our agents learned to grasp and lift each object up to discover how much they weigh. Designing reward functions for grasping is a challenging problem [82] and it was very interesting to see these emergent behaviors using Causal Curiosity.

I also used Causal Curiosity to detect out-of-distribution (OOD) environments. RL agents are known to be brittle to distribution shifts [165] a fact which I verified in our follow-up work, **GalilAI, published at AISTATS 2022** [157]. Thus, detecting environments that are out-of-distribution is a step in the direction of deploying existing RL algorithms in-the-wild safely. GalilAI uses Causal Curiosity to perform experiments to determine whether a new environment is within distribution or not. Perhaps one of the most interesting emergent behaviors we obtained was when we set up an experiment where the agent interacted with a set of environments with varying masses and at test time was provided an environment with a different gravitational acceleration as an OOD environment. Using GalilAI, our agents learn to discern between high and low gravitational environments - while being invariant to their mass - through a free-falling behavior. These behaviors mimic Galileo's experiments (dropping objects of different masses from the top of the Tower of Pisa to discover that they fall at the same speed) that ultimately led to the experimental discovery of the Equivalence Principle!

The second half of this thesis, in the *post-foundation model era*, explored how to use large pretrained models to improve sequential decision-making. This portion of the dissertation asks: instead of having agents explore and generate knowledge, can we inject agents with human knowledge so that they learn faster or achieve higher asymptotic performance or generalize to hundreds of tasks? My first work in this direction, **SHERLock, published at ICPR 2022** [137] explored how to learn unstructured event representations from real-world demonstration videos. I followed it up with Video2Skill [159], where I adapted SHERLock's representations into a robotic embodiment. I found that our agent was able to perform complex skills like stirring and pouring motions simply by watching cooking video demonstrations.

During this phase of my Ph.D., I was also fortunate to contribute to some seminal work in building new foundation models for robot learning: RT-1 [20] and Q-Transformer [29], works that won a **best demo paper at RSS 2023**. This direction also yielded my **NeurIPS 2023 paper**, RoboCLIP, One Demonstration is Enough to Learn Robot Policies. This work provided a solution to a multitude of open problems in robot learning — **(1)** learning from *very* few demonstrations, **(2)** learning from human demonstrations and **(3)** task specification using language.

The remainder of this dissertation details these learnings one chapter at a time. In the final chapter, I enlist my thoughts for the final project I hope to conduct in this PhD, and pose possibly the most challenging question I have asked thus far — how do we build autonomous physical agents that have a common sense understanding of the world?

I hope you enjoy reading this thesis as much as I have enjoyed working on it.

Chapter 2

RoboCLIP: One Demonstration is Enough to Learn Robot Policies

2.1 Introduction

Sequential decision-making problems typically require significant human supervision and data. In the context of online reinforcement learning [163], this manifests in the design of good reward functions that map transitions to scalar rewards [4, 66]. Extant approaches to manual reward function definition are not very principled and defining rewards for complex long-horizon problems is often an art requiring significant human expertise. Additionally, evaluating reward functions often requires knowledge of the true state of the environment. For example, imagine a simple scenario where the agent must learn to lift an object off the ground. Here, a reward useful for task success would be proportional to the height of the object from the ground — a quantity non-trivial to obtain without full state information. Thus, significant effort has been expounded in developing methods that can learn reward functions either explicitly or implicitly from demonstrations, i.e., imitation learning [130, 114, 1, 189]. With these methods, agent policies can either be directly extracted from the demonstrations or trained to optimize rewards functions learned from them.

Imitation learning (IL), however, only somewhat alleviates the need for expert human intervention. First, instead of designing complex reward functions, expert supervision is needed to collect massive datasets such as RT-1 [20], Bridge Dataset [47], D4RL [55], or Robonet [41]. The performance of imitation learning algorithms and their ability to generalize hinges on the coverage and size of data [90, 91], making

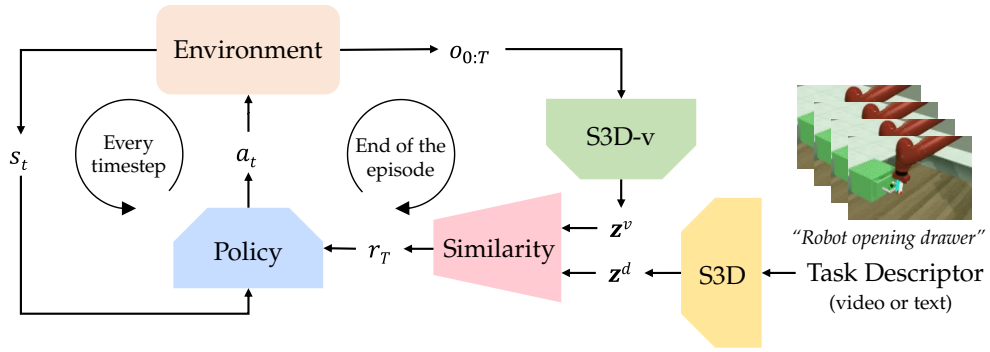


Figure 2.1: RoboCLIP Overview. A Pretrained Video-and-Language Model is used to generate rewards via the similarity score between the encoding of an episode of interaction of an agent in its environment, $\mathbf{z}^v$ with the encoding of a task specifier $\mathbf{z}^d$ such as a textual description of the task or a video demonstrating a successful trajectory. The similarity score between the latent vectors is provided as reward to the agent.

the collection of large datasets imperative. Second and most importantly, the interface for collecting demonstrations for IL is tedious, requiring expert robot operators to collect thousands of demonstrations. On the contrary, a more intuitive way to define rewards would be in the form of a textual description (e.g., "*robot grasping object*"), or in the form of a naturalistic video demonstration of the task performed by a human actor in an environment separate from the robotic environment. For example, demonstrating to a robot how to open a cabinet door in one's own kitchen is more naturalistic than collecting many thousands of trajectories via teleoperation in the target robotic environment.

Thus, there exists an unmet need for IL algorithms that 1) require very few demonstrations and 2) allow for a natural interface for providing these demonstrations. For instance, algorithms that can effectively learn from language instructions or human demonstrations without the need for full environment state information. Our key insight is that by leveraging Video-and-Language Models (VLMs)—which are already pretrained on large amount of video demonstration and language pairs—we do not need to rely on large-scale and in-domain datasets. Instead, by harnessing the power of VLM embeddings, we treat the mismatch between a single *instruction's* embedding (provided as a language command or a video demonstration) and the embedding of the video of the current policy's rollout as a proxy reward that will guide the policy towards the desired *instruction*.

To this end, we present RoboCLIP, an imitation learning algorithm that learns and optimizes a reward function based on a single language or video demonstration. The backbone model used in RoboCLIP is S3D [177] trained on the Howto100M dataset [105], which consists of short clips of humans performing activities with textual descriptions of the activities. These videos typically consist of a variety of camera angles, actors, lighting conditions, and backgrounds. We hypothesize that VLMs trained on such diverse videos are invariant to these extraneous factors and generate an actor-agnostic semantically-meaningful representation for a video, allowing them to generalize to unseen robotic environments.

We present an overview of RoboCLIP in Figure 2.1. RoboCLIP computes a similarity score between videos of online agent experience with a task descriptor, i.e., a text description of the task or a single human demonstration video, to generate trajectory-level rewards to train the agent. We evaluate RoboCLIP on the Metaworld Environment suite [181] and on the Franka Kitchen Environment [64], and find that policies obtained by pretraining on the RoboCLIP reward result in $2-3\times$ higher zero-shot task success in comparison to state-of-the-art imitation learning baselines. Additionally, these rewards require no experts for specification and can be generated using naturalistic definitions like natural language task descriptions and human demonstrations.

2.2 Related Work

Learning from Human Feedback. Learning from demonstrations is a long-studied problem that attempts to learn a policy from a dataset of expert demonstrations. Imitation learning (IL) methods, such as those based on behavioral cloning [130], formulate the problem as a supervised learning over state-action pairs and typically rely on large datasets of expert-collected trajectories directly demonstrating how to perform the target task [20, 102]. However, these large demonstration datasets are often expensive to collect. Another IL strategy is *inverse* RL, i.e., directly learning a reward function from the demonstrations [114,

1, 189, 53]. Inverse RL algorithms are typically difficult to apply when state and action spaces are high-dimensional. Methods such as GAIL [74], AIRL [56], or VICE [57] partially address these issues by assigning rewards which are proportional to the probability of a given state being from the demonstration set or a valid goal state as estimated by a learned discriminator network. However these discriminator networks still require many demonstrations or goal states to train to effectively distinguish between states from agent-collected experience and demonstration or goal states. On the other hand, RoboCLIP's use of pretrained video-and-language models allows us to train agents that learn to perform target tasks with just *one demonstration* in the form of a video or a language description. Other works instead use human feedback in the form of pairwise comparisons or rankings to learn preference reward functions [34, 140, 14, 109, 15, 22, 13, 95, 69]. These preferences may require less human effort to obtain than reward functions, e.g., through querying humans to simply rank recent trajectories. Yet individual trajectory preferences convey little information on their own (less than dense reward functions) and therefore humans need to respond to many preference queries for the agent to learn useful reward functions. In contrast, RoboCLIP is able to extract useful rewards from a single demonstration or single language instruction.

Large Vision and Language Models as Reward Functions. [92] and [76] propose using large language models (LLMs) for designing and regularizing reward functions that capture human preferences. These works study the reward design problem in text-based games such as negotiations or card games, and thus are not grounded in the physical world. RoboCLIP instead leverages video-and-language models to assess if video demonstrations of robot policies align with an expert demonstration. Prior work has demonstrated that video models can be used as reward functions. For example, [30] learn a visual reward function using human data and then utilize this reward function for visual model-based control of a robot. However, they require training the reward model on paired human and robot data from the deployment environment. We demonstrate that this paired data assumption can be relaxed by utilizing large-scale vision-language models pretrained on large corpora of human-generated data. The most well-known of these is CLIP [132], which

is trained on pairs of images and language descriptions scraped from the internet. While CLIP is trained only on images, video-language-models (VLMs) trained on videos of humans performing daily tasks such as S3D [177] or XCLIP [116] are also widely available. These models utilize language descriptions while training to supervise their visual understanding so that *semantically* similar vision inputs are embedded close together in a shared vector space. A series of recent works demonstrate that these VLMs can produce useful rewards for agent learning. [51] finetune CLIP on YouTube videos of people playing Minecraft and demonstrate that the finetuned CLIP model can be used as a language-conditioned reward function to train an agent. DECKARD [119] then uses the fine-tuned reward function of [51] to reward an agent for completing tasks proposed by a large-language model and abstract world model. PAFF [58] uses a fine-tuned CLIP model to align videos of policy rollouts with a fixed set of language skills and relabel experience with the best-aligned language label. We demonstrate that *videos* and multi-modal task specifications can be utilized to learn reward functions allowing for training agents. Additionally, we present a method to test the alignment of pretrained VLMs with deployment environments.

2.3 Method

Overview. RoboCLIP utilizes pretrained video-and-language models to generate rewards for online RL agents. This is done by providing a sparse reward to the agent at the end of the trajectory which describes the similarity of the agent's behavior to that of the demonstration. We utilize video-and-language models as they provide the flexibility of defining the task in terms of natural language descriptions or video demonstrations sourced either from the target robotic domain or other more naturalistic domains like human actors demonstrating the target task in their own environment. Thus, a demonstration (textual or video) and the video of an episode of robotic interaction are embedded into the semantically meaningful latent space of S3D [177], a video-and-language model pretrained on diverse videos of human actors performing everyday tasks taken from the HowTo100M dataset [105]. The two vectors are subsequently

multiplied using a scalar product generating a similarity score between the 2 vectors. This similarity score (without scaling) is returned to the agent as a reward for the episode.

Notation. We formulate the problem in the manner of a POMDP (Partially Observable Markov Decision Process) with ($\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, ϕ, θ, r, T, γ) representing an observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, transition function ϕ, emission function θ, reward function r, time horizon T, and discount factor γ. An agent in state $\mathbf{s}_t$ takes an action $\mathbf{a}_t$ and consequently causes a transition in the environment through $\phi(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. The agent receives the next state $\mathbf{s}_{t+1}$ and reward $r_t = r(\mathbf{o}_t, \mathbf{a}_t)$ calculated using the observation $\mathbf{o}_t$. The goal of the agent is to learn a policy π which maximizes the expected discounted sum of rewards, i.e., $\sum_{t=0}^{T} \gamma^t r_t$. Note that all of our baselines utilize the true state for reward generation and for policy learning. To examine the effect of using a video-based reward, we also operate our policy on the state space while using the pixel observations for reward generation. Thus, r_t uses $\mathbf{o}_t$ while π uses $\mathbf{s}_t$ for RoboCLIP while for all other baselines, both r_t and π utilize $\mathbf{s}_t$. This of course is unfair to our method, but we find that in spite of the advantage provided to the baselines, RoboCLIP rewards still generate higher zero-shot success.

Reward Generation. During the pretraining phase, we supply the RoboCLIP reward to the agent in a sparse manner at the end of each episode. This is done by storing the video of an episode of the interaction of the agent with the environment into a buffer as seen in Figure 2.1. A sequence of observations of length 128 are saved in a buffer corresponding to the length of the episode. S3D is trained on videos length 32 frames and therefore the episode video is subsequently downsampled to result in a video of length $T = 32$. The video is subsequently center-cropped to result in frames of size $(250, 250)$. This is done to ensure that the episode video is preprocessed to match the specifications of the HowTo100M preprocessing used to

train the S3D model. Thus the tensor of a sequence of T observations $\mathbf{o}_{0:T}$ is encoded as the latent video vector $\mathbf{z}^v$ using

$$\mathbf{z}^v = S3D^{\text{video-encoder}}(\mathbf{o}_{0:T}) \tag{2.1}$$

The task specification is also encoded into the same space. If it is defined using natural language, the language encoder in S3D encodes a sequence of K textual tokens $\mathbf{d}_{0:K}$ into the latent space using:

$$\mathbf{z}^d = S3D^{\text{text-encoder}}(\mathbf{d}_{0:K}) \tag{2.2}$$

If the task description is in the form of a video of length K, then we preprocess and encode it using the video-encoder in S3D just as in Equation (2.1). For intermediate timesteps, i.e., timesteps other than the final one in an episode, the reward supplied to the agent is zero. Subsequently, at the end of the episode, the similarity score between the encoded task descriptor $\mathbf{z}^d$ and the encoded video of the episode $\mathbf{z}^v$ is used as reward $r^{\text{RoboCLIP}}(T)$. Thus the reward is:

$$r^{\text{RoboCLIP}}(t) = \begin{cases} 0, & t \neq T \\ \mathbf{z}^d \cdot \mathbf{z}^v & t = T \end{cases}$$

where $\mathbf{z}^d \cdot \mathbf{z}^v$ corresponds to the scalar product between vectors $\mathbf{z}^d$ and $\mathbf{z}^v$.

Agent Training. Using r^{RoboCLIP} defined above, we then train an agent online in the deployment environment with any standard reinforcement learning (RL) algorithm by labeling each agent experience trajectory with r^{RoboCLIP} after the agent collects it. In our paper, we train with PPO [151], an on-policy RL algorithm, however, RoboCLIP can also be applied to off-policy algorithms. After training with this reward, the agent can be zero-shot evaluated or fine-tuned on true environment reward on the target task in the deployment environment.

2.4 Experiments

We test out each of the hypotheses defined in Section 2.1 on simulated robotic environments. Specifically, we ask the following questions:

1. *Do existing pretrained VLMs semantically align with robotic manipulation environments?*
2. *Can we utilize natural language to generate reward functions?*
3. *Can we use videos of expert demonstrations to generate reward functions?*
4. *Can we use out-of-domain videos to generate reward functions?*
5. *Can we generate rewards using a combination of demonstration and natural language?*
6. *What aspects of our method are crucial for success?*

We arrange this section to answer each of these questions. Both RoboCLIP and baselines utilize PPO [151] for policy learning.

Baselines. We use 2 state-of-the-art methods in inverse reinforcement learning: **GAIL**, or Generative Adversarial Imitation Learning [74] and **AIRL** or Adversarial Inverse Reinforcement Learning [56]. Both of these methods attempt to learn reward functions from demonstrations provided to the agent. Subsequently, they train an agent using this learned reward function to imitate the expert behavior. Both methods receive a single demonstration, consistent with our approach of using a single video imitation. However, since they both operate on the ground-truth environment state, we provide them with a **trajectory of states**, instead of images, thereby providing them privileged state information that our method does not receive.

2.4.1 Domain Alignment

Pretrained vision models are often trained on a variety of human-centric activity data, such as Ego4D [62]. Since we are interested in solving robotic tasks with view from third person perspectives, we utilize the

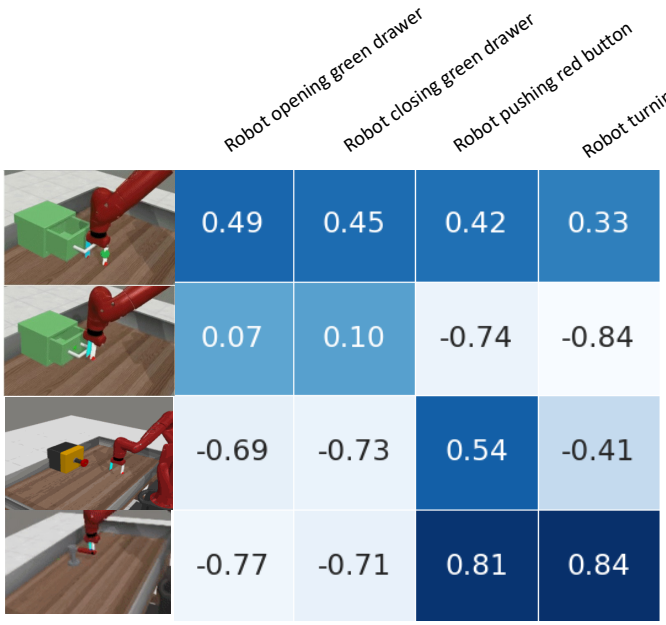


Figure 2.2: Domain Alignment Confusion Matrix. We perform a confusion matrix analysis on a subset of the data on collected on Metaworld [181] environments by comparing the pair-wise similarities between the latent vectors of the strings describing the videos and those of the videos. We find that Metaworld is well-aligned with higher scores along the diagonal than along the off-diagonal elements.

S3D [177] VLM pretrained on HowTo100M [105], a dataset of short third-person clips of humans performing everyday activities. This dataset, however, contains no robotic manipulation data.

To analyze the alignment of the VLM to different domains, we perform a confusion matrix analysis using videos from Metaworld [181]. We collect 10 videos per task with varying values of true reward. For each video, we also collect the true reward. We then compute the RoboCLIP reward for each video using VLM alignment between the textual description of the task and the video. We visualize the correlations between the RoboCLIP and true rewards in the form of an $n \times n$ matrix where entry (i, j) corresponds to the correlation between the true reward and the RoboCLIP reward generated for the i^{th} task using the j^{th} text description. As one can see, for a given task, the highest correlation in the matrix is for the correct textual description. We visualize one such similarity matrix in Figure 2.2 for Metaworld. We find that Metaworld seems to align well in the latent space of the model with a more diagonal-heavy confusion matrix. The objects are all correctly identified.

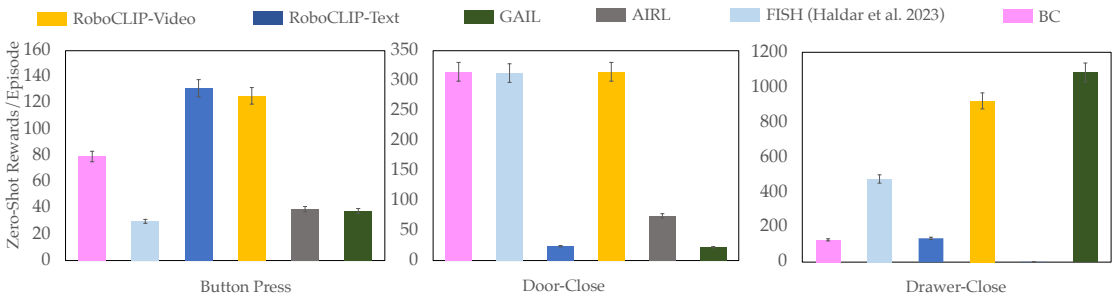


Figure 2.3: Language-Conditioned Reward Generation. The pretrained VLM is used to generate rewards via the similarity score of the encoding of an episode of interaction of an agent in its environment, $\mathbf{z}^v$ with the encoding of a task specifier $\mathbf{z}^d$ specified in natural language. We use the strings, "*robot closing black box*", "*robot closing green drawer*" and "*robot pushing red button*" for conditioning for the 3 environments respectively. We find that agents pretrained on these language-conditioned rewards outperform imitation learning baselines like GAIL [74] and AIRL [56].

2.4.2 Language for Reward Generation

The most naturalistic way to define a task is through natural language. We do this by generating a sparse reward signal for the agent as described in Section 2.3: the reward for an episode is the similarity score between its encoded video and the encoded textual description of the expected behavior in the VLM's latent space. The reward is provided to the agent at the end of the episode. For RoboCLIP, GAIL, and AIRL, we first pretrain the agents online with their respective reward functions and then perform finetuning with the true task reward in the deployment environment.

We perform this analysis on 3 Metaworld Environments: `Drawer-Close`, `Door-Close` and `Button-Press`. We use the textual descriptions, "*robot closing green drawer*", "*robot closing black box*", and "*robot pushing red button*" for each environment, respectively. Figure 2.3 plots returns on the target tasks while finetuning on the depoloyment environment after pretraining (with the exception of the Dense Task Reward baseline). Our method outperforms the imitation learning baselines with online exploration in terms of true task rewards in all environments. Additionally our baselines utilize the full state information in the environment for reward generation where RoboCLIP uses only the pixels to infer state. RoboCLIP also achieves more than double zero-shot rewards in all environments — importantly, the RoboCLIP-trained agent is able to complete the tasks even before finetuning on true task rewards.

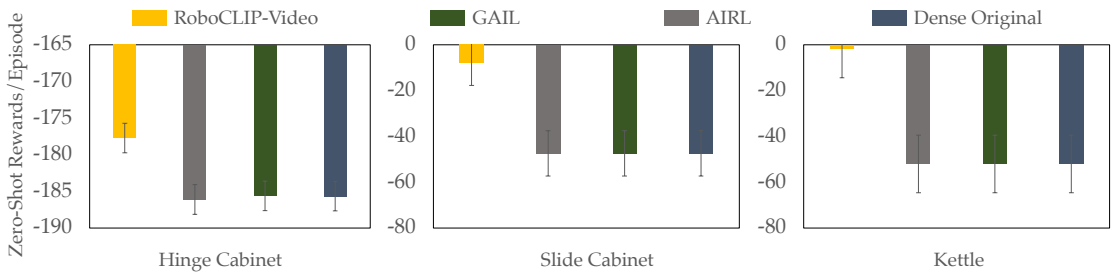


Figure 2.4: Using In-Domain Videos for Reward Generation. The pretrained VLM is used to generate rewards via the similarity score of the encoding of an episode of interaction of an agent in its environment, $\mathbf{z}^v$ with the encoding of a video demonstration of expert behavior in the same environment. The similarity score between the latent vectors is provided as reward to the agent and is used to train online RL methods. We study this setup in the `Kettle`, `Hinge` and `Slide` Tasks in the Franka Kitchen Environment [64]. We find that policies trained on the RoboCLIP reward are able to learn to complete the task in all three setups without any need for external rewards using just a single in-domain demonstration.

2.4.3 In-Domain Videos for Reward Generation

Being able to use textual task descriptors for reward generation can only work in environments where there is domain alignment between the pretrained model and the visual appearance of the environment. Additionally, VLMs are large models often with billions of parameters making it computationally expensive to fine tune for domain alignment. The most naturalistic way to define a task in such a setting is in the form a single demonstration in the robotic environment which can be collected using teleoperation. We study how well this works in the Franka Kitchen [64] environment. We consider access to a single demonstration per task whose video is used to generate rewards for online RL.

Quantitative Results. We measure the zero-shot task reward, which increases as the task object (i.e., Kettle, Slide and Hinge Cabinets) gets closer to its goal position. This reward does not depend on the position of the end-effector, making the tasks difficult. Figure 2.4 shows the baselines perform poorly as they generally do not interact with the target objects, while RoboCLIP is able to solve the task using the reward generated using the video of a single demonstration.

Qualitative Results. We find that RoboCLIP allows for mimicking the "style" of the source demonstration, with idiosyncrasies of motion from the source demonstration generally transferring to the policy generated.

Figure 2.5: Qualitative Inspection of Imitation. The first row in each subfigure shows the visualizations of the demonstration video used for reward generation via the VLM. The second rows are videos taken from policy recovered from training on the RoboCLIP reward generated using the videos in the first rows. The quick swiping motion demonstrated in the `Slide` demonstration is mimicked well in the resultant policy while the wrist-rotational "trick-shot" behavior in the demonstration for `Hinge` appears in the resultant learned policy.

We find this to occur in the kitchen environment's `Slide` and `Hinge` task as seen in Figure 2.5. The first row of the subfigures in Figure 2.5 are visualizations of the demonstration video used to condition the VLM for reward generation. The bottom rows correspond to the policies that are trained with the generated rewards of RoboCLIP. As can be seen, the `Slide` demonstration consists of a wide circular arc of motion. This is mimicked in the learned policy, although the agent misses the cabinet in the first swipe and readjusts to make contact with the handle.

This effect is even more pronounced in the `Hinge` example where the source demonstration consists of twirling wrist-rotational behavior, which is subsequently imitated by the learned policy. The downstream policy misses the point of contact with the handle but instead uses the twirling motion to open the hinged cabinet in an unorthodox manner by pushing near the hinge. We posit that the VLMs used in RoboCLIP

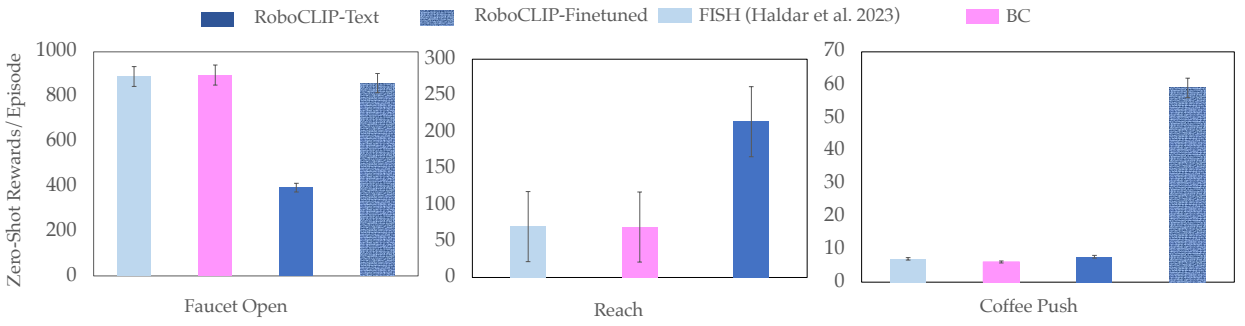


Figure 2.6: Finetuning for Harder Environments: In harder environments, like `Coffee-Push` and `Faucet-Open`, we find that RoboCLIP rewards do not solve the task completely. We test whether providing a single demonstration in the environment (using states and actions) is enough to finetune this pretrained policy, a setup identical to our baselines. Thus, we pre-train on the RoboCLIP reward from language and then finetune using a single robotic demonstration. This improves performance by $\sim 200\%$. See videos on our website.

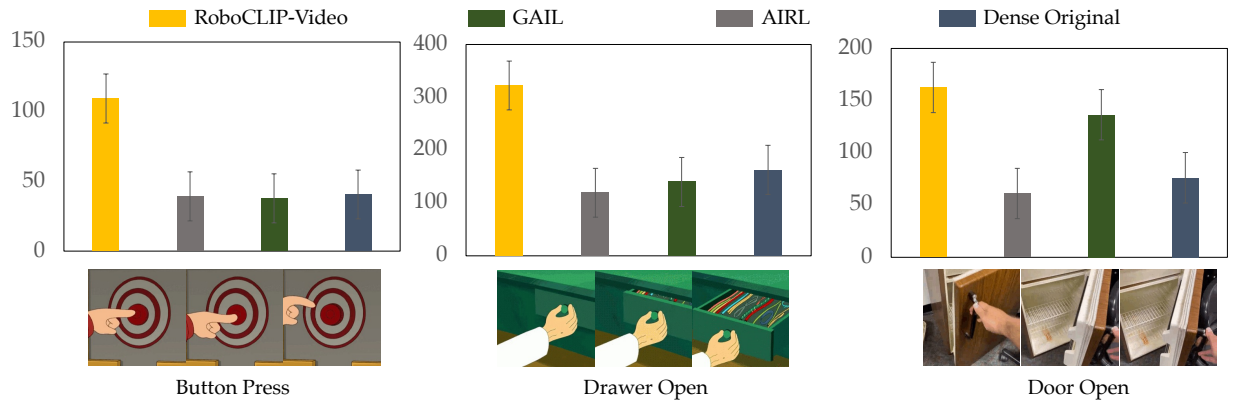


Figure 2.7: Using Out-of-Domain Videos for Reward Generation. A Pretrained Video-and-Language Model is used to generate rewards via the similarity score of the encoding of an episode of interaction of an agent in its environment, $\mathbf{z}^v$ with the encoding of a task specifier $\mathbf{z}^d$ in the form of a video of a human or an animated character demonstrating a task in their own environment. The similarity score between the latent vectors is provided as reward to the agent and is used to train online RL methods. The frames below the graphs illustrate the video used for reward generation.

contain a rich latent space encoding these various motions, and so even if they cannot contain semantically meaningful latent vectors in the Franka Kitchen environments due to domain mismatch, they are still able to encode motion information allowing them to be used for RoboCLIP with a single demonstration video.

