Towards Generalist Robots through Visual World Modeling

Boyuan Chen

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2022

Abstract

Towards Generalist Robots through Visual World Modeling

Boyuan Chen

Moving from narrow robots specializing in specific tasks to generalist robots excelling in multiple tasks in various environmental conditions is the future of next-generation robotics. The key to generalist robots is the ability to learn world models that are reusable, generalizable, and adaptable. Having a general understanding of how the physical world works will enable robots to acquire transferable knowledge across different tasks, predict possible outcomes of future actions before execution, and constantly update their knowledge through continual interactions. While the majority of robot learning frameworks tend to mix task-related and task-agnostic components altogether throughout the learning process, these two components are often not intertwined when one of them is changed. For example, a task-agnostic component such as the computational model of the robot body remains the same even under different task settings, while a task-related component such as the dynamics of a moving object remains the same for different embodiments.

This thesis studies the key steps towards building generalist robots by decomposing the world modeling problem into task-agnostic and task-related elements: (1) robot self-modeling; (2) robot modeling other agents; and (3) robot modeling the physical environment. This framework has produced powerful and efficient learning-based robotic systems for a variety of tasks and physical embodiments, such as computational models of physical robots that can be reused and adapted to numerous task objectives and changing environments, behavior modeling frameworks for complex multi-robot applications, and dynamical system understanding algorithms to distill compact physics knowledge from high-dimensional and multi-modal sensory data. The approach in this thesis could help catalyze the understanding, prediction, and control of increasingly complex systems.

Table of Contents

List of Figures

List of Tables

Acknowledgements

There are so many people that I am grateful who have made my Ph.D. journey so exciting and memorable. I will never forget their love and support.

I want to thank my advisor Prof. Hod Lipson. I sincerely think Hod is much more beyond my advisor. I have learned so much from him. It is not just about how to conduct research and teaching, but how to think like a great scientist, how to communicate science, how to advise students, how to become a leader, how to take responsibility, and how to overcome significant challenges. The list goes on. His creative mind on research and teaching always inspires me to think deeper and unconventionally. I cannot think of any place where Hod has not given a research talk. His passion on continuously challenging conventional thinking and communicating state-of-the-art science to general audiences encourages me to follow his steps. I am very proud to be his student, and I feel so happy that I can call him a friend. I have already started to miss my time with him. I will always be grateful for his guidance and support. I also wish to pass on these valuable spirits in my future career paths.

I have received countless guidance and support from faculties at Columbia. I want to thank Prof. Carl Vondrick. Carl has been a close mentor and collaborator. Carl spent plenty of time sitting with me to brainstorm crazy ideas, discuss connections of research topics from various areas, and guide me through paper writing to storytelling in many late nights before conference deadlines. I have learned a lot from him, both in small but critical details and big picture on how to become a successful researcher. Thank you for always being there to support me whenever I seek guidance and advice. Prof. Shuran Song has continuously offered valuable advice on my research and career path. She can always provide a different angle to my problem at hand and show me how to think and present research uniquely. It is enjoyable and eye-opening to hear her thoughts. Thank you also for providing open opportunities to learn from you and work with you. I am grateful to Prof. Paul Sajda for always being supportive, open-minded, and encouraging. I thank Prof Qiang Du for countless guidance on rigorous mathematical thinking and intuition in complex physics and dynamics. I sincerely appreciate my Ph.D. committee, Prof. Hod Lipson, Prof. Carl Vondrick, Prof. Shuran Song, Prof. Paul Sajda, and Prof. Tony Dear, for serving on my committee and providing valuable advice on my thesis work.

I have plenty great time with students here at Columbia. I want to thank everyone in Creative Machines Lab for being always supportive and helpful: Philippe Wyder, Boxi Xia, Yuhang Hu, Robert Kwiatkowski, Judah Goldfeder, Jonathan Blutinger, Joni Mici, Oscar Chang, Siyuan Chen, Yazmin Feliz, John Whitehead, Hayley McClintock, and Richa Batra. It has been my great pleasure to work with or mentor students on many exciting projects at Columbia: Sunand Raghupathi, Margaret Qian, Xiaotian Hu, Dave Epstein, Sara Cummings, Kuang Huang, Yu Li, Benjamin Kolber, Yiqing Liang, Ling Lan, Jack Shi, Mia Chiquier, Horus Wu, Iretiayo Akinola, Jake Varley, Lianfeng Li, Harvey Wu, Warren Mo, Dídac Surís, Ruoshi Liu, Abby Lu, Ishaan Chandratreya, Chengzhi Mao, Huy Ha, Zhenjia Xu, Zhanpeng He, Cheng Chi, Shubham Agrawal, Jingxi Xu, and Samir Gadre.

I would love to thank my wife, Yinuo, for her endless love, understanding, and support. Knowing you is the best thing that has ever happened to me. Thank you for bearing with me on my first version of the research ideas, the first version of manuscripts, and countless accompanies and beliefs. Thank you for everything that you have done for this family. You know me better than I do. I would not get this far without you.

Finally, I would love to thank my parents for their selfless love. Thank you for everything. You are my internal drive and forever role models. Thank you for never giving up on me and always believing in me. Thank you for giving me the courage and power to move forward fearlessly.

To my family.

Chapter 1: Introduction

1.1 Motivation

Despite the accelerating progress in robotics, robots today remain relatively narrow in their capabilities. Current robots are typically designed to be specialized in one particular task, such as part pick and place, navigation, and locomotion. Though robots are becoming better and better at these tasks, they are pushed towards "specialists robots" in one particular environmental setting or task setting.

Consider an assistive robot offering to open the door when both our hands are occupied. It turns out that this task is trivial for humans but extremely difficult for the current robots to do. In order to offer help like this in everyday human activities, robots need to know how to grasp and manipulate the door, recognize the need to provide help, understand the perspective of the human and take the right motion plan for safe operations. All these subtle but critical capabilities are essential for robots to work seamlessly alongside humans. In other words, robots not only need to be good at a specific task, but they also need to be good at multiple tasks in various environmental conditions. Such future robots are "generalist robots".

The gap from "specialists robots" to "generalist robots" highlights a fundamental challenge of the current robot design: instead of learning each new task from scratch, robots need to maintain a consistent belief of the world, adapt this belief to different tasks and environments, and update the belief. Therefore, the fundamental challenge is centered around how to build a **world model** and leverage the model for various task planning.

The standard paradigm for robot learning seeks to model the world by learning task-specific knowledge and task-agnostic knowledge altogether from interaction data. This paradigm inherits the widely adopted end-to-end learning framework in computer vision and natural language processing. However, robotics poses several unique challenges that make this paradigm difficult to scale in robot learning domain. First, there is not enough human power to supervise robots. Therefore, robots need to learn on their own and quickly adapt to different scenarios. Second, the world changes constantly. Furniture may be moved around and new objects may be introduced. In this case, the learned knowledge should recognize such changes and generalize to new settings. Third, there are often the presence of other agents such as humans, animals and other robots. Robots need to take into account the behavior, goals and perspectives of these other agents to plan accordingly. Finally, the world comes with complex physics and dynamics. Robots need to distill compact physics knowledge to reuse them in different future settings to be truly adaptable.

This thesis approaches the challenging world modeling problem by decomposing the problem into three major aspects aiming to learn disentangled representation of the task-agnostic models and task-specific models:

- **Robot Self-Modeling.** Robots learn what they can and cannot do. These self-models allow the robots to predict the possible outcomes of multiple future actions without trying them out in the physical reality. By learning a consistent self-model, robots can carry over and update the self-model under different changes.

- **Robot Modeling of Other Agents.** Robots learn to model the behaviors, goals and perspectives of other agents. The models of others enable robots to move and plan by taking other agents into the consideration for multi-agent settings.

- **Robot Modeling the Physical Environment.** Robots distill compact physical knowledge from noisy and high-dimensional data. These data often comes from natural and unlabelled sensory data from multiple modalities.

1.2 Thesis Outline

This thesis is structured as follows:

Chapter 2 introduces a method to scale data-driven forward-kinematics self-modeling approach to robotic platforms with complex kinematics and dynamics. In particular, this chapter introduces both the hardware and software design of an animatronic robot face with soft skin and a visual perception system. The robot can first produce a self-image to predict what itself would look like if it were going to imitate a given facial expression and then use an inverse model to produce the actual motor commands to execute. Once the self-model is acquired, the robot can imitate diverse facial expressions across multiple human subjects.

Chapter 3 presents a novel perspective on data-driven self-modeling termed query-based self-modeling. Query-based self-model learns to model the full-body robot morphology and kinematics by answering pose-conditioned space occupancy queries of the robot body. The learning data source purely comes from visual observations. Thus, this self-model is also known as a visual self-model. Such self-model can then be reused in multiple control and motion planning tasks in real-time without any retraining. When there is self-damage, this self-model can first detect the change, identify the specific type of the change, and finally recover from the change.

Chapter 4 studies how to generalize the idea of self-modeling to modeling of other agents. This chapter introduces a long-term visual predictive model to learn the behavior of another robot solely from visual inputs. Several experiments such as prediction accuracy evaluation, false-belief test and counterfactual perturbation are proposed to demonstrate the effectiveness of the proposed method on a physical robot platform.

Chapter 5 describes an approach to have one robot to further model the visual perspective of another robot, an ability known as visual perspective taking. Moving beyond behavior modeling to perspective modeling enables the robot to perform tasks that involve complex spatial reasoning among multiple agents. This chapter focuses on the task of visual hide and seek. With the predictions of what the 3D environment will look like from the moving opponent robot's point of view, the hider robot can accurately evaluate the safety level of all possible locations in the room and plan an optimal trajectory to hide.

Chapter 6 develops a method to automatically discover the fundamental state variables hidden in experimental data. While the previous four chapters have focused on modeling the embodied self and others, this chapter starts to explore how to model the physical environment by distilling compact physical knowledge from high-dimensional noisy data. The proposed two-stage approach first identifies the minimum number of variables needed to fully describe the system dynamics and then extracts the values of these variables, all directly from videos. The extracted state variables preserve significant properties for long-term dynamics predictions which is valuable for numerous robotics applications.

Chapter 7 goes beyond modeling of the physical environment with visual sensory modality and presents a learning-based method to reconstruct the visual contents of the scene from acoustic vibrations. Though visual signals have served as a major source of robot perception, other modalities such as sound are also essential for robots to perceive the rich environmental information for state estimation and decision making. This chapter presents a model to estimate the state of a dropped object from the acoustic vibrations caused by its motion.

Conclusion offers conclusion, discussion and possible future directions.

1.3 Related Works

1.3.1 Robot Self-Modeling

Robot self-modeling aims to model the computational aspects of the robot body and decouple the embodied model from task-related components. Different approaches model different aspects of robots for specific downstream tasks, such as the tilt angle of the robot [1], the position of end effectors [2], the velocity of motor joints [3], the mirror image of animatronic faces [4], or the contact locations as well as joint configurations of robot grippers [5]. These approaches are also known as data-driven forward-kinematics self models.

1.3.2 Predictive Vision

There is a substantial body of work on predictive vision [6, 7, 8, 9, 10, 11, 12, 13, 14]. The strategy presented in this paper is also inspired by the successful applications of these approaches.

Some research groups are also attempting to utilize this idea on robot motion planning [15]. In these approaches, possible future scene representations are predicted in a recurrent manner, i.e., after the next possible frame is predicted, it is fed into the input loop to predict subsequent frames. What is missing when these approaches can be effective for building Robot Theory of Mind, they must be capable of predicting both possible future actions and goals of other robots over a relatively long-term horizon.

1.3.3 Self-supervised learning

Our work uses unlabeled video to learn useful representations without manual supervision. In recent years, self-supervision, which predicts information naturally present in data by manipulating or withholding part of the input, has become a popular paradigm for unsupervised learning. Various types of self-supervised signals have been used to learn strong visual representations, such as spatial arrangement [16], contextual information [17, 18], color [19, 20], the arrow of time [21, 22, 23, 24, 25, 26], future prediction [6, 8, 11, 27], consistency in motion [28, 29], view synthesis [30, 31], spatio-temporal coherence [32, 33, 34, 35, 36], and predictive coding [27, 37]. Learned representations are then used for other downstream tasks such as image classification, object detection, video clip retrieval, and action recognition.

1.3.4 Machine Theory of Mind

Recently, there is an emerging body of work studying the behavior of machines to better understand machine behavior so as to better control it in the future [38]. Researchers have tried to build Theory of Mind that would apply to robots or machines, as this would in their view allow robots to better emulate the ability to grasp a perspective adopted by others. Robot Theory of Mind is also posited to lead to a better understanding of the meta-cognition mechanism and build machines that could exhibit human like behaviors [39].