2.4.4 Out-of-Domain Videos for Reward Generation

Another natural way to define a task is to demonstrate it yourself. To this end, we try to use demonstrations of humans or animated characters acting in separate environments as task specification.

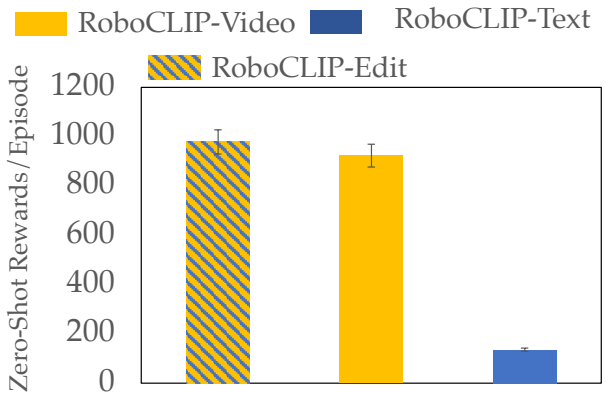


Figure 2.8: Multimodal Task Specification. We study whether video demonstrations of expert demonstrations can be used to define tasks. We use the latent embedding of a video demonstration of a robot pushing a button and subtract from it the embedding of the text "*red button*" and add to it the embedding of the text "*green drawer*". This modified latent is used to generate rewards in the Drawer-Close environment. We find that the policy trained using this modified vector outperforms string-only manipulation in the zero-shot setting.

For this, we utilize animated videos of a hand pushing a red button and opening a green drawer and a real human video of opening a fridge door (see Figure 2.7). The animated videos are collected from stock image repositories and the human video is collected using a phone camera in our lab kitchen. Using the encodings of these video, we test out RoboCLIP in the 3 corresponding Metaworld tasks - `Button-Press`, `Drawer-Open` and `Door-Open`. We follow the same setup as in Section 2.4.2 by first pretraining methods with their respective reward functions and then finetuning in the deployment environment with target task reward.

We compare the performance of the policy trained with these rewards to GAIL [74] and AIRL [56] trained using the same single expert demonstration as RoboCLIP on these rewards with state information. These methods are known to be data-hungry, requiring multiple demonstrations to train their reward functions. Consequently, they perform much worse than RoboCLIP, even with 2-3x worse zero-shot task performance, as can be seen from Figure 2.7.

2.4.5 Multimodal Task Specification

Using videos to specify a task description is possible when either there is access to a robot for teleoperation as in Section 2.4.3 or a human can demonstrate a behavior in their own environment as in Section 2.4.4. When these are not the case, a viable alternative is to utilize multimodal demonstrations. For example, consider a scenario where the required task is to push a drawer to close it, but only a demonstration for pushing a button is available. In this situation, being able to edit the video of the off-task demonstration is useful. This way, one can direct the agent to move its end-effectors to push the drawer instead of the button.

We do this by algebraically modifying the encoding of the video demonstration:

$$\mathbf{z}^{\text{edited}}(\text{push drawer}) = \mathbf{z}^{video}(\text{push button}) - \mathbf{z}^{text}(\text{button}) + \mathbf{z}^{text}(\text{drawer}) \tag{2.3}$$

where $\mathbf{z}^{\text{edited}}(\text{push drawer})$ is the vector used to generate rewards in the `Drawer-Close` environment, $\mathbf{z}^{video}(\text{push button})$ is the vector of the encoding of the video of the robot pushing a button, $\mathbf{z}^{text}(\text{button})$ is the encoding of the string *button* and $\mathbf{z}^{text}(\text{drawer})$ is the encoding of the string *drawer*. As can be seen in Figure 2.8, defining rewards in such a multimodal manner results in a higher zero-shot score than the dense task reward and also pretraining on the string-only task reward.

2.4.6 Finetuning

In harder environments, and with rewards from OOD videos and language, the robot policy sometimes approaches the target object, but fails to complete the task. Thus, we tested whether providing a single demonstration (using states and actions) was enough to finetune this pretrained policy.

Thus, for this experiment we first (1) pretrain on the RoboCLIP reward from human videos or language descriptions and then (2) finetune using a single demonstration. As seen in Figure 2.6, we find that this

converts each of the partially successful policies into complete success and improves the rewards attained by the policies by 200%. This fine-tuning setup is especially useful in harder tasks like like `Coffee-Push` and `Faucet-Open` and is competitive with state-of-the-art approaches like FISH [67].

2.4.7 Ablations

Finally, we investigate the effects of various design decisions in RoboCLIP. First, we study the effect of additional video demonstrations on agent performance. We also examine the necessity of using a pre-trained VLM. Recent works like RE3 [152, 170] have shown that randomly initialized networks often contain useful image priors and can be used to supply rewards to agents to encourage exploration. Therefore, we test whether a *randomly initialized* S3D VLM can supply useful pretraining rewards in the in-domain video demonstration setup as in Section 2.4.3. Finally, we study our choice of pre-trained VLM. We examine whether a pretrained CLIP [132], which encodes single images instead of videos and was trained on a different dataset from S3D, can be used to generate rewards for task completion. In this setup, we record the last image in an episode of interaction of the agent in its environment and feed it to CLIP trained on ImageNet [138] (i.e., not trained on videos). We then specify the task in natural language and use the similarity between the embeddings of the textual description of the task and the final image in the episode to generate a reward that is fed to the agent for online RL.

As seen in Figure 2.9, using a single video demonstration provides the best signal for pretraining. We posit that our method performs worse when conditioned on multiple demonstrations as the linear blending of multiple video embeddings, which is used due to the scalar product, does not necessarily correspond to the embedding of a successful trajectory.

Crucially, we also find that using the static image version of CLIP does not provide any useful signal for pretraining. The zero-shot performance is very poor, which we posit is because it does not contain any information about the dynamics of motion and task completion although it contains semantic meaning

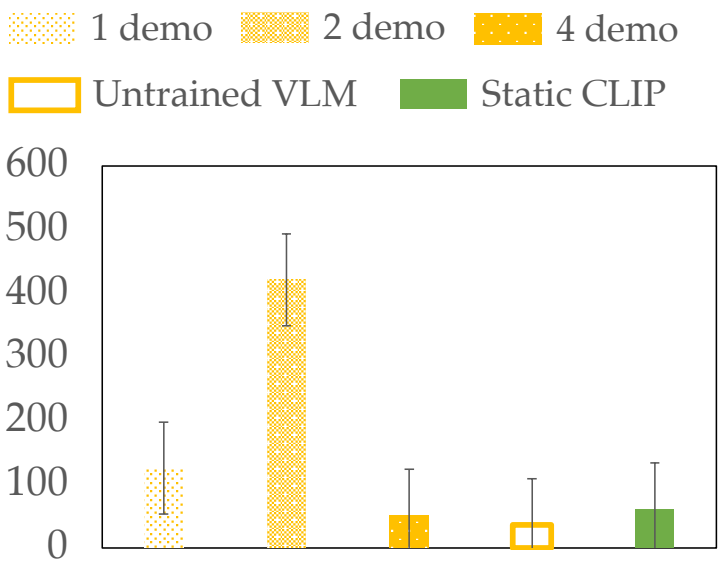


Figure 2.9: Ablations. We study the effects of varying the number of demonstrations provided to the agent can have on downstream rewards. We also study the effects of the training provided to the VLM on the downstream rewards. Finally, we study whether using CLIP trained on static images provides good rewards for pretraining.

about objects in the frame. On the other hand, video contrastive learning approaches do contain this information. This is further evidenced by the fact that inspite of poor domain alignment between Franka Kitchen and the VLM, we find that encodings of in-domain video demonstrations are still good for providing a pretraining reward signal to the agent.

2.5 Conclusion

Summary. We studied how to distill knowledge contained in large pretrained Video-and-Language-Models into online RL agents by using them to generate rewards. We showed that our method, RoboCLIP, can train robot policies using a single video demonstration or textual description of the task, depending on how well the domain aligns with the VLM. We further investigated alternative ways to use RoboCLIP, such as using out-of-domain videos or multimodal demonstrations. Our results showed RoboCLIP outperforms the baselines in various robotic environments.

Limitations and Broader Impact. Since we are using VLMs, the implicit biases within these large models could percolate into RL agents. Addressing such challenges is necessary, especially since it is unclear what the form of biases in RL agents might look like. Currently, our method also faces the challenge of stable finetuning. We find that in some situations, finetuning on downstream task reward results in instabilities as seen in the language conditioned reward curve in Figure 2.8. This instability is potentially due to the scale of rewards provided to the agent. Rewards from the VLM are fairly low in absolute value and subsequently, the normalized Q-values in PPO policies are out-of-shape when finetuned on task rewards. In our experiments, this is not a big problem since the RoboCLIP reward is already sufficient to produce policies that complete tasks without any deployment environment finetuning, but this will be essential to solve when deploying this for longer horizon tasks.

Another limitation of our work is that there is no fixed length of pretraining. Our current method involves pretraining for a fixed number of steps and then picking the best model according to the true task reward. This is of course difficult when deploying RoboCLIP in a real-world setup as a true reward function is unavailable and a human must monitor the progress of the agent. We leave this for future work.

In this work, we saw how to reward RL agents using the commonsense available within VLMs. But can robots have their own intrinsic motivation to explore their world? Causal curiosity (the upcoming chapter) studies this.

Chapter 3

Causal Curiosity: RL Agents Discovering Self-supervised Experiments for Causal Representation Learning

3.1 Introduction

Discovering causation in environments an agent might encounter remains an open and challenging problem for reinforcement learning [11, 149]. In physical systems, causal factors such as gravity or friction affect the outcome of behaviors an agent might perform. Thus, there has been recent interest in attempting to train agents to be robust or invariant against varying values of such causal factors, allowing them to learn modular behaviors that are useful across tasks. Most model-based approaches take the form of Bayes Adaptive Markov Decision Processes (BAMDPs) [190] or Hidden Parameter MDPs (Hi-Param MDPs) [46, 179, 87, 127] which condition the transition and/or reward function of each environment on hidden parameters.

Formally, let $\mathbf{s} \in \mathcal{S}$, $\mathbf{a} \in \mathcal{A}$, $\mathbf{r} \in \mathcal{R}$, $\mathbf{h} \in \mathcal{H}$ where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, and $\mathcal{H}$ are the set of states, actions, rewards and admissible causal factors, respectively. In the physical world, examples of the parameter $\mathbf{h}_j \in \mathcal{H}$ might include gravity, coefficients of friction, masses and sizes of objects. Hi-Param MDP or BAMDP approaches treat each $\mathbf{h}_j \in \mathcal{H}$ as a latent variable for which an embedding is learnt during training (often using variational methods [89, 79]). Let $\mathbf{a}_{0:T}$ be a sequence of actions taken by an agent to maximize an

external reward resulting in a state trajectory $\mathbf{s}_{0:T}$. The above approaches define a probability distribution over the entire observable sequence (i.e., rewards, states, actions) as $p(\mathbf{r}_{0:T}, \mathbf{s}_{0:T}, \mathbf{a}_{0:T-1})$ which factorizes as

$$\prod_{t=1}^{T-1} p(\mathbf{r}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{z}) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}) p(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z})$$

conditioned on the latent variable $\mathbf{z}$, a representation for the unobserved causal factors. At test time, in a new environment, the agent infers $\mathbf{z}$ by observing the trajectories produced by its initial actions issued by the latent conditioned policy obtained during training.

In practice, discovering causal factors in a physical environment is prone to various challenges that are due to the disjointed nature of the influence of these factors on the produced trajectories. More specifically, at each time step, the transition function is affected by a subset of global causal factors. This subset is implicitly defined on the basis of the current state and the action taken. For example, if a body in an environment loses contact with the ground, the coefficient of friction between the body and the ground no longer affects the outcome of any action that is taken. Likewise, the outcome of an upward force applied by the agent to a body on the ground is unaffected by the friction coefficient.

Without knowledge of how independent causal mechanisms affect the outcome of a particular action in a given state in an environment, it becomes impossible for the agent to conclude where an encountered variation came from. Unsurprisingly, Hi-Param and BAMDP approaches fail to learn a disentangled embedding of the causal factors, making their behaviors uninterpretable. For example, if, in an environment, a body remains stationary under a particular force, the Hi-Param or BAMDP agent may apply a higher force to achieve its goal of perhaps moving the body, but will be unable to conclude whether the "un-movability" was caused by a high friction coefficient, or high mass. Additionally, these approaches require substantial reward engineering, making it difficult to apply them outside the simulated environments they are tested in.

Our goal is, instead of focusing on maximizing reward for a particular task, to allow agents to discover causal processes through exploratory interaction. During training, our agents discover self-supervised experimental behaviors which they apply to a set of training environments. These behaviors allow them to learn about the various causal mechanisms that govern the transitions in each environment. During inference in a novel environment, they perform these discovered behaviors sequentially and use the outcome of each behavior to infer the embedding for a single causal factor (Figure 3.1), allowing us to recover a disentangled embedding describing the causal factors of an environment.

The main challenge while learning a disentangled representation for the causal factors of the world is that several causal factors may affect the outcome of behaviors in each environment. For example, when pushing a body on the ground, the outcome, i.e., whether the body moves, or how far the body is pushed, depends on several factors, e.g., mass, shape and size, frictional coefficients, etc. However, if, instead of pushing on the ground, the agent executes a perfect grasp-and-lift behavior, only mass will affect whether the body is lifted off the ground or not.

Thus, it is clear that not all experimental behaviors are created equal and that the outcomes of some behaviors are caused by fewer causal factors than others. Our agents learn these behaviors without supervision using *causal curiosity*, an intrinsic reward. The outcome of a single such experimental behavior is then used to infer a binary quantized embedding describing the single isolated causal factor. While causal factors of variation in a physical world are easily identifiable to humans, a concrete definition is required to formalize our proposed method.

Definition 1 (Causal factors)**.** *Consider the POMDP $(\mathcal{O}, \mathcal{S}, \mathcal{A}, \phi, \theta, r)$ with observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, the transition function ϕ, emission function θ, and the reward function r. Let $\mathbf{o}_{0:T} \in \mathcal{O}^T$ denote a trajectory of observations of length T. Let $d(\cdot, \cdot) : \mathcal{O}^T \times \mathcal{O}^T \to \mathbb{R}_+$ be a distance function defined on the space of trajectories of length T. The set $H = \{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{K-1}\}$ is called a set of $\epsilon-$causal factors if for*

every $\mathbf{h}_j \in H$, there exists a unique sequence of actions $\mathbf{a}_{0:T}$ that clusters the observation trajectories into m disjoint sets $C_{1:m}$ such that $\forall C_a, C_b$, a minimum separation distance of ϵ is ensured:

$$\min\{d(\mathbf{o}_{0:T}, \mathbf{o}'_{0:T}) : \mathbf{o}_{0:T} \in C_a, \mathbf{o}'_{0:T} \in C_b\} > \epsilon \tag{3.1}$$

and that $\mathbf{h}_j$ is the cause of the obtained trajectory of states i.e. $\forall v \neq v'$,

$$p(\mathbf{o}_{0:T}|do(\mathbf{h}_j = v), \mathbf{a}_{0:T}) \neq p(\mathbf{o}_{0:T}|do(\mathbf{h}_j = v'), \mathbf{a}_{0:T}) \tag{3.2}$$

where $do(\mathbf{h}_j)$ corresponds to an intervention on the value of the causal factor $\mathbf{h}_j$.

According to Def. 2, a causal factor $\mathbf{h}_j$ is a variable in the environment the value of which, when intervened on (i.e., varied) using $do(\mathbf{h}_j)$ over a set of values, results in trajectories of observations that are divisible into disjoint clusters $C_{1:m}$ under a particular sequence of actions $\mathbf{a}_{0:T}$. These clusters represent the quantized values of the causal factor. For example, mass, which is a causal factor of a body, under an action sequence of a grasping and lifting motion, may result in 2 clusters, liftable (low mass) and not-liftable (high mass).

However, such an action sequence is not known in advance. Therefore, discovering a causal factor in the environment boils down to finding a sequence of actions that makes the effect of that factor prominent by producing clustered trajectories for different values of that environmental factor. For simplicity, here we assume binary clusters. For an introduction to causality and $do(\cdot)$ notation, see [126, 161, 150, 48].

Our contributions of our work are as follows:

- **Causal POMDPs:** We extend Partially Observable Markov Decision Processes (POMDPs) by explicitly modelling the effect of causal factors on observations.

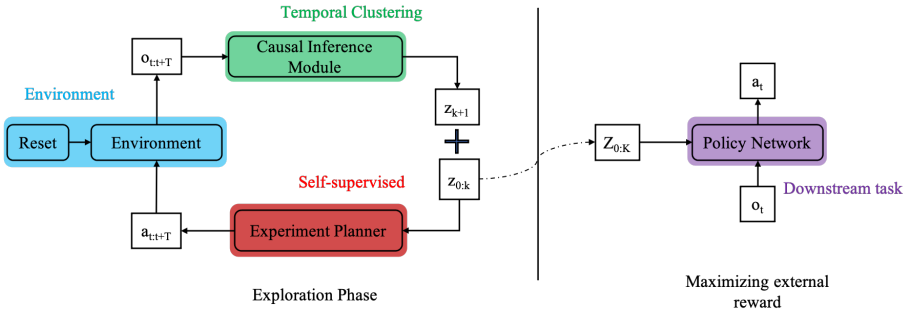


Figure 3.1: Overview of Inference. The exploration loop produces a series of K experiments allowing the agent to infer the representations for K causal factors. After exploration, the agent utilizes the acquired knowledge for downstream tasks.

- **Unsupervised Behavior:** We equip agents with the ability to perform experiments and behave in a semantically meaningful manner in a set of environments in an unsupervised manner. These behaviors can expose or obfuscate specific independent causal mechanisms that occur in the world surrounding the agent, allowing the agent to "experiment" and learn.

- **Disentangled Representation Learning:** We introduce an minimalistic intrinsic reward, *causal curiosity*, which allows our agents to discover these behaviors without human-engineered complex rewards. The outcomes of the experiments are used to learn a disentangled quantized binary representation for the causal factors of the environment, analogous to the human ability to conclude whether objects are light/heavy, big/small etc.

- **Sample Efficiency:** Through extensive experiments, we conclude that knowledge of the causal factors aids sample efficiency in two ways - first, that the knowledge of the causal factors aids transfer learning across multiple environments; second, the learned experimental behaviors can be re-purposed for downstream tasks.

3.2 Method

Consider a set of N environments $\mathcal{E}$ with $\mathbf{e}^i \in \mathcal{E}$ where $\mathbf{e}^i$ denotes the i^{th} environment. Each causal factor $\mathbf{h}_j \in H$ is itself a random variable which assumes a particular value for every instantiation of an environment. Thus, every environment $\mathbf{e}^i$ is represented with the values assumed by its causal factors $\{\mathbf{h}_j^i, j = 0, 1, \ldots, K-1\}$. For each environment $\mathbf{e}^i$, $(\mathbf{z}_0^i, \mathbf{z}_1^i ... \mathbf{z}_{K-1}^i)$ represents the disentangled embedding vector corresponding to the physical causal factors where $\mathbf{z}_j^i$ encodes $\mathbf{h}_j^i$.

3.2.1 POMDP Setup

3.2.1.1 Classical POMDPs

Classical POMDPs ($\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, ϕ, θ, r) consist of an observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, the transition function ϕ, emission function θ, and the reward function r. An agent in an unobserved state $\mathbf{s}_t$ takes an action $\mathbf{a}_t$ and consequently causes a transition in the environment through $\phi(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The agent receives an observation $\mathbf{o}_{t+1} = \theta(\mathbf{s}_{t+1})$ and a reward $\mathbf{r}_{t+1} = r(\mathbf{s}_t, \mathbf{a}_t)$.

3.2.1.2 Causal POMDPs

Our work divides the unobserved state $\mathbf{s} \in \mathcal{S}$ at each timestep into two portions - the *controllable state* $\mathbf{s}^c$ and the *uncontrollable state* $\mathbf{s}^u$. The uncontrollable portion of the state $\mathbf{s}^u$ consists of the causal factors of the environment. We assume that these remain constant during the interaction of the agent with a single instance of the environment. For example, the value of the gravitational acceleration does not change for a single environment. For the following discussion, we refer to the uncontrollable state as causal factors as in Def 2 i.e., $\mathbf{s}^u$ = $\mathcal{H}$.

The controllable state $\mathbf{s}^c$ consists of state variables such as positions and orientations of objects, location of end-effectors of the agent etc. Thus, by executing particular action sequences the agent can manipulate this portion of the state, which is hence controllable by the agent.

3.2.1.3 Transition Probability

A trajectory of the controllable state is dependent on both the action sequence that the agent executes and a subset of the causal factors. At each time step, only a subset of the causal factors of an environment affect the transition in the environment. This subset is implicitly selected by the employed policy for every state of the trajectory (depicted as a Gated Causal Graph (Figure 3.2)). For example, the outcome of an upward force applied by the agent to a body on the ground is unaffected by the friction coefficient between the body and the ground.

Thus the transition function of the controllable state is:

$$\phi(\mathbf{s}^c_{t+1}|\mathbf{s}^c_t, f_{sel}(\mathcal{H}, \mathbf{s}^c_t, \mathbf{a}_t), \mathbf{a}_t) \tag{3.3}$$

where f_{sel} is the implicit Causal Selector Function which selects the subset of causal factors affecting the transition defined as:

$$f_{sel} : \mathcal{H} \times \mathcal{S} \times \mathcal{A} \to \wp(\mathcal{H}) \tag{3.4}$$

where $\wp(\mathcal{H})$ is power-set of $\mathcal{H}$ and $f_{sel}(\mathcal{H}, \mathbf{s}^c_t, \mathbf{a}_t) \subset \mathcal{H}$ is the set of effective causal factors for the transition $\mathbf{s}_t \to \mathbf{s}_{t+1}$ i.e., $\forall v \neq v'$ and $\forall \mathbf{h}_j \in f_{sel}(\mathcal{H}, \mathbf{s}^c_t, \mathbf{a}_t)$:

$$\phi(\mathbf{s}^c_{t+1}|do(\mathbf{h}_j = v), \mathbf{s}^c_t, \mathbf{a}_t) \neq \phi(\mathbf{s}^c_{t+1}|do(\mathbf{h}_j = v'), \mathbf{s}^c_t, \mathbf{a}_t) \tag{3.5}$$

where $do(\mathbf{h}_j)$ corresponds to an external intervention on the factor $\mathbf{h}_j$ in an environment.

Intuitively, this means that if an agent takes an action $\mathbf{a}_t$ in the controllable state $\mathbf{s}^c_t$, the transition to $\mathbf{s}^c_{t+1}$ is caused by a subset of the causal factors $f_{sel}(\mathcal{H}, \mathbf{s}^c_t, \mathbf{a}_t)$. For example, if a body on the ground (i.e., state $\mathbf{s}^c_t$) is thrown upwards (i.e., action $\mathbf{a}_t$), the outcome $\mathbf{s}_{t+1}$ is caused by the causal factor gravity (i.e., $f_{sel}(\mathcal{H}, \mathbf{s}^c_t, \mathbf{a}_t) = \{\text{gravity}\}$), a singleton subset of the global set of causal factors. The $do()$ notation

expresses this causation. If an external intervention on a causal factor is performed, e.g., if somehow the value of gravity was changed from v to v', the outcome of throwing the body up from the ground, $\mathbf{s}_{t+1}$, would be different.

3.2.1.4 Emission Probability

The agent neither has access to the controllable state, nor to the causal factors of each environment. It receives an observation described by the function:

$$\mathbf{o}_{t+1} = \theta(\mathbf{s}_t^c, f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t), \mathbf{a}_t) \tag{3.6}$$

where f_{sel} is the implicit Causal Selector Function.

3.2.2 Training the Experiment Planner

The agent has access to a set of training environments with multiple causal factors varying simultaneously. Our goal is to allow the agent to discover action sequences $\mathbf{a}_{0:T}$ such that the resultant observation trajectory $\mathbf{o}_{0:T}^i$ is caused by a single causal factor i.e., $\forall t < T, f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t) = \text{constant}$ and $|f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t)| = 1$. Consequently, $\mathbf{o}_{0:T}^i$ can be used to learn a representation $\mathbf{z}_j^i$ for the causal factor $f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t)$ for each environment $\mathbf{e}^i$.

We motivate this from the perspective of algorithmic information theory [80]. Consider the Gated Directed Acyclic Graph of the observed variable $\mathbf{O}$ and its causal parents (Figure 3.2). Each causal factor has its own causal mechanism, jointly bringing about $\mathbf{O}$. A central assumption of our approach is that causal factors are independent, i.e., the Independent Mechanisms Assumption [6, 122, 150]. The information in $\mathbf{O}$ is then the sum of information "injected" into it from the multiple causes, since, loosely speaking, for information to cancel, the mechanisms would need to be algorithmically dependent [80]. Intuitively, the information content in $\mathbf{O}$ will be greater for a larger number of causal parents in the graph. Interestingly, a

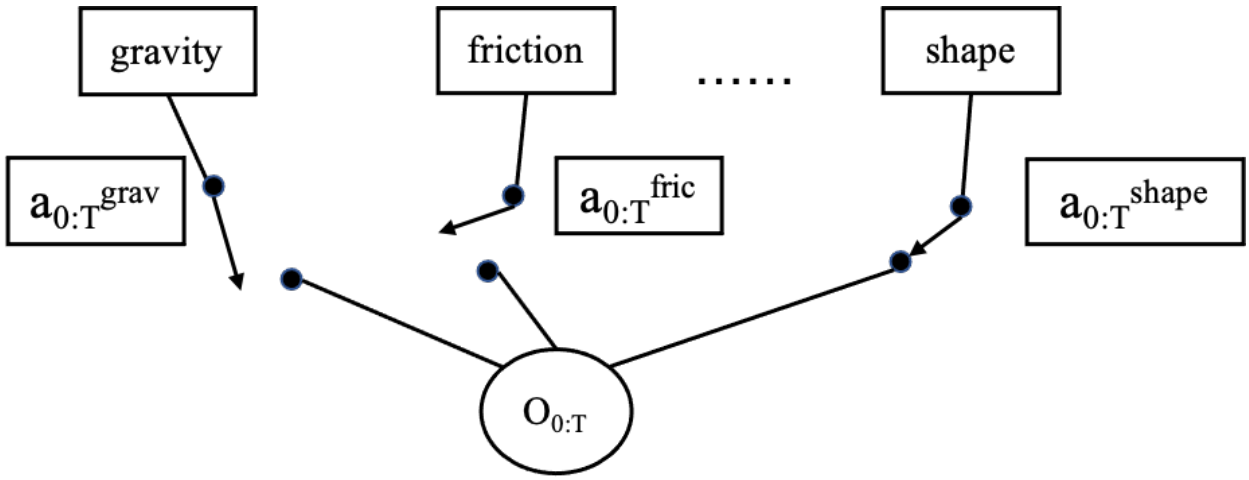


Figure 3.2: Gated Causal Graph. A subset of the unobserved parent causal variables influence the observed variable $\mathbf{O}$. The action sequence $\mathbf{a}_{0:T}$ serves a gating mechanism, allowing or blocking particular edges of the causal graph using the implicit Causal Selector Function (Equation 4.4).

similar argument has been made to justify the thermodynamic arrow of time [28, 81]: while a microscopic time evolution is invertible, the assumption of algorithmic independence for initial conditions and forward mechanisms generates an asymmetry. To invert time, the backward mechanism would need to depend on the initial state.

Thus we attempt to find an action sequence $\mathbf{a}_{0:T}$ for which the number of causal parents of the resultant observation $\mathbf{O}$ is low, i.e., the complexity of $\mathbf{O}$ is low. One could conceive of this by assuming that the generative model for $\mathbf{O}$, $\mathbf{M}$ has low Kolmogorov Complexity. Here, a low capacity bi-modal model is assumed. We utilize Minimum Description Length $L(\cdot)$ (MDL) as a tractable substitute of the Kolmogorov Complexity [136, 63]).

Causal curiosity solves the following optimization problem.

$$\mathbf{a}_{0:T}^{*} = \arg\min_{\mathbf{a}_{0:T}} (L(\mathbf{M}) + L(\mathbf{O}|\mathbf{M})) \tag{3.7}$$

where each observed trajectory $\mathbf{O} = \mathbf{O}(\mathbf{a}_{0:T})$ is a function of the action sequence. As mentioned earlier, the model is fixed in this formulation; hence, the first term $L(\mathbf{M})$ is constant and not a function of the

Algorithm 1 Recursive Training Scheme

Initialize $j = 0$
Initialize training environment set $Envs$ **Train**(j, $\mathbf{z}^i_{0:j}$, $Envs$)
if $j == K$ **then**
  Return
end if
for iteration m to M **do**
  Sample experimental behavior $\mathbf{a}_{0:T} \sim$ CEM($\cdot|\mathbf{z}^i_{0:j}$)
  for i^{th} env in $Envs$ **do**
    Apply $\mathbf{a}_{0:T}$ to env
    Collect $\mathbf{O}^i = \mathbf{o}^i_{0:T}$
    Reset env
  end for
  Calculate $-L(\mathbf{O}|\mathbf{M})$ given that $\mathbf{M}$ is bimodal clustering model
  Update CEM($\cdot$) distribution with highest reward trajectories
end for
Use learnt $q_M(\mathbf{z}^i_j|\mathbf{O}, \mathbf{z}^i_{0:j})$ for cluster assignment of each env in $Envs$ i.e. $\mathbf{z}^i_j = q_M(\mathbf{z}|\mathbf{O}^i, \mathbf{z}^i_{0:j})$
Update $j = j + 1$
Train(j, $\mathbf{z}^i_{0:j}$, $Envs = \{\mathbf{e}^i : \mathbf{z}^i_{j-1} = 0\}$)
Train(j, $\mathbf{z}^i_{0:j}$, $Envs = \{\mathbf{e}^i : \mathbf{z}^i_{j-1} = 1\}$)

actions. The MDL of the trajectories given binary categorization model, $-L(\mathbf{O}|\mathbf{M})$, is the inherent reward function that is fed back to the RL agent. We regard this reward function as *causal curiosity*.