In Human Robot Interaction, Scassellati [40] implemented functions such as face finding and face recognition for different animate and inanimate entities, while attributing them visual attention inherited from humans [41, 42]. Systems that try to detect and infer the intentions of humans have also been developed [41, 42]. Understanding the behavior of other agents by building and evolving self-models [43, 44, 45, 46, 47] is also a key step in the development of Theory of Mind pertaining specifically to robots.

Authors of recent studies [48, 49, 50, 51, 52] have also attempted to build a mind network directly for the observer robot, however all with symbolic reasoning processes. We have previously [49] conducted both simulations and real-world experiments involving a real robot that reverse-engineers the actor's policy with neural networks based on collected trajectories. However, the trajectories of the actor robot were given to the observer network directly and the definition of the final goal of the actor robot and the dynamics of the scene were not learned. Rabinowitz [48] presented a similar problem as a few-shot learning frameworks where the next step action was predicted. Nevertheless, the authors adopted discrete action spaces and experimented on a small (11×11) grid world setting. These constraints do not apply to real robotics settings and must be overcome in order to have real life applications.

1.3.5 Embodiment Navigation and Manipulation

Our work builds on research for embodied agents that learn to navigate and manipulate environments. Embodied agents with extensive training experience are increasingly able to solve a large number of problems across manipulation, navigation, and game-playing tasks [53, 54, 55, 56, 57, 58, 59, 60, 61]. Extensive work has demonstrated that, after learning with indirect supervision from a reward function, rich representations for their task automatically emerge [62, 63, 64, 60, 65, 66]. Several recent works have created 3D embodiment simulation environment [67, 68, 69, 70, 71] for navigation and visual question answering tasks. To train these models, visual navigation is often framed as a reinforcement learning problem [72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 57]. Moreover, by incorporating multiple embodied agents into the environment, past work has explored how to learn diverse strategies and behaviors in multi-agent visual navigation tasks [65, 84]. For a full review of multi-agent reinforcement learning, please see [85, 86, 87, 88, 89].

1.3.6 Sound in Robotics

Most related research to our work is on audiovisual object understanding for robot perception and control. [90] investigates vision and sound in a robot setting to predict which robot actions caused the given sound. [91] leverages audio signals from ball bouncing motions to calibrate the stochastic dynamical events for “sim2real” tasks. [92] predicts the trajectory of a falling cube for robot object retrieval outside visible region. Instead of predicting the 2D trajectory for a single cube, this paper learns to predict the entire visual scene under fully occluded conditions by outputting RGB and depth images on three different blocks.

Chapter 2: Robot Forward-Kinematics Self-Modeling

Building computational self-models of robot bodies, or the ability of a robot to simulate its physical self, is an essential requirement for robot motion planning and control. Similar to humans and animals [93, 94], robots can use self-models to anticipate future outcomes of various motion plans without explicitly trying them out in the physical world. Predictions obtained using a self-model can be utilized in decision criteria for future actions. Importantly, a consistent self-model, once acquired, can be re-purposed for many different tasks and thus can serve for lifelong learning.

Most available robotic systems rely on dedicated physical simulators for task planning and control [95, 96, 97, 98, 99, 100]. Yet, these simulators require extensive human effort to develop, calibrate and maintain over the lifetime of the robot. In contrast, fully data-driven self-modeling enables machines to learn their forward kinematics directly in situ using task-agnostic interaction data.

In this chapter, we will introduce our efforts [4, 101] on driving an animatronic robot face with a soft skin and complex kinematics and dynamics with data-driven forward-kinematics self-modeling. The majority of past works on forward-kinematics self-modeling focus only on rigid body robots. However, directly scaling the previous approaches to hybrid mechanical embodiment do not work. For example, the state of the soft skin of the robot cannot be easily defined as 6-DoF poses. Therefore, a more general data-driven forward-kinematics self-modeling approach that works across both rigid body and soft body robots require extensive re-design of the learning algorithms. Here, we propose to use the mirror image of a robot, known as the “self-image” of the robot, as the intermediate representation to actuate an animatronic robot face to mimic diverse human facial expressions across various human subjects without any human supervision during training.

2.1 Background

Facial expressions are an essential aspect of nonverbal communication. In our day-to-day lives, we rely on diverse facial expressions to convey our feelings and attitudes to others and interpret other people's emotions, desires and intentions [102]. Facial mimicry [103, 104, 105, 106] is also recognized as a vital stepping stone towards the early development of social skills for infants. Therefore, building robots that can automatically mimic diverse human facial expressions [107, 108, 109, 110] will facilitate more natural robotic social behaviors and further encourage stronger engagement in human-robot interactions. Mimicking human facial expressions is also the first step towards achieving adaptive facial reactions in robots. Despite the practical value of such systems, extant research in this domain mostly focuses on the hardware design and pre-programmed facial expressions, allowing robots to select one of the facial expressions from a predefined set. Generalizing across various human expressions has remained challenging.

Current robotic face systems cannot mimic human facial expressions adaptively. The key limitation is the lack of a general learning framework that can learn from limited human supervision. Some traditional methods [111, 112, 113, 114, 115, 116, 117] define a set of pre-specified facial expressions.

Others generalize this process to search for closest match from a database [118] or by following an fitness function [118, 119]. However, as human expressions are highly diverse, these approaches have limited value in practical robot-human interactions.

In this work, we present Eva 2.0 (Fig. 2.1) with significant upgrades to our previous Eva 1.0 [101] platform with more flexible and stable control. We further propose a general learning-based framework to learn facial mimicry from visual observations that can generalize well to different human subjects and diverse expressions. Importantly, our approach does not rely on human supervisions to provide ground-truth robot commands. Our key idea is to decompose the problem into two stages: (1) given normalized human facial landmarks, we first use a generative model to synthesize a corresponding robot self-image with the same facial expression and then (2) leverage

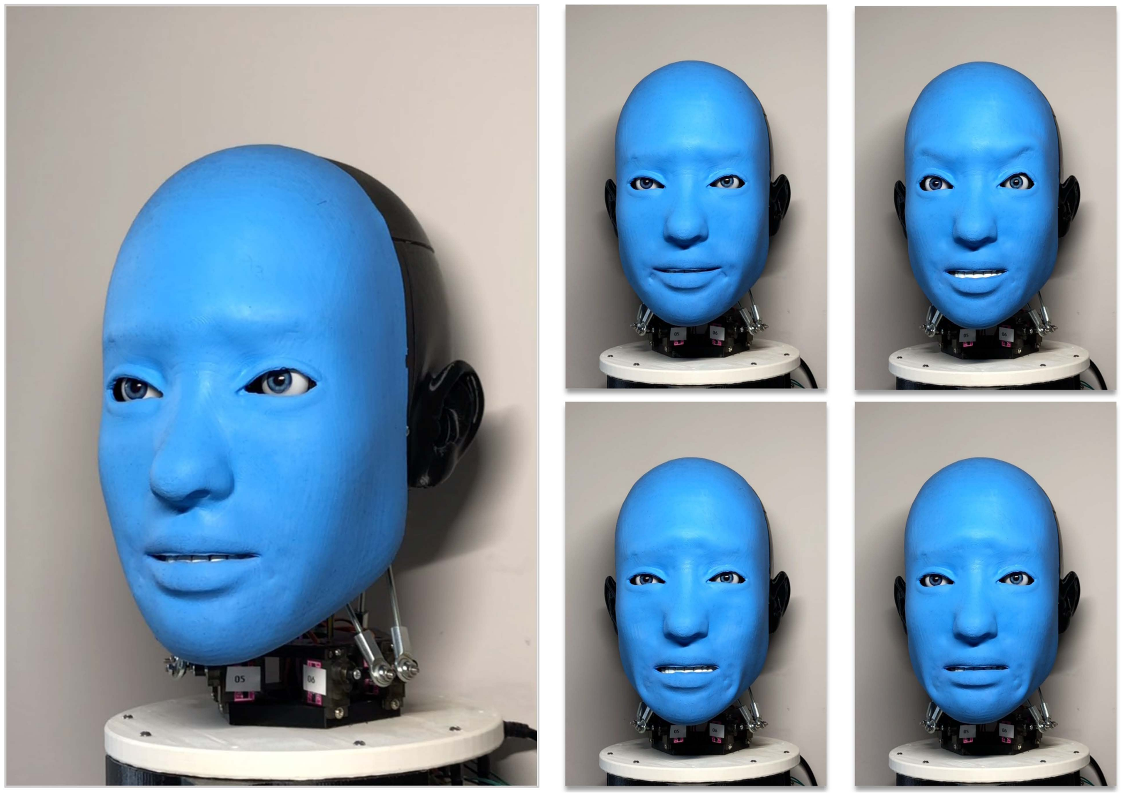


Fig. 2.1: **Eva 2.0** is a general animatronic robotic face for facial mimicry. The robot does so by learning the correspondence between facial landmarks and self-images as well as a learned inverse kinematic model. The entire learning process relies on the robot's motor babbling in a self-supervised manner. Our robot can mimic varieties of human expressions across many human subjects.

an inverse network to output the set of motor commands from the synthesized image.

Our experiments suggest that our approach outperforms previous nearest-neighbor search-based algorithms and direct mapping from human face to action methods. Moreover, quantitative visualizations of our robot imitations demonstrate that, when presented with diverse human subjects, our method generates appropriate and accurate facial expression imitations.

Our primary contributions are threefold. First, we present an animatronic robotic face with soft skin and flexible control mechanisms. Second, we propose a vision-based learning framework for robot facial mimicry that can be trained in a self-supervised manner. Third, we construct a human facial expression dataset from previous database, YouTube videos and real-world human subjects. Our approach enables strong generalization over 12 human subjects and nearly 400 salient natural expressions. Our robot has a response time within 0.18 s, indicating a strong capability of real-time reaction.

2.2 Robot Platform Design

Our robot design — based on our previous Eva 1.0 [101] with significant upgrades — consists of two sub-assembly modules: facial movement module and neck movement module. We use micro servo motors (MG90S) to actuate all the components of our robotic face. All parts in our design are based on off-the-shelf hardware components that can be easily purchased online or 3D printed. To accelerate research, we open-source the design and step-by-step assembly process on our website. An overview of our hardware design is shown in Fig. 2.2.

Facial Movement Module Our facial movement module can be further divided into skull frame, eye module, muscle module and jaw module. Compared to Eva 1.0, we redesign the 3D shape of the skull frame to enlarge the maneuvering space and thus allow for more flexible movements. The new skull frame also facilitates tighter connection to the skin with smoother and more natural looks on the face surface.

By adding a pair of ball joints and three pairs of parallelogram mechanisms, we now also ensure that the 6DoF eyeball module can move freely within a ±20° range both horizontally and vertically, making the movements more stable. The rotation of each eyeball is controlled by two motors and three ball joint linkages. We also upgraded the eyelid design based on human face.

Our robot skin is attached to the front skull. The muscles driving facial expressions are attached to the inner side of the skin with a piece of fiber fabric, a strand of nylon cord, and a servo actuator inside the back of the skull. By pulling different nylon cords, a specific skin region can be deformed. We use the same region selection as Eva 1.0. To reduce the noise for muscle control, we run each nylon cord through a transparent vinyl tube linking the skull front and the motor. Our design ensures that the deformation in each muscle is proportional to the motor's rotation angle. Our design enables a large possible space of facial expressions by manipulating 10 pairs of symmetrical muscles in different proportions.

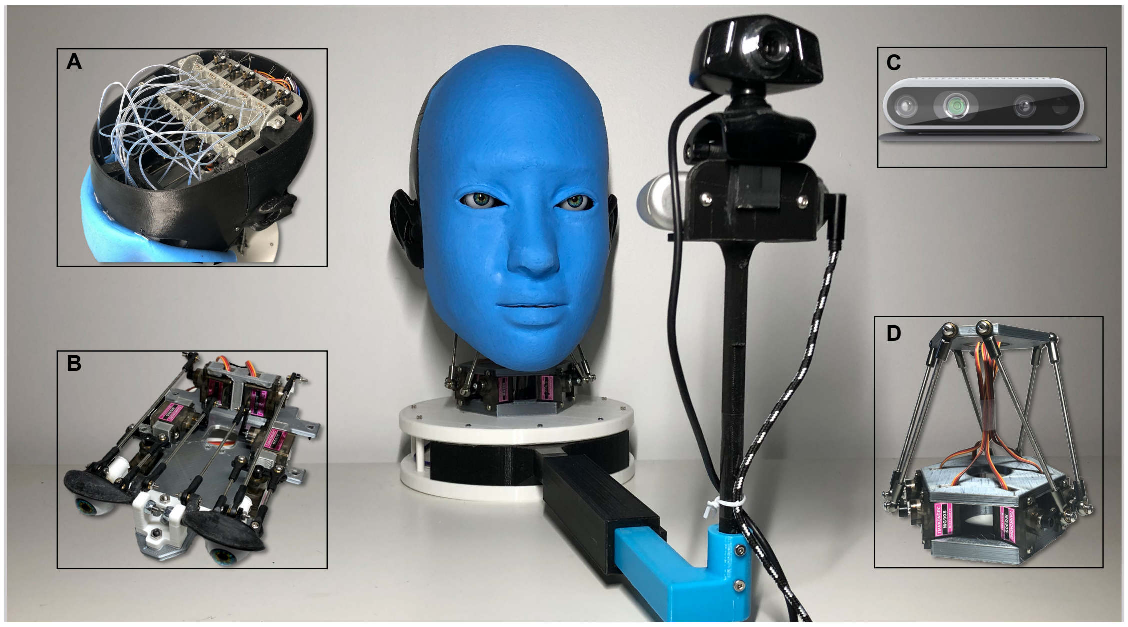


Fig. 2.2: **Mechanical Design:** our robot is actuated by the motor servo module (A) controlled by a Raspberry Pi 4 located at the bottom. The soft skin is connected to 10 motors via nylon cord. Our 6DoF eye module (B) is decoupled from the front skull. The RGB camera (C) is only used for random data collection of robot self-images but not for testing. The 6DoF neck module (D) follows Steward platform.

The jaw module is responsible for deforming the two muscles around the mouth. This is similar to the aforementioned skin movement design, but benefits from two additional coupled motors that control the movement, allowing the jaw to open up to 20°.

Neck Movement Module Our 6DoF neck module design is inspired by Stewart platform. Six motors are arranged in 3 pairs evenly distributed on 3 sides of the hexagon base. Each pair comprises two motors in a mirrored arrangement that are connected to a ball joint linkage.

2.3 Self-Modeling through Self-Image

We propose a learning-based framework for controlling the animatronic robotic face to mimic varieties of human facial expressions. An overview of our learning algorithms is shown in Fig. 2.3. We consider the following problem setup: given an image of human face displaying a natural facial expression, the model needs to output the motor commands to actuate the robot to imitate the given facial expression.

Without human pre-programming for different expressions, the problem poses several key challenges and desired properties. First, we hope that the learning algorithm can generalize to diverse unseen human faces. We leveraged the recent advances in human facial landmark detection to obtain abstract representations from high-dimensional image frames (Section 2.3.1) which can be shared among different human subjects.

The second challenge in attempting to achieve facial mimicry stems from the lack of ground-truth pairs of human expressions and robot motor commands. Without hard-coding and extensive trail-and-error, obtaining such a pair is not practical. In this chapter, we overcome this issue by adopting a two stage learning-based method: (1) a generative model which first synthesizes a robot self-image from facial landmarks processed by a proposed normalization algorithm (Section 2.3.3 and Section 2.3.2), (2) and an inverse model that is trained to produce desired motor commands from the generated robot image (Section 2.3.3). As we will show, the ground-truth labels for both models can be acquired with one round of self-supervised data collection without any human input.

2.3.1 Representation of Facial Expression

We capture facial expressions via facial landmarks, as this has been shown as an effective means of representing the underlying emotions. More importantly, facial landmarks provide a unified abstraction from diverse high-dimensional human images under different lighting, background and poses.

Specifically, we extract facial landmarks with OpenPose [120, 121] software. The output is a vector of 53×3 size representing the spatial position of 53 landmarks on human faces, with the last dimension being confidence scores. We also extracted the head pose for direct neck movement control. As ground-truth pairs for generating motor commands directly from the extracted human landmarks are not available, we adopt a different strategy, as discussed below.

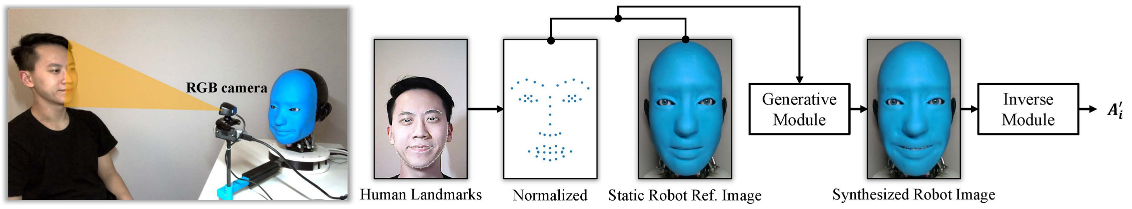


Fig. 2.3: **Model Overview:** our two-stage framework consists of two major modules: a generative network and an inverse network. Given an image captured by a regular RGB camera, we first extract facial landmarks with OpenPose. We then normalize the human landmarks to the robot scale and embed it on an image. Together with a reference static robot self-image, these two images are concatenated to the generative network to synthesize a robot self-image as if the robot makes the same expression. The inverse model takes the synthetic robot self-image to output the final motor commands for execution.

2.3.2 Landmark Normalization

Before we send the landmarks from human faces for robot learning, we need to normalize the spatial locations of the landmark vector to the robot domain. This is necessary due to potential variations in the scale or landmark arrangements in the captured human faces. Formally we normalize each landmark coordinate from human space L_H to robot space L_R with:

$$L_R = \frac{(L_H - H_{min})(R_{max} - R_{min})}{H_{max} - H_{min}} + R_{min}$$

where $H_{min}, H_{max}, R_{min}, R_{max}$ represent the value ranges of the spatial location per corresponding landmark in the sampled human and robot image frames.

2.3.3 Generative Model and Inverse Model

The generative model takes in the normalized human landmarks and generates a synthetic RGB image allowing the robot to conceive the same facial expression, which we denote as robot's self-image. We parametrize the generative model G with a deep neural network with parameter θ.

A key challenge here is to map the coordinate vector to a high-dimensional image. This could be accomplished by encoding the input vectors with a fully-connected network, which is thus used by the robot to learn the entire spatial mapping. However, this simple approach requires a network of strong capacity and optimization algorithm and does not perform well in practice [122].

To this end, we propose to encode the spatial coordinates of the landmarks to a two-channel image mask $\mathbf{M_i^{w\times h\times 2}}$ [123]. The first channel has the value of 1 if there is a landmark at the particular location and 0 otherwise. The second channel is a greyscale image indicating the confidence score returned by the landmark detection algorithm. This encoding matches the size of the robot image, which helps with ensuring correct spatial correspondence.

Furthermore, instead of relying on the network to directly regress the absolute value in the output image, it only needs to output the "change" in the image introduced by landmark displacements. In practice, we achieve this by conditioning the network with a static robot self-image $\mathbf{I_s^{w\times h\times 3}}$ whereby the two images are concatenated along the depth channel. Our generative model can be expressed as: $\mathbf{I_i} \leftarrow G(\mathbf{M_i}, \mathbf{I_s})$.

Implementation Details We use a fully convolutional encoder-decoder architecture [124, 125] where the resolution of the decoder network is enhanced by several hierarchical feature refinement convolutional layers [126]. Since the network is fully convolutional, we can preserve all the spatial information with high-quality outputs. Our network is optimized with a simple pixel-wise mean-squared error loss using Adam [127] optimizer and a learning rate of 0.001. We train the network for 200 epochs with batch size 196 until convergence on a validation dataset. The formal objective function that we minimize is:

$$\mathcal{L}_G = \mathrm{MSE}(G(\mathbf{M}, \mathbf{I_s}), \mathbf{I})$$

The inverse model F maps the synthetic robot self-image to motor commands in order to learn an inverse mapping from the goal image to actions.

Action representation Our robot utilizes N motors to actuate the face muscles. Given the robot face image, it is straightforward to frame this problem as continuous value regression. That is, given an input image, the network needs to output N values for N motor encoders. However, this framing can quickly become impractical in complex real-world scenarios and may lead to over-engineering. In practice, we observe that the motors have certain angle threshold to produce salient and stable motion. As a result, we can achieve salient motions with a discrete parameterized angle encoder without loss of accuracy.

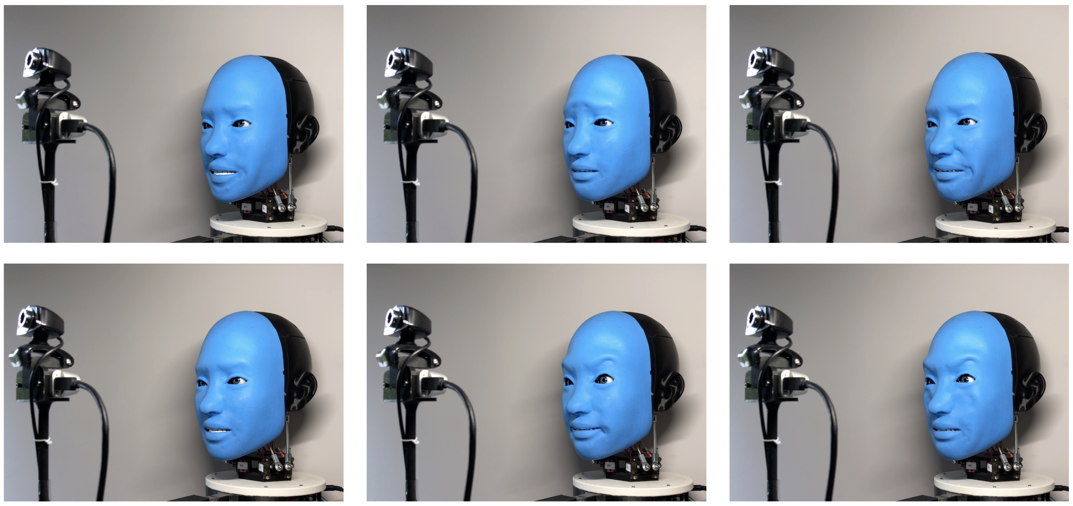


Fig. 2.4: **Training Data Collection:** the whole training process of our generative model and the inverse model rely on a single robot dataset without human supervision. We collect our training data with random motor babbling in a self-supervised manner whereby the camera facing the robot is used solely for gathering the training data, *i.e.*, it is not used during evaluation

We normalized and discretized the motor values to the $[0, 1]$ range with 0.25 step size, resulting in 5 values per motor. This discretization converts the original problem to a multi-class classification problem. Given an input robot image, the inverse model will output $5 \times N$ numbers, where every five numbers represents the probability of choosing each motor angle for one actuator. Our inverse model can be expressed as: $\mathbf{A_i} \leftarrow F(\mathbf{I_i})$.

Implementation Details Our architecture has 6 convolutional layers followed by several fully-connected layers to adapt the output dimension to be $5 \times N$. We train our network with a multi-class Cross-Entropy loss with Adam optimizer and a learning rate of 0.00005 for 58 epochs. We have $14,000$ pairs for training and $1,000$ pairs for validation and testing respectively. The formal objective function is:

$$\mathcal{L}_F = \Sigma_{n=0}^{N-1} \mathrm{CE}(F_n(\mathbf{I}), \mathbf{A_n})$$

2.3.4 Training Data Collection

Both models can be trained separately with the same data collected via self-supervised motor babbling (Fig. 2.4). We randomized the angles of the 10 motors ranging from 0 to 1 with an interval of 0.25 for $16,000$ steps. We used Intel RealSense D435i to capture RGB images and cropped the image to 480×320 to center the robot head. For each step i, we recorded the motor command values $\mathbf{A_i}$, the corresponding robot images $\mathbf{I_i}$, as well as the extracted landmark $\mathbf{L_{R,i}}$ from the robot with OpenPose. We thus obtain the training pairs of the generative model as ($\mathbf{L_{R,i}}$, $\mathbf{I_i}$) and the training pairs of the inverse model as ($\mathbf{I_i}$, $\mathbf{A_i}$). Since the data collection is purely random, the process does not require any human labeling.

2.3.5 Inference

Once the generative model and the inverse model are trained, we can use them jointly to perform inference. Given the input human image captured by an RGB camera, we first extract the landmark coordinates. After our normalization procedure, we can send it to our generative model which then outputs a synthetic robot self-image. This synthetic robot image serves as input for the inverse model which outputs the motor commands. All the network training and testing can be accomplished on a single NVIDIA 1080Ti GPU, whereas the motor commands on the robot are executed via WiFi.

2.4 Experiments

2.4.1 Evaluation Dataset

Even though the training of our models do not require any human face dataset, we constructed a human facial dataset with salient facial expressions to perform extensive evaluations. Our human expression dataset is a combination of MMI Facial Expression database [128] and a set of online videos featuring eight subjects. We uniformly down-sampled the original videos to obtain 380 salient frames covering a variety of facial expressions and human subjects.