3.2.3 Causal Inference Module

By maximizing the causal curiosity reward it is possible to achieve behaviors which result in trajectories of states only caused by a single hidden parameter. Subsequently, we utilize the outcome of performing these experimental behaviors in each environment to infer a representation for the causal factor isolated by the experiment in question.

We achieve this through clustering. An action sequence $\mathbf{a}_{0:T} \sim$ CEM($\cdot|\mathbf{z}^i_{0:j-1}$) is sampled from the Model Predictive Control Planner [26] and applied to each of the training environments. The learnt

clustering model $\mathbf{M}$ is then used to infer a representation for each environment using the collected outcome $\mathbf{O}^i$ obtained by applying $\mathbf{a}_{0:T}$ to each environment.

$$\mathbf{z}_j^i = q_M(\mathbf{z}|\mathbf{O}^i, \mathbf{z}_{0:j-1}^i) \tag{3.8}$$

The learnt representation $\mathbf{z}$ is the cluster membership obtained from the learnt clustering model $\mathbf{M}$. It is binary in nature. This corresponds to the quantization of the continuous spectrum of values a causal factor takes in the training set into high and low values.

3.2.4 Interventions on beliefs

Having learnt about the effects of a single causal factor of the environment we wish to learn such experimental behaviors for each of the remaining hidden parameters that may vary in the training environments. To achieve this, in an ideal setting, the agent would require access to the generative mechanism of the environments it encounters. Ideally, it would hold the values of the causal factor already learnt about a constant i.e. $do(\mathbf{h}_j = constant)$, and intervene over (vary the value of) another causal factor over a set of values V i.e. $do(\mathbf{h}_{j'} = v)$ such that $v \in V$. For example, if a human scientist were to study the effects of a causal factor, say mass of a body, they would hold the values of all other causal factors constant (e.g., interact with cubes of the same size and external texture), and vary only mass to see how it affects the outcome of specific behaviors they apply to each body.

However, in the real world the agent does not have access to the generative mechanism of the environments it encounters, but merely has the ability to act in them. Thus, it can intervene on the representations of a causal factor of the environment i.e. $do(\mathbf{z}_i = constant)$. For, example having learnt about gravity, the agent picks all environments it believes have the same gravity, and uses them to learn about a separate causal factor say, friction.

Thus, to learn about the j^{th} causal factor, the agent proceeds in a tree-like manner, dividing each of the 2^{j-1} clusters of training environments into two sub-clusters corresponding to the binary quantized values of the j^{th} causal factor. Each level of this tree corresponds to a single causal factor.

$$Envs = \{\mathbf{e}^i : \mathbf{z}^i_{j-1} = k\}, k \in \{0, 1\} \tag{3.9}$$

This process continues iteratively (Algorithm 1 and Figure 3.3), where for each cluster of environments, a new experiment learns to split the cluster into 2 sub-clusters depending on the value of a hidden parameter. At level n, the agent produces 2^n experiments, having already intervened on the binary quantized representations of n causal factors.

3.3 Related Work

Curiosity for robotics is not a new area of research. Pioneered by Schmidhuber in the 1990s, [144, 145, 146, 147], [115], [125] curiosity is described as the motivation behind the behavior of an agent in an environment for which the outcome is unpredictable, i.e., an intrinsic reward that motivates the agent to explore the unseen portions of the state space (and subsequent transitions). [46] define a class Markov Decision Processes where transition probabilities $p(s_{t+1}|s_t, a_t; \theta)$ depend on a hidden parameter θ, whose value is not observed, but its effects are felt. [87] and [179] utilize these Hidden Parameter MDPs (Markov Decision Processes) to enable efficient policy transfer, assuming that transition probabilities across states are a function of hidden parameters. [127] relax this assumption, allowing both transition probabilities and reward functions to be functions of hidden parameters. [190] approach the problem from a Bayes-optimal policy standpoint, defining transition probabilities and reward functions to be dependent on a hidden parameter characteristic of the MDP in consideration. We utilize this setup to define causal factors.

Substantial attempts have been made at unsupervised disentanglement, most notably, the β-VAE [72] [24],

where a combination of factored priors and the information bottleneck force disentangled representations. [88] enforce explicit factorization of the prior without compromising on the mutual information between the data and latent variables, a shortcoming of the β-VAE. [31] factor the KL divergence into a more explicit form, highlighting an improved objective function and a classifier-agnostic disentanglement metric. [100] show theoretically that unsupervised disentanglement (in the absence of inductive biases) is impossible and highly unstable, susceptible to random seed values. They follow this up with [101] where they show, both theoretically and experimentally, that pair-wise images provide sufficient inductive bias to disentangle causal factors of variation. However, these works have been applied to supervised learning problems whereas we attempt to disentangle the effects of hidden variables in dynamical environments, a relatively untouched question.

3.4 Experiments

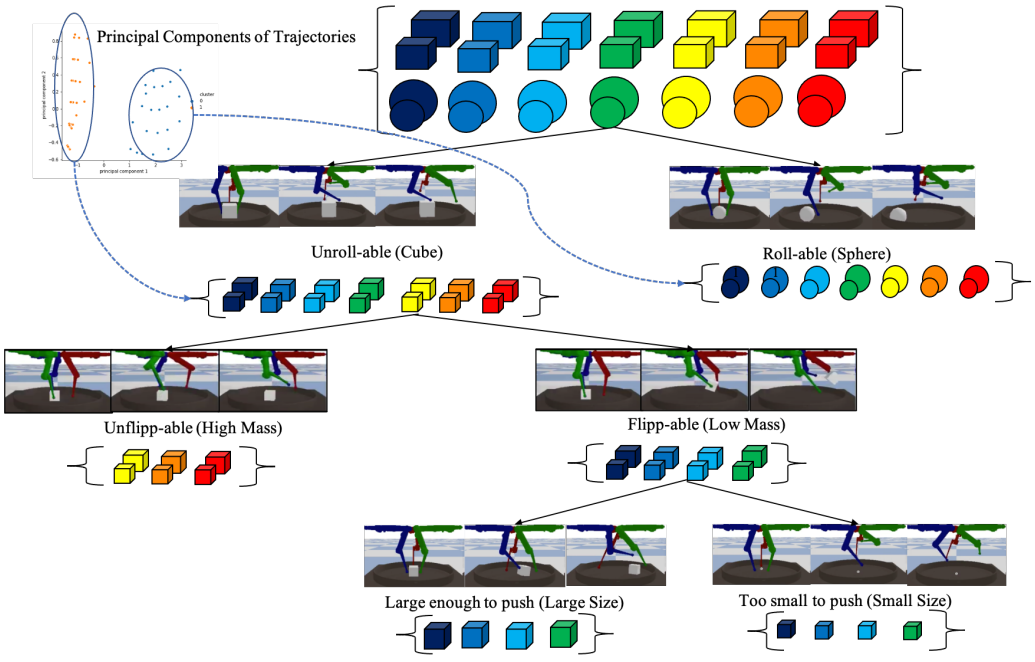


Figure 3.3: Discovered hierarchical latent space. The agent learns experiments that differentiate the full set of blocks in `ShapeSizeMass` into hierarchical binary clusters. At each level, the environments are divided into 2 clusters on the basis of the value of a single causal factor. We also show the principal components of the trajectories in the top left. For brevity, the full of extent of the tree is not depicted here. For each level of hierarchy k, there are 2^k number of clusters.

Our work has 2 main thrusts - the discovered *experimental behaviors* and the *representations* obtained from the outcome of the behaviors in environments. We visualize these learnt behaviors and verify that they are indeed semantically meaningful and interpretable. We quantify the utility of the learned behaviors by using the behaviors as pre-training for a downstream task. In our experimental setup, we verify that these behaviors are indeed invariant to all other causal factors except one.

We visualize the representations obtained using these behaviors and verify that they are indeed the binary quantized representations for each of the ground truth causal factors that we manipulated in our experiments. Finally, we verify that the knowledge of the representation does indeed aid transfer learning and zero-shot generalizability in downstream tasks.

Causal World. We use the Causal World Simulation [3] based on the Pybullet Physics engine to test our approach. The simulator consists of a 3-fingered robot, with 3 joints on each finger. We constrain each environment to consist of a single object that the agent can interact with. The causal factors that we manipulate for each of the objects are size, shape and mass of the blocks. The simulator allows us to capture and track the positions and velocities of each of the movable objects in an environment.

Mujoco Control Suite. We optimize causal curiosity on 4 articulated agents that try to learn locomotion - Ant, Half-Cheetah, Hopper, and Walker. For each agent type, we train with agent body masses from $0.5\times$ to $1.5\times$ the default.

3.4.1 Visualizing Discovered Behaviors

We would like to analyze whether the discovered experimental behaviors are human interpretable, i.e., *are the experimental behaviors discovered in each of the setups semantically meaningful?* We find that our agents learn to perform several useful behaviors without any supervision. For instance, to differentiate between objects with varying mass, we find that they acquire a perfect grasp-and-lift behavior with an upward force. In other random seed experiments, the agents learn to lift the blocks by using the wall of the environment

for support. To differentiate between cubes and spheres, the agent discovers a pushing behavior which gently rolls the spheres along a horizontal direction. Qualitatively, we find that these behaviors are stable and predictable. See videos of discovered behaviors `here` (website under construction).

Concurrent with the objective they are trained on, we find that the acquired behaviors impose structure on the outcome when applied to each of the training environments. The outcome of each experimental behavior on the set of training environments results in dividing it into 2 subsets corresponding to the binary quantized values of a single factor, e.g., large or small, while being invariant to the values of other causal factors of the environments. We also perform ablation studies where instead of providing the full state vector, we provide only one coordinate (e.g., only x, y or z coordinate of the block). We find that causal curiosity results in behaviors that differentiate the environments based on outcomes along the direction provided. For example, when only the x coordinate was provided, the agent learned to evaluate mass by applying a pushing behavior along the x direction. Similarly, a lifting behavior was obtained when only the z coordinate was supplied to the curiosity module (Figure 3.4).

Causal curiosity also yields semantically meaningful behaviors that test out agent mass in Mujoco: the Half-Cheetah learns a front-flip, the Hopper learns to hop to gauge its own mass, in the absence of external rewards (Fig 3.7).

3.4.2 Utility of learned behaviors for downstream tasks

While the behaviors acquired are semantically meaningful, we would like to quantify their utility as pre-training for downstream tasks. We analyze the performance on `Lifting` where the agent must grasp and lift a block to a predetermined height and `Travel`, where the agent must impart a velocity to the block along a predetermined direction. We re-train the learnt planner using an external reward for these tasks (**Curious**). We implement a baseline vanilla Cross Entropy Method optimized Model Predictive Control Planner [42] (**Vanilla CEM**) trained using the identical reward function and compare the rewards per

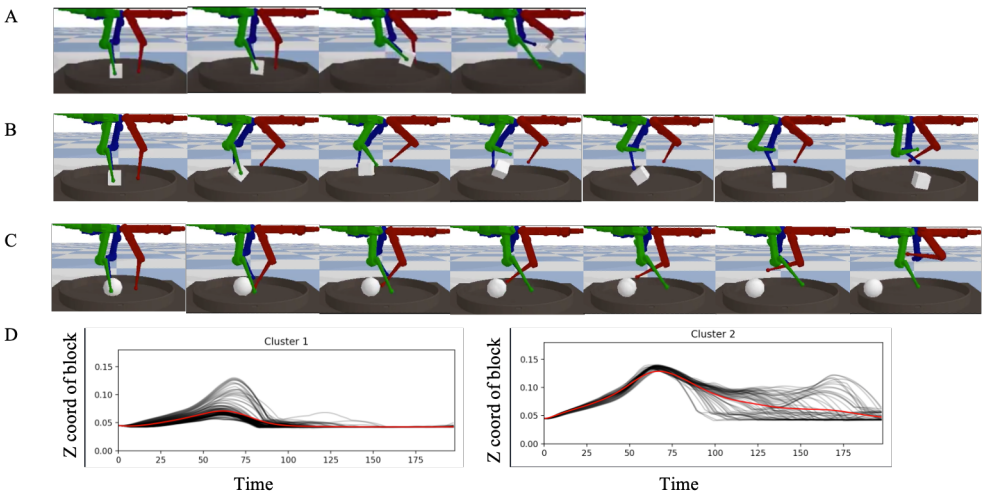
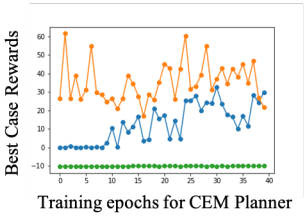
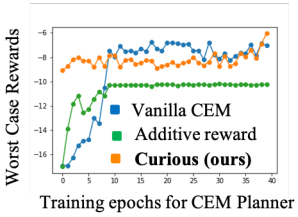


Figure 3.4: Examples of discovered behaviors. The agent discovers experimental behaviors that allow it to characterize each environmental object in a binary manner, e.g., heavy/light, big small, rollable/not rollable, etc. These behaviors are acquired without any external supervision by maximizing the causal curiosity reward. **A, B, C** correspond to self-discovered toss, lift-and-spin and roll behaviors respectively. **D** shows an ablation study, where the agent is only provided the z coordinate of the block in every environment. Each line corresponds to one environment and the z coordinate of the block is plotted with time when the discovered behavior is applied. It learns a lifting behavior, where cluster 1 represents the heavy blocks (z coordinate does not change much) and cluster 2 represents the light blocks (z increases as block is lifted and then falls when dropped and subsequently increases again when it bounces).

trajectory during training. We also run a baseline (**Additive reward**) which explores whether the agent recieves both the causal curiosity reward and the external reward. We find high zero-shot generalizability and quicker convergence as compared to the vanilla CEM (Figure 3.5). We also find that additive rewards, achieves suboptimal performance due to competing objectives.





Discovered Experiments as Pre-training	
Method	**Time Steps of training**
Vanilla CEM Planner	$15 \times 10^5 \pm 122.47$
Curious (Ours)	6×10^5

Figure 3.5: Utility of discovered behaviors. We find that the behaviors discovered by the agents while optimizing causal curiosity show high zero-shot generalizability and converge to the same performance as conventional planners for downstream tasks. We also analyze the worst case performance and find that the pre-training ensures better performance than random initialization. The table compares the time-steps of training required on an average to acquire a skill with the time steps required to learn a similar behavior using external reward. We find that the unsupervised experimental behaviors are approximately 2.5 times more sample efficient. We also find that maxizing both curiosity and external reward in our experimental setups results in sub-optimal results.

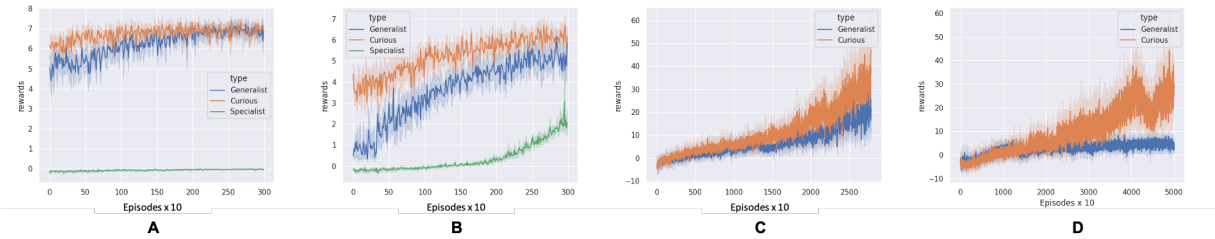


Figure 3.6: Knowledge of causal factors aids transfer. We find that knowledge of the causal representation allows agents to generalize to unseen environments with high zero-shot performance. We find that as the number of varying causal factors increase, the difference in zero-shot performance of the Causally-curious agent and the Generalist increases, showing that the CC agents are indeed robust to multiple varying causal factors. Panes **A** and **B** are represent reward curves for `TransferMass` and `TransferSizeMass`. Pane **C** shows the training rewards curve for the new `StackingTower` experiment and Pane **D** shows the reward curves during adaptation to an unseen value of causal factors.

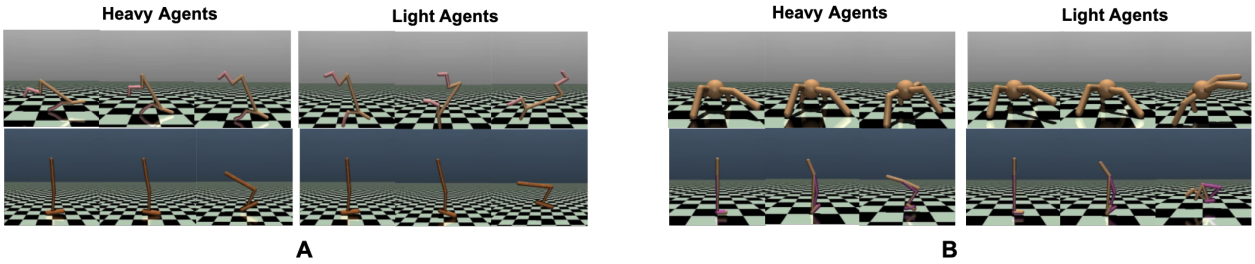


Figure 3.7: Mujoco Experiments. Pane **A** shows discovered experimental behaviors on Half-Cheetah and Hopper, while Pane **B** shows discovered experimental behaviors on Ant and Walker. The Half-Cheetah learns to front-flip, the Hopper learns to hop, and the Ant learns to rear up on its hind legs to gauge its mass.

3.4.3 Visualization of hierarchical binary latent space

Our agents discover a disentangled latent space such that they are able to isolate the sources of causation of the variability they encounters in their environments. For every environment, they learn a disentangled embedding vector which describes each of the causal factors.

To show this, we use 3 separate experimental setups - `Mass`, `SizeMass` and `ShapeSizeMass` where each of the causal factors are allowed to vary over a range of discrete values.

During training, the agent discovers a hierarchical binary latent space (Figure 3.3), where each level of hierarchy corresponds to a single causal factor. The binary values at each level of hierarchy correspond to the high/low values of the causal factor in question. To our knowledge, we obtain the first interpretable

latent space describing the various causal processes in the environment of an agent. This implies that it learns to quantify each physical attribute of the blocks it encounters in a completely unsupervised manner.

3.4.4 Knowledge of causal factors aids transfer

Next, we test whether knowledge of the causal factors does indeed aid transfer and zero-shot generalizability. To this end, we supply the representations obtained by the agent during the experimental behavior phase as input to a policy network in addition to the state of the simulator, and train it for a place-and-orient downstream task (Figure 3.1). We define 2 experimental setups - `TransferMass` and `TransferSizeMass` where mass and size of the object in each environment is varied. We also test our agent in a separate task, `StackingTower`, where the agent is provided 2 blocks which it must use to build a stable tower configuration. These blocks vary in mass and the agent must use causal representations to learn to build towers with a heavy base for stability. In each of the setups, the agent learns about the varying causal mechanisms by optimizing causal curiosity. Subsequently, using the causal representation along with the state for each environment, it is trained to maximize external reward.

After training, the agents are exposed to a set of unseen test environments, where we analyze their zero-shot generalizability. These test environments consist of unseen masses and sizes and their unseen combinations. This corresponds to "Strong Generalization" as defined by [127]. We report results averaged over 10 random seeds.

For each setup, we train a PPO-optimized Actor-Critic Policy (referred to as **Causally-curious agent**) with access to the causal representations and an observation vector from the environment i.e., $\mathbf{a}_t \sim \pi(\cdot|\mathbf{o}_t, \mathbf{z}_{0:K})$. Similar to [127], we implement 2 baselines - the **Generalist** and the **Specialist**. The **Specialist** consists of an agent with identical architecture as **Causally-curious agent**, but without access to causal representations. It is initialized randomly and is trained only on the test environments, serving as a benchmark for complexity of the test tasks. It performs poorly, indicating that the test tasks are complex.

The architecture of the **Generalist** is identical to the **Specialist**. Like the **Specialist**, the **Generalist** also does not have access to the causal representations, but is trained on the same set of training environments that the Causally-curious agent is trained on. The poor performance of the generalist indicates that the tasks distribution of training and test tasks differs significantly and that memorization of behaviors does not yield good transfer. We find that causally-curious agents significantly outperform the both baselines indicating that indeed, knowledge of the causal representation does aid zero-shot generalizability.

3.5 Conclusion

Our work introduces a causal viewpoint of POMDPs, where unobserved static state variables (i.e., causal factors) affect the transition of dynamic state variables. Causal curiosity rewards experimental behaviors an agent can conduct in an environment that underscore the effects of a subset of such global causal factors while obfuscating the effects of others. Motivated by the popular One-Factor-at-a-Time (OFAT) [54, 71, 40], our agents study the effects causal factors have on the dynamics of an environment through active experimentation and subsequently obtain a disentangled causal representation for causal factors of the environment. Finally, we show that knowledge of causal representations does indeed improve sample efficiency in transfer learning.

Causal curiosity can help RL agents adapt to new values of seen causal factor variations. But what if a new causal factor is varied at inference? How can RL agents adapt? The next chapter GalilAI studies such settings.

Chapter 4

GalilAI: Out-of-Task Distribution Detection using Causal Active Experimentation for Safe Transfer RL

4.1 Introduction and Related Work

Generalization to near-distribution shifts caused by natural perturbations and **Detection** of out-of-distribution shifts caused by artificial perturbations (adversarial attacks) are central desiderata of modern decision-making systems. Significant advances have been made in supervised learning systems on both fronts - with work in transfer/meta-learning aiding the ability of ML systems to generalize across shifts in input distributions [148, 141, 52]. Such methods learn internal representations which are invariant to perturbations occurring in data [10]. These invariant representations are subsequently used for domain adaptation [184, 108], with applications in music [16] and speech [153]. Out-of-distribution Detection for the supervised learning domain has also made significant advances [70, 45, 98, 59], with the development of both training-time methods [176] (alterations to typical supervised training to make models robust to OOD inputs) and inference-time methods (utilizing the features of a fully trained model to detect OOD samples)[75].

While attempts have been made in generalization in the space of sequential long-horizon RL and decision-making [52, 111, 65, 123, 133, 190], Out-of-Distribution Detection is fairly unexplored. To our knowledge, our work is the first that offers a concrete causal framework for OOD Detection.

We motivate the need for OOD Detection in RL with an example. Consider an agent that has learnt to land an aircraft for various values of directions and velocity of crosswinds. Now consider the situation when one of the airplane's engines fails when the agent is deployed. Current RL systems would assume that the observations they receive from this test environment were caused by perhaps high crosswinds and would subsequently increase fuel flow to the engines - a potentially disastrous strategy. On the contrary, a seasoned pilot might perform an experiment - perhaps yawing the aircraft from side-to-side, concluding that due to the low controllability of the aircraft, the engine was somehow compromised. Our work extends that of Sontakke et al. by utilizing advances in algorithmic information theory and curiosity-based reinforcement learning to "encourage" the RL agent to perform such experimental behaviors and conclude whether a test-time environment is out-of-training-distribution or not.

During our experiments, we find that our agent discovers the **Galilean Equivalence Principle**, managing to successfully decouple the effect of mass and gravitational acceleration. For this reason, we refer to the agent as GalilAI (pronounced Galilei). The contributions of our work are as follows:

- **Causal transfer:** We offer a causal perspective on transfer learning in RL and provide a theoretical framework for defining various classes of transfer RL problems.
- **Causal active experimentation (GalilAI) for safe transfer RL:** We extend the work of Sontakke et al. to provide an algorithm aimed at improving the safety of transfer reinforcement learning by detecting whether a given test environment is out-of-distribution or not. If an environment is detected as OOD, the agent could relinquish control of a system to a human operator [4].

- **Probabilistic baseline:** Due to a lack of prior work in the field, we propose a simple probabilistic neural network baseline for OOD Detection of environments in RL. We compare GalilAI and the PNN in complex robotic domains such as the Causal World [3] and Mujoco [166].

4.2 Preliminaries

Definition 2 (Causal factors)**.** *Consider the POMDP ($\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, ϕ, θ, r) with observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, the transition function ϕ, emission function θ, and the reward function r. Let $\mathbf{o}_{0:T} \in \mathcal{O}^T$ denote a trajectory of observations of length T. Let $d(\cdot,\cdot) : \mathcal{O}^T \times \mathcal{O}^T \to \mathbb{R}_+$ be a distance function defined on the space of trajectories of length T. The set $H = \{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{K-1}\}$ is called a set of $\epsilon-$causal factors if for every $\mathbf{h}_j \in H$, there exists a unique sequence of actions $\mathbf{a}_{0:T}$ that clusters the observation trajectories into m disjoint sets $C_{1:m}$ such that $\forall C_a, C_b$, a minimum separation distance of ϵ is ensured:*

$$\min\{d(\mathbf{o}_{0:T}, \mathbf{o}'_{0:T}) : \mathbf{o}_{0:T} \in C_a, \mathbf{o}'_{0:T} \in C_b\} > \epsilon \tag{4.1}$$

and that $\mathbf{h}_j$ is the cause of the obtained trajectory of states i.e. $\forall v \neq v'$,

$$p(\mathbf{o}_{0:T}|do(\mathbf{h}_j = v), \mathbf{a}_{0:T}) \neq p(\mathbf{o}_{0:T}|do(\mathbf{h}_j = v'), \mathbf{a}_{0:T}) \tag{4.2}$$

where $do(\mathbf{h}_j)$ corresponds to an intervention on the value of the causal factor $\mathbf{h}_j$.

According to Definition 2, a causal factor $\mathbf{h}_j$ is a variable in the environment the value of which, when intervened on (i.e., varied) using $do(\mathbf{h}_j)$ over a set of values, results in trajectories of observations that are divisible into disjoint clusters $C_{1:m}$ under a particular sequence of actions $\mathbf{a}_{0:T}$. These clusters represent the quantized values of the causal factor. For example, mass, which is a causal factor of a body, under an

action sequence of a grasping and lifting motion with fixed force, may result in 2 clusters, liftable (low mass) and not-liftable (high mass).

4.2.1 POMDPs and Causal POMDPs

Classical POMDPs ($\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, ϕ, θ, r) consist of an observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, the transition function ϕ, emission function θ, and the reward function r. An agent in an unobserved state $\mathbf{s}_t$ takes an action $\mathbf{a}_t$ and consequently causes a transition in the environment through $\phi(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The agent receives an observation $\mathbf{o}_{t+1} = \theta(\mathbf{s}_{t+1})$ and a reward $\mathbf{r}_{t+1} = r(\mathbf{s}_t, \mathbf{a}_t)$. **Causal POMDPs** explicitly model the effects of causal factors on the transition and emission functions by dividing the state into the controllable state $\mathbf{s}_t^c$ and the causal factor, $\mathcal{H}$. The causal factors of an environment cannot be manipulated by the agent, but their values affect the outcome of an action taken by the agent. Thus the transition function of the controllable state is:

$$\phi(\mathbf{s}_{t+1}^c|\mathbf{s}_t^c, f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t), \mathbf{a}_t) \tag{4.3}$$

where f_{sel} is the implicit Causal Selector Function which selects the subset of causal factors affecting the transition defined as:

$$f_{sel} : \mathcal{H} \times \mathcal{S} \times \mathcal{A} \to \wp(\mathcal{H}) \tag{4.4}$$

where $\wp(\mathcal{H})$ is power-set of $\mathcal{H}$ and $f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t) \subset \mathcal{H}$ is the set of effective causal factors for the transition $\mathbf{s}_t \to \mathbf{s}_{t+1}$ i.e., $\forall v \neq v'$ and $\forall \mathbf{h}_j \in f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t)$:

$$\phi(\mathbf{s}_{t+1}^c|do(\mathbf{h}_j = v), \mathbf{s}_t^c, \mathbf{a}_t) \neq \phi(\mathbf{s}_{t+1}^c|do(\mathbf{h}_j = v'), \mathbf{s}_t^c, \mathbf{a}_t) \tag{4.5}$$

where $do(\mathbf{h}_j)$ corresponds to an external intervention on the factor $\mathbf{h}_j$ in an environment.

Intuitively, this means that if an agent takes an action $\mathbf{a}_t$ in the controllable state $\mathbf{s}_t^c$, the transition to $\mathbf{s}_{t+1}^c$ is caused by a subset of the causal factors $f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t)$. For example, if a body on the ground (i.e., state $\mathbf{s}_t^c$) is thrown upwards (i.e., action $\mathbf{a}_t$), the outcome $\mathbf{s}_{t+1}$ is caused by the causal factor gravity (i.e., $f_{sel}(\mathcal{H}, \mathbf{s}_t^c, \mathbf{a}_t) = \{\text{gravity}\}$), a singleton subset of the global set of causal factors. The $do()$ notation expresses this causation. If an external intervention on a causal factor is performed, e.g., if somehow the value of gravity was changed from v to v', the outcome of throwing the body up from the ground, $\mathbf{s}_{t+1}$, would be different.