2.4.2 Baselines

We are interested in evaluating the effectiveness of our approach for the generative model (GM), the inverse model (IM), and finally the entire pipeline. To this end, we design our baseline and ablation studies for each of them. For the generative model, we compared our method with a randomly sampled (RS) image from a collection of real images. This comparison aims to ascertain whether the synthetic image can outperform the real image and whether our model can predict reasonable motion on the static robot template.

Even though our method successfully convert the inverse model to a classification problem, a purely random baseline only has a 20% success rate. We also compared our model with another two less random simple baselines — a randomly initialized network (RI) and a model trained for about 100 iterations (RI-100).

For the entire pipeline, we evaluated different combinations of the above baselines. For example, we obtained one baseline by combining our generative model with an inverse model that has been trained for 100 iterations. Additionally, we present another three baselines. The first one is to perform a nearest neighbor retrieval (NN) with the landmarks extracted from the output of our generative model. We can directly search within our robot dataset. This baseline replaces our inverse model with a NN model. Similarly, we can perform such NN operation with human landmarks as a direct input. Lastly, we assessed whether we can directly generate the motor commands without our two-stage algorithm by training a network to output motor commands from our embedded input landmarks.

We used the same architecture for all the above baselines while varying only the number of channels in the first layer for different input dimensions. We trained all the models with three random seeds and report the mean and standard errors.

2.4.3 Evaluation Metrics

Our evaluation metrics cover both the pixel-wise accuracy of synthetic image and the accuracy of the output commands. We also extracted landmarks from our synthetic image to measure if the

Table 2.1: Accuracy of the Generative Model

Method	Image Distance ↓ ($\times 10^{-5}$)	Landmark Distance ↓
GM (ours)	$\mathbf{3.47 \pm 0.009}$	$\mathbf{0.46 \pm 0.002}$
RS	6.47 ± 0.039	0.84 ± 0.007

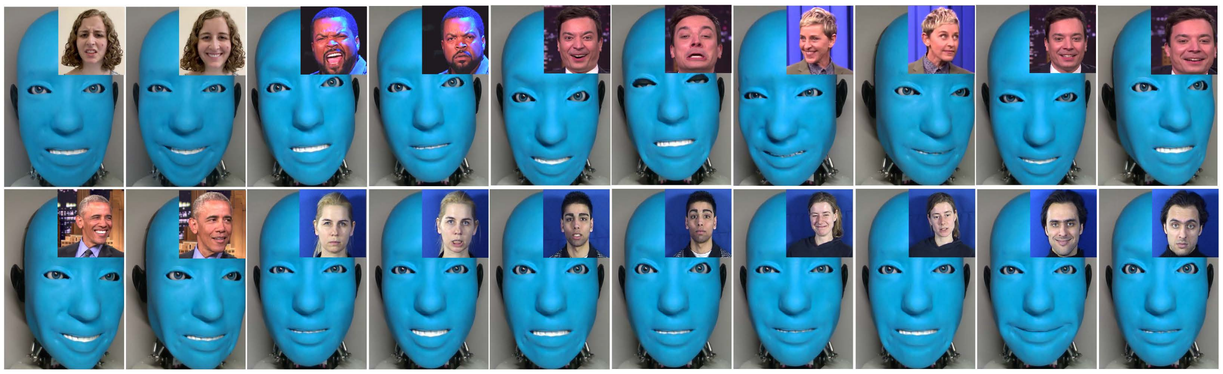


Fig. 2.5: **Robot Visualizations:** we executed the output motor commands on our physical robot to demonstrate that our method supports accurate facial mimicry of a variety of human expressions across multiple human subjects.

model successfully learns the facial expression than simply copying the static image. Moreover, we provide qualitative visualizations for the final pipeline. We also extracted landmarks from our entire pipeline execution to compare against the input human landmarks. We use L2 metric for both the image and landmark distances.

2.4.4 Results

Generative Model Tab. 2.1 shows the quantitative evaluation for our generative model. Our generative model outperforms the random baseline by a large margin. Note that the image distance is normalized by the total number of pixel values (480×320×3) which range from 0 and 1, whereas the landmark distance is normalized by the total number of landmarks (53).

Inverse Model Tab. 2.2 shows the evaluation result for our inverse model. Compared with the random initialized model, the model trained for 100 iterations as well as the purely random baselines (20% accuracy), our inverse model produces much more accurate motor commands.

Two-Stage Pipeline By combining the generative and inverse model, we can use the synthetic output from the generative model as the input for the inverse model, allowing us to evaluate the en-

Table 2.2: Accuracy of the Inverse Model

Method	Command Distance ↓	Command Accuracy ↑ (%)
IM (ours)	$\mathbf{0.53 \pm 0.009}$	$\mathbf{75.86 \pm 0.042}$
RI	1.39 ± 0.009	30.34 ± 0.038
RI-100	0.90 ± 0.010	56.40 ± 0.041

Table 2.3: Accuracy of the Entire Two-Stage Pipeline

Method		Command Distance ↓	Command Accuracy ↑ (%)
GM	**IM**	$\mathbf{0.83 \pm 0.008}$	$\mathbf{54.57 \pm 0.048}$
GM	NN	1.06 ±0.009	45.74 ±0.045
GM	RI	1.39 ±0.009	30.34 ±0.039
GM	RI-100	0.98 ±0.009	53.48 ±0.041
Landmark to Motor		1.03 ±0.011	52.81 ±0.048
Landmark NN		0.98 ±0.095	49.3 ±0.045

tire two-stage pipeline. As shown in Tab. 2.3, our two-stage strategy outperforms all baselines, but is inferior to the performance of its component models. This is to be expected, as the inverse model takes the synthetic output from the generative model as its input when these are evaluated jointly, whereas individual evaluations are based on real images from our robot dataset. Nonetheless, our method still achieves the best predictions and is particularly advantageous compared to two single model baselines, indicating that decomposing the problem into two sub-steps is a prudent design choice.

Pipeline Execution We executed the output motor commands from the above two-stage pipeline on our physical robot and computed the landmark distance extracted from the resulted robot face with the ground-truth normalized landmarks. To provide qualitative evaluations, we visualized the final robot face together with the input human face in Fig. 2.5.

Our evaluation was conducted on 380 salient frames which covers 10 video clips from 8 subjects. We show the quantitative results compared with a random baseline in Fig. 2.6. Our approach demonstrates a flexible learning-based framework to mimic human facial expression. Our method also generalizes across various human subjects without any human supervision.

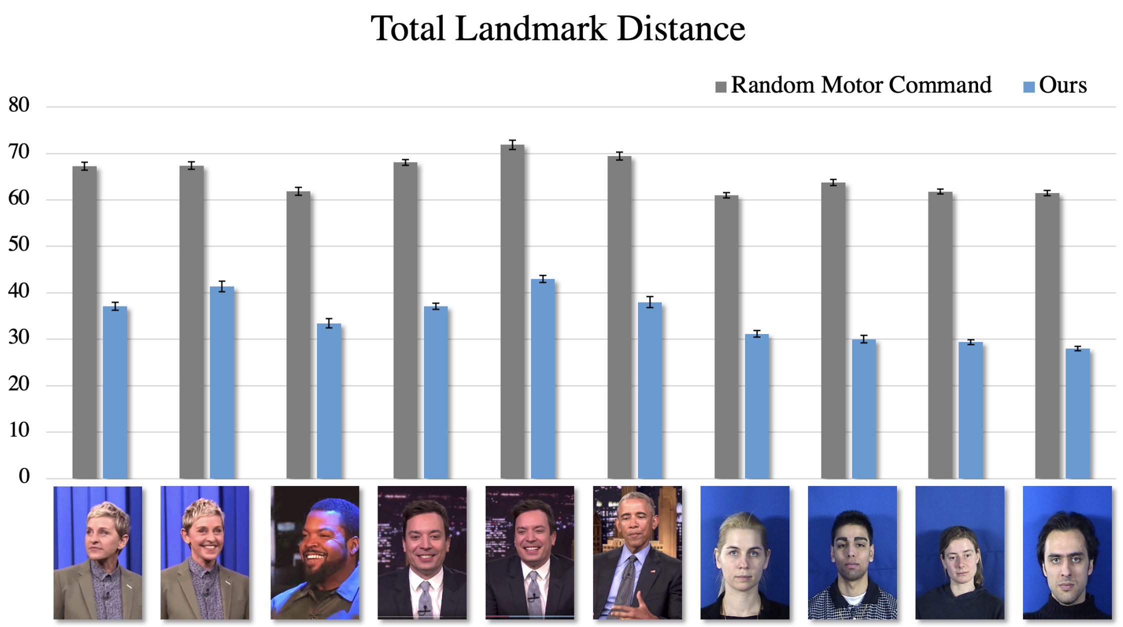


Fig. 2.6: **Pipeline Execution:** we executed the motor commands output by our entire two-stage pipeline and extracted the landmarks from the resulting physical robot face to compare against the ground-truth human landmarks as well as a random baseline to benchmark the task difficulty. The results show that our robot can imitate different human expressions accurately.

Overall, we present a new animatronic robot face design with soft skin and visual perception system. We also introduce a two-stage self-supervised learning framework for general face mimicry. Our experiments demonstrate that the two-stage algorithm improves the accuracy and diversity of imitating human facial expressions under various conditions. Our method enables real-time planning and opens new opportunities for practical applications.

Chapter 3: Robot Query-Based Self-Modeling

The last chapter has discussed the contributions toward scaling data-driven forward-kinematics self-modeling approach to robots with complex kinematics and dynamics, such as an animatronic robotic face with a visual perception system and soft skin. However, data-driven forward-kinematics self-models must know in advance what aspects of the robot need to be modeled, such as the tilt angle of the robot [1], the position of end effectors [2], the velocity of motor joints [3], the mirror image of animatronic faces [4], or the contact locations, as well as joint configurations of robot grippers [5]. The restricted predictive scope of traditional data-driven self-models limits the general applicability of these self-models to future, yet unknown, 3D spatial planning tasks. For example, a data-driven self-model focusing only on predicting the position of an end effector may not be useful for tasks involving operation in a crowded workspace, where full-body collisions must be factored into the planning. Ensuring that the entire robot arm motion will be collision-free is a critical aspect for numerous safe robot operations such as object retrieval, trajectory planning, and human-robot interaction. Data-driven modeling of the entire robot morphology and kinematics, without prior knowledge of what aspects of the morphology are relevant to future tasks, has remained a major challenge.

In this chapter, we present a full-body visual self-modeling approach [129] (Fig.3.1) which captures the entire robot morphology and kinematics using a single implicit neural representation. Rather than predicting positions and velocities of prespecified robot parts, this implicit system is able to answer space occupancy queries, given the current state (pose) or the possible future states of the robot. For example, the query-driven visual self-model can answer queries as to whether a spatial position (x, y, z) will be occupied if the joints move to some specified angles. Since both the spatial and robot state inputs are real values, our visual self-model allow continuous queries in the domain of both control signals and spatial locations. Furthermore, the learning process only

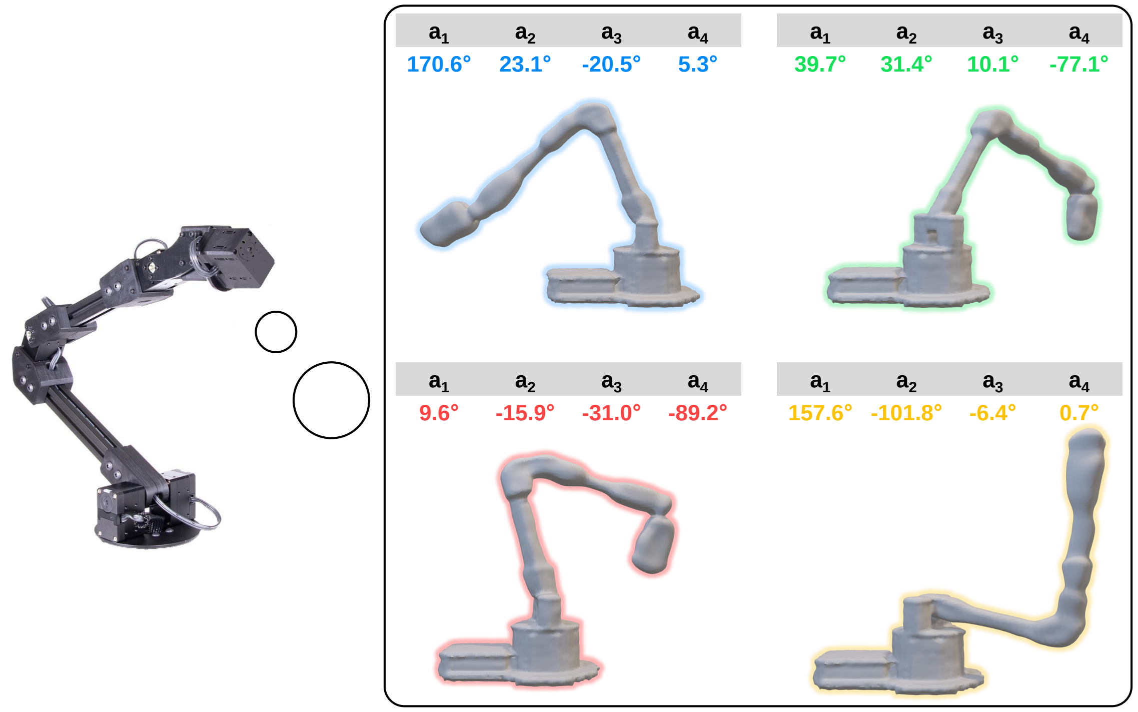


Fig. 3.1: **Visual self-modeling robots.** We equip the robot with the ability to model its entire morphology and kinematics in 3D space only given joint angles, known as visual self-model. With the visual self-model, the robot can perform variety of motion planning and control tasks by simulating the potential interactions between itself and the 3D world. Our visual self-model is continuous, memory efficient, differentiable and kinematic aware.

requires joint angles and sparse multi-view depth images, which enables generalizable and scalable data acquisition without human supervision.