4.2.2 Algorithmic Information Theoretic View on Causality

Causality can be motivated from the perspective of algorithmic information theory [80]. Consider the Gated Directed Acyclic Graph of the observed variable $\mathbf{O}$ and its causal parents (Figure 4.1). Each causal factor has its own causal mechanism, jointly bringing about $\mathbf{O}$. The action sequence $\mathbf{a}_{0:T}$ serves a gating mechanism, allowing or blocking particular edges of the causal graph using the implicit Causal Selector Function (Equation (4.4)). A central assumption of our approach is that causal factors are independent, i.e., the Independent Mechanisms Assumption [6, 122, 150]. The information in $\mathbf{O}$ is then the sum of information "injected" into it from the multiple causes, since, loosely speaking, for information to cancel, the mechanisms would need to be algorithmically dependent [80]. Thus, the information content in $\mathbf{O}$ will be greater for a larger number of independent causal parents in the graph.

$$L(\mathbf{O}) \propto |PA(\mathbf{O}))| \tag{4.6}$$

where $L(\cdot)$ is the Minimum Description Length (MDL), a tractable substitute of the Kolmogorov Complexity [136, 63]).

4.2.3 Causal Curiosity

Causal curiosity [158] allows an RL agent to discover sequences of actions that bring out the effect of a single causal factor while ignoring the effects of all other. This is similar to how a human scientist studying multiple mechanisms in their environment would behave whilst following the One-Factor-at-a-Time (OFAT) paradigm of experiment design [54]. For e.g., when interacting with objects of varying mass and shape, a human scientist will learn a perfect lifting sequence that grasps all shapes and then use it to test out the mass of each object.

Thus, *Causal Curiosity selects one among multiple competing causal mechanisms* and generates a sequence of actions that bring out the effect of the selected mechanism. This is done by attempting to learn a simple model of the environment with capacity low enough to learn about only a single causal mechanism at a time. One could conceive of this by assuming that the generative model for $\mathbf{O}$, $\mathbf{M}$ has low Kolmogorov Complexity. A low capacity bi-modal model is assumed. The Minimum Description Length (MDL), $L(\cdot)$ is utilized as a tractable substitute of the Kolmogorov Complexity [136, 63]). Subsequently, the following optimization problem is solved.

$$\mathbf{a}^*_{0:T} = \arg\min_{\mathbf{a}_{0:T}} (L(\mathbf{M}) + L(\mathbf{O}|\mathbf{M})) \tag{4.7}$$

where each observed trajectory $\mathbf{O} = \mathbf{O}(\mathbf{a}_{0:T})$ is a function of the action sequence. Thus the resulting action sequence from the optimization in Equation (4.7) will result in an action sequence that brings out the effect of a single causal factor. Having established this, we now introduce a causal perspective on transfer.

4.2.4 Causal Perspective on Transfer

Consider the set of POMDPs $P = \{\mathbf{p}_0, \mathbf{p}_1, \ldots\}$ parameterized by the tuple ($\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, ϕ, θ, r, $H' \subset H$) with observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, the transition function ϕ, emission function θ, and the

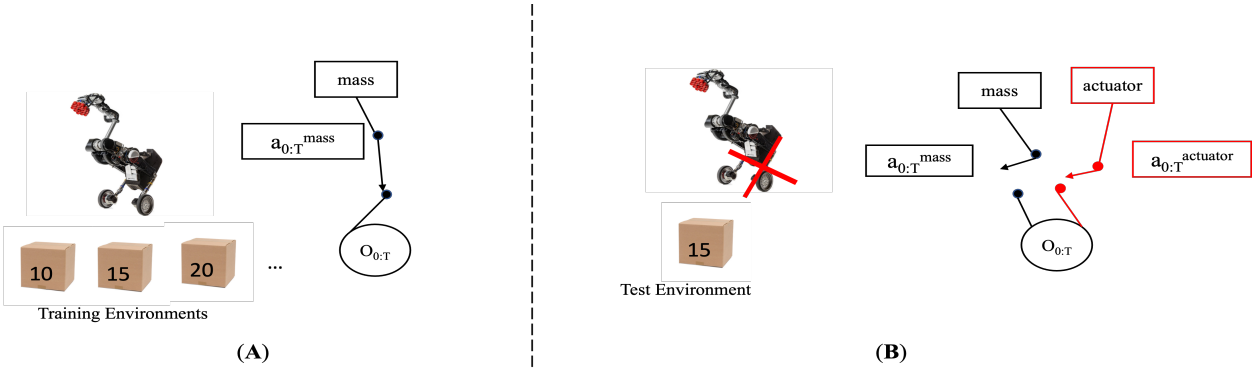


Figure 4.1: Out-of-Task Distribution Transfer. Pane **A** shows a training-time scenario where an agent learns to interact with environments containing objects for varying values of mass. The causal graph is gated as particular action sequences either obfuscate or underscore the effects of certain causal factors. Pane **B** represents the inference-time scenario where the a causal factor, actuator health, held constant during training is varied at inference.

reward function r, with the set of causal factors $H' \subset H$, i.e., subset of the global causal factors, varied over a range of values and the remaining $H - H'$ held constant.

Definition 3 (In-Task-Distribution Transfer)**.** *An in-task-distribution transfer occurs when an agent trained on P is launched into a POMDP $\mathbf{p}'$ where $\forall\ \mathbf{h} \in H - H'$ the values assumed by $\mathbf{h}$ remain unchanged (assume the same values as in P).*

Definition 4 (Out-of-Task-Distribution Transfer)**.** *An Out-of-Task-distribution transfer occurs when an agent trained on P is launched into a POMDP $\mathbf{p}'$ where $\exists\ \mathbf{h} \in H - H'$ which assumes a value different from the value it had in P.*

Consider a transfer learning agent training to lift cubes with varying masses and sizes, i.e., $H' = \{mass, size\}$. An In-Task-Distribution Transfer scenario occurs if at test-time it encounters an cube of an unseen mass/size combination. An Out-of-Task-Distribution scenario occurs if it is required to lift a cube with a broken actuator. This is because the causal factor $actuator \in H - H'$ which was held constant during training (agent trained is using a healthy actuator) is required to lift using a broken actuator at test-time. We would like to be able to detect such faults while generalizing to known causal factors.

4.3 Method

Setup We consider the scenario where a learning agent is trained on a set of POMDPs $P = \{\mathbf{p}_0, \mathbf{p}_1, \ldots\}$ parameterized by the tuple ($\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, ϕ, θ, r, $H' \subset H$) with observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, the transition function ϕ, emission function θ, and the reward function r, with the set of causal factors $H' \subset H$, i.e., subset of the global causal factors, varied over a range of values and the remaining $H - H'$ held constant.

We assume that the learning agent is able to learn $\mathbf{z} = Z_\phi(\mathbf{p})$, called *belief function*, using each of the training environments which generates a representation for the intervened causal factors, i.e., $H' \subset H$. This assumption is quite general - the RL systems that are capable of performing well over different environments can be assumed to either explicitly model such representations (as in [133, 190, 127]) or implicitly (as in [52, 111]). At test time, the agent is launched into a novel environment $\mathbf{p}'$ which is either an In-Task-Distribution Transfer (see Definition 3) or Out-of-Task-Distribution Transfer (see Definition 4).

4.3.1 Construction of the Belief Set

The agent performs inference in the novel test environment $\mathbf{p}'$ using $Z_\phi(\mathbf{p}')$. We assume that the agent has access to $\{Z_\phi(\mathbf{p}) : \mathbf{p} \in P\}$, i.e., the belief representation for the training environments. The agent then collects all training environments that lie near $\mathbf{p}'$ in the space of the learned belief functions into the ball $\mathcal{B}$ called the *belief set* defined as,

$$\mathcal{B} := \{\mathbf{p}_i : d(q_\phi(\mathbf{z}|\mathbf{p}')||q_\phi(\mathbf{z}|\mathbf{p}_i)) < \epsilon\} \tag{4.8}$$

where $d(\cdot|\cdot)$ is a distance function (e.g., Euclidean) in the latent space and ϵ is a design hyperparameter.

Thus, for example, in a lifting task of cubes of various masses, if the agent fails to lift a cube at test time, it constructs the belief ball consisting of the training environments with close representations, i.e.,

heavy cubes and adds them to the belief set $\mathcal{B}$. Depending on the cause for the failure of the agent in lifting the cube, the situation goes into one of the following branches: (1) The test environment requires an **In–Task-Distribution Transfer**, i.e., the test-time block is actually a heavy block or (2) The test environment requires **Out-of-Task-Distribution Transfer**, i.e., a broken actuator makes a light block seem heavy.

4.3.2 Belief Verification

Subsequently, the agent optimizes causal curiosity on $\mathcal{B} \cup \{\mathbf{p}'\}$. As in Equation (4.7), a low capacity binary clustering model is considered. Thus, the following optimization procedure is implemented:

$$\begin{aligned} \arg\max_{\mathbf{a}_{0:T} \in \mathcal{A}^T} [\min\{d(\mathbf{o}_{0:T}, \mathbf{o'}_{0:T}) : \mathbf{o}_{0:T} \in C_1, \mathbf{o'}_{0:T} \in C_2\} - \\ \max\{d(\mathbf{o}_{0:T}, \mathbf{o''}_{0:T}) : \mathbf{o''}_{0:T}, \mathbf{o}_{0:T} \in C_1\} - \\ \max\{d(\mathbf{o'}_{0:T}, \mathbf{o'''}_{0:T}) : \mathbf{o'}_{0:T}, \mathbf{o'''}_{0:T} \in C_2\}] \end{aligned} \tag{4.9}$$

where $\mathbf{O}$ is the observation obtained by applying action sequence $\mathbf{a}_{0:T}$. Clusters C_1 and C_2 represent the bimodal model.

In-Task Distribution If the test environment $\mathbf{p}'$ is In-Task Distribution, then the variance of values assumed by the causal factors H' in the set of environments $\mathcal{B} \cup \{\mathbf{p}'\}$ is small and the clusters are *not well-separated*. Thus optimizing causal curiosity as in Equation (4.9) will produce action sequences that result in observations that cluster in a distributed manner as in pane **A** of Figure 4.2.

Intuitively, if the agent has learnt to interact with blocks of various masses and at test time is presented with a heavy block, the outcome of its interaction with the test block (i.e., $\mathbf{p}'$) will not differ significantly in comparison with the heavy blocks it interacted with during training.

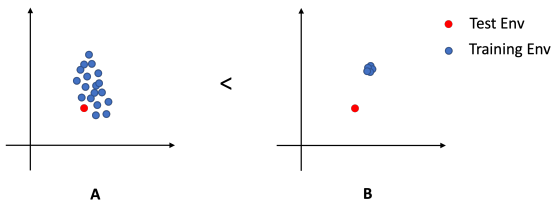


Figure 4.2: Visualization of the Observation O. Pane **A** represents the observation variables obtained after the optimization of Equation (4.9) during an In-Task-Distribution Transfer. Causal Curiosity will be quite low in such a case as the bi-modal clustering would be poor. Pane **B** represents the case when OOTD transfer occurs - the causal curiosity reward would be high as the bi-modal clustering would be near-perfect.

Out-of-Task-Distribution However, during the optimization of Equation (4.9) in the OOTD case, 2 competing causal mechanisms will exist - one induced by the set H' and the other from the set $H - H'$. The mechanism caused by $H - H'$ will however dominate as all environments in $\mathcal{B}$ will have the same values for $H - H'$ while $\mathbf{p}'$ will have a different value. Thus, the resulting clusters for the causal mechanism from $H - H'$ will be *well-separated*. Subsequently, the causal curiosity reward (Equation (4.9)) will be higher for selecting the causal mechanism induced by $H - H'$.

Intuitively, as in the above example of an agent interacting with blocks of varying masses but constant size ξ, if at test time, the agent is provided with a block of low mass and a new size $\xi' \neq \xi$, the causal curiosity reward for the size mechanism will be higher because a perfect binary clustering is possible (as in pane **B** in Figure 4.2) where one cluster contains observations from training environments (blue cluster) corresponding to size s while the other cluster corresponds to the test environment with size s' (red cluster). Thus, if the test environment $\mathbf{p}'$ lies in its own cluster after optimizing causal curiosity on $\mathcal{B} \cup \{\mathbf{p}'\}$, then GalilAI concludes $\mathbf{p}'$ to be OOTD, i.e.,

$$\text{Is\_OOTD}(p') = \begin{cases} 1, & \text{if } \mathbf{p}' \text{ lies in its own cluster} \\ 0, & \text{otherwise} \end{cases}$$

Note, the causal curiosity for a known causal factor, (In-Task-Distribution Transfer) will be less than the causal curiosity for an unknown causal factor (OOTD Transfer) as seen in Figure 4.2.

4.3.3 Probabilistic Baselines

A natural extension of Model-based learning methods for OOTD is possible. We question whether such an extension yields good results. We test whether OOTD Detection is possible by simply learning a model of the training and test environments and using using the discrepancy of the outputs to detect whether the learnt test-time model represents a task from OOTD.

We utilize an ensemble of probabilistic neural networks (PNNs) [93, 36], which is a generative neural network whose output neurons parameterize a probability distribution $p_\theta(y|\mathbf{x})$; a mean value corresponds to the believed label $\hat{y}$ along with some degree of uncertainty θ.

For an environment $\mathbf{p}$, we estimate the environment transition function $\phi_{\mathbf{p}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ using an ensemble of PNNs $f_\phi^{\mathbf{p}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t))$.

We are interested in the disagreement between a novel test environment $\mathbf{p}'$relative to a training environment $\mathbf{p}$ which we measure using the relative entropy between $f_\phi^{\mathbf{p}'}$ and $f_\phi^{\mathbf{p}}$ given by $KL(f_\phi^{\mathbf{p}'}||f_\phi^{\mathbf{p}})$:

$$\begin{aligned} D_{KL}(f_\phi^{\mathbf{p}'}||f_\phi^{\mathbf{p}}) =& \frac{1}{2}[\log \frac{|\Sigma(f_\phi^{\mathbf{p}})|}{|\Sigma(f_\phi^{\mathbf{p}'})|} - k + \\ & (\mu(f_\phi^{\mathbf{p}'}) - \mu(f_\phi^{\mathbf{p}'}))^T \Sigma^{-1}(f_\phi^{\mathbf{p}})(\mu(f_\phi^{\mathbf{p}'}) - \mu(f_\phi^{\mathbf{p}})) + \\ & \mathrm{Tr}\Big\{\Sigma^{-1}(f_\phi^{\mathbf{p}})\Sigma(f_\phi^{\mathbf{p}'})\Big\}] \end{aligned} \tag{4.10}$$

where k is the dimensionality of the environment's observation space; $\mathbf{s}_t \in \mathbb{R}^k$, $\Sigma(\cdot)$ is covariance and $\mu(\cdot)$ is mean and $\mathrm{Tr}(\cdot)$ is trace.

We use Negative Log Loss as a scoring rule for PNNs, and KL divergence as a measure of distribution disagreement. We utilize a thresholding technique here to detect OOTD. We train $f_\phi^{\mathbf{p}'}$ and $f_\phi^{\mathbf{p}}$ over multiple

random seeds and use the mean $KL(f_\phi^{\mathbf{p}'}||f_\phi^{\mathbf{p}})$ of the first k seeds as the threshold. Detection is then performed using:

$$\text{Is\_OOTD}(p') = \begin{cases} 1, & \text{if } KL(f_\phi^{\mathbf{p}'}||f_\phi^{\mathbf{p}}) > threshold \\ 0, & \text{otherwise} \end{cases}$$

4.4 Experiments

Our work has 2 main thrusts - the discovered *experimental behaviors* and the *Out-of-Task-Distribution Detection* obtained from the outcome of the behaviors in environments. We visualize these learnt behaviors and verify that they are indeed semantically meaningful and interpretable. We quantify the utility of the learnt behaviors to perform OOTD detection. **Causal World.** We use the Causal World Simulation [3] based on the Pybullet Physics engine to test our approach. The simulator consists of a 3-fingered robot, with 3 joints on each finger. We constrain each environment to consist of a single object that the agent can interact with. The causal factors that we manipulate for each of the objects are size and mass of the blocks and the damping factor and control frequency of the robotic motors. The simulator allows us to capture and track the positions and velocities of each of the movable objects in an environment.

Mujoco Control Suite. We also perform OOTD Detection on 3 articulated agents that try to learn locomotion - Half-Cheetah, Hopper, and Walker. For each agent type, the causal factors that we intervene on include the mass of the robot, and wind and gravity in the environment, and the friction between the robot actuators and the ground.

4.4.1 Generalized Experimental Setup

To test our approach, we train a transfer RL algorithm - in our case, Causal Curiosity [158] on multiple environments with causal factor A assuming values $A = a \in \mathbb{A}$ where $\mathbb{A}$ is a set of values causal factor A can assume. For example, we train an agent to interact with blocks of varying masses (here mass is causal

factor A). At test time, we generate a range of values of a causal factor B previously held fixed. Thus, in the above example, having been trained on varying values of mass, we now generate a range of values for the control frequency (causal factor B) of the robot actuators (previously held constant at some η). For each pair of values of (B, A) causal factors, we report the accuracy of detection over 10 random seed experiments. Thus, in the above example, all environments with $(control = \eta, mass = m)$ are considered In-Task-Distribution Transfer while all others are Out-of-Task Distribution.

Interpreting Results For each pair of test-time and training-time causal factors, we vary both over a range of values. Consider for example Figure 4.3, where we vary the mass of the blocks the robot interacts with during training and at test-time, it receives environments with a different perception frequency. During training in Figure 4.3 Pane **C**, the perception frequency was at 1 (corresponding to the column at Perception Frequency = 1). For each pair of (Control Frequency, Mass), we run our method over 10 random seeds. The value at each (x, y) position corresponds to the number of times during the 10 runs, GalilAI concluded that the test environment was OOTD. Figure 4.3 Pane **C** is an example of perfect detection - no false positives (column above $x = 1$ is zero) and 100% detection when Perception Frequency is varied. Other experiments depict varying degrees of detection success.

4.4.2 Causal World Experiments

During training, we vary either the mass or size of the block in an environment. At test time, the agent interacts with an environment with 3 possible errors - **(1) Distributional Shift of the Environment**: Change in the physical features of the block (2) **Perception Defect**: Frequency of perception changes (i.e., framerate of sensors) which affects the perception-to-action loop and **(3) Actuator Defect**: the damping coefficient of the arm actuators is varied, which affects the dynamics of the robotic arm.

Figure 4.3 depicts experiments with mass and size of blocks as training-time causal factors and Perception Frequency, Damping Factor and Mass as test-time causal factors. Each of these experiments yield no false

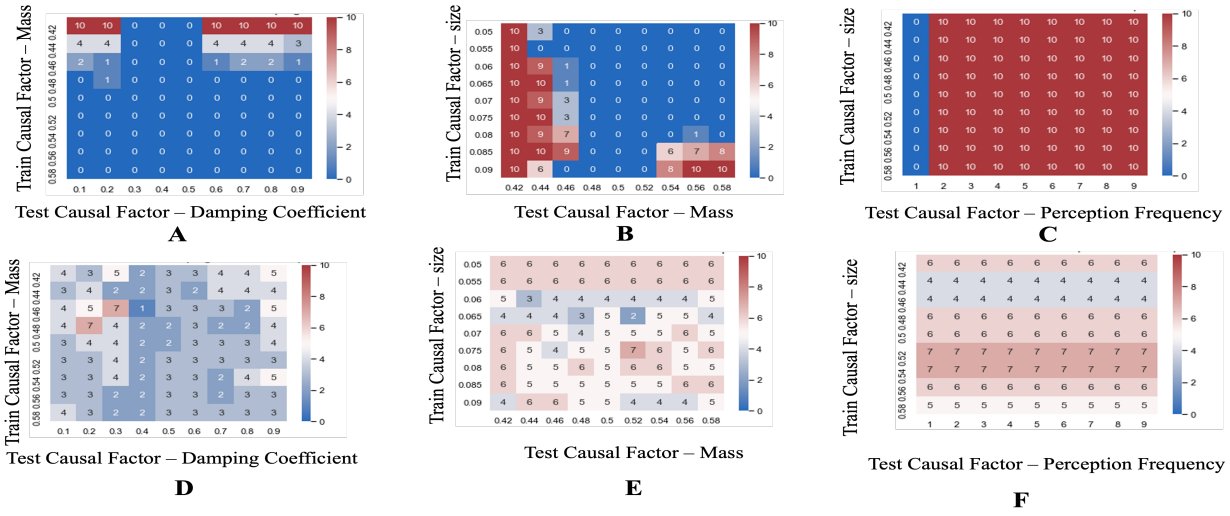


Figure 4.3: Causal World experiments. Subfigures **A** – **C** refer to GalilAI, and subfigures **D** – **F** refer to the probabilistic baseline. Each (x, y) pair on the plot corresponds to an $(unseen, seen)$ pair of causal factors. The value at each (x, y) pair depicts the performance of each method across measure performance as the number of correctly classified environments on 10 different random seeds. In-distribution value of mass is 0.5; Damping Coefficient is 0.5; Perception frequency is 1.0. Ideally, the column above the training value of the OOTD causal factor should be 0, while all other columns should be 10 as is the case in Pane **C**.

positives as the columns above in the fixed value of the test-time causal factor have zero detections. Figure 4.3 Pane **C** shows the agent has a perfect detection performance in detecting the perception defect. Figure 4.3 panes **A** and **B** show that detection is successful when the test-time value of the unseen causal factor is some distance away from its constant value during training. The agent is more likely to detect an unknown causal factor when we make a larger change to it (larger values near the left and right border).

4.4.3 Mujoco Experiments

We perform experiments with the Mujoco control suite [167] as well. During training, we manipulate the mass of the friction and the friction coefficients between the agent actuators and ground. At test time, we manipulate the wind and gravity in the environment and the mass of the agent. The in-distribution value of wind is 0.0, gravity is -9.8 and mass is 1.0. Discerning wind while being invariant to agent mass (Panes **A** and **D** in each sub-figure of Figure 4.4) is a relatively easy endeavour with the half-cheetah resulting in the highest accuracy across each of the random seeds. The task of discerning mass while being invariant

to friction also yields high accuracy of detection when the test mass varies significantly in comparison with training time mass (red columns on right and left edges of Panes **C** and **F** in Figure 4.4). However, it suffers from poor detection at $0.8\times$ and $1.2\times$ the default mass for cheetah and hopper. The hardest task is that of discerning gravity while being invariant to mass - a task that requires discovering the **Galilean Equivalence Principle**, i.e., that the acceleration due to gravity is independent of mass. While the success of GalilAI is limited when gravity is in the vicinity of 9.8, it begins to successfully learn to detect changes in gravity as it deviates from 9.8.

4.4.4 Interpretation of Learned Behaviours

For visualizations of our method, see **here.** We analyze whether the discovered experimental behaviors are actually semantically meaningful. We find that the agent is able to discover many semantically meaningful behaviors that underscore the effect of a new causal factor previously held constant during training. Chiefly, we find that the 17th century philosopher Galileo Galilei and his namesake GalilAI agree that mass and gravitational acceleration are decoupled - GalilAI learns a free-falling behavior that mimics Galileo's experiments of dropping objects to discern the gravity of an environment whilst being invariant to the agent mass.

In Mujoco Experiments, for mass as training time causal factor and wind as test-time factor, the agent learnt to use its body as a sail and allow the wind to carry it along. It also learnt to do front-flips and rolls in the direction of the wind, using the wind to help it along. For friction as training causal factor and mass as the test time causal factor, the agent also learnt to perform headstands to test out its mass while avoiding any horizontal locomotion allowing it to be invariant to the friction coefficients.

In Causal World Experiments, for size as training time causal factor and mass as test-time factor, the agent learnt a relay-kick action when one of the finger push the object to the other finger, who makes a

further push on it. This relay can only be finished on small mass blocks, thus distinguishing the causal factor.

4.5 Conclusion

In this work, we propose a novel task - that of Out-of-Task Distribution (OOTD) Detection and offer a causally inspired solution for the same. We find that simplistic extensions of existing model-based methods result in suboptimal performance with either low-detection accuracy and high false positive rate. We show the efficacy of our method in both a variety of embodied robotic environments spanning 2 simulation engines. We find GalilAI has the ability learn complex causal mechanisms and is a first step towards safer transfer/meta-RL.

While curiosity-based methods can allow one to learn behaviors without external supervision, they can be inefficient for long-horizon tasks. In such settings, demonstrations can help. The next chapter, SHERLock studies how to learn event representations from human demonstrations.

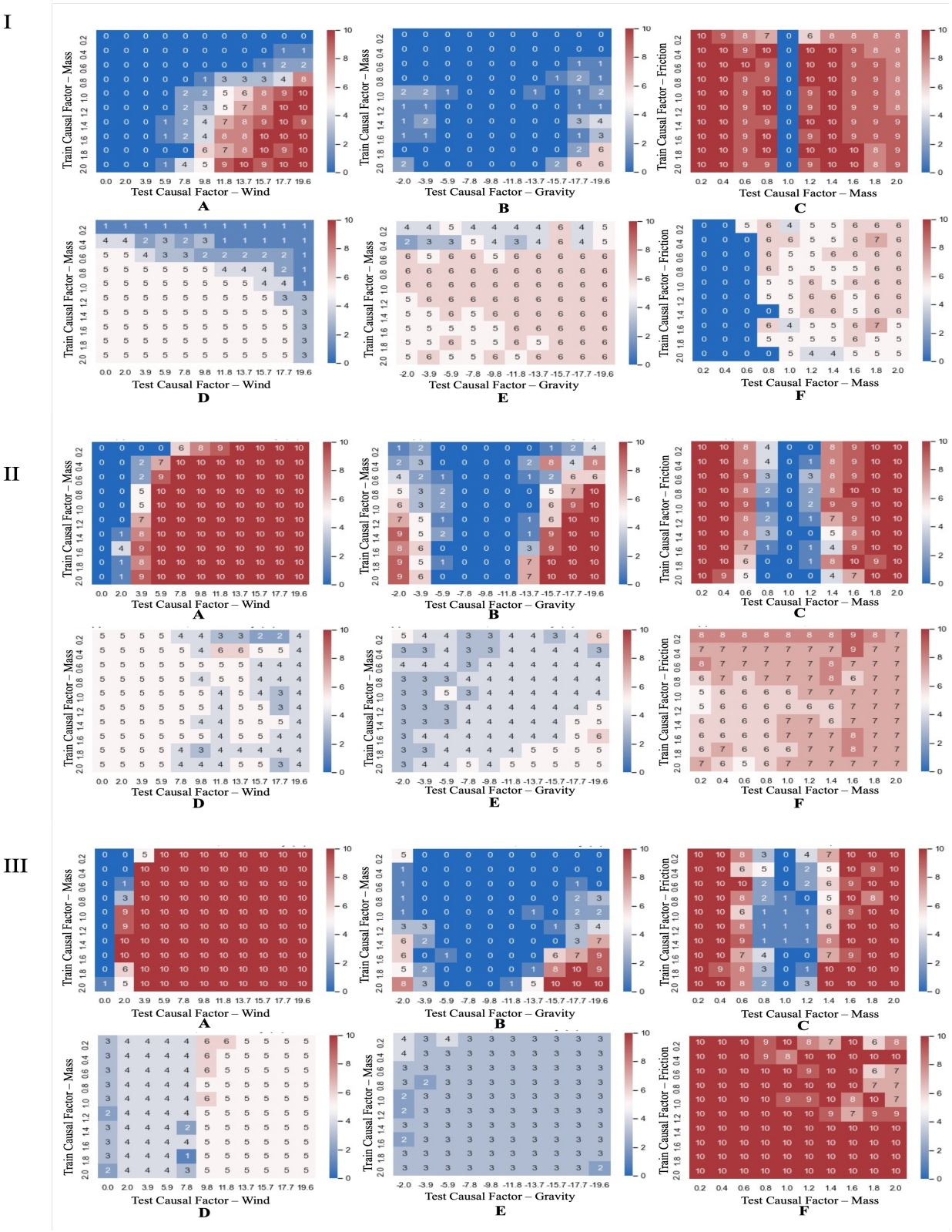


Figure 4.4: Mujoco experiments. Plots **I, II** and **III** correspond to Hopper, Walker and Cheetah environments respectively. Within each, subfigures **A** – **C** refer to GalilAI, and subfigures **D** – **F** refer to the probabilistic baseline. Each (x, y) pair on the plot corresponds to an $(unseen, seen)$ pair of causal factors. The value at each (x, y) pair depicts the performance of each method across measure performance as the number of correctly classified environments on 10 different random seeds. In-distribution value of wind is 0.0; gravity is -9.8; mass is 1.0.

Chapter 5

SHERLock: Self-Supervised Hierarchical Event Representation Learning

5.1 Introduction and Related Work

Multi-modal Event Representation learning in demonstration videos is a challenging problem. Existing methods attempt to chunk videos into events using prohibitively expensive, heavily annotated datasets containing labels for objects per frame and timestamps for activities in the video. Further, existing methods suffer from sub-optimal performance when learning event representations for long sequences. In contrast, humans excel in such scenarios - given a video demonstration (Figure 5.1) of a complex task (such as cooking), humans can subconsciously abstract events (such as boiling, frying, pouring, etc.) that succinctly encode sub-sequences in such demonstrations [121]. These events are hierarchical in nature - lower-level events are building blocks for higher-level events [113].

To learn such hierarchical events, we propose an end-to-end trainable Seq2Seq architecture, SHERLock (Self-supervised Hierarchical Event Representation Learning), for multi-modal hierarchical representation learning from demonstrations. SHERLock takes a long-horizon sequence of demonstration images (in our case, chess, tutorial, and cooking) and commentary as input. It can then isolate semantically meaningful subsequences in input trajectories. Through ablations, we show how variants of SHERLock discover meaningful subsequences using only a sequence of images. Our method *does not require* timestamps of video and commentary, nor does it need any alignment annotation between video and textual inputs.

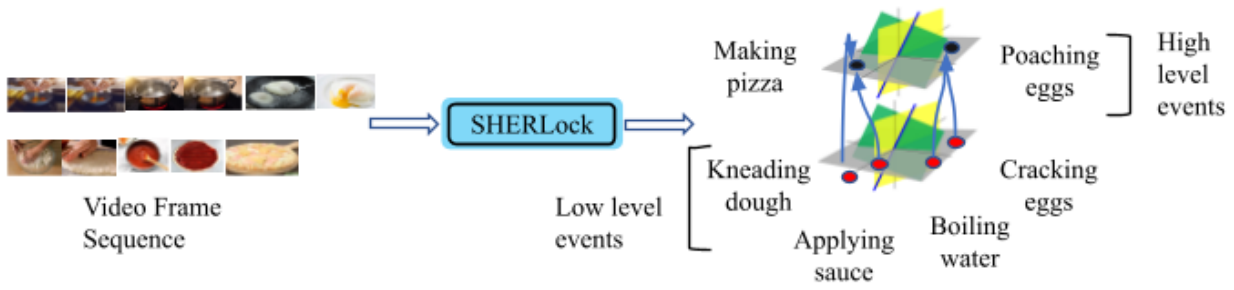


Figure 5.1: Overview of our approach. SHERLock learns a hierarchical latent space of events describing long horizon tasks like cooking using tutorial video and associated textual commentary.

We only assume the order of the events and narration are preserved in the input data. Our architecture discovers event representations along with their hierarchical organization without any supervision.

SHERLock improves upon the state-of-the-art of related work in the following ways:

1. **Self-supervised:** State-of-the-art approaches in allied fields [32, 17, 168, 104, 188, 162] (skill learning, event detection etc.) require large datasets of demonstrations, with expensive human annotations for timestamps corresponding to each event. [154] discover motor primitives from demonstrations but in a non-hierarchical fashion. SHERLock, on the other hand, abstracts hierarchical event representations from multimodal data, i.e. it divides long-horizon trajectories into a hierarchy of semantically meaningful subsequences, without requiring any temporal annotations.

2. **Long Horizon:** Long-horizon tasks remain the bane of learning systems, due to an aggregation of sub-optimal behavior over a horizon [117]. Previous works in imitation learning [142, 49, 124, 118, 5, 129], show how agents can learn representations for events in simple tasks like cart-pole from demonstrations. More recently, [143] shows that agents can learn action representation using a large corpus of observation data, i.e., trajectories of states and a relatively smaller corpus of interaction data, i.e., trajectories of state-action pairs. However, these approaches all restrict themselves to short horizons, while SHERLock is able to generate meaningful event representations for long-horizon tasks like cooking and chess.

3. **Offline abstraction:** Recent works in unsupervised skill/event discovery [50, 155, 178, 77] require costly interactions with an environment to discover skill sequences, an infeasible assumption in domains such as cooking or healthcare, where exploration is potentially dangerous. [50] learn a large number of low-level sequences of actions by forcing the agent to produce skills that are different from those previously acquired. Similarly, [155] attempt to learn skills such that their transitions are almost deterministic in a given environment. However, these approaches require access to an environment while SHERLock discovers these representations from offline demonstrations, utilizing large amounts of demonstration data.

4. **View Invariance:** SHERLock abstracts events from demonstrations of a variety of cooking tasks. The demonstration videos originate from a number of sources, varying in camera angles, instructional styles, etc. Recent works in unsupervised skill/event learning are more restrictive [154, 50, 155], requiring that demonstration data originate from a single viewpoint, with coincident state and action spaces.

5. **Multi-modality and Interpretability:** SHERLock learns a joint latent space for events utilizing both textual and visual inputs which are available in typical human demonstrations. This allows us to both visualize the physical manifestation of a temporal event, and describe in words the outcome. This is an improvement upon recent works in unsupervised skill learning, which utilize demonstrations corresponding to low dimensional state spaces and simple control signals [129, 118, 178, 77].

6. **Hierarchical:** We find that hierarchical events abstracted by SHERLock are indeed more semantically meaningful and align more closely with ground-truth annotations for events in real-world datasets (YouCook2 [185] , Chess Opening and TutorialVQA [38]) than other non-hierarchical approaches [154]. See *Non-Hierarchy* in Table 5.1.

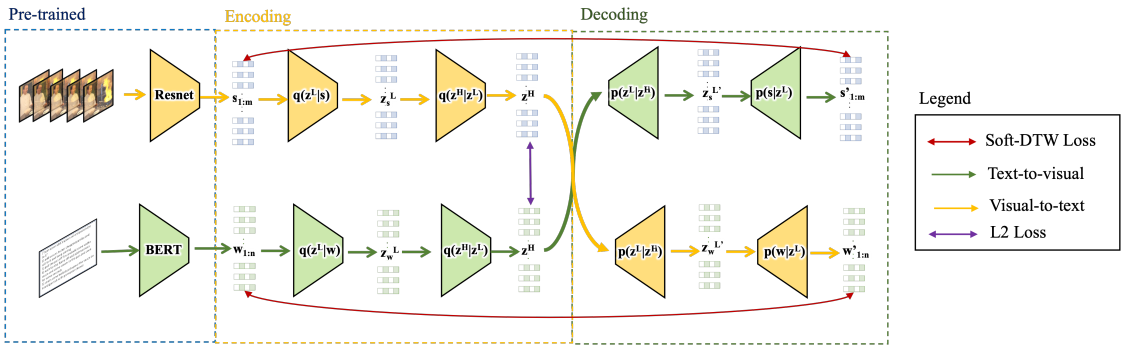


Figure 5.2: Overview of our approach. SHERLock learns a semantically-meaningful hierarchical embedding space which allows it to perform complex downstream tasks such as temporal event segmentation and label prediction. We start by encoding video and text streams into latent representations separately ($\mathbf{s}_{0:n}$ and $\mathbf{w}_{0:m}$). These are then further encoded into a hierarchy, currently consisting of low-level ($\mathbf{z}^L$) and high-level ($\mathbf{z}^H$) events. During training, we swap the representation hierarchies between video and text, such that the training loss Soft-dtw($\mathbf{s}_{0:m}, \mathbf{s}'_{0:m}$) and Soft-dtw($\mathbf{w}_{0:n}, \mathbf{w}'_{0:n}$) will align both representations.

5.2 Approach

5.2.1 Overview

We explain the motivation for SHERLock with an example in the domain of cooking demonstrations. Consider a long horizon demonstration for example, of an Eggs Benedict recipe. Here, low-level events might include boiling water or addition of eggs to water. Several such low-level events may combine to produce a high-level event - e.g., poaching an egg, which consists of boiling water, addition of egg to water, and finally removal after two minutes of cooking. SHERLock learns embeddings for such low and high-level events.

Broadly, SHERLock (Figure 5.2) can be described as a multi-modal, hierarchical, sequence-to-sequence model. The model receives as input a sequence of pre-trained ResNet-50 Embeddings ([68]), in addition to a sequence of pre-trained BERT-base [44] embeddings. The two modalities are encoded separately by two transformer models into a pair of sequences of low-level latent event embeddings (e.g., boiling water or placing eggs in water, derived from either video or text). Such low-level sequences are further encoded by another pair of transformers that generate sequences of high-level event embeddings (e.g., poaching an egg). The embedding pairs are aligned through an L2 loss, forcing both representations to correspond

to one another. Subsequently, a cross-modal decoding scheme is implemented: visual embeddings are used to re-generate word / BERT-base embeddings, while textual embeddings are used to generate video frame ResNet embeddings. After successful training, the system hence is expected to generate modality and domain invariant embeddings for temporal events. Those embeddings could subsequently be used for event classification and robotic skill learning (to be developed in future work).

5.2.2 Hierarchical Events

Intuitively, we define an event as a short sequence of states which may occur repeatedly across several demonstration trajectories. Events have an upper limit on their length in time steps. They can be obtained from both a sequence of demonstration images ($\mathbf{S} = \mathbf{s}_{0:m}$) and from the associated textual description ($\mathbf{W} = \mathbf{w}_{0:n}$). Additionally, they are hierarchical in nature - thus, low-level and high-level events representations are denoted by $\mathbf{z}^L$ and $\mathbf{z}^H$, respectively (while the following discussion is restricted to two levels, we explore the effect of more levels in Table 5.2). Given a low-level event representation, an associated sequence (of words or images) can be obtained using a decoder Φ^{x-dec}:

$$\begin{aligned} \mathbf{x}_t|\mathbf{z}_t^L &\sim \mathcal{N}(\mu_{x,t}, \sigma_{x,t}^2) \\ \text{where } [\mu_{x,t}, \sigma_{x,t}^2] &= \Phi^{x-dec}(\mathbf{z}_t^L, \mathbf{x}_{\leq t-1}) \end{aligned} \tag{5.1}$$

where $\mathbf{X} = \mathbf{x}_{0:T}$ may correspond to the flattened embedding of words $\mathbf{W}$ or images $\mathbf{S}$, and $\mathcal{N}(\cdot|\cdot)$ is a Gaussian distribution (assume prior) with parameters generated by the neural network Φ^{H-dec}. Events also exhibit a temporal hierarchy. High-level events are generated as:

$$\begin{aligned} \mathbf{z}_t^H|\mathbf{z}_{\leq t-1}^H &\sim \mathcal{N}(\mu_{H,t}, \sigma_{H,t}^2) \\ \text{where } [\mu_{H,t}, \sigma_{H,t}^2] &= \Phi^{H-dec}(\mathbf{z}_{\leq t-1}^H) \end{aligned} \tag{5.2}$$

Given such a high-level event $\mathbf{z}_t^H$, the associated sequence of low-level events can be approximated through a function Φ^{L-dec} as:

$$\begin{gathered} \mathbf{z}_t^L | \mathbf{z}_t^H, \mathbf{z}_{\leq t-1}^L \sim \mathcal{N}(\mu_{L,t}, \sigma_{L,t}^2) \\ \text{where } [\mu_{L,t}, \sigma_{L,t}^2] = \Phi^{L-dec}(\mathbf{z}_t^H, \mathbf{z}_{\leq t-1}^L) \end{gathered} \tag{5.3}$$

Thus, the resulting joint model mapped over trajectories of images $p(\mathbf{S}, \mathbf{z}^L, \mathbf{z}^H)$ factorizes as:

$$p(\mathbf{s}_0) \prod_{t=1}^{m} p(\mathbf{s}_t | \mathbf{z}_{\leq t}^L, \mathbf{s}_{<t}) p(\mathbf{z}_t^L | \mathbf{z}_{<t}^L, \mathbf{z}_{<t}^H) p(\mathbf{z}_t^H | \mathbf{z}_{<t}^H) \tag{5.4}$$

and the resulting joint model mapped over trajectories of words $p(\mathbf{W}, \mathbf{z}^L, \mathbf{z}^H)$ factorizes as:

$$p(\mathbf{w}_0) \prod_{t=1}^{n} p(\mathbf{w}_t | \mathbf{z}_{\leq t}^L, \mathbf{w}_{<t}) p(\mathbf{z}_t^L | \mathbf{z}_{<t}^L, \mathbf{z}_{<t}^H) p(\mathbf{z}_t^H | \mathbf{z}_{<t}^H) \tag{5.5}$$

The transition functions $p(\mathbf{z}_t^L | \mathbf{z}_{<t}^L, \mathbf{z}_{<t}^H)$ and $p(\mathbf{z}_t^H | \mathbf{z}_{<t}^H)$ are also learned using fixed length transformer models [172].

5.2.3 Architecture

SHERLock consists of 2 pairs of encoding transformers - one pair for each of the modalities. For a modality X, where $X \in$ set of modalities $M = \{\text{images}, \text{words}\}$, the pair of encoders consists of $q(\mathbf{z}_x^L | \mathbf{X})$, which encodes the modality X into low-level events $\mathbf{z}_x^L$ and $q(\mathbf{z}_x^H | \mathbf{z}_x^L)$ which encodes low-level events $\mathbf{z}_x^L$ into high level events $\mathbf{z}_x^H$.

$$\mathbf{z}_x^L = q(\mathbf{z}_x^L | \mathbf{X}) \text{ and } \mathbf{z}_x^H = q(\mathbf{z}_x^H | \mathbf{z}_x^L) \tag{5.6}$$

Analogously, SHERLock also contains 2 pairs of decoding transformers - one pair for each of the modalities. Decoding occurs in a cross-modal manner - textual events generate video and visual events generate text.

Thus, for a modality X, where $X \in$ set of modalities $M = \{\text{images}, \text{words}\}$, $p({\mathbf{z}'_x}^L | \mathbf{z}^H_{M-\{x\}})$ generates low-level events from high-level events of the modality $M - \{X\}$ and $p(\mathbf{x}' | {\mathbf{z}'_x}^L)$ regenerates the modality X.

$$\mathbf{z}'_x = p({\mathbf{z}'_x}^L | \mathbf{z}^H_{M-\{x\}}) \text{ and } \mathbf{x}' = p(\mathbf{x}' | {\mathbf{z}'_x}^L) \tag{5.7}$$

5.2.4 Training Metrics

5.2.4.1 Soft-Dynamic Time Warping (Soft-DTW)

Given two trajectories $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n)$ and $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_m)$, the soft-DTW$(\mathbf{x}, \mathbf{y})$ ([39]) computes the discrepancy between $\mathbf{x}$ and $\mathbf{y}$ as,

$$\text{soft-DTW}(\mathbf{x}, \mathbf{y}) = min^\gamma \{\langle A,\ \Delta(\mathbf{x}, \mathbf{y}) \rangle, A \in\ \mathcal{A}_{n,m}\} \tag{5.8}$$

where $A \in \mathcal{A}_{n,m}$ is the alignment matrix, $\Delta(\mathbf{x}, \mathbf{y}) = [\delta(\mathbf{x}_i, \mathbf{y}_i)]_{ij} \in \mathcal{R}^{n \times m}$ and δ being the cost function. min^γ operator is then computed as,

$$min^\gamma \{\mathbf{a}_1, \cdots, \mathbf{a}_n\} = \begin{cases} min_{\ i \leq n}\ \mathbf{a}_i, & \gamma = 0, \\ -\gamma \log \sum_{i=1}^n e^{-\mathbf{a}_i / \gamma}, & \gamma > 0. \end{cases} \tag{5.9}$$

For our experiments, we use L_2 distance as δ and $\gamma = 1$.

5.2.4.2 Learning Objective

We emphasize that we do not require supervision for hierarchical temporal segmentation, i.e., we do not require annotations which demarcate the beginning and ending of a event, both in language and in the

Figure 5.3: t-SNE of low-level events and their corresponding high-level mappings discovered by SHERLock on the YouCook2 dataset. We obtain clusters of low-level events such as frying, pouring while frying, seasoning etc. We also obtain two high-level events that correspond to events that require heating and those that do not.

space of frame's timestamps. Our approach uses several loss terms between network outputs to achieve our objective.

$$\begin{aligned}\mathcal{L}_{dyn} = \text{soft-DTW}(\mathbf{Z}_w^L, \mathbf{Z'}_w^L) + \text{soft-DTW}(\mathbf{Z}_s^L, \mathbf{Z'}_s^L) + \\ \text{soft-DTW}(\mathbf{S}, \mathbf{S'}) + \text{soft-DTW}(\mathbf{W}, \mathbf{W'}) \\ + \text{soft-DTW}(\mathbf{Z}_s^H, \mathbf{Z}_w^H)\end{aligned} \tag{5.10}$$

$$\mathcal{L}_{static} = L_2(\mathbf{Z}_s^H, \mathbf{Z}_w^H) + L_2(\mathbf{Z'}_s^L, \mathbf{Z'}_w^L) \tag{5.11}$$

We then define our total loss as, $\mathcal{L}_{total} = \mathcal{L}_{dyn} + \beta * \mathcal{L}_{static}$. We posit that this loss function provides the inductive bias necessary for learning the event latent space. The term soft-DTW$(\mathbf{S}, \mathbf{S'})$ ensures reconstruction of demonstration frames from the textual events, while soft-DTW$(\mathbf{W}, \mathbf{W'})$ ensures the generation of textual description from visual events. $L_2(\mathbf{Z}_s^H, \mathbf{Z}_w^H)$ and $L_2(\mathbf{Z'}_s^L, \mathbf{Z'}_w^L)$ aligns the textual and visual event spaces.

5.2.5 Evaluation Metrics

The ground-truth events in the dataset and the events generated by SHERLock may differ in number, duration, and start-time. To evaluate the efficacy of SHERLock in generating events that align with the

human-annotated events in our dataset, it is imperative that we utilize a metric that measures the overlap between generated events and ground truths and also accounts for this possible temporal mismatch.

Consider the search series $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 ... \mathbf{x}_M)$ and target series $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3 ... \mathbf{t}_N)$ where $\mathbf{X}$ corresponds to the end-of-event time stamp for each event as generated by SHERLock for a single long-horizon demonstration trajectory. Thus, the i^{th} event abstracted from SHERLock starts at time $\mathbf{x}_{i-1}$ and end at time $\mathbf{x}_i$. Similarly, T corresponds to the end-of-event time stamp for each ground-truth event in the demonstration trajectory, where the j^{th} ground truth event starts at time $\mathbf{t}_{j-1}$ and ends at time $\mathbf{t}_j$. Note that both $\mathbf{x}_0$ and $\mathbf{t}_0$ are equal to zero i.e. we measure time starting at zero for all demonstration trajectories.

To meaningfully compute the **intersection over union (IoU)** between ground truth and outputs from SHERLock, we first need to align the two representations using dynamic time warping (DTW; [12]). This implies calculating $\Delta(\mathbf{X}, \mathbf{T})$, solving the following DTW optimization problem ([12]), $\Delta(\mathbf{X}, \mathbf{T}) = \min_{\mathbf{P} \in \mathcal{P}} \sum_{\mathbf{m},\mathbf{n} \in \mathbf{P}} \delta(\mathbf{x}_m, \mathbf{t}_n)$,

where the $\mathbf{X}$ and $\mathbf{T}$ correspond to the search and target series respectively and δ corresponds to a distance metric (in our case the L_2 norm), measuring time mismatch.

$\Delta(\mathbf{X}, \mathbf{T})$ therefore corresponds to the trajectory discrepancy measure defined as the matching cost for the optimal matching path $\mathbf{P}$ among all possible valid matching paths $\mathcal{P}$ (i.e., paths satisfying monotonicity, continuity, and boundary conditions). From this optimal trajectory we can also obtain the warping function W such that $W(\mathbf{x}_i) = \mathbf{t}_j$, i.e. we find the optimal mapping between the i^{th} event ending at time $\mathbf{x}_i$ and the j^{th} event ending at time = $\mathbf{t}_j$. The resulting Intersection over Union for a single long-horizon trajectory, Time-warped IoU (TW-IoU), is:

$$\sum_{t_i} \frac{\sum_{x_j : W(x_j) = t_i} \min(t_i, x_j) - \max(t_{i-1}, x_{j-1})}{\max_{x_j : W(x_j) = t_i} \{\max(t_i, x_j)\} - \min_{x_j : W(x_j) = t_i} \{\min(t_{i-1}, x_{j-1})\}} \tag{5.12}$$

Method	TW-IoU
Non-Hierarchical [154]	14.47 ± 1.10
Non-Hierarchical w/ comment	14.84 ± 1.08
GRU Change Point Prediction	22.85 ± 0.74
Clustering (ResNet32) ([68])	31.22 ± 0.05
Clustering (HowTo100M) ([105])	32.17 ± 0.04
SHERLock-GRU w/o comment (ours)	35.99 ± 1.13
SHERLock w/o comment (ours)	39.45 ± 1.25
SHERLock w/ comment (ours)	$\mathbf{47.44 \pm 1.64}$
GRU-Supervised Segment Prediction	53.12 ± 1.09

Table 5.1: TW-IoU scores for single level events predicted by the baselines along with the TW-IoU scores for the high-level events abstracted by our proposed technique SHERLock.

5.2.6 Alignment during Inference

We calculate the DTW path [12] ($\gamma = 0$ case in eqn. (9)) between a decoded sequence and a ground truth video to obtain the optimal alignment between ground truth video frames and predicted video frames (high & low-level). This alignment is subsequently used during the calculation of the TW-IoU scores.

5.3 Experiments

Datasets: YouCook2([185]) dataset comprises of instructional videos for 89 unique food recipes. **Recommending Chess Openings**[*] dataset consists of opening moves in the game of Chess. **TutorialVQA** ([38]) consists of 76 tutorial videos pertaining to an image editing software.

5.3.1 Visualizing Hierarchy

Here we analyze whether the discovered events are human interpretable i.e. *are the temporal clusters within a single demonstration semantically meaningful?*. We find that SHERLock abstracts several useful

[*]https://www.kaggle.com/residentmario/recommending-chess-openings

Ablation Variants	TW-IoU
SHERLock w/o comment w/o L2 loss	38.46
SHERLock w/o comment w/o cross-decoding	37.67
SHERLock Single-level Decoding	20.33
SHERLock w/o comment w/o low-align loss	18.99
SHERLock Three-level Hierarchy	37.61
SHERLock w/o comment (200 Frames)	39.45
SHERLock w/o comment (64 Frames)	33.41
SHERLock w/o comment (32 Frames)	12.79

Table 5.2: TW-IoU scores for ablation experiments. Note that the reported TW-IoU scores are calculated with reference to high-level annotations available in the dataset (low-level annotations are unavailable).

Method	TW-IoU
Non Hierarchy ([154])	4.47 ± 1.12
SHERLock w/o comment (ours)	40.69 ± 1.66
SHERLock w/ comment (ours)	$\mathbf{52.66 \pm 1.72}$

Table 5.3: TW-IoU scores on the TutorialVQA dataset

human interpretable events without any supervision. See Figure 5.4 and 5.5 for results. For instance, in a pasta-making demonstration in YouCook2, a single event corresponding to the description "*heat a pan add 1 spoon oil and prosciutto to it*", is divided into low level events corresponding to "*heat pan*", "*add oil*" and "*prosciutto*". Also in Figure 5.3, a single high-level event corresponding to "*editing image text*" is divided into low level events like "*changing text color*", "*text font*", "*typekit font*", etc. Note that no explicit event time labels were provided to SHERLock, indicating that our model can abstract such coherent sub-sequences, thus taking the first step towards video understanding. Figure 5.3 shows the t-SNE [103] for the low-level event representations abstracted by SHERLock. The low-level events aggregate into clusters corresponding to frying, pouring while heating, seasoning. We also visualize the events abstracted by SHERLock when trained on chess opening data. The events learnt here also produce coherent, human-interpretable results.

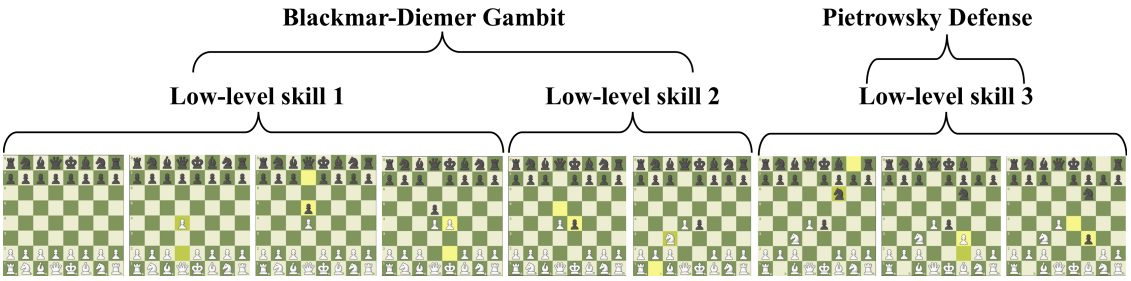


Figure 5.4: Hierarchy of events discovered by SHERLock using openings data in Chess. At a high level, SHERLock correctly identifies events corresponding to the Blackmar-Diemer Gambit and the Pietrowsky Defense. At a low level, it identifies events such as "*d4 d5..* and *e4 d3 ..*" that are used across several openings.

5.3.2 Comparison with Baselines

We evaluate the performance of SHERLock quantitatively on YouCook2 and TutorialVQA and quantify its ability to generate coherent events that align with the human annotated ground truths using the TW-IoU metric. We compare our approach with 6 baselines.

GRU Time Stamp Prediction: A supervised baseline comprising of a GRU-based encoder [33] that sequentially processes the ResNet features corresponding to frames in a video followed by a decoder GRU [8] that attends to encoder outputs and is trained to sequentially predict end-of-event timestamps of each meaningful segment (variable in number) in the video.

Non-Hierarchical w/o comment: We implement the [154] approach (SOTA in unsupervised skill learning w/o environment) which takes as input a sequence of video frames and discovers a single level of events without any hierarchy.

Non-Hierarchical w/ comment: A modified multi-modal version of Non-Hierarchical where frames and words are utilized to form a non-hierarchical latent event representation. This baseline ascertains the effect of both hierarchical and multi-modal learning on the representations obtained.

Clustering - ResNet32 Embeddings: Given an input sequence of frames, we define the weight function based on their temporal position in the sequence and also the L_2 distance between the frame embeddings. Then we use standard K-means algorithm (we find best K=4) to cluster the frames based on the weighting

function defined and use the clusters formed to predict the temporal boundaries.

Clustering - HowTo100M Embeddings: We utilize the pre-trained embeddings from the supervised action recognition dataset and method [105] and apply a K-means (we find best K=4) clustering on them.

GRU Supervised Segment Prediction: Instead of predicting end time stamps of each segment (as in GRU Time Stamp Prediction), the decoder is trained to predict/assign identical ids to frames which are part of the same segment. Further, the model's decoder is trained to assign different ids to frames part of different segments while frames not part of any meaningful segment in the ground truth are trained to have a default null id - 0.

Table 5.1 summarises and compares the TW-IoU computed between ground truth time stamp annotations and predicted/discovered segments. SHERLock achieves the highest TW-IoU when compared with all other unsupervised baselines. We find that SHERLock discovers events that align better with the ground truth events (**SHERLock performs** $\sim 23\%$ **better**) compared to Non-Hierarchical [154] performing at par with the supervised baselines.

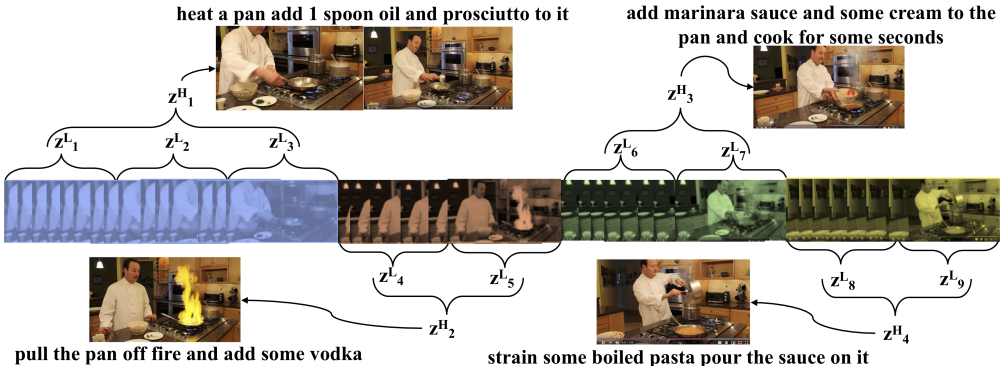


Figure 5.5: Example Hierarchy of events discovered by SHERLock on the YouCook2 dataset.

5.3.3 Ablation Experiments

Effect of Sampling Rate on the quality of hierarchy For YouCook2, we cap the length of a frame sequence to 200 frames (down-sampled from the original frames provided in the dataset due to memory

constraints). Subsequently, we analyze the trade-off between sequence length and performance. This provides an insight into granularity of information required to discover naturalistic hierarchies. Interestingly, we don't observe a linear drop in performance with a reduction in the number of frames (refer Table 5.2).

Effect of Guidance through Commentary We study the effect that language has on event discovery by comparing SHERLock without comment, which discovers event hierarchy using just frames and SHERLock which additionally uses word embeddings as a guide (as in Figure 5.2). **Language improves the TW-IoU by** $\sim 10\%$, indicating that using commentary enables SHERLock to detect more precisely the boundaries of segments corresponding to various events in a trajectory. Further, we find that the implicitly hierarchical nature of the language provides inductive bias to the model to learn a more natural hierarchy of events.

Number of Levels in Hierarchy We explore the effect of a third level of hierarchy, through additional transformers during the encoding and decoding phase. Thus, our architecture generates 16 low-level, 8 mid-level and 4 high level events. We find that this third level of event **provides only a marginal improvement** over the TW-IoU scores which we report in table 5.2. Additionally, we find that this increases the GPU memory requirements during training due to the increased number of model parameters in memory along with the additional losses (calculating Soft-DTW losses means solving a dynamic programming problem).

Model Complexity: SHERLock uses the Transformer architecture for modeling Φ and $p()$ which makes the model a bit heavy to train. So, we experiment by replacing all the transformer modules with simpler GRU modules keeping same number of layers (See SHERLock-GRU in Table 5.1). We observe that there is **not much difference** in performance ($\sim$ 3.5%). Also it still outperforms all other unsupervised baselines.