Once learned, the responses from this single visual self-model to a series of queries can then be used for a variety of 3D motion planning and control tasks, even though the self-model was only trained with task-agnostic random motor movements. Because of our fully differentiable parameterization, the robot can directly perform efficient parallel gradient-based optimization on top of the self-model to search for the best plans in real-time. We can also combine the self-model in a seamless manner with existing motion planning techniques. Moreover, when the robot sustains physical damage, such as broken motors or changed morphology, our self-model can detect, identify and recover from these changes. Since our self-model is inherently visual, it can provide real-time human-interpretable visualization of the robot's internal belief of its current 3D morphology and state. This ability to sense pose-conditioned space occupancy is perhaps not unlike our natural human ability to "see in our mind's eye" whether our body could fit through a narrow passage without actually trying it out in reality [130].

3.1 Query-Based Self-Modeling of Robots

3.1.1 Implicit Visual Self-Model Representation

Robots operate in a 3D world, and therefore being morphologically and kinematically aware in 3D space is essential for them to successfully interact with the physical environments as well as adapt to potential changes in the field. Traditionally, robot engineers build a physical simulator and integrate it with CAD models of the robot. However, designing a simulation environment is not trivial. Accurate CAD models that reflect the real as-built robot geometry may not be easily available, especially for robots that have been modified due to damage, adaptation, wear and repair. This challenge will likely become more acute as the variety and complexity of robotic systems continues to increase in the future, and especially as robots must operate with less human supervision, maintenance, and oversight.

We therefore aim to learn the self-model of robots directly through task-agnostic data with minimal human supervision or domain knowledge. Our goal is to learn a visual self-model which can capture the entire body morphology and kinematics, without prior knowledge of the body configurations such as joint placements, part geometry, motor axis and joint types. With the visual self-model, a robot should be able to plan its future actions by rolling out the self-model before executing any actions in the physical world. We can also visualize its final plan from different viewing angles, because the model itself is three-dimensional.

There are two major challenges when designing a visual self-modeling process. First, we need to carefully decide how to represent the 3D geometry of the robot body. Most existing 3D representations are explicit, such as point cloud, tessellated triangle meshes, or voxelized occupancy grids. However, such approaches come with several limitations. Point clouds, meshes, and grids often consume large amounts of memory to store even a single geometry, let alone a kinetic geometry dependent on input DoF. Point clouds also lose structural connectivity, while voxel representa-

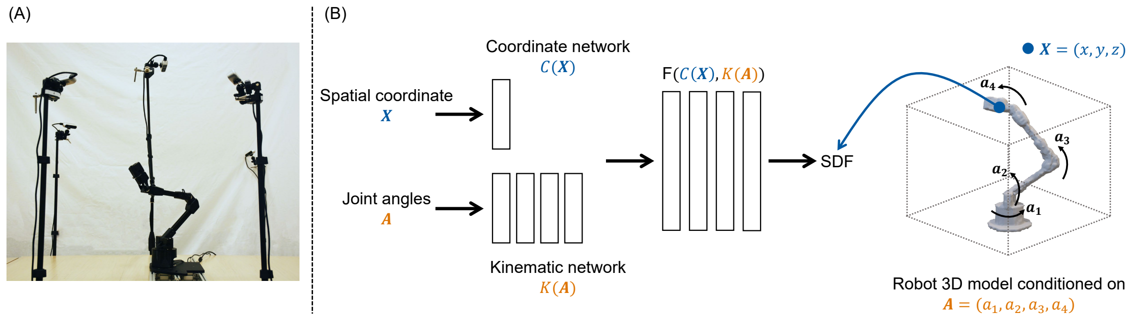


Fig. 3.2: **Implicit visual self-model representation.** (A) Real-world setup for data collection. We fused sparse views from five depth cameras to capture the point cloud of the robot body. As the robot arm randomly moved around, we recorded pairs of the robot joint angles and its 3D point cloud. (B) We show the computational diagram of our visual self-model. The coordinate network takes in the spatial coordinate and the kinematic network extracts kinematic features from the input joint angles. We then concatenated the spatial features and the kinematic features into a few layers of MLPs to output the zero-level set SDF values. The implicit representation can be queried at arbitrary continuous 3D spatial coordinates and different sets of joint angles.

tions lose continuous resolutions. These limitations are amplified in kinematic tasks, since the self-models are expected to be dependent on trajectories of multiple degrees of freedom of the robot.

The second challenge concerns the computational efficiency of leveraging the learned visual self-model for downstream task planning. Once a visual self-model is formed, we hope that the same model can be used for many tasks. In other words, the model must be task-agnostic. Furthermore, real-time planning and control is critical for many robotic applications. Therefore, the ideal representation should render the 3D model in a parallel and memory efficient manner using GPU hardware. The model should also provide fast inference capability to solve common inverse problems in robotics, such as inverse kinematics. Lastly, not every component of the robot body weights equally in all tasks, so it should be possible to query different spatial components of the visual self-model as needed. For example, the full 3D knowledge of the robot base and 3D geometry of other arm components are not required when calculating the inverse kinematic solution of a robot arm trying to reach a 3D object with its end effector.

We overcame the above challenges by proposing a state conditioned implicit visual self-model that is continuous, memory efficient, differentiable, and kinematics aware. The key idea is that the model does not simply predict future robot states explicitly; instead, it is able to answer spatial and kinematic queries about the geometry of the robot under various future states.

To construct a query-answering self-model, we leverage implicit neural representations to model the 3D body of the robot as shown in Fig.3.2. Given a spatial query point coordinate $\boldsymbol{X} \in \mathbb{R}^3$ normalized based on scene boundary, and a robot joint state vector $\boldsymbol{A} \in \mathbb{R}^N$ specifying all the N joint angles, the visual self-model can be represented by a neural network to produce the zero-level set Signed Distance Function (SDF) of the robot body at the given query point $\boldsymbol{X}$.

We use SDF as a representation of 3D shapes [131]. An SDF is a continuous field in which each point is associated with a magnitude value representing its closest distance to a surface, and a sign ($-$ or $+$) indicating if the point is inside or outside the surface boundary. Through this network, the robot morphology is represented as the zero-level iso-surface of the function.

Formally, the model can be expressed as:

$$SDF = \{\boldsymbol{X} \in \mathbb{R}^3, \boldsymbol{A} \in \mathbb{R}^N | F(C(\boldsymbol{X}), K(\boldsymbol{A}))\},$$

where C is the coordinate neural network with several layers of MLPs to encode the spatial coordinate features, K is the kinematic neural network with several layers of MLPs to encode the robot kinematic features, and F is the last few layers of MLPs to fuse the features from both the coordinate network C and kinematic network K after concatenating their outputs to produce the final SDF values conditioned on the queried spatial coordinates and current joint angles. We omit the batch size here for simplicity. For nonlinear activation functions, we used Sine functions to preserve the details on the 3D models [132].

We trained the network by formulating the problem as an Eikonal boundary value problem. Instead of supervising the network with the ground-truth SDF, similar to SIREN network [132], we directly used point clouds and surface normals obtained by fusing observations from sparse RGB-D camera views as labels, as indicated in Fig.3.2. In both simulation and real-world setup, we used five RGB-D cameras to capture pairs of data for training, namely the joint angles and the fused point cloud. During testing, the only available robot-related information to our visual

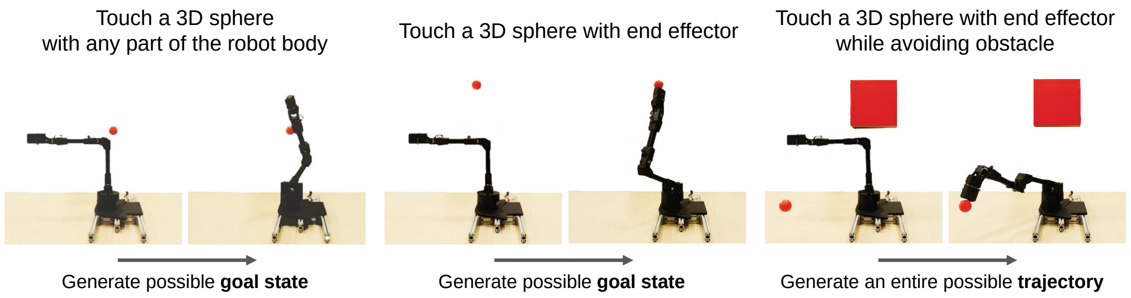

Touch a 3D sphere
with any part of the robot body
Touch a 3D sphere with end effector
Touch a 3D sphere with end effector
while avoiding obstacle
Generate possible goal state
Generate possible goal state
Generate an entire possible trajectory

Fig. 3.3: **3D self-aware motion planning tasks.** We present an overview of three different tasks. Touch 3D sphere with any part of the robot body (Left) asks the robot to generate a set of target joint angles such that some part of the robot body needs to be in contact with a randomly placed target sphere. Touch a 3D sphere with end effector (Middle) requires the robot to generate a set of target joint angles such that the robot needs to touch a randomly placed target sphere with its end effector link. Touch a 3D sphere with end effector while avoiding obstacle (Right) tasks the robot to propose an entire set of collision-free trajectories in the form of intermediate joint angles to touch a randomly placed target sphere using its end effector. The three tasks gradually becomes harder with more constraints.

self-model is a set of joint angles. More details of the network architectures and loss functions are discussed in the Materials and Methods section.

Overall, our visual self-model is formed by several layers of MLPs that implicitly captures the entire morphology and kinematics of the robot body. We implemented the network with differentiable deep learning framework so that it can be easily deployed on GPUs with end-to-end differentiable capabilities. Notably, though the entire self-model only consumes 1.1 MB to store its weights, our visual self-model can represent the 3D morphology of the robot body with different kinds of joint angles at various continuous spatial locations. By separating the kinematic feature encoder and coordinate feature encoder into two sub-networks, each sub-network captures independent semantic meaning. As we will show next, this property allows the self-model to learn rich kinematic features useful for downstream tasks.

3.1.2 3D Self-Aware Motion Planning

We aim to utilize the learned visual self-model in various motion planning tasks in 3D space. In this section, we will present algorithm designs to show the use cases for three sample tasks (Fig. 3.3). However, our model is not limited by only those three tasks. Rather, we use them as representative examples for demonstration purposes and we expect that the model can generalize to other possible tasks.

- **Touch a 3D sphere with any part of the robot body** The goal of this task is to touch a 4cm diameter sphere using any part of the robot body. To solve this problem, the robot needs to calculate inverse kinematics in 3D without constraints on which specific body piece touches the target object.
- **Touch a 3D sphere with end effector** This task not only requires the robot to touch a target sphere, but it also asks the robot to touch it with the end effector link. This is a harder task since the robot needs to solve inverse kinematics in 3D with a particular link constraint. The solution space is quickly reduced.
- **Touch a 3D sphere with end effector, while Avoiding an Obstacle** In this task, we ask the robot to go beyond computing a target end state with or without link constraints. Instead, in order to succeed at this task, the robot needs to perform precise motion planning in 3D to touch the final target while avoiding a large obstacle shown as red block in Fig.3.3. Overall, the robot is tasked to propose an entire safe trajectory from its initial state to the target state. During the execution of the proposed trajectory, the robot will fail the task if any part of the robot body collides with the obstacle.