This indicates that the attention mechanism in Transformers does help us learn better representations but most of the gain can be attributed to the model architecture.

Components and Losses: We perform ablation experiments to ascertain the need for each of the modules and losses used in SHERLock. We remove the soft-DTW($\mathbf{Z}_s^L, {\mathbf{Z}'_s}^L$) loss from our SHERLock to highlight its importance in maintaining the fidelity of the reconstruction scheme. This loss guides the alignment between the encoded low-level events ($\mathbf{Z}_s^L$) and the reconstructed low-level events (${\mathbf{Z}'_s}^L$). We find that **removing this loss reduces the TW-IoU scores drastically** (see *SHERLock w/o comment w/o low-align loss* in Table 5.2).

We also evaluate a simplified version of the SHERLock w/o commentary model, where we remove the $\mathbf{Z'}^L = {\mathbf{z}'_{0:7}}^L \sim p(\mathbf{z'}^L|\mathbf{z}^H)$ modules and re-generate the word and visual sequence embeddings from the high-level events as $\mathbf{X}' = \mathbf{x}'_{0:T} \sim p(\mathbf{x}'|\mathbf{z}^H)$. We see this results in the drop of TW-IoU (Table 5.2), thus confirming our need for the step-wise encoding-and-decoding scheme used. We call this the *SHERLock Single-level Decoding* baseline.

5.4 Conclusion

In this paper, we provide a self-supervised method (SHERLock) capable of hierarchical and multi-modal learning. It can discover events and organize them in a meaningful hierarchy using only demonstration data from chess openings, tutorials, and cooking. We also show that this discovered hierarchy of events helps predict textual labels and temporal event segmentations for the associated demonstrations.

One limitation is that we can't use longer video sequences for training since computing soft-DTW requires solving a DP problem in quadratic space. Also sometimes specific nouns like *"lobster"* are replaced with more commonly appearing nouns like *"patty"*, which is due to the fact that grounding of nouns using a few images is very difficult. This could be an interesting direction for future work. Also, we would explore

curriculum learning where the discovered event hierarchy by SHERLock is used to generate curricula where lower-level events would be taught first followed by higher-level events and also be used for option discovery and training in reinforcement learning.

Learning event representations is useful, but how can we adapt them into something actionable for a robot to execute? The next chapter Video2Skill studies how to adapt event representations to robot skills.

Chapter 6

Video2Skill: Adapting Events in Demonstration Videos to Skills in an Environment using Cyclic MDP Homomorphisms

6.1 Introduction and Related Work

Offline reinforcement learning has been of substantial interest to the community with continued efforts in attempting to teach RL agents to perform tasks simply from a corpus of expert demonstration data ([96, 86, 2]). While offline RL holds promise, it is challenging because it requires making counter-factual queries under distributional shift (i.e., the agent cannot explore the effects of hypothetical action sequences not present in the training data; [96]). Additionally, current offline RL formulations make 2 constraining assumptions - first, they require **domain coincidence**, i.e., that the state and action spaces of the demonstrations and the downstream agent being trained coincide. This can be restrictive especially in applications to domains such as robotics, where expert demonstrations in the same domain may not be available. Consider, for example, attempting to teach a robot to perform medical surgery. In such a scenario, current offline RL formulations would fail as they would require a dataset of demonstrations from an expert robot performing surgery. However, generating such an expert agent using vanilla RL in a safety-critical application would be disastrous due to exploration - a classic chicken-or-egg conundrum. Instead, it is more likely that

demonstrations from a human expert would be available, and thus we need new methods to become able to exploit them.

Second, offline RL assumes **task coincidence**, i.e., it attempts to train agents to perform the same tasks as made available in the demonstration dataset, e.g., a demonstration of a robotic manipulation task, say grasping, will result in a policy that enables an agent to grasp. Combined, these assumptions mean that offline RL assumes that the MDPs on which the expert demonstrates its behavior and on which the downstream agent are trained to behave in are the same.

Hence, these assumptions make offline RL difficult to apply to practical robotic scenarios. In this work, we attempt to relax these assumptions. We utilize the large corpus of online human video tutorials of complex long-horizon tasks to teach a robotic agent to perform semantically meaningful behaviors in its own environment. Inspired by [187], our work attempts to learn adaptable short-horizon motion representations - called events - using domain randomization on human demonstrations. We then utilize a small amount of environment-specific demonstration data to adapt this latent space for domain-specific behavior.

We improve upon the state-of-the-art in the following ways:

Unsupervised Event Representation Learning: Learning temporal representations from demonstrations ([32], [17], [168]) (e.g., skill learning, event detection, etc.) typically requires large datasets of demonstrations, with expensive human annotations for timestamps corresponding to each event. Video2Skill, on the contrary, learns event representations without temporal supervision, i.e., it divides long-horizon trajectories into semantically meaningful subsequences, without access to any temporal annotations that splits these trajectories.

Domain and Task Invariance: Video2Skill abstracts events from demonstrations of a variety of cooking tasks. Additionally, these videos originate from a number of sources, varying in camera angles, instructional styles, etc. Thus, through domain randomization, our architectures generate domain

invariant event representation. Unsupervised skill learning typically has more restrictive assumptions ([154],[50],[155]) requiring that demonstration data originate from a single domain, with the same state and action spaces.

Offline and Reward-free Skill Learning: Unsupervised skill discovery ([50, 155, 178]), [77]) also typically requires costly interactions with an environment to discover skill sequences. Such assumptions can be infeasible in domains such as healthcare, where active exploration may not only be impossible, but potentially dangerous. [50] learn a large number of low-level sequences of actions by enforcing that the corpus of skills acquired is diverse. Similarly, [155] attempt to learn skills such that under a skill, subsequent transitions are almost deterministic in a given environment. Video2Skill first discovers event representations from freely available human demonstration data, and subsequently adapts them to learn environment-specific skills.

Long Horizon Learning from Demonstration: Long-horizon tasks remain the bane of decision-making algorithms, especially in the offline-learning scheme, due to an aggregation of sub-optimal behaviors over a horizon ([117]). Imitation learning ([49], [5], [142], [124], [129], [118]) has shown how agents can learn simple tasks from demonstrations. More recently, [143] shows that agents can learn to maximize external reward using a large corpus of observation data, i.e., trajectories of states, and a relatively smaller corpus of interaction data, i.e., trajectories of state-action pairs. However, such approaches are restricted to short horizons, while Video2Skill is able to generate skills for long-horizon tasks like cooking.

Multi-modal World Models: Video2Skill learns representations for events occurring free-flowing tutorial videos utilizing both textual and visual inputs which are available in typical human demonstrations. Once adapted to a specific environment, it describes the model of the environment. We show that these World Models outperform typical model ([73, 112, 94, 37]) learning methods on multi-step prediction.

Incorporating Prior Knowledge into Decision Making: We propose a adaptation based method to incorporate prior knowledge into decision making - both for model-based and model-free RL. We pre-train

a Backbone network on real world cooking data and subsequently learn environment-specific adapter functions to model dynamics of a kitchen environment. We show how the pre-training aids efficient dynamics learning and yields semantically meaningful representations.

6.2 Methods

Tutorial videos contain informative demonstrations of complex real-world tasks. These consist of humans acting as expert agents in an abstract Markov Decision Process (MDP). While we do not have access to the state and action spaces of such an abstract MDP, we do, however, have access to proxies for them through the video frames and textual commentary in the tutorial video, i.e., **state**$\rightarrow$ **action** is like **video frame**$\rightarrow$ **commentary**.

These video demonstrations consist of several events which are described in words and viewed through short sequences of video frames. We utilize such real world human demonstration data to learn environment agnostic event representations. We do this using domain randomization - by training a multi-modal temporal auto-encoder-style architecture (called **Backbone network**) on human cooking demonstrations consisting of a variety of cooking recipes and tasks. Additionally, our data comes from many sources - with a variety of camera angles, lighting, etc. The temporal autoencoder thus generates domain and task independent embeddings for sequences of videos and words. Thus, given a human demonstration of say, poaching eggs, our architecture can isolate semantically meaningful subsequences, like cracking an egg, pouring water, etc. These event representations encode both a sequence of observations in the domain of the human cooking videos, and the associated "action sequences" in the form of textual tutorial commentary.

We then utilize a small amount of demonstration data in the environment of a real robotic agent. This data consists of sequences of states and actions of an expert robot demonstrating related tasks in the environment; for example, the robot arm opening cabinet doors, etc., but not cooking. We then force the representations of these robotic demonstrations to be in same space as those of human cooking

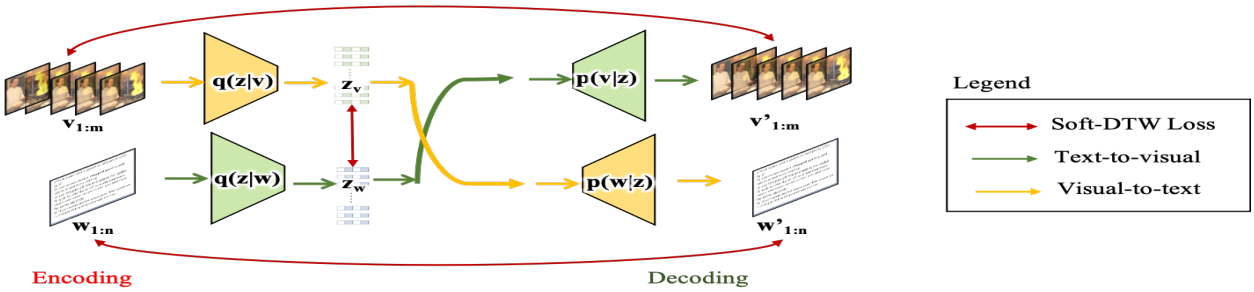


Figure 6.1: Backbone Network. Video2Skill contains a backbone temporal autoencoder which learns a semantically-meaningful embedding space, encoding events that occur in natural free-flowing tutorial videos, without explicit temporal supervision for event start and end timestamps. Section 6.2 contains details of components.

demonstrations. This is achieved using a pair of MDP Homomorphisms which map the robotic MDP to the human abstract MDP and vice-versa. The homomorphisms are learnt in a cyclical manner, thereby requiring no supervision during training. Using the homomorphisms, we can later translate cooking events (e.g., cracking an egg) into the target robotic space and *vice-versa*, resulting in zero-shot skill generation. Thus, **state**$\rightarrow$ **action**$\rightarrow$ **skill** is analogous to **video-frame**$\rightarrow$ **commentary** $\rightarrow$ **event**.

6.2.1 Event Representation Learning from Demonstration Videos

Intuitively, we define an event as a short sequence of states which may occur repeatedly across several demonstration trajectories. Events have an upper limit on their length in time steps. They can be obtained from both a sequence of demonstration images (video data) ($\mathbf{V} = \mathbf{v}_{0:m}$) and from the associated textual description ($\mathbf{W} = \mathbf{w}_{0:n}$). Given an event representation, an associated sequence (of words or images) can be obtained using a decoder Φ^{x-dec}:

$$\mathbf{x}_t|\mathbf{z}_t \sim \mathcal{N}(\mu_{x,t}, \sigma^2_{x,t}) \text{ where } [\mu_{x,t}, \sigma^2_{x,t}] = \Phi^{x-dec}(\mathbf{z}_t, \mathbf{x}_{t-1}) \tag{6.1}$$

where $\mathbf{X} = \mathbf{x}_{0:T}$ may correspond to the flattened embedding of words $\mathbf{W}$ or images $\mathbf{V}$, and $\mathcal{N}(\cdot|\cdot)$ is a Gaussian distribution (assume prior) with parameters generated by the neural network Φ^{x-dec}. Thus, the resulting joint model mapped over trajectories $p(\mathbf{X}, \mathbf{z})$ factorizes as:

$$p(\mathbf{x}_0) \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{z}_{\leq t}, \mathbf{x}_{<t}) p(\mathbf{z}_t|\mathbf{z}_{<t}) \tag{6.2}$$

The functions Φ^{x-dec} and the transition function $p(\mathbf{z}_t^H|\mathbf{z}_{<t}^H)$ are approximated by sequence-to-sequence models (in this case transformers ([172])).

Encoding: An input sequence of video frames is downsampled to 200. The visual encoder $\mathbf{Z}_v = \mathbf{z}_{0:16} \sim q(\mathbf{z}_v|\mathbf{V})$ generates a sequence of event representations such that each event $\mathbf{z}_v \in \mathbb{R}^{768}$. Similarly, textual events $\mathbf{Z}_w = \mathbf{z}_{0:16} \sim q(\mathbf{z}_w|\mathbf{W})$ are also generated using seq2seq transformer models.

Decoding: We decode in a cross-modal manner, where the events abstracted from the visual domain are used to re-generate the textual description and *vice-versa*. In what follows, prime notation refers to a re-generated value. Thus, the visual events are used to regenerate words using $\mathbf{W}' = \mathbf{w}'_{0:n} \sim p(\mathbf{w}'|\mathbf{z}_v)$ and textual events are used to subsequently regenerate demonstration frame embedding $\mathbf{V}' = \mathbf{v}'_{0:m} \sim p(\mathbf{v}'|\mathbf{z}_w)$.

Learning Objective: We emphasize that we do not require supervision for temporal segmentation, i.e., we do not require annotations which demarcate the beginning and ending of a event, both in language and in the space of video frame timestamps. Our approach uses several loss terms between network outputs to achieve our objective. The soft-DTW ([39]) is used to compute the match between two sequences of varying length. It is calculated between several sequences to generate the pre-training loss term, $\mathcal{L}_{pretrain}$.

$$\mathcal{L}_{pretrain} = \text{soft-DTW}(\mathbf{V}, \mathbf{V}') + \text{soft-DTW}(\mathbf{Z}_v, \mathbf{Z}_w) + \text{soft-DTW}(\mathbf{W}, \mathbf{W}') \tag{6.3}$$

We posit that this loss function provides the inductive bias necessary for learning the event latent space. The term soft-DTW$(\mathbf{S}, \mathbf{S}')$ ensures reconstruction of demonstration frames from the textual events, while

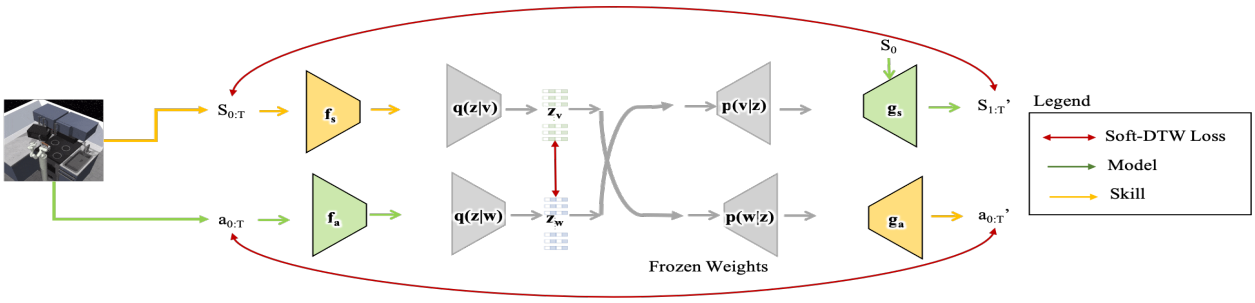


Figure 6.2: Distillation. We freeze the weights of the Backbone network and learn MDP homomorphisms from the MDP of the robotic kitchen domain to the abstract MDP of human demonstrations (g), and *vice-versa* (f), in a self-supervised manner. The latent space simultaneously contains event representations from human cooking videos and skills from the robotic domain.

soft-DTW$(\mathbf{W}, \mathbf{W}'))$ ensures the generation of textual description from visual events. soft-DTW$(\mathbf{Z}_s, \mathbf{Z}_w)$ aligns the textual and visual event spaces.

6.2.2 Skill Learning using Cyclical Homomorphisms

After pre-training on cooking videos of human demonstrations using Eq. 6.8, the weights of the encoders ($q(\mathbf{z}_v|\mathbf{v})$ and $q(\mathbf{z}_w|\mathbf{w})$) and decoders ($p(\mathbf{w}'|\mathbf{z}_v)$ and $p(\mathbf{v}'|\mathbf{z}_w)$) are frozen. Subsequently, offline demonstration (i.e., sequences of states and actions) in the robotic domain is used. This data consists of demonstrations by an expert robot performing tasks in the environment. In our case, the robot demonstration data consists of how to open a microwave oven, open cabinets, turn on the light, etc. Adapter functions f and g are then learnt which map demonstration trajectories of states and actions from a trained robot performing these tasks in the environment onto the same space of video and word embeddings as used for human cooking.

6.2.2.1 Skills and Environment Dynamics

As in [85], we define a skill as a sequence actions that may be executed in and of itself, without sensory feedback. Additionally, when a skill is applied to an environment, it results in a sequence of transitions that uniquely identifies the skill. For example, a skill which lifts an object in an environment is identified by both

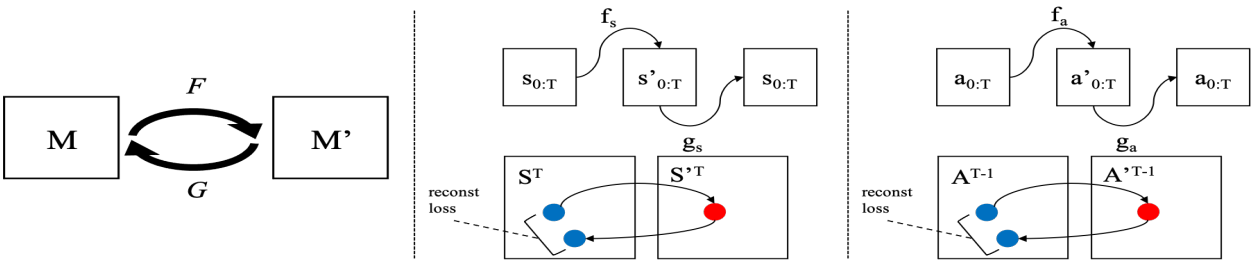


Figure 6.3: Cyclical Homomorphisms. During pre-training, Video2Skill learns an embedding space which encodes events occurring in an abstract MDP in which the human demonstrator behaves. This is done using video frames and textual commentary which serve as proxies for states and actions. Subsequently, a pair of homomorphisms map the robotic MDP into and out of the abstract MDP. These are learnt using a reconstruction loss.

the sequence of actions applied by the agent and the resultant sequence of transitions in the environment.

Thus, a latent vector $\mathbf{z}$ contextualizes both the policy (π_θ) and the subsequent model ($\mathbf{p}_\phi$):

$$\mathbf{a}_t \sim \pi_\theta(\mathbf{s}_t, \mathbf{z}_t) \text{ and } \mathbf{s}_{t+1} \sim \mathbf{p}_\phi(\mathbf{s}_t, \mathbf{z}_t) \tag{6.4}$$

Thus, the models over trajectories of states and actions in demonstration data factorize as:

$$p(\mathbf{s}_0)\prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{z}_t, \mathbf{s}_t)p(\mathbf{z}_t|\mathbf{z}_{<t}) \text{ and } p(\mathbf{a}_0)\prod_{t=1}^{T-1} p(\mathbf{a}_{t+1}|\mathbf{z}_t, \mathbf{a}_{\leq t})p(\mathbf{z}_t|\mathbf{z}_{<t}) \tag{6.5}$$

6.2.2.2 Cyclical Homomorphisms

Upon inspection, one can find that models in Eq. 6.2 and in Eq. 6.5 consist of the same structure, i.e., knowledge of the latent representation and of the history of a sequence determines the transition (with the exception of the state models in Eq. 6.5 due to the Markov assumption, where rather than history, the current state is sufficient). Consider the pair of MDPs - $\mathbf{M}$ in the robotic domain ($\mathcal{S}$, $\mathcal{A}$) with state space $\mathcal{S}$, and action space $\mathcal{A}$ and second, $\mathbf{M}'$, the abstract MDP with state space $\mathcal{S}' = \omega(\mathcal{V}) \times \xi(\mathcal{W})$ and action space $\mathcal{A}' = \Omega(\mathcal{V}) \times \Xi(\mathcal{W})$ where $\mathcal{V}$ and $\mathcal{W}$ represent the spaces of video frames and words and $\Omega(.)$,$\omega(.)$,$\xi(.)$ and $\Xi(.)$ are unknown mappings from text and videos to state and action representations (in what follows, knowing their exact form is not necessary). We exploit the shared structure between these MDPs to learn a

pair of MDP homomorphisms - from the robotic MDP to the abstract MDP and *vice-versa.* As defined in [134]:

Definition 5 (MDP Homomorphisms). *A Deterministic MDP homomorphism from an MDP* $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ *to an MDP* $\mathcal{M}' = (\mathcal{S}', \mathcal{A}', \mathcal{T}', \mathcal{R}')$ *is a tuple of functions,* $\mathcal{F} = (\mathbf{f}_s, \mathbf{f}_a)$ *with:*

- $\mathbf{f}_s : \mathcal{S} \to \mathcal{S}'$ *the state embedding function, and*
- $\mathbf{f}_a : \mathcal{A} \to \mathcal{A}'$ *the action embedding function*

such that the following identities hold:

$$\forall \mathbf{s}_t, \mathbf{s}_{t+1} \in \mathcal{S}, \mathbf{a}_t \in \mathcal{A} \text{ if } \mathbf{s}_{t+1} = \mathcal{T}(\mathbf{s}_t, \mathbf{a}_t) \text{ then: } \mathbf{f}_s(\mathbf{s}_{t+1}) = \mathcal{T}'(\mathbf{f}_s(\mathbf{s}_t), \mathbf{f}_a(\mathbf{a}_t)) \tag{6.6}$$

$$\forall \mathbf{s}_t \in \mathcal{S}, \mathbf{a}_t \in \mathcal{A}, \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t) = \mathcal{R}'(\mathbf{f}_s(\mathbf{s}_t), \mathbf{f}_a(\mathbf{a}_t)) \tag{6.7}$$

Thus we learn a pair of MDP homomorphisms - $\mathcal{F} : \mathcal{M} \to \mathcal{M}'$ and $\mathcal{G} : \mathcal{M}' \to \mathcal{M}$ such that $d(\mathcal{G}(\mathcal{F}(\mathcal{M})), \mathcal{M})$ is minimized, where $d(\cdot, \cdot) : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_+$ is a suitable distance metric in the space of MDPs.

6.2.2.3 Cyclical Homomorphic Objective

We learn each of the homomorphisms by learning the state and action embedding functions separately. This is done by freezing the weights of the encoders ($q(\mathbf{z}_v|\mathbf{v})$ and $q(\mathbf{z}_w|\mathbf{w})$) and decoders ($p(\mathbf{w}'|\mathbf{z}_v)$ and $p(\mathbf{v}'|\mathbf{z}_w)$) in the backbone network pre-trained on the human demonstrations. Subsequently, $\mathbf{f}_s$, $\mathbf{f}_a$, $\mathbf{g}_s$, and $\mathbf{g}_a$ are learnt such that $\mathbf{f}_s$ and $\mathbf{f}_a$ (i.e., the forward homomorphism) map sequences of states and actions from demonstrations in the kitchen environment into the spaces of video frames and words respectively. Next the pre-trained encoders, i.e., ($q(\mathbf{z}_v|\mathbf{v})$ and $q(\mathbf{z}_w|\mathbf{w})$), generate the latent skill vectors for these trajectories. The skill vectors are fed into the pre-trained decoders ($p(\mathbf{w}'|\mathbf{z}_v)$ and $p(\mathbf{v}'|\mathbf{z}_w)$) to regenerate the sequences in the

space of words and video frames. Finally, these sequences are fed back into the inverse MDP homomorphism ($\mathbf{g}_s$, and $\mathbf{g}_a$) to generate the original input sequence (Fig. 6.3).

$$\mathcal{L}_{distil} = \text{soft-DTW}(\mathbf{S}, \mathbf{S}') + \text{soft-DTW}(\mathbf{A}, \mathbf{A}') \tag{6.8}$$

6.3 Experiments

Our architecture is versatile, in that it results in the simultaneous learning of a latent conditioned dynamics model of the environment (green arrows in Fig. 6.2) and a latent conditioned policy or skill network (yellow arrows in Fig. 6.2). Through experiments, we present three main thrusts — **representation learning**, **dynamics learning** and **skill learning**. We study the utility of our approach in learning adaptable event representations. We show that our work results in unsupervised analogy learning of motion sequences. Subsequently, we study the ability of our latent conditioned model to utilize prior knowledge to quickly learn a long-horizon model of its environment, outperforming several state-of-the-art sophisticated baselines. Finally, we study the ability of Video2Skill to generate skills simply from human demonstration. We show that our agent performs complex motion behaviors in the robot kitchen environment akin to stirring, grasping, pouring, etc., which were demonstrated by humans in the cooking videos.

We train our backbone network on the **YouCook2** ([185]) dataset which comprises instructional videos for 89 unique recipes ($\sim$22 videos per recipe) containing labels that separate the long horizon trajectories of demonstrations into events - with explicit time stamps for the beginning and end of each event along with the associated commentary. Subsequently, we train the cyclical homomorphic objective on demonstrations from **d4RL** dataset ([55]) of the **Franka Kitchen** environment ([64]). The goal of the FrankaKitchen environment is to interact with the various objects to reach a desired state configuration. The objects the agent can interact with include the position of a kettle, flipping a light switch, opening and closing a microwave and cabinet doors, or sliding another cabinet door. The desired goal configuration for all 3 tasks

is to complete 4 subtasks: open the microwave, move the kettle, flip the light switch, and slide open the cabinet door.

6.3.1 Long Horizon Dynamics Learning

Dynamics learning, especially in an offline manner is a challenging endeavour. Dynamics learning using expressive models such as neural networks has proven to be challenging due to uncertainty stemming from insufficient data (epistemic uncertainty) and from the inherent stochasticity of an environment (aleatoric uncertainty). Further, long horizon dynamics modelling has long remained the bane of model-based reinforcement learning systems. Without an adequate model to rely upon while planning in the long horizon, model-based RL systems, although interpretable and simple have fallen behind recent advances in model-free RL. The root cause of many failures of long-horizon planning is error aggregation, i.e., sub-optimal predictions at each time step during inference results in a trajectory of states that moves increasingly further from the ground truth transitions in an environment with time.

Here, instead of inferring a long-horizon trajectory of states in an auto-regressive manner by passing actions into the model one-by-one, we propose to feed a whole sequence of actions into Video2Skill. Video2Skill decodes a sequence of skill representations from such a sequence of actions and subsequently generates the expected trajectory of resultant states conditioned on a starting state.

We compare our approach to four popular model/dynamics learning approaches currently used as state-of-the-art. As defined in [37]:

Probabilistic Neural Network (PNN): A probabilistic NN is a network whose output neurons simply parameterize a probability distribution function, capturing aleatoric uncertainty. We use the negative log prediction probability as our loss function, i.e., $loss_{PNN} = \sum_{i=1}^{N} log(\mathbf{f}_\theta(\mathbf{s}_t + 1|\mathbf{s}_t, \mathbf{a}_t))$ and choose the output distribution to be Gaussian with a diagonal covariance matrix.

Determinstic Neural Network (DNN): A deterministic NN is a special case of a probabilistic network

that outputs delta distributions centered around point predictions denoted by $\mathbf{f}_\theta(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{f}_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = Pr(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \delta(\mathbf{s}_{t+1} - \mathbf{f}_\theta(\mathbf{s}_t, \mathbf{a}_t))$. It is trained using $loss_{DNN} = \sum_{i=1}^{N} ||\mathbf{s}_{t+1} - \mathbf{f}_\theta(\mathbf{s}_t, \mathbf{a}_t))||_2$. MSE can be interpreted as $loss_{PNN}$ with a Gaussian model of fixed unit variance, but cannot be used in practice for propagation.

Ensembles - PE and DE: As in [37], we consider ensembles of B-many bootstrap models, using θ_B to refer to the parameters of our b^{th} model $\mathbf{f}_{\theta_b}$. Ensembles can consist of probabilistic NNs or deterministic NN both with effective probabilty distributions as $\mathbf{f}_\theta = \frac{1}{b}\sum_1^b \mathbf{f}_{\theta_b}$.

Video2Skill is pre-trained on the YouCook2 [186] demonstrations. Subsequently, we provide each of the baselines and Video2Skill with a dataset of demonstrations in the robot kitchen environment. Each of the baselines and Video2Skill are trained for 1, 5, and 10 epochs and subsequently evaluated on unseen data for a 2-step, 5-step and 180-step (full sequence) next-state prediction. We repeat training over 10 random seeds and report standard error across the seeds.

We find that our model is robust to long-horizon error aggregation. In Table 6.1, we compare the ability of our model to quickly adapt to dynamics of the kitchen environment when the backbone network is pre-trained on cooking videos. We find that our model outperforms all current state-of-the-art approaches in learning dynamics of the environment faster and maintaining performance over longer horizons.

6.3.2 Unsupervised Analogy Learning

Video2Skill is trained on two separate datasets - **YouCook2** and **d4RL-FrankaKitchen**. During pre-training, the agent learns environment-agnostic event representations encoding the associated sequences of video frames and textual commentary from **YouCook2**. During homomorphism learning, it learns to map sequences of states and actions in the robotic environment into the same latent space using the **d4RL-FrankaKitchen** dataset. Thus, we obtain a **shared latent space**, which contains event representations from cooking videos and kitchen environment skill representations. In Fig. 6.5, we plot a reduced dimensional

Table 6.1: Long Horizon Dynamics Learning. We study the ability of Video2Skill to learn long horizon models of an environment in an offline manner. We pre-train the Backbone networks and on the cyclical homomorphic objective for $1, 5$ and 10 epochs. We find that Video2Skill performs up to 10 times better over long-horizon sequences **(lower RMSE is better)**.