To solve these motion planning tasks, one immediate thought is to obtain the entire robot body meshes and load them into existing robot simulators. This can be done by traversing all possible spatial points under certain precision and different sets of joint angles through the implicit neural representation, and rendering the entire 3D robot mesh with post-processing algorithms [133]. Such usage of the visual self-model seems to be a straightforward solution to bypasses the need to construct robot kinematic and geometric models such as CAD and URDF files. However, in practice, we found that constantly loading new robot meshes and destroying old robot meshes in commonly available robot simulators costed a significant amount of time. This limits the possibility of applying this method for real-time planning and control.

We propose to frame the first two tasks as constrained optimization problems by leveraging the differentiability of the visual self-model as well as its capability of answering partial queries on spatial coordinates. Specifically, for the first task, we initialize thousands sets of joint angles. We then sample P points uniformly on the surface of the target object. Since the output of the visual self-model will be zero when the queried spatial point is on the surface, the overall objective is to find the set of joint angles which can minimize the total sum of output values across all the sampled points on the target object. By freezing the weights of the learned visual self-model, we can perform gradient descent from the output surface predictions with respect to the input joint angles, under the constraint that the motor angles has to be within the range of $[-\pi, \pi]$.

Formally, the constrained optimization problem can be expressed as:

$$\boldsymbol{A}^* = \min \mathbb{E}_{\boldsymbol{A}^b} \left[\Sigma_{\boldsymbol{T}^p} F(C(\boldsymbol{T}^p), K(\mathbf{A}^b)) \right] \quad \textbf{s.t.} - \pi \leqslant A_i^b \leqslant \pi, i = 1, 2, 3, 4$$

where $\boldsymbol{T} \in \mathbb{R}^{P \times 3}$ is the sampled points on the target object, i is the motor index, and $b = 1, 2, ..., B$ is the index of each sampled set of joint angles with the maximum value B to be the batch size on a single GPU. Since the visual self-model runs parallelly on a GPU with small consumption of memories, the entire optimization process can produce accurate solutions within a short period of time. With more GPUs, the process can be further sped up.

To solve the second task, we need additional information about where the end effector locates relative to the entire robot body. Since the current visual self-model was only trained to capture the overall body geometries, similar to other works in self modeling, we can supervise the visual self-model to predict the end effector location at the same time. The good news is that our visual self-model already has a specialized sub-network that implicitly captures the robot kinematics. Therefore, we can directly use the pretrained weights of the kinematic sub-network, and train only two nonlinear layers of MLPs E attached to the end of the sub-network with little additional efforts. In fact, as we will show in the experiment section, our visual self-model provides a strong semantic proxy to pre-train the kinematic sub-network, leading to superior performance than training a specialized network to predict the end effector position from scratch. Without our decomposition formulation of kinematic sub-network, the acquired kinematics information may not be easily distilled as an independent feature for future use.

Similar to the first task, we now can formalize the solution of the second task by adding another objective function to make sure the resulted end effector reaches the target object. The overall optimization problem can be formalized as follows:

$$\boldsymbol{A}^{*} = \min \mathbb{E}_{\boldsymbol{A}^{b}} \left[\Sigma_{\boldsymbol{T}^{p}} \lambda_{\text{EE}} E(K(\mathbf{A}^{b})) + \lambda_{\text{SDF}} F(C(\boldsymbol{T}^{p}), K(\mathbf{A}^{b})) \right] \textbf{ s.t.} -\pi \leqslant A_{i}^{b} \leqslant \pi, i = 1, 2, 3, 4$$

As discussed above, the objective function includes two terms weighted by hyperparameters λ_{EE} and λ_{SDF}. The first term ensures the end effector touches the target object and the second term encourages the robot body to touch the target object. We found that adding a small λ_{SDF} consistently achieves better results.

Regarding the third task, our visual self-model can directly work with existing motion planning algorithms with minimal changes. There have been great success [134] on motion planning algorithms to solve obstacle avoidance problem in high-dimensional state and action spaces. We thus combine our visual self-model with the existing algorithms in a plug-and-play manner. Specifically, we use RRT* [135] as our backbone algorithm due to its popularity, probabilistic completeness and computational efficiency. Generally speaking, there are two major components in RRT* that require physical inference with robot bodies. The first component is to calculate the goal state, and the second component is to check whether a collision will happen given a particular state of the robot. With these two components, various planning algorithms can narrow the search space to the final solution without having to explicitly query robot status again.

Traditionally, these two components require a dedicated robot simulator and pre-defined robot bodies. With our visual self-model, we can reach the final solution by simply performing fast parallel inference on the learned model. Specifically, the goal state can be obtained by running the same optimization procedure as in the second task. For collision detection, we can pass uniformly sampled points on the obstacle surface as well as the given set of joint angles A_{query} through our visual self-model as shown below. If the total sum of the output values over all the sampled points is equal to or below a small threshold τ , then there is a collision. Otherwise, the robot will not collide with the obstacle object.

$$\text{Collision} = \begin{cases} \text{True,} \ \Sigma_{\boldsymbol{O}^i} F(C(\boldsymbol{O}^i), K(\boldsymbol{A}_{query})) \leqslant \tau; \\ \text{False,} \ \Sigma_{\boldsymbol{O}^i} F(C(\boldsymbol{O}^i), K(\boldsymbol{A}_{query})) > \tau. \end{cases}$$

3.1.3 Damage Identification and Recovery

One major promise of machines that can model or identify themselves is the capability of recognizing and inspecting damage or changes, and then quickly adapting to these changes. In this section, we present our method to identify and recover from damage using the learned visual self-model.

Our approach involves three steps. The robot first detects a damage or change on its body compared to its original (intact) geometry. Then the robot can identify which specific type of damage or change is happening. Finally, the robot will gather new information about itself with limited data and computational resources to quickly adapt its self-model to the new changes.

Overall, our approach introduces several significant advantages over previous methods. First, being able to recognize the specific type of damage or change enables the robot to provide additional feedback information. Previous works have shown that it was possible to detect a damage. However, they were not able to provide additional information to identify the source of the change or which specific type of damage has happened. This information is extremely helpful when the damage requires hardware repair. Instead of relying on a domain expert to perform a series of inspections, our method can automatically generate information about specific damage such as "the second joint motor is broken".

Another advantage is that our approach performs modeling in the 3D visual world. This means that we can visualize and render the internal belief of the visual self-model in a straightforward and interactive fashion. As we will show in the results, one can immediately tell which section of the internal belief of the robot body does not match the real-world counterpart. We can further tell visually if the internal belief has been updated to match the new changes after learning from new observations.

In the following sections, we begin by describing the specific algorithms, and then we follow with real-world results in the next section. In the first step, we measure the current prediction error and the original prediction error. The current prediction error is computed by comparing the internal belief expressed by the learned visual self-model with the current observed 3D mesh of the robot, while the original prediction error is computed by comparing the same internal belief with the previously observed 3D robot body. Both cases share the same joint conditions. By comparing these two prediction errors, a large gap can inform us about a significant change or damage to the robot body.

In the second step, we aim to identify the specific type of damage happening on the robot. Based on the robot arm platform we are experimenting with, we assume two types of potential changes: (1) broken motor and (2) changed topology.

To reveal which specific type of the current damage is, our key idea is to solve the inverse problem with the learned visual self-model. Concretely, based on a single current observation of the robot body, we infer the best joint angles that the robot should have executed to result to the current 3D observation. This is a very challenging problem because an ideal joint angle set needs to give accurate 3D reconstruction of the entire robot body. Relying on previous gradient-based optimization algorithm is inefficient since the final gradient computation requires the sum over all the sampled points on the whole robot body. This process takes a large amount of memory and computation resources to perform a single gradient step due to the large volume of the robot mesh. Instead, we propose to use random search to locate the best possible joint angles. The simple random search algorithm works very well in this case. It does not require the accumulation of any gradient information so that larger batch of queries can fit on a single forward pass of visual self-model.

With the inferred joint angles, we can quantify the damage by comparing them with the actual input motor commands. If a specific inferred joint angle is always different from the actual input and that particular inferred angle always stays as a constant value or some other random values, then we can tell that the corresponding motor is broken. When all the inferred joint angles match closely to the actual input commands, the wrong belief of the 3D body then comes from a topology change and all the motors function well. We leave the research where both changes happen at the same time or more complex changes as future directions.

Finally, we also evaluate if our visual self-model can quickly recover from the changes by adapting on several new observations. For this step, the main purpose is to demonstrate the resiliency of the model, rather than proposing a new algorithm for continual adaption. Therefore, we follow common approaches by collecting a few more 3D observations after the changes to keep training the network for several epochs. We then check if the new visual self-model can successfully update its internal belief to match the current robot body both quantitatively and qualitatively.

3.2 Experiments

In this section, we aim to evaluate the performance of the learned visual self-model, demonstrate the results of using the visual self-model in various motion planning tasks, and test the resiliency of the learned model under real-world damages. To this end, in the first two subsections, we present quantitative and qualitative evaluations as well as baseline comparisons in simulation. For all three subsections, we also demonstrate the fidelity of directly learning and using the visual self-model in the real-world setup.

3.2.1 Visual Self-Model Estimation

We used the WidowX 200 Robot Arm as our experimental platform both in simulation and real-world. In order to obtain the ground truth point cloud data, we mounted five RealSense D435i RGB-D cameras around the robot as shown in Fig.3.2(A). Four cameras were around each side of the robot to capture side views. One camera was on the top to capture the top-down view. All cameras were calibrated. The depth images were first projected to point clouds which were then fused into a single point cloud based on the camera extrinsic parameters. The final point cloud was generated by clipping the scene with a pre-defined scene boundary.

During data collection, we randomly moved the robot arms to get pairs of joint values and its corresponding point cloud. For each pair of data, the simulation needed less than 1 second and the real-world collection took around 8 seconds. In total, we collected 10,000 data points in simulation with PyBullet [98] and 7,888 data points in the physical setup. We partitioned the data into training set (90%), validation set (5%) and testing set (5%).

To evaluate the prediction accuracy, we ran several forward passes on the learned visual self-model to obtain the whole body mesh of the robot on the testing set. On a single GPU (NVIDIA RTX 2080Ti), this process took about 2.4 seconds. Following previous works on implicit neural representations of 3D models [131, 132], we calculated the Chamfer-L1 distance between the predicted mesh and the ground truth mesh as our metric. All units in this chapter are in meter.

In simulation, the point cloud fusion was nearly perfect due to noiseless depth image and exact camera calibrations. In the real-world experiments, we noticed that the point cloud fusion was very noisy due to imprecise depth information introduced by internal sensor errors, noisy camera calibrations, and importantly, very sparse view points. We did not increase the number of views since the current ground truth scan can already reflect the overall pose of the robot, so we tested the fidelity of our algorithm directly on the noisy real-world data in exchange of adding more resources and time cost. The gap of the ground truth data quality between the simulation and real world suggests that the final results in the real-world setup can be greatly improved with better future 3D scanning techniques.

Fig.3.4 (B) visualizes pairs of predicted meshes and the ground truth meshes. In both simulation and real-world cases, our learned visual self-model produced accurate estimations of the robot morphology and kinematics, given only unseen joint angles as input. We also compared our algorithm with a random search baseline and a nearest neighbor baseline. For the random search, we randomly selected a robot mesh from the training set as the prediction. For the nearest neighbor

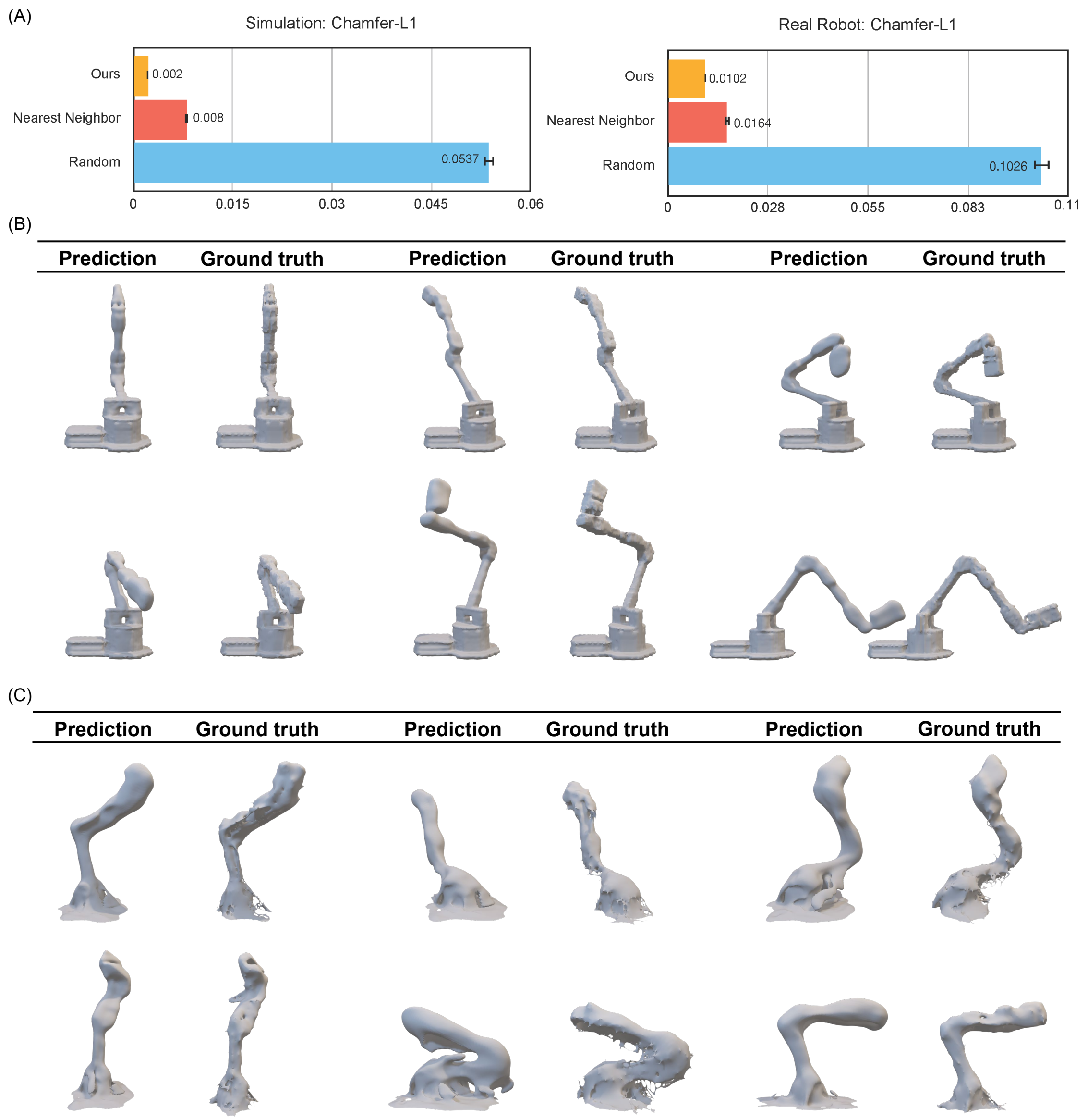


Fig. 3.4: **visual self-model predictions.** (A) Quantitative evaluations of our visual self-model predictions in both simulated and noisy real-world environments. Our visual self-model outperforms nearest neighbor and random baselines suggesting that the visual self-model learns a generalizable representation of the robot morphology beyond the training samples. (B) With simulated training data, our visual self-model can produce high quality 3D body predictions given a diverse set of novel joint angles. (C) When the training data becomes highly noisy in the real world due to imprecise depth information, noisy camera calibrations and super sparse view points, our visual self-model can still accurately match the ground truth to reflect the overall robot body morphology and kinematics.

baseline, we compared the testing joint angles with all the joint angles in the training set using L2 distance metric, and then used the robot mesh corresponding to the closest joint angles as the final

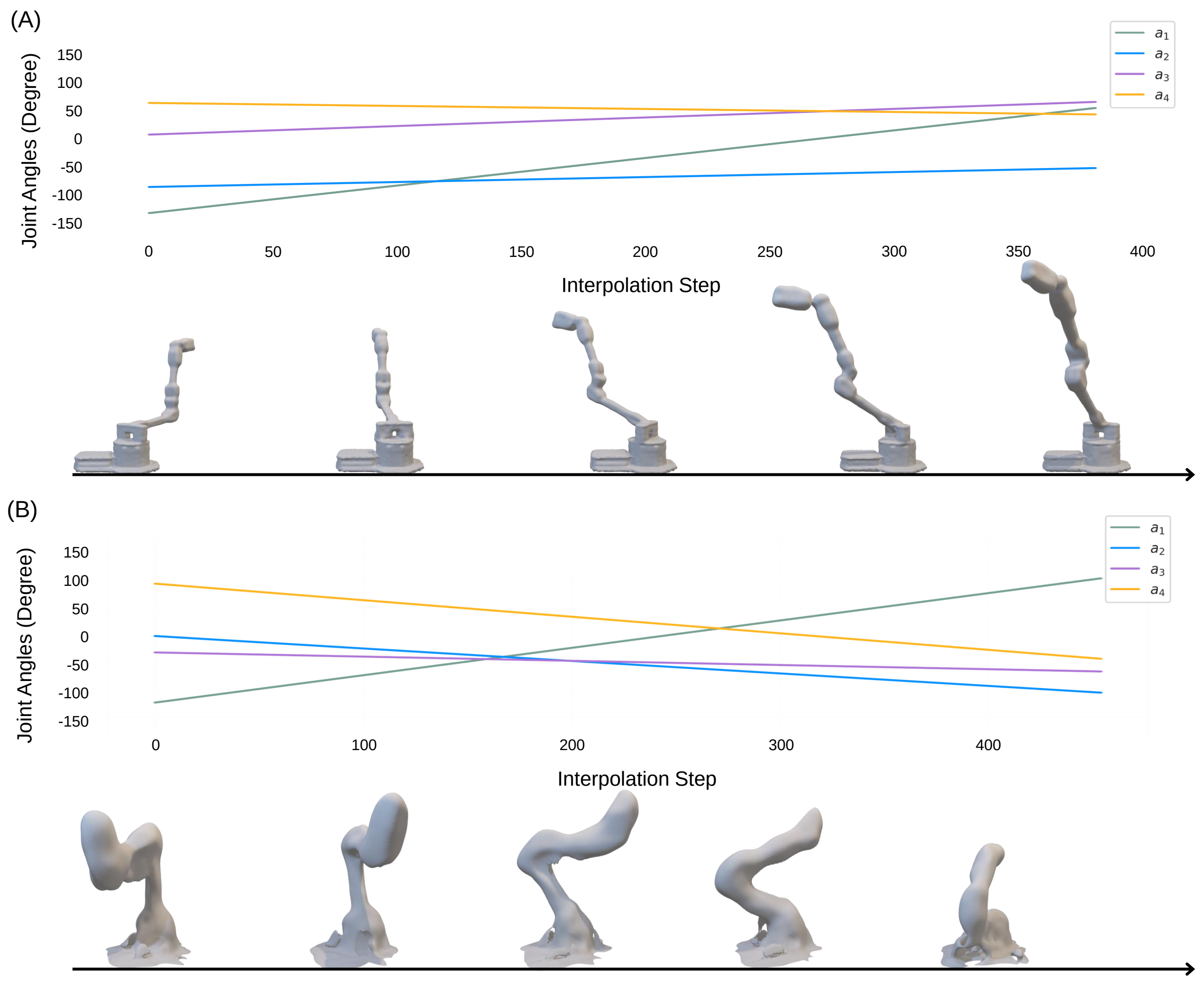


Fig. 3.5: **Interpolation between joint angles.** We demonstrate that our learned visual self-model can smoothly interpolate between different joint angles. (A) shows the results trained in simulation and (B) shows the results trained on real-world data.

prediction.

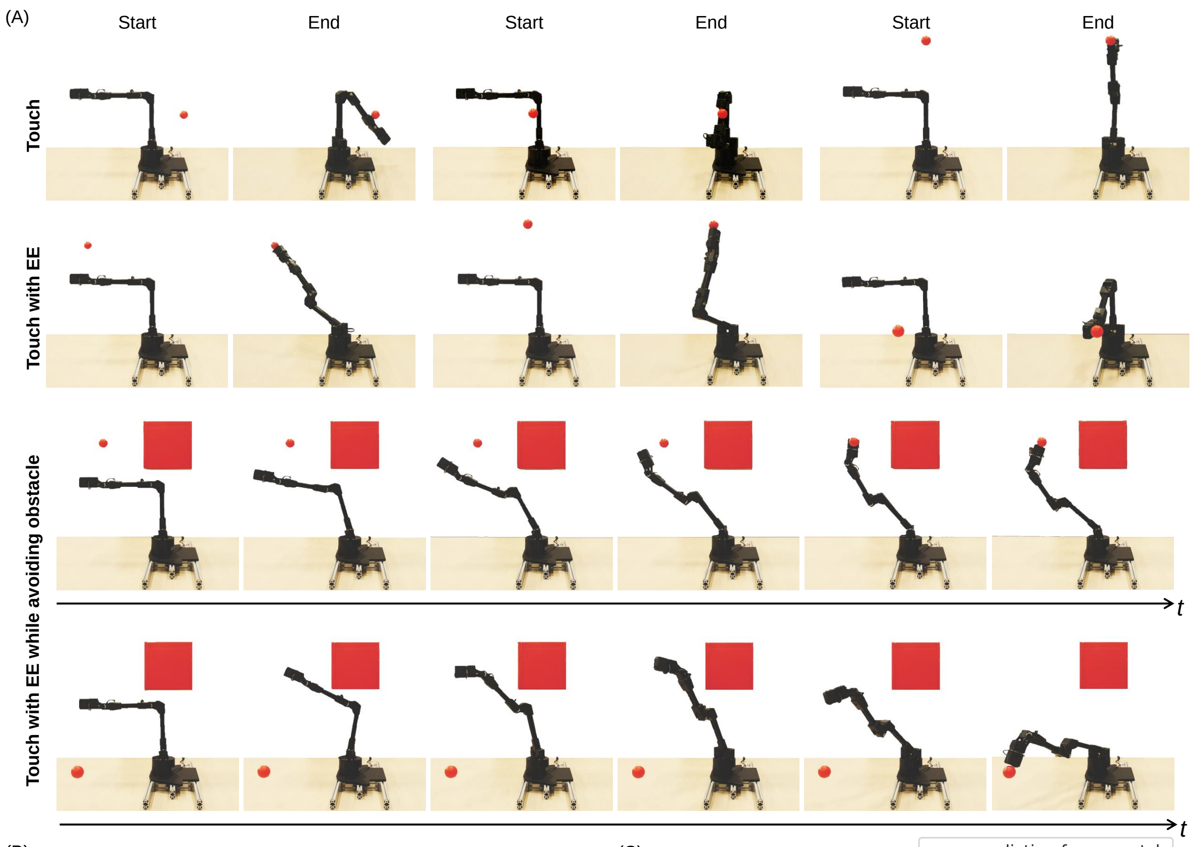
We presented the quantitative results in Fig.3.4 (A). Our method reached around 0.002 and 0.0102 meter Chamfer-L1 distance in simulated and real-world experiments, respectively. Our self-model outperformed both baselines suggesting that the our visual self-model learns the generalizable correspondence between the joint angles and the robot morphology as well as kinematics rather than memorizing the training set distribution. Since the workspace of the physical robot is around $0.9 \times 0.9 \times 0.9$ meter. Our visual self-model is accurate to about one percent of the workspace calculated as $0.0102/0.9 \approx 1.1\%$.

In addition to the predictions on individual set of joint angles, we also visualize the predictions over joint angle trajectories by linearly interpolating between sets of starting joint angles and sets of target joint angles. Both the starting and target joint angles are randomly sampled. As shown in Fig.3.5, our visual self-model can generate smooth interpolations of robot morphologies between small changes of joint angles. As we will show next, this property allows our visual self-model to generate accurate trajectories for downstream motion planning tasks.

3.2.2 3D Self-Aware Motion Planning

In this subsection, we aim to evaluate the performance of using our visual self-model and 3D Self-Aware motion planning algorithms for three representative downstream tasks: teach a 3D sphere with any part of the robot body, touch a 3D sphere with end effector and touch a 3D sphere with end effector while avoiding obstacle. Detailed illustrations of the tasks and algorithms have been discussed above. For all three tasks, we present qualitative visualizations of our solutions obtained through the visual self-model in the real-world system in Fig.3.6. We then introduce our quantitative evaluation results in the simulation setup.