Method	2-Step	5-Step	Full Sequence
PNN: [73]			
1 Epoch	1.0065 ± 0.2719	0.6447 ± 0.0886	0.9131 ± 0.0126
5 Epoch	0.7297 ± 0.1462	0.9212 ± 0.25	0.8612 ± 0.0132
10 Epoch	0.7658 ± 0.2858	1.2529 ± 0.3738	0.6937 ± 0.0131
DNN: [112]			
1 Epoch	0.4184 ± 0.0009	0.4189 ± 0.0009	0.4502 ± 0.0013
5 Epoch	0.3897 ± 0.0027	0.407 ± 0.0016	0.4334 ± 0.0017
10 Epoch	0.1327 ± 0.0076	0.2496 ± 0.0113	0.3357 ± 0.0081
DE: [94]			
1 Epoch	0.4201 ± 0.0011	0.4205 ± 0.001	0.451 ± 0.0011
5 Epoch	0.3912 ± 0.0018	0.4078 ± 0.0013	0.4343 ± 0.0014
10 Epoch	0.1384 ± 0.0036	0.257 ± 0.0054	0.341 ± 0.0043
PE: [37]			
1 Epoch	0.5339 ± 0.0376	0.5061 ± 0.02	0.57 ± 0.0041
5 Epoch	0.5124 ± 0.0524	0.5633 ± 0.06	0.5434 ± 0.005
10 Epoch	0.3585 ± 0.0757	0.3925 ± 0.0581	0.4327 ± 0.0026
Video2Skill (ours)			
1 Epoch	0.2872 ± 0.0069	0.2946 ± 0.006	0.3077 ± 0.0012
5 Epoch	0.0818 ± 0.003	0.0675 ± 0.0023	0.0655 ± 0.0001
10 Epoch	0.0552 ± 0.003	0.052 ± 0.0037	0.0511 ± 0.0001

t-SNE plot ([171]). We then explore the overlapping latent vectors in the plot and decode them to visualize the analogies discovered by the architecture. We find that Video2Skill models motion programs successfully across domains without any supervision. The model learns to pick up on analogies between a spreading motion in the cooking videos to a horizontal sliding motion in the kitchen demonstration sequences. In other instances, it discovers analogies between circular hinge motions in the kitchen environment and circular stirring motions in the cooking videos. We emphasize, no supervision was provided to map the individual domains to one another. The cyclical pair of MDP homomorphisms resulted in an unsupervised analogy discovery.

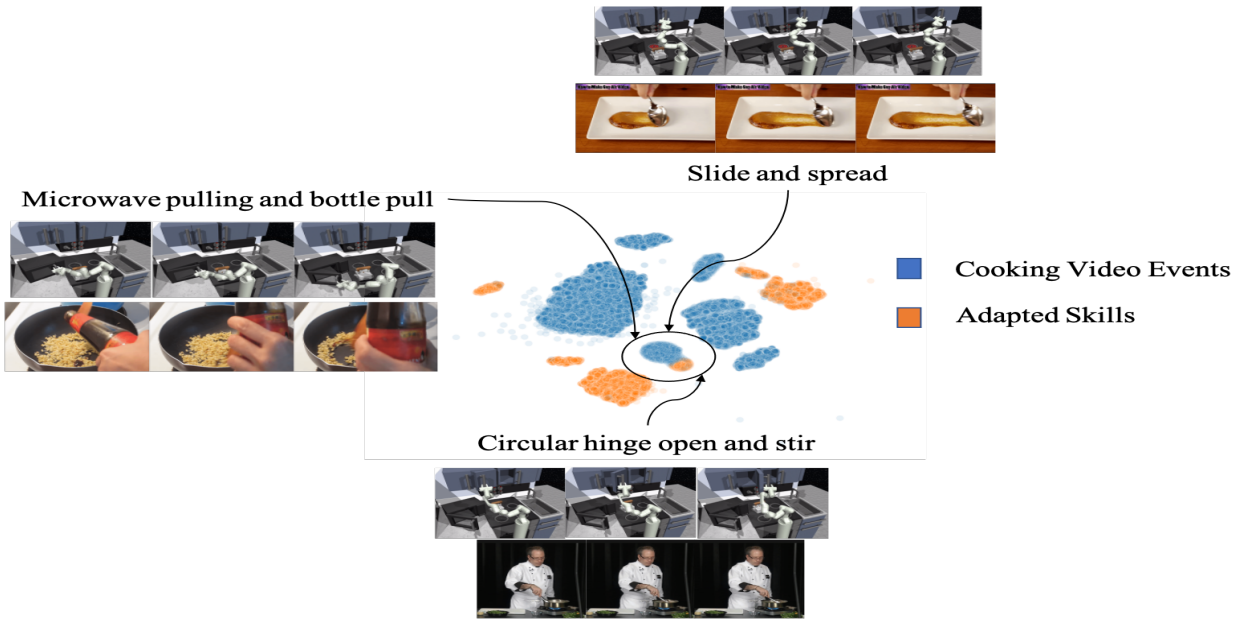


Figure 6.4: Unsupervised Analogy Learning. Using the cyclical homomorphisms, we embed events from the human cooking demonstration videos and skills from the robotic kitchen environment into the same latent space. We explore the representation learning capacity by finding overlapping regions of the latent space and exploring their semantic meaning. Video2Skill produces semantically meaningful analogies.

6.3.3 Zero-shot Skill Generation

As both video demonstration event representations and skill vectors are embedded in a **shared latent space**, we explore the ability of our architecture to generate useful and semantically meaningful skills in a zero-shot manner. To do this, we sample event representation vectors and pass them as input to the textual decoders $p(\mathbf{w}'|\mathbf{z}_v)$ and subsequently, to the action embedding of the Inverse Homomorphism $\mathcal{G}' =, \mathbf{g}_a$. The resultant actions are then applied to the environment to visualize how knowledge acquired from the cooking videos can be used to learn long horizon action sequences that are semantically meaningful.

We find that the model is able to generate complex skills that were never seen in the robotic demonstration data, but were demonstrated by humans in the cooking video data. For example, our model produces a robotic stirring motion both clockwise and counter-clockwise. Other skills include motion sequences that could be used for grasping, pouring, etc. if the robot was given extra artifacts like cups, water, etc. **This link** shows discovered skills from human demonstrations.

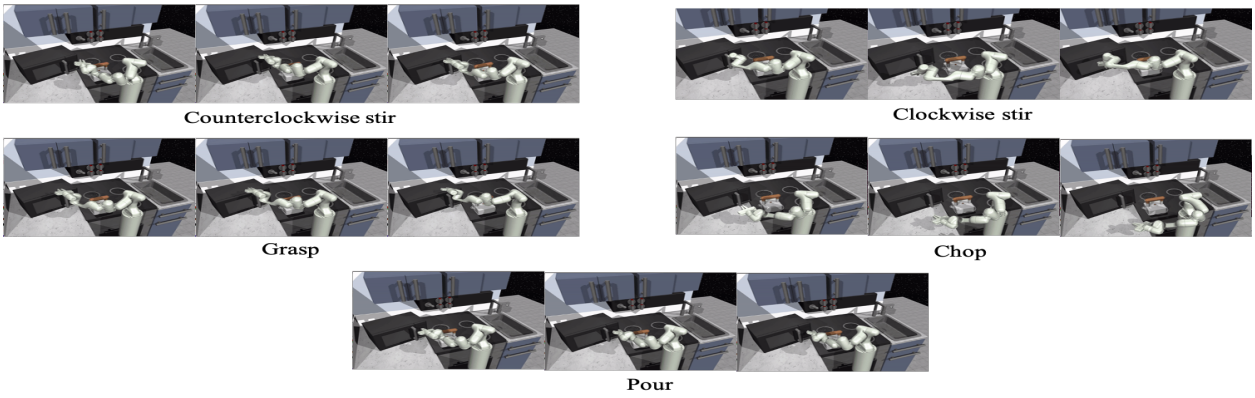


Figure 6.5: Qualitative Evaluation of Generated Skills. Video2Skill generates several significant semantically meaningful skills merely from human demonstrations. These motions are learnt in a reward free manner and can be used for complex tasks. Click **here** to view gifs of discovered skills.

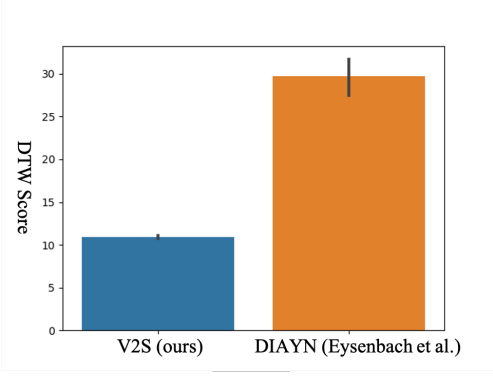


Figure 6.6: Quantitative Evaluation of Generated Skills. We study the time-warped sequence distance **(lower is better)** between demonstrations from an expert agent in the kitchen environment and the skills generated by Video2Skill. We find that Video2Skill generates skills closer to expert trajectories than [50], a state-of-the-art unsupervised skill learning approach.

6.3.4 Quantitative Skill Assesment

In Fig. 6.6, we study the utility of the skills generated by Video2Skill in being able to effectively manipulate objects and successfully complete tasks in the environment. To this end, we decode each of the generated skills from our architecture in terms of control signals and find their smallest sequence discrepancy in the demonstration data. This discrepancy calculation is performed using the Dynamic Time-Warping loss proposed by [39]. This allows us to calculate the sequence matches between 2 sequences of varying lengths. We compare the quality of our skills to those generated by DIAYN ([50]). We find that our skills are up to

three times closer to demonstration trajectories than those generated by DIAYN. We repeat experiments over 6 random seeds.

Video2Skill demonstrates how to map human demonstrations to learnable robot skills. But can we use human videos to gain a commonsense understanding of the world? The upcoming chapter, RoboGPT studies this.

6.4 Conclusion

We propose a reward-free approach to skill learning, which utilizes prior knowledge to aid decision-making in a complex environment. We show that our architecture results in powerful long-horizon models and semantically meaningful skills and uses human demonstration data to aid both. A drawback of our architecture is the size and training time (several GPU-months of training means significant energy expenditure); work towards leaner models will be beneficial. Additionally, there is still a gap between demonstration and generated skills (Fig. 6.6). Work towards bridging this gap is necessary.

Chapter 7

RoboGPT: Embedding Commonsense into Embodied Foundation Models

7.1 Introduction

Foundation Models [18, 182] have displayed impressive emergent capabilities [23, 97] bringing us ever closer to the promised land of generalist instruction-following agents. These trillion-parameter models have shown impressive sequential reasoning capabilities [174, 175, 183], code generation [180, 156], and embodied decision-making abilities [78, 21]. However, these models have remained *in silico*, interacting with the world programmatically through APIs, *never physically acting in it.* To bridge this gap, this proposal focuses on **(1) building new foundation models** and **(2) repurposing existing foundation models** for generalist-embodied decision-making in the physical world.

The fundamental resource that has delivered the early signs of generalist intelligence displayed by digital foundation models is the internet scale of data on which they are trained. Data, ironically, is also the bane of existence of *embodied* foundation models. As it turns out, acquiring some kinds of data is *much harder* than others. Gathering text data for instance is relatively easy — online interactions, textbooks, and the general human need to document have resulted in a large corpus of textual data. Gathering robot data is a different story — RT-1 [19], a pioneering foundation model for robotics utilized $\sim 10^5$ demonstrations to train, collected painstakingly over nearly 2 years by a team of robot operators. In comparison, LLAMA-2 [169], a textual foundation model utilized in the order of 10^{12} number of tokens in the training data.

Collecting the same amount of robot data at the same rate as RT-1 would take longer than humans have been on Earth! Thus, there exists an unmet need to build foundation models for embodied agents that rely on less robot-specific data and can leverage human data (i.e., text/videos/images) to learn skills.

With this data disparity in mind, in this document, I will first discuss two recent advances in foundation models for embodied decision-making which I helped develop — RT-1 [19], which was a **Best Demo Paper Finalist at RSS 2023** and RoboCLIP [160], (accepted at NeurIPS 2023) which utilizes existing foundation vision-and-language models (VLMs) and distills their knowledge into robotic policies by using them to define rewards. Finally, I will *propose* RoboGPT, a roadmap towards universally-capable embodied generalist agents that can learn from unstructured human videos.

7.2 Related Works

7.2.1 RT-1: Large Scale Behavior Cloning for Embodied AI

RT-1 is a classical behavior cloning method that utilizes a large Transformer decoder [172] to parameterize the policy π. In RT-1, we collect a large corpus of language-annotated expert observation-action demonstrations and train a transformer using supervised learning on this data to learn an optimal policy. Thus, a long horizon control problem is treated as a supervised learning problem as done in previous imitation learning works [130].

Model: Generally speaking, a Transformer is a sequence model mapping an input sequence $\{\xi\}_{h=0}^{H}$ using combinations of self-attention layers and fully-connected neural networks to an output sequence $\{y\}_{k=0}^{K}$. We parameterize π by first mapping language instruction i and observations $\{x_j\}_{j=0}^{t}$ to a sequence $\{\xi\}_{h=0}^{H}$ and the corresponding action outputs a_t to a sequence $\{y_k\}_{k=0}^{K}$. We then learn the mapping $\{\xi\}_{h=0}^{H} \rightarrow \{y_k\}_{k=0}^{K}$ using vanilla supervised learning. The observations are first encoded using a FiLM EfficientNet-B3 [164] pretrained on Imagenet [43] and the language instruction is encoded using a Universal

Sentence Encoder [27].

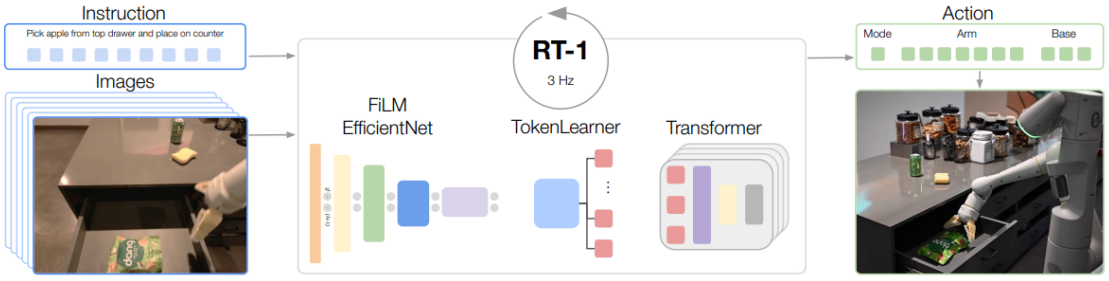


Figure 7.1: Using Out-of-Domain Videos for Reward Generation. RT-1 takes images and natural language instructions and outputs discretized base and arm actions. Despite its size (35M parameters), it does this at 3 Hz, due to its efficient yet high-capacity architecture: a FiLM [128] conditioned EfficientNet [164], a TokenLearner [139], and a Transformer [172].

Dataset: Our primary dataset consists of $\sim$ 130k robot demonstrations, collected with a fleet of 13 robots over the course of 17 months. The current set of skills includes picking, placing, opening and closing drawers, getting items in and out drawers, placing elongated items up-right, knocking them over, pulling napkins and opening jars. The skills were chosen to demonstrate multiple behaviors with many objects to test aspects of RT-1 such as generalization to new instructions and ability to perform many tasks.

Results: Our results show that RT-1 can perform over 700 training instructions at 97% success rate, and can generalize to new tasks, distractors, and backgrounds 25%, 36% and 18% better than the next best baseline, respectively. **Limitations:** The biggest limitation of RT-1 is the data collection loop. To collect a mere $\sim$ 130k trajectories, teams of robot operators worked for 17 months on a fleet of 13 robots. Thus, RT-1 needs (1) experts to (2) collect many thousand demonstrations, i.e., it requires significant expert supervision and a large dataset of demonstrations. Instead what if robots could learn from a single demonstration like humans can? What if they could also learn from demonstrations outside their own domain, the way we do by watching recipe videos shot in someone else's kitchen? What if they could learn to do a task by simply reading a DIY manual?

RoboCLIP can.

7.2.2 RoboCLIP: Imitation from One Demonstration

Given RT-1's limitations, there exists an unmet need for foundation models that 1) require very few demonstrations and 2) allow for a natural interface for providing these demonstrations. For instance, algorithms that can effectively learn from language instructions or human demonstrations. Our key insight is that by leveraging Video-and-Language Models (VLMs)—which are already pretrained on large amount of video demonstration and language pairs—we might not need to rely on large-scale and in-domain datasets. Instead, by harnessing the power of VLM embeddings, we can treat the alignment between a single *instruction's* embedding (provided as a language command or a video demonstration) and the embedding of the video of the current robot policy's rollout as a proxy reward that can potentially guide the robot towards the desired *instruction*.

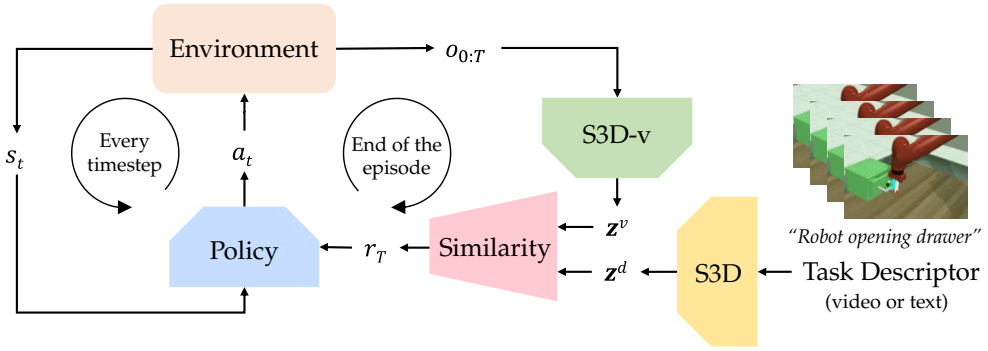


Figure 7.2: RoboCLIP Overview. A Pretrained Video-and-Language Model is used to generate rewards via the similarity score between the encoding of an episode of interaction of an agent in its environment, $\mathbf{z}^v$ with the encoding of a task specifier $\mathbf{z}^d$ such as a textual description of the task or a video demonstrating a successful trajectory. The similarity score between the latent vectors is provided as reward to the agent.

Method: RoboCLIP utilizes pretrained video-and-language models to generate rewards for online RL agents. This is done by providing a sparse reward to the agent at the end of the trajectory which describes the similarity of the agent's behavior to that of the demonstration. We utilize video-and-language models as they provide the flexibility of defining the task in terms of natural language descriptions or video demonstrations sourced either from the target robotic domain or other more naturalistic domains like human actors demonstrating the target task in their own environment. Thus, a demonstration (textual or

video) and the video of an episode of robotic interaction are embedded into the semantically meaningful latent space of S3D [177], a video-and-language model pretrained on diverse videos of human actors performing everyday tasks taken from the HowTo100M dataset [105]. The two vectors are subsequently multiplied using a scalar product generating a similarity score between the 2 vectors. This similarity score (without scaling) is returned to the agent as a reward for the episode.

Thus, at the end of the episode, the similarity score between the encoded task descriptor $\mathbf{z}^d$ and the encoded video of the episode $\mathbf{z}^v$ is used as reward $r^{\text{RoboCLIP}}(T)$. There the RoboCLIP reward is:

$$r^{\text{RoboCLIP}}(t) = \begin{cases} 0, & t \neq T \\ \mathbf{z}^d \cdot \mathbf{z}^v & t = T \end{cases}$$

where $\mathbf{z}^d \cdot \mathbf{z}^v$ corresponds to the scalar product between $\mathbf{z}^d$, the encoding of the task descriptor and $\mathbf{z}^v$, the encoding of the video of an episode of interaction of the robot in its environment.

Results: We find that RoboCLIP utilizes a single demonstration to learn policies and is able to outperform strong imitation learning baselines like FISH [67], GAIL [74] and AIRL [56] which utilize the full state action demonstration to learn a policy/reward. RoboCLIP only uses the video demonstration or textual description of the task to learn a policy providing a natural interface with the agent and allowing for domain non-experts to teach the agent. Additionally, this low data requirement means that RoboCLIP does not require fleets of robots and teams of operators for data collection.

Limitations: The biggest limitation of RoboCLIP is the requirement for online interaction. The robot must interact in the environment and is required to explore to learn a policy using the RoboCLIP reward. This is difficult to do in safety-critical situations, e.g., in robotic surgery, where exploration can be disastrous. What if robots could learn from a single demonstration like RoboCLIP, but do so in a completely offline manner?

RoboGPT just might.

7.3 RoboGPT: Future Prediction for Commonsense Understanding within Embodied Foundation Models

To recap, we would like our embodied foundation models to do the following: **(1)** learn from human data and utilize copious amounts of unstructured text and video data, **(2)** use as little robot data as possible allowing the foundation model to scale to thousands of tasks and **(3)** learn in an offline manner allowing the foundation model to be deployed in domains where online exploration is not possible. Inspired by the GPT-style [120] of pretraining which has demonstrated impressive results on language tasks, we intend to build its robotic counterpart, *RoboGPT*. RoboGPT will consist of a next-token self-supervised prediction objective that facilitates the learning of a world model. This will be followed by finetuning the model to incorporate human preferences [35].

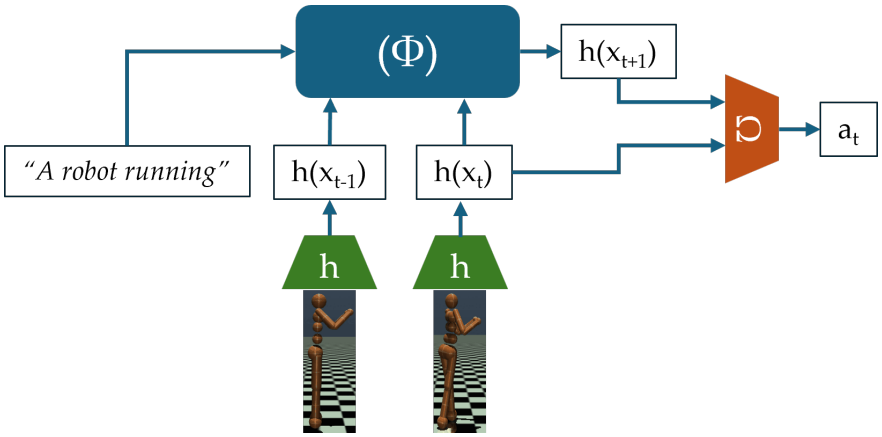


Figure 7.3: Overview of RoboGPT. **(Left)** We will first learn a language-conditioned model of the world Φ using a large corpus of language-annotated human videos. The objective for this pretraining will be a temporal next token prediction error, i.e., predicting what happens H steps into the future given the present. This dynamics model will operate on the embedding space of a pretrained vision model h. This model will be pretrained on a large amount of unstructured human video data. **(Right)** For a single demonstration of a given task the robot will learn an inverse dynamics model that maps 2 consecutive observation embeddings to the action that caused the transition.

Pretraining: We will first learn a language-conditioned model of the world Φ using a large corpus of language-annotated human videos. The objective for this pretraining will be a temporal next-token

prediction error, i.e., conditioned on a sequence of embeddings of K past observations $\{h(x_{t-k})\}_{k=K}^{0}$ and a textual description of the video i, the model will learn to predict embeddings for the next H steps into the future, i.e., $\{h(x_{t+j})\}_{j=1}^{H}$. We will utilize a pretrained image-encoder h that maps high-dimensional observations x_t into latent vectors. Specifically we will learn,

$$\{h(x_{t+j})\}_{j=1}^{H} = \Phi(\{h(x_{t-k})\}_{k=K}^{0}, i) \tag{7.1}$$

This will endow the model with knowledge about the physics of the world and result in the acquisition of general-purpose commonsense/prior about what task completion looks like.

Finetuning: Given a small amount of demonstration data for a target task, we will learn an inverse dynamics model Ω in the embedding space which maps $\{h(x_t), h(x_{t+1})\} \rightarrow a_t$ where a_t is the action taken at timestep t. Cheaply learning this model will be possible due to the fact that $h(\cdot)$ maps a high-dimensional observation x_t into a low-dimensional space. One could conceive of not needing any task-specific demonstrations and instead learning this inverse-dynamics model using data collected by a random policy.

Inference: The combined models will be used during inference whereby a sequence of previous and current observations $\{h(x_{t-1}), h(x_t)\}$ along with the instruction i describing the task will generate the embedding for the next observation $h(x_{t+1})$ through Φ. The learned inverse dynamics model Ω will take as input the embeddings for the current observation and the predicted next observation $\{h(x_t), h(x_{t+1})\}$ and produce the required action a_t.

7.4 Experiments

7.4.1 Training the Backbone

We train the backbone using a reconstruction loss in pixel space. We experiment with different trajectory lengths for training - namely predicting either the next token in video space, i.e., the next image, or predicting the next 18 images in a video conditioned on 2 input frames. The start point of the video is sampled randomly between the first frame of the video and the $len(video) - len(trajectory)$. The number 20 for the trajectory length was chosen using the mean of the video lengths in the Something-something-v2 dataset [60].

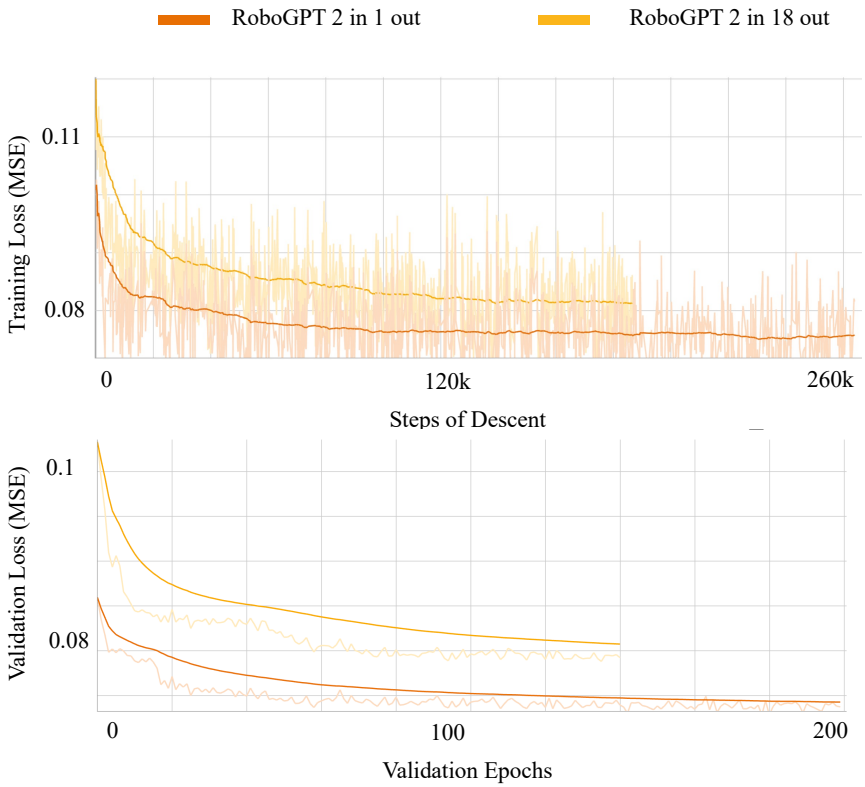


Figure 7.4: Backbone Training Curves. (Top) We train the backbone for $\sim$ 200 epochs on the Something-Something-v2 [60] Dataset. We vary the sequence length on which the method is trained. We find that the training loss remains higher for a sequence length of 20 which is expected as learning dynamics is a harder task. **(Right)** After every epoch, we perform validation on the validation split of the dataset and utilize the best model as the backbone.

The model is trained using the Something-something-v2 [60] dataset of humans interacting with everyday household objects. The data has a high diversity of objects, camera views, and backgrounds, allowing us to learn generalizable representations using the videos.

Baseline Our primary baseline is Voltron[84] which utilizes the same Something-something-v2 dataset for pretraining. Voltron uses a dual language generation and image reconstruction objective. The image reconstruction objective attempts to reconstruct masked pairs of images sampled randomly from videos in Something-something-v2. The language generation objective attempts to use masked images to generate the language description of the task the robot is attempting to perform. The representation is shown by the authors to improve downstream embodied intelligence tasks like language-conditioned imitation learning, single-task imitation learning in simulation, etc.

7.4.2 One Demonstration Behavior Cloning

We train a policy head on top of the representation generated by RoboGPT and Voltron. This consists of a 4-layer MLP with an internal representation size of 512. The representation generated is first fed into a LayerNorm [7] and then subsequently into the MLP. The output of the network is the 4-DoF action space corresponding to the robot's end effector position and gripper open/close command in the Metaworld environment [181]. We experiment with how much demonstration data is needed to learn an effective policy using each of the representations. We do not freeze the representation generated by each of the methods, but instead fine-tune the backbone and the policy head jointly on the demonstration data.

While statistically significant, we that the performance improvement is not very large with an approximate 2% task success increase across the metaworld suite. This is potentially due to the small amount of data utilized during the training of the policy (we are using only one demonstration to learn the policy). Subsequently, in the next subsection, we study how adding more demonstrations affects performance.

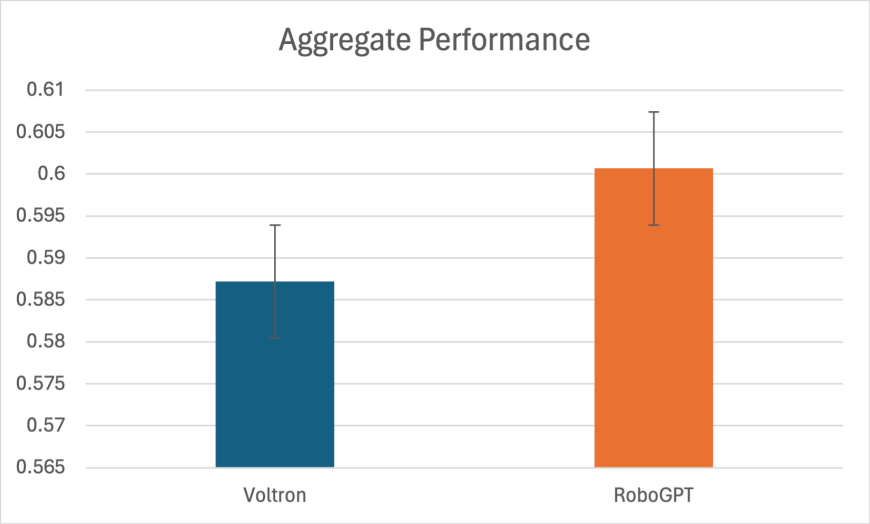


Figure 7.5: One Demonstration Imitation Learning on 31 Metaworld Environments. We train the backbone along with the policy head on one image-based demonstration trajectory for $\sim$ 3000 epochs. Subsequently, we run evaluations on 5 separate random seeds. We repeat this process over 3 random seeds. We report results aggregated on 31 Metaworld environments. RoboGPT outperforms the strongest baseline - Voltron[84] by $\sim$ 2% task success.

7.4.3 Three Demonstration Behavior Cloning

Adding more demonstrations significantly improves performance as can be seen in Figure 7.6. In this setting, we also study how performance varies by increasing the output sequence length utilized during training. We find that the backbone that yields the highest performance is the version of RoboGPT with 2 input frames and 18 output frames. This demonstrates that dynamics prediction indeed is a useful inductive bias for downstream policy learning. This is likely because a longer output sequence forces the backbone to model more dynamics and action information making the representation encode more task-relevant information than simple language-based image reconstruction.

7.5 Conclusion

RoboGPT allows us to leverage a large amount of human data available to gain a common sense understanding of the world achieving **(1)**. The learned policy maps the robot embodiment into the world model allowing for generalization to unseen tasks within a robotic embodiment. This is especially useful in settings

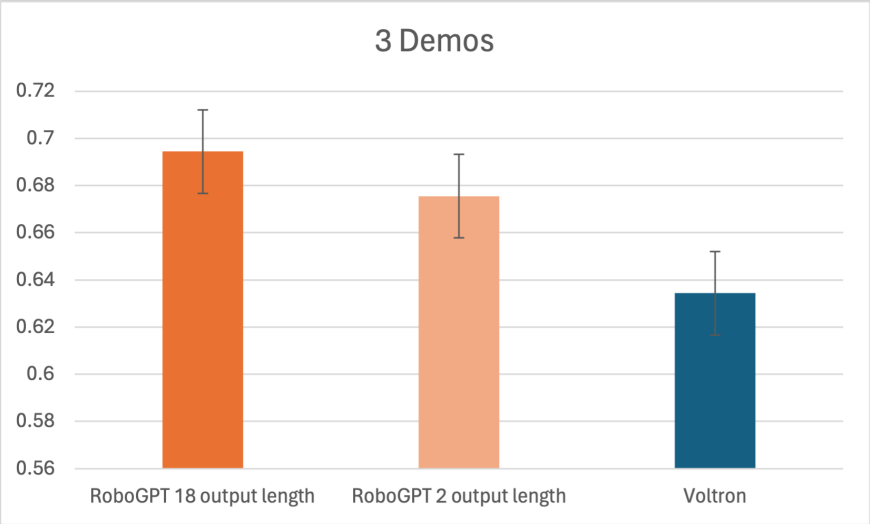


Figure 7.6: Three Demonstration Imitation Learning on 6 Metaworld Environments. We train the backbone along with the policy head on three image-based demonstration trajectories for $\sim$ 3000 epochs. Subsequently, we run evaluations on 5 separate random seeds. We repeat this process over 3 random seeds. We report results aggregated on 31 Metaworld environments. RoboGPT outperforms the strongest baseline - Voltron[84] by $\sim$ 7% task success. The gap between RoboGPT and Voltron is increased by the longer sequence length.

where collecting robot trajectories is difficult but human data is easier. For instance, whilst deploying a home robot, collecting robot trajectories in every home will be difficult, however, YouTube data with thousands of hours of human demonstrations can be utilized for policy learning using RoboGPT.

Notes on Utility and Risks: In this modern age of immersive experience, embodied agents will become ubiquitous. Humans will interact with virtual embodied agents in mixed/augmented reality scenarios and generalist agents within these worlds will become a necessity. One can imagine a scenario in which these embodied agents can learn from passive internet-scale datasets of human videos and gain some generalist capabilities using RoboGPT. Subsequently, they can learn new tasks quickly through natural language or a single demonstration using RoboCLIP through human interactions. Utilizing this large amount of human data/models trained on human data may result in biases percolating through models — a risk that we must be mindful of.

Chapter 8

Conclusions

Through this thesis, I studied the ability of sequential decision-making agents to first explore their environment in a structured and efficient way inspired by causality. Following this, I focused on how such exploration strategies can help the generalization of RL agents to test-time environments with unseen variations.

The second half of the thesis focuses on how to use human data to improve robot learning. SHERLock, Video2Skill, and RoboCLIP study how to use the wealth of human data available on the internet, in the form of video-language pairs to improve robot learning allowing robots to acquire skills from human demonstrations and/or language task descriptions. RT-1 and Q-Transformer study how to use a large dataset of teleoperated robot behaviors to train generalist robot policies.

The future of robot learning presents significant opportunities to help overcome labor shortages and improve the quality of life of aging populations. However, significant challenges remain — how to do we collect enough robot data to scalably to robot policies that can generalize in noisy unstructured, and unseen environments? How can we build commonsense mechanisms into robot policies? Finally, even if we can build generalist policies using large amounts of demonstration data, how can we make them efficient at inference and reduce the time required to sample actions?

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 1. ISBN: 1581138385. DOI: 10.1145/1015330.1015430.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. "An optimistic perspective on offline reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 104–114.

[3] Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wüthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer. "Causalworld: A robotic manipulation benchmark for causal structure and transfer learning". In: *arXiv preprint arXiv:2010.04296* (2020).

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[5] Christopher G Atkeson and Stefan Schaal. "Robot learning from demonstration". In: *ICML*. Vol. 97. Citeseer. 1997, pp. 12–20.

[6] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. M. Mooij. "On Causal and Anticausal Learning". In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*. 2012, pp. 1255–1262.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[9] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. "Autonomous navigation of stratospheric balloons using reinforcement learning". In: *Nature* 588.7836 (2020), pp. 77–82.

[10] Yoshua Bengio. "Deep learning of representations: Looking forward". In: *International conference on statistical language and speech processing*. Springer. 2013, pp. 1–37.

[11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[12] Donald J Berndt and James Clifford. “Using dynamic time warping to find patterns in time series.” In: *KDD workshop*. Seattle, WA, USA: 1994, pp. 359–370.

[13] Erdem Biyik, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. “Active Preference-Based Gaussian Process Regression for Reward Learning”. In: *Proceedings of Robotics: Science and Systems (RSS)*. July 2020. DOI: 10.15607/rss.2020.xvi.041.

[14] Erdem Biyik, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. “Asking Easy Questions: A User-Friendly Approach to Active Reward Learning”. In: *Proceedings of the 3rd Conference on Robot Learning (CoRL)*. 2019.

[15] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. “Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences”. In: *The International Journal of Robotics Research* 41.1 (2022), pp. 45–67.

[16] Thomas Blumensath and Mike Davies. “Sparse and shift-invariant representations of music”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (2005), pp. 50–57.

[17] Angie W Boggust, Kartik Audhkhasi, Dhiraj Joshi, David Harwath, Samuel Thomas, Rogério Schmidt Feris, Danny Gutfreund, Yang Zhang, Antonio Torralba, Michael Picheny, et al. “Grounding Spoken Words in Unlabeled Video.” In: *CVPR Workshops*. 2019, pp. 29–32.

[18] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. “On the opportunities and risks of foundation models”. In: *arXiv preprint arXiv:2108.07258* (2021).

[19] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. “Rt-1: Robotics transformer for real-world control at scale”. In: *arXiv preprint arXiv:2212.06817* (2022).

[20] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. “RT-1: Robotics Transformer for Real-World Control at Scale”. In: *arXiv preprint arXiv:2212.06817*. 2022.

[21] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. “Do as i can, not as i say: Grounding language in robotic affordances”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 287–318.

[22] Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. “Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 783–792.

[23] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. “Sparks of artificial general intelligence: Early experiments with gpt-4”. In: *arXiv preprint arXiv:2303.12712* (2023).

[24] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. “Understanding disentangling in β-VAE. arXiv 2018”. In: *arXiv preprint arXiv:1804.03599* (2018).

[25] Roberto Cabeza and Lars Nyberg. “Imaging cognition II: An empirical review of 275 PET and fMRI studies”. In: *Journal of cognitive neuroscience* 12.1 (2000), pp. 1–47.

[26] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[27] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. “Universal sentence encoder”. In: *arXiv preprint arXiv:1803.11175* (2018).

[28] R. Chaves, L. Luft, T.O. Maciel, D. Gross, D. Janzing, and B. Schölkopf. “Inferring latent structures via information inequalities”. In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*. Ed. by N. L. Zhang and J. Tian. Corvallis, OR: AUAI Press, 2014, pp. 112–121.

[29] Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. “Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 3909–3928.

[30] Annie S. Chen, Suraj Nair, and Chelsea Finn. *Learning Generalizable Robotic Reward Functions from "In-The-Wild" Human Videos*. 2021. arXiv: `2103.16817 [cs.RO]`.

[31] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. “Isolating sources of disentanglement in variational autoencoders”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 2610–2620.

[32] Po-Yi Chen, Alexander H Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. “Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation”. In: *CVPR*. 2019, pp. 2624–2632.

[33] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *EMNLP*. Oct. 2014.

[34] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. *Deep reinforcement learning from human preferences*. 2023. arXiv: 1706.03741 [stat.ML].

[35] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[36] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models". In: *CoRR* abs/1805.12114 (2018). arXiv: 1805.12114. URL: http://arxiv.org/abs/1805.12114.

[37] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models". In: *arXiv preprint arXiv:1805.12114* (2018).

[38] Anthony Colas, Seokhwan Kim, Franck Dernoncourt, Siddhesh Gupte, Daisy Zhe Wang, and Doo Soon Kim. "TutorialVQA: Question Answering Dataset for Tutorial Videos". In: *LREC* (2020).

[39] Marco Cuturi and Mathieu Blondel. "Soft-DTW: a differentiable loss function for time-series". In: *arXiv preprint arXiv:1703.01541* (2017).

[40] Veronica Czitrom. "One-factor-at-a-time versus designed experiments". In: *The American Statistician* 53.2 (1999), pp. 126–131.

[41] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. "Robonet: Large-scale multi-robot learning". In: *arXiv preprint arXiv:1910.11215* (2019).

[42] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. "A tutorial on the cross-entropy method". In: *Annals of operations research* 134.1 (2005), pp. 19–67.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[45] Terrance DeVries and Graham W Taylor. "Learning confidence for out-of-distribution detection in neural networks". In: *arXiv preprint arXiv:1802.04865* (2018).

[46] Finale Doshi-Velez and George Konidaris. "Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations". In: *IJCAI: proceedings of the conference*. Vol. 2016. NIH Public Access. 2016, p. 1432.

[47] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. "Bridge data: Boosting generalization of robotic skills with cross-domain datasets". In: *arXiv preprint arXiv:2109.13396* (2021).

[48] Felix Elwert. “Graphical causal models”. In: *Handbook of causal analysis for social research*. Springer, 2013, pp. 245–273.

[49] Nasser Esmaili, Claude Sammut, and GM Shirazi. “Behavioural cloning in control of a dynamic system”. In: *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*. Vol. 3. IEEE. 1995, pp. 2904–2909.

[50] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. “Diversity is all you need: Learning skills without a reward function”. In: *arXiv preprint arXiv:1802.06070* (2018).

[51] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. “MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge”. In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022.

[52] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135.

[53] Chelsea Finn, Sergey Levine, and Pieter Abbeel. “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML’16. New York, NY, USA: JMLR.org, 2016, pp. 49–58.

[54] Ronald Aylmer Fisher. “Design of experiments”. In: *Br Med J* 1.3923 (1936), pp. 554–554.

[55] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. “D4rl: Datasets for deep data-driven reinforcement learning”. In: *arXiv preprint arXiv:2004.07219* (2020).

[56] Justin Fu, Katie Luo, and Sergey Levine. “Learning robust rewards with adversarial inverse reinforcement learning”. In: *arXiv preprint arXiv:1710.11248* (2017).

[57] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. “Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[58] Yuying Ge, Annabella Macaluso, Li Erran Li, Ping Luo, and Xiaolong Wang. *Policy Adaptation from Foundation Model Feedback*. 2023. arXiv: 2212.07398 [cs.LG].

[59] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).

[60] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. “The" something something" video database for learning and evaluating visual common sense”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5842–5850.

[61] Scott T Grafton, Eliot Hazeltine, and Richard Ivry. “Functional mapping of sequence learning in normal humans”. In: *Journal of cognitive neuroscience* 7.4 (1995), pp. 497–510.

[62] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. "Ego4d: Around the world in 3,000 hours of egocentric video". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18995–19012.

[63] Peter Grunwald. "A tutorial introduction to the minimum description length principle". In: *arXiv preprint math/0406077* (2004).

[64] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning". In: *arXiv preprint arXiv:1910.11956* (2019).

[65] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. "Meta-reinforcement learning of structured exploration strategies". In: *arXiv preprint arXiv:1802.07245* (2018).

[66] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. "Inverse reward design". In: *Advances in neural information processing systems* 30 (2017).

[67] Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. "Teach a Robot to FISH: Versatile Imitation from One Minute of Demonstrations". In: *arXiv preprint arXiv:2303.01497* (2023).

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

[69] Joey Hejna and Dorsa Sadigh. "Few-Shot Preference Learning for Human-in-the-Loop RL". In: *Proceedings of the 6th Conference on Robot Learning (CoRL)*. 2022.

[70] Dan Hendrycks and Kevin Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *arXiv preprint arXiv:1610.02136* (2016).

[71] Charles Robert Hicks. "Fundamental concepts in the design of experiments". In: (1964).

[72] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In: ().

[73] Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek. "Synthesizing neural network controllers with probabilistic model-based reinforcement learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2538–2544.

[74] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in neural information processing systems* 29 (2016).

[75] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10951–10960.

[76] Hengyuan Hu and Dorsa Sadigh. “Language Instructed Reinforcement Learning for Human-AI Coordination”. In: *40th International Conference on Machine Learning (ICML)*. 2023.

[77] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. “Neural task graphs: Generalizing to unseen tasks from a single video demonstration”. In: *CVPR*. 2019.

[78] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. “Inner monologue: Embodied reasoning through planning with language models”. In: *arXiv preprint arXiv:2207.05608* (2022).

[79] Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. “DIVA: Domain Invariant Variational Autoencoders”. In: *arXiv preprint arXiv:1905.10427* (2019).

[80] D. Janzing and B. Schölkopf. “Causal inference using the algorithmic Markov condition”. In: *IEEE Transactions on Information Theory* 56.10 (2010), pp. 5168–5194.

[81] Dominik Janzing, Rafael Chaves, and Bernhard Schölkopf. “Algorithmic independence of initial condition and dynamical law in thermodynamics and causal inference”. In: *New Journal of Physics* 18.9 (2016), p. 093052.

[82] Shirin Joshi, Sulabh Kumra, and Ferat Sahin. “Robotic grasping using deep reinforcement learning”. In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2020, pp. 1461–1466.

[83] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[84] Siddharth Karamcheti, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. “Language-driven representation learning for robotics”. In: *arXiv preprint arXiv:2302.12766* (2023).

[85] Steven W Keele. “Movement control in skilled motor performance.” In: *Psychological bulletin* 70.6p1 (1968), p. 387.

[86] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. “Morel: Model-based offline reinforcement learning”. In: *arXiv preprint arXiv:2005.05951* (2020).

[87] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. “Robust and efficient transfer learning with hidden parameter markov decision processes”. In: *Advances in neural information processing systems*. 2017, pp. 6250–6261.

[88] Hyunjik Kim and Andriy Mnih. “Disentangling by factorising”. In: *arXiv preprint arXiv:1802.05983* (2018).

[89] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. “Semi-supervised learning with deep generative models”. In: *Advances in neural information processing systems*. 2014, pp. 3581–3589.

[90] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. “Stabilizing off-policy q-learning via bootstrapping error reduction”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[91] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. “When should we prefer offline reinforcement learning over behavioral cloning?” In: *arXiv preprint arXiv:2204.05618* (2022).

[92] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. “Reward Design with Language Models”. In: *International Conference on Learning Representations (ICLR)*. 2023.

[93] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[94] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *arXiv preprint arXiv:1612.01474* (2016).

[95] Kimin Lee, Laura Smith, and Pieter Abbeel. *PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training*. 2021. arXiv: 2106.05091 [cs.LG].

[96] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. 2020. arXiv: 2005.01643 [cs.LG].

[97] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. “Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models”. In: *arXiv preprint arXiv:2301.12597* (2023).

[98] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. “Enhancing the reliability of out-of-distribution image detection in neural networks”. In: *arXiv preprint arXiv:1706.02690* (2017).

[99] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[100] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. “Challenging common assumptions in the unsupervised learning of disentangled representations”. In: *arXiv preprint arXiv:1811.12359* (2018).

[101] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. “Weakly-Supervised Disentanglement Without Compromises”. In: *arXiv preprint arXiv:2002.02886* (2020).

[102] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. *Interactive Language: Talking to Robots in Real Time*. 2022. arXiv: 2210.06407 [cs.RO].

[103] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[104] Antoine Miech, Jean Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. “End-to-End Learning of Visual Representations from Uncurated Instructional Videos”. In: *CVPR* (2020).

[105] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. “Howto100m: Learning a text-video embedding by watching hundred million narrated video clips”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2630–2640.

[106] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[107] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[108] K. Muandet, D. Balduzzi, and B. Schölkopf. “Domain Generalization via Invariant Feature Representation”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. JMLR Workshop and Conference Proceedings. 2013, pp. 10–18.

[109] Vivek Myers, Erdem Biyik, Nima Anari, and Dorsa Sadigh. “Learning Multimodal Rewards from Rankings”. In: *5th Annual Conference on Robot Learning*. 2021.

[110] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. “Data-efficient hierarchical reinforcement learning”. In: *Advances in neural information processing systems* 31 (2018).

[111] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning”. In: *arXiv preprint arXiv:1803.11347* (2018).

[112] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7559–7566.

[113] Michelangelo Naim, Stefano Mikhail, and Misha Tsodyks. “Emergence of hierarchical organization in memory for random material”. In: *Scientific Reports* 9.1 (2019), pp. 1–10.

[114] Andrew Y. Ng and Stuart J. Russell. “Algorithms for Inverse Reinforcement Learning”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. ISBN: 1558607072.

[115] Hung Ngo, Matthew Luciw, Alexander Forster, and Juergen Schmidhuber. “Learning skills from play: artificial curiosity on a katana robot arm”. In: *The 2012 international joint conference on neural networks (IJCNN)*. IEEE. 2012, pp. 1–8.

[116] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. "Expanding Language-Image Pretrained Models for General Video Recognition". In: *European Conference on Computer Vision (ECCV)*. 2022.

[117] Monica N Nicolescu and Maja J Mataric. "Natural methods for robot task learning: Instructive demonstrations, generalization and practice". In: *AAMAS*. 2003, pp. 241–248.

[118] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. "Learning and generalization of complex tasks from unstructured demonstrations". In: *IROS*. 2012.

[119] Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. "Do Embodied Agents Dream of Pixelated Sheep?: Embodied Decision Making using Language Guided World Modelling". In: *arXiv preprint arXiv:2301.12050* (2023).

[120] R OpenAI. "GPT-4 technical report". In: *arXiv* (2023), pp. 2303–08774.

[121] Chandrasekhar Pammi, Bapi Krishna, Raju S, and Kenji Doya. "Chunking phenomenon in complex sequential skill learning in humans". In: *International Conference on Neural Information Processing*. Springer. 2004, pp. 294–299.

[122] Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. "Learning independent causal mechanisms". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4036–4044.

[123] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. "Actor-mimic: Deep multitask and transfer reinforcement learning". In: *arXiv preprint arXiv:1511.06342* (2015).

[124] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. "Learning and generalization of motor skills by learning from demonstration". In: *ICRA*. IEEE. 2009, pp. 763–768.

[125] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. "Curiosity-driven exploration by self-supervised prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 16–17.

[126] Judea Pearl. *Causality*. Cambridge university press, 2009.

[127] Christian F Perez, Felipe Petroski Such, and Theofanis Karaletsos. "Generalized Hidden Parameter MDPs: Transferable Model-based RL in a Handful of Trials". In: *AAAI Conference On Artifical Intelligence* (2020).

[128] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. "Film: Visual reasoning with a general conditioning layer". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[129] Jan Peters, Jens Kober, Katharina Mülling, Oliver Krämer, and Gerhard Neumann. "Towards robot skill learning: From simple skills to table tennis". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 627–631.

[130] Dean A Pomerleau. "Alvinn: An autonomous land vehicle in a neural network". In: *Advances in neural information processing systems* 1 (1988).

[131] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. "Temporal difference models: Model-free deep rl for model-based control". In: *arXiv preprint arXiv:1802.09081* (2018).

[132] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[133] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. "Efficient off-policy meta-reinforcement learning via probabilistic context variables". In: *International conference on machine learning*. PMLR. 2019, pp. 5331–5340.

[134] Balaraman Ravindran and Andrew G Barto. "Approximate Homomorphisms: A framework for non-exact minimization in Markov Decision Processes". In: (2004).

[135] Siddharth Reddy, Anca D Dragan, and Sergey Levine. "Sqil: Imitation learning via reinforcement learning with sparse rewards". In: *arXiv preprint arXiv:1905.11108* (2019).

[136] Jorma Rissanen. "Modeling by shortest data description". In: *Automatica* 14.5 (1978), pp. 465–471.

[137] Sumegh Roychowdhury, Sumedh A Sontakke, Nikaash Puri, Mausoom Sarkar, Milan Aggarwal, Pinkesh Badjatiya, Balaji Krishnamurthy, and Laurent Itti. "Unsupervised Hierarchical Concept Learning". In: *arXiv preprint arXiv:2010.02556* (2020).

[138] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115 (2015), pp. 211–252.

[139] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. "Tokenlearner: Adaptive space-time tokenization for videos". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12786–12797.

[140] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. "Active Preference-Based Learning of Reward Functions". In: *Proceedings of Robotics: Science and Systems (RSS)*. July 2017. DOI: 10.15607/RSS.2017.XIII.053.

[141] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. "Meta-learning with memory-augmented neural networks". In: *International conference on machine learning*. PMLR. 2016, pp. 1842–1850.

[142] Stefan Schaal. "Learning from demonstration". In: *Advances in neural information processing systems*. 1997, pp. 1040–1046.

[143] Karl Schmeckpeper, Annie Xie, Oleh Rybkin, Stephen Tian, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. "Learning Predictive Models From Observation and Interaction". In: *arXiv preprint arXiv:1912.12773* (2019).

[144] Jürgen Schmidhuber. “A possibility for implementing curiosity and boredom in model-building neural controllers”. In: *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*. 1991, pp. 222–227.

[145] Jürgen Schmidhuber. “Curious model-building control systems”. In: *Proc. international joint conference on neural networks*. 1991, pp. 1458–1463.

[146] Jürgen Schmidhuber. “Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts”. In: *Connection Science* 18.2 (2006), pp. 173–187.

[147] Jürgen Schmidhuber. “Formal theory of creativity, fun, and intrinsic motivation (1990–2010)”. In: *IEEE Transactions on Autonomous Mental Development* 2.3 (2010), pp. 230–247.

[148] Jürgen Schmidhuber. “Gödel machines: Fully self-referential optimal universal self-improvers”. In: *Artificial general intelligence*. Springer, 2007, pp. 199–226.

[149] B. Schölkopf. “Artificial intelligence: Learning to see and act (News & Views)”. In: *Nature* 518.7540 (2015), pp. 486–487.

[150] Bernhard Schölkopf. “Causality for machine learning”. In: *arXiv preprint arXiv:1911.10500* (2019).

[151] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[152] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. “State entropy maximization with random encoders for efficient exploration”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9443–9454.

[153] Dmitriy Serdyuk, Kartik Audhkhasi, Philémon Brakel, Bhuvana Ramabhadran, Samuel Thomas, and Yoshua Bengio. “Invariant representations for noisy speech recognition”. In: *arXiv preprint arXiv:1612.01928* (2016).

[154] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. “Discovering Motor Programs by Recomposing Demonstrations”. In: *ICLR*. 2020. URL: https://openreview.net/forum?id=rkgHY0NYwr.

[155] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. “Dynamics-aware unsupervised discovery of skills”. In: *arXiv preprint arXiv:1907.01657* (2019).

[156] Noah Shinn, Beck Labash, and Ashwin Gopinath. “Reflexion: an autonomous agent with dynamic memory and self-reflection”. In: *arXiv preprint arXiv:2303.11366* (2023).

[157] Sumedh A Sontakke, Stephen Iota, Zizhao Hu, Arash Mehrjou, Laurent Itti, and Bernhard Schölkopf. “GalilAI: Out-of-Task Distribution Detection using Causal Active Experimentation for Safe Transfer RL”. In: *arXiv preprint arXiv:2110.15489* (2021).

[158] Sumedh A Sontakke, Arash Mehrjou, Laurent Itti, and Bernhard Schölkopf. “Causal curiosity: RL agents discovering self-supervised experiments for causal representation learning”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9848–9858.

[159] Sumedh A Sontakke, Sumegh Roychowdhury, Mausoom Sarkar, Nikaash Puri, Balaji Krishnamurthy, and Laurent Itti. "Video2Skill: Adapting Events in Demonstration Videos to Skills in an Environment using Cyclic MDP Homomorphisms". In: *arXiv preprint arXiv:2109.03813* (2021).

[160] Sumedh A Sontakke, Jesse Zhang, Sebastien Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. "RoboCLIP:One Demonstration is Enough to Learn Robot Policies". In: *Unreleased and Under Submission At NeurIPS* (2023).

[161] Peter Spirtes. "Introduction to causal inference." In: *Journal of Machine Learning Research* 11.5 (2010).

[162] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. "VideoBERT: A joint model for video and language representation learning". In: *ICCV* (2019).

[163] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[164] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[165] Matthew E Taylor, Peter Stone, and Yaxin Liu. "Value functions for RL-based behavior transfer: A comparative study". In: *AAAI*. Vol. 5. 2005, pp. 880–885.

[166] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[167] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[168] Fabio Tosi, Filippo Aleotti, Pierluigi Zama Ramirez, Matteo Poggi, Samuele Salti, Luigi Di Stefano, and Stefano Mattoccia. "Distilled semantics for comprehensive scene understanding from videos". In: *CVPR*. 2020, pp. 4654–4665.

[169] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[170] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.

[171] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[172] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[173] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards". In: *arXiv preprint arXiv:1707.08817* (2017).

[174] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. "Self-consistency improves chain of thought reasoning in language models". In: *arXiv preprint arXiv:2203.11171* (2022).

[175] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24824–24837.

[176] Zhisheng Xiao, Qing Yan, and Yali Amit. "Likelihood regret: An out-of-distribution detection score for variational auto-encoder". In: *arXiv preprint arXiv:2003.02977* (2020).

[177] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 305–321.

[178] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. "Neural task programming: Learning to generalize across hierarchical tasks". In: *(ICRA)*. IEEE. 2018, pp. 1–8.

[179] Jiayu Yao, Taylor Killian, George Konidaris, and Finale Doshi-Velez. "Direct policy transfer via hidden parameter markov decision processes". In: *LLARLA Workshop, FAIM*. Vol. 2018. 2018.

[180] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. "React: Synergizing reasoning and acting in language models". In: *arXiv preprint arXiv:2210.03629* (2022).

[181] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning". In: *Conference on robot learning*. PMLR. 2020, pp. 1094–1100.

[182] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. "Florence: A new foundation model for computer vision". In: *arXiv preprint arXiv:2111.11432* (2021).

[183] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. "Multimodal chain-of-thought reasoning in language models". In: *arXiv preprint arXiv:2302.00923* (2023).

[184] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. "On learning invariant representations for domain adaptation". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7523–7532.

[185] Luowei Zhou, Chenliang Xu, and Jason J Corso. "Towards automatic learning of procedures from web instructional videos". In: *arXiv preprint arXiv:1703.09788* (2017).

[186] Luowei Zhou, Yingbo Zhou, Jason J. Corso, Richard Socher, and Caiming Xiong. "End-to-End Dense Video Captioning With Masked Transformer". In: *CVPR*. June 2018.

[187] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[188] Linchao Zhu and Yi Yang. "ActBERT: Learning Global-Local Video-Text Representations". In: *CVPR* (2020).

[189] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 9781577353683.

[190] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. "VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning". In: *arXiv preprint arXiv:1910.08348* (2019).