For the “Touch a 3D sphere with any part of the body” task, our evaluation metric measures the Euclidean distance of the closest points between the robot surface and the target object surface. We sampled 100 tasks where the target sphere is placed at different 3D locations within the reachable space of the robot. If the robot was already in contact with the target sphere at initialization, we discarded that task and re-sampled another task. Our results are shown in Fig.3.6(B). We compared our visual self-model with several other baselines. To reflect the task difficulty, we first measured the initial distance between the robot surface body at its home location and all sampled target sphere surfaces. We also compared with a random trial baseline where the only input was also the joint angles, similarly to our visual self-model. In this case, the robot randomly selected a set of joint angles as its final solution. This baseline gave even worse performance than initial distance indicating that the robot needs to perform careful inverse kinematics calculation with considerations of its entire morphology and kinematics. Overall, our method produces much more



(B)

Table 1: Touch 3D Sphere

Method	Distance (meter)
Initial distance	0.1045 ± 0.0082
Random trial	0.1413 ± 0.0108
Visual self-model (ours)	0.0207 ± 0.0076

Table 2: Touch 3D Sphere with End Effector

Method	Distance (meter)
Initial distance	0.3624 ± 0.0163
Random trial	0.3731 ± 0.0191
End-effector prediction	0.2551 ± 0.0223
Visual self-model (ours), $\lambda_{\text{EE}} = 1.0$, $\lambda_{\text{SDF}} = 0.0$	0.0647 ± 0.0164
Visual self-model (ours), $\lambda_{\text{EE}} = 0.9$, $\lambda_{\text{SDF}} = 0.1$	0.0284 ± 0.0098
Visual self-model (ours), $\lambda_{\text{EE}} = 0.8$, $\lambda_{\text{SDF}} = 0.2$	**0.0274 ± 0.0070**
Visual self-model (ours), $\lambda_{\text{EE}} = 0.5$, $\lambda_{\text{SDF}} = 0.5$	0.0304 ± 0.0087
Visual self-model (ours), $\lambda_{\text{EE}} = 0.2$, $\lambda_{\text{SDF}} = 0.8$	0.0410 ± 0.0110

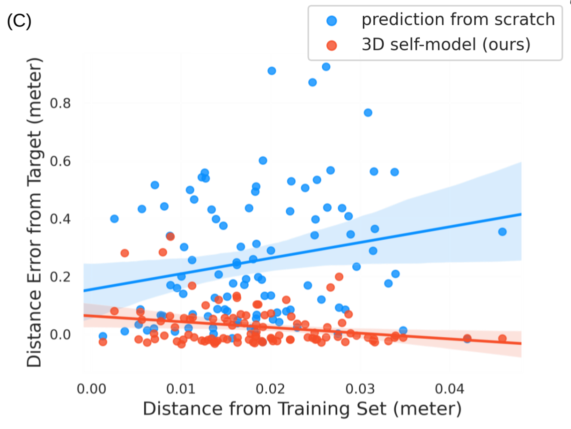


Fig. 3.6: **3D self-aware motion planning results.** (A) For each of the three tasks, we show the real-world demos by executing the proposed plans from our visual self-model. (B) Our visual self-model outperformed all the baselines by a large margin. Overall, our visual self-model can produce accurate solutions for both tasks. (C) We found that our visual self-model enables the kinematic network to gain better generalization performance on downstream tasks than a plain kinematic self-model trained from scratch.

accurate solutions. Furthermore, our method was also time efficient during the search stage. Each solution took 2.92 seconds on average on a single GPU after 500 optimization iterations. Note that the solutions generated in this sample application satisfy the constraints but are not necessarily optimal. Additional optimization criteria, such as energy or peak speed minimization could be added, and many established path planning techniques could be used. We leave such explorations for future work.

For the “Touch a 3D sphere with end effector task”, our evaluation metric measures the Euclidean distance between the end effector link and the closest point on the target sphere surface. We sampled 100 tasks and made sure that the robot was not in contact with the target sphere at its home configuration. Our results are shown in Fig.3.6(B). Similar to the first task, we compared our approach with the initial distance and random trial baselines. Both baselines were poor at this task with about 36 cm to 37 cm errors. This is even worse than the first task because the presented task requires more accurate solutions to consider both the 3D body geometry and the end effector position.

We have hypothesized that our visual self-model encourages strong semantic knowledge of robot kinematics in the kinematic sub-network. To verify this hypothesize, we re-used the pre-trained weights of the kinematic sub-network, and appended two nonlinear layers of MLPs to perform further training only on the newly added layers, in order to regress the end effector link position. The quantity of the data and the strategy of data splits followed the same definition with our original visual self-model. The test error was around 0.5cm. We also trained a network with the exact same architecture without pre-trained weights from our visual self-model to predict the end effector position. The test error of this model was 1.3cm which was nearly three times higher. Moreover, when applying these two models separately with our motion planning pipeline in Fig.3.6(B), our method reached nearly ten times higher accuracy than the model trained from scratch denoted as “end-effector prediction” in the table. These results suggest the importance of considering the kinematic structure of the robot together with its 3D morphology. In terms of time efficiency, our method took 4.93 seconds on average on a single GPU after 500 optimization iterations because of the fast parallel inference property.

Furthermore, we found that learning the kinematic structure, together with our visual self-model to learn the entire robot morphology, brought stronger generalization capability to downstream tasks. In Fig.3.6(C), every dot represents a task sample. The y-axis indicates the error measurements on the task of “touch a 3D sphere with end effector”, and the x-axis denotes the closest distance between each sampled task and their nearest neighbor in the training set. Larger values on the x-axis means that the sampled task is farther away from the training data distribution. Therefore, the errors of the method with strong generalization capability should not raise with the increased distance from the training data distribution. We thus also plotted a linear regression model fit in the same figure. By comparing our visual self-model denoted as red dots and the model trained from scratch indicated as blue dots, we can tell that our visual self-model obtains a much stronger generalization capability, while the model trained from scratch will have a much higher error when the data is away from the training set.

Finally, we also provide results of using different values of λ_{Link} and λ_{SDF} in the objective function. We found that $\lambda_{\text{Link}} = 0.8$ and $\lambda_{\text{SDF}} = 0.2$ gives the best results. Therefore, adding a small regularization with the original SDF objective can help achieve better performance in this task.

For the “Touch a 3D sphere with end effector while avoiding obstacle” task, since the target joint states are generated and evaluated through the above task, we are now interested in evaluating the capability of generating collision-free trajectory when combing existing motion planners with our visual self-model as collision prediction function. Again, we sampled 100 tasks with initial states being contact free with the robot body. We placed a block 40cm above the robot base as the obstacle object. The block has a dimension of 20cm $\times$ 20cm $\times$ 20cm. In total, after running the motion planner with our visual self-model, we received 95 out of 100 trajectories which the model believes no collision will happen along each trajectory. We then executed these trajectories and found that 92 out of the 95 trajectories successfully passed around the obstacle towards the target object without any collision. This is 96.84% success rate over all the output trajectories. Our method took 7.43 seconds on average to produce an entire trajectory which includes the time for both inferring the target state as presented in the second task and running the motion planners. This fast inference time enables our method to provide real-time planning and control solutions.

3.2.3 Resiliency Tests

Being able to identify potential damages or changes to the robot body and quickly recover from these changes is a critical capability of intelligent machines in the real world. We made two type of changes to the robot body as depicted by Fig.3.7(A). In the first change, we broke the second motor to the end effector link by disconnecting the data transfer cable from the motor, which results the corresponding joint always stayed at 90°. Motor broken can happen due to various reasons such as loosing cables, over heating or hardware damage, but the common resulted observation is that the motor does not respond to any commands. The second change applies to the topology change of the end effector link. We attached a 3D-printed plastic stick to the end effector so that the reachable space of the robot arm was extended. This is also a representative change in practical applications when different tasks demand new attachments of tools to the robot body or switch different grippers on a robot arm.

With our proposed algorithm and the learned visual self-model, we tested the applicability of our method directly on these real world changes. Fig.3.8 presents several example results. The first step is to detect the change. As shown in the first column, our algorithm detected a clear gap between the original prediction errors and the current prediction errors. The obvious gaps suggest that our visual self-model can capture the changes happening on the robot body.

The second step is to identify the specific type of change. In the first two examples, no matter what the input commands were to the robot, the second last joint was always inferred to be around 90° by solving the inverse problem with the newly observed morphology. This consistent mismatch indicates that the second last motor was broken and the angle stayed at 90°. In the last two examples, even though we can detect that there were some changes from the first step, the inferred joint angles were still well-aligned with the input commands. Following our discussions earlier, our algorithm identified that there was a topology change on the robot body. Our results suggest that our visual self-model can be used to effectively solve inverse problems to help identify what body change or damage might have taken place. Importantly, our approach only requires a single 3D observation of the current robot to produce the above results to detect and further identify the

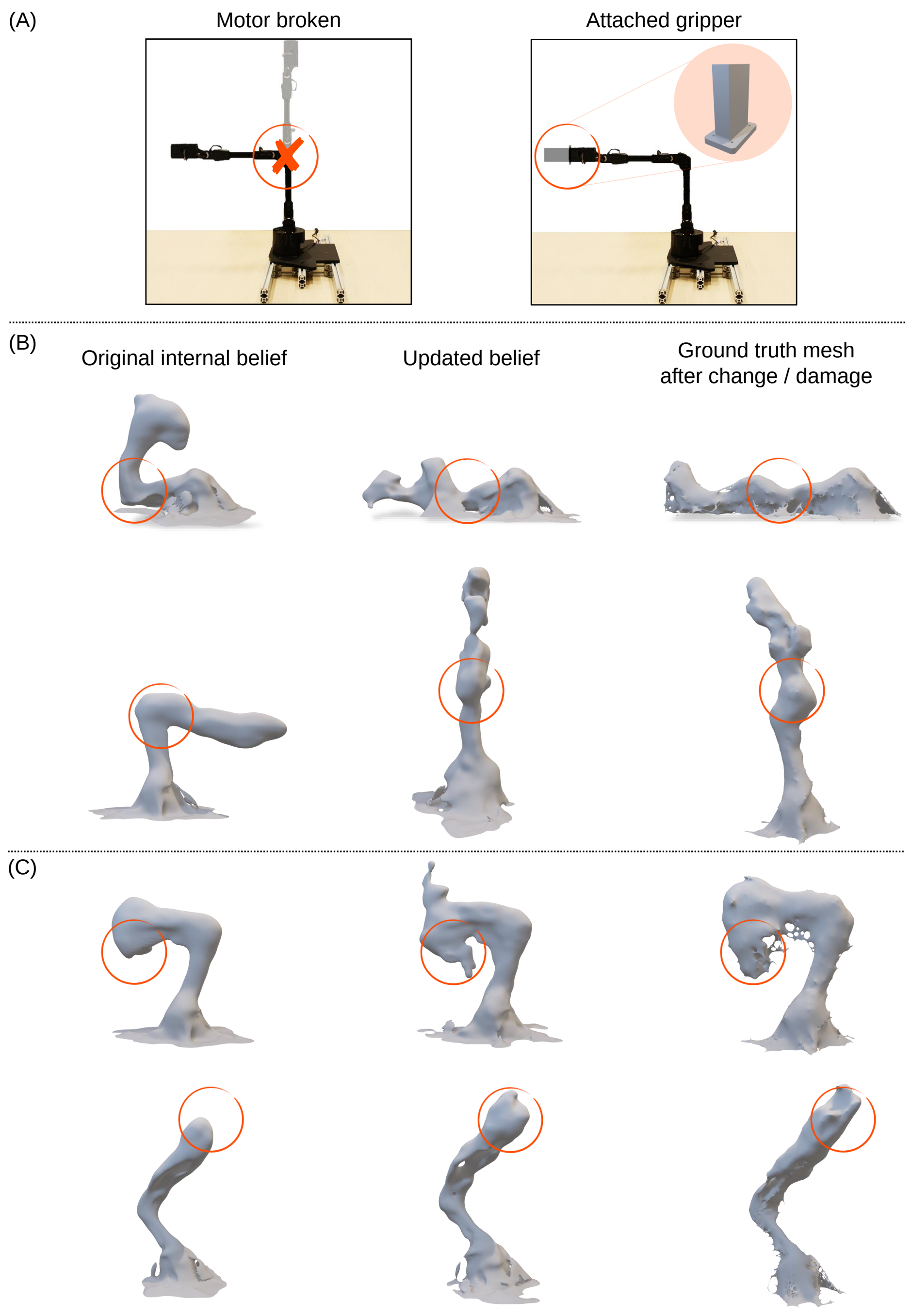


Fig. 3.7: **Potential change or damage on the robot and visualizations** (A) Two types of potential changes. The left scenario is motor broken where the joint will always stay at 90°. In the right scenario, we attached a 3D-printed plastic stick. (B) Motor broken: we can visualize the robot's original internal belief, its updated belief after continual learning and the current robot morphology. (C) Extended robot link visualizations.

damage.

In the final step, our goal is to evaluate whether our visual self-model can quickly recover from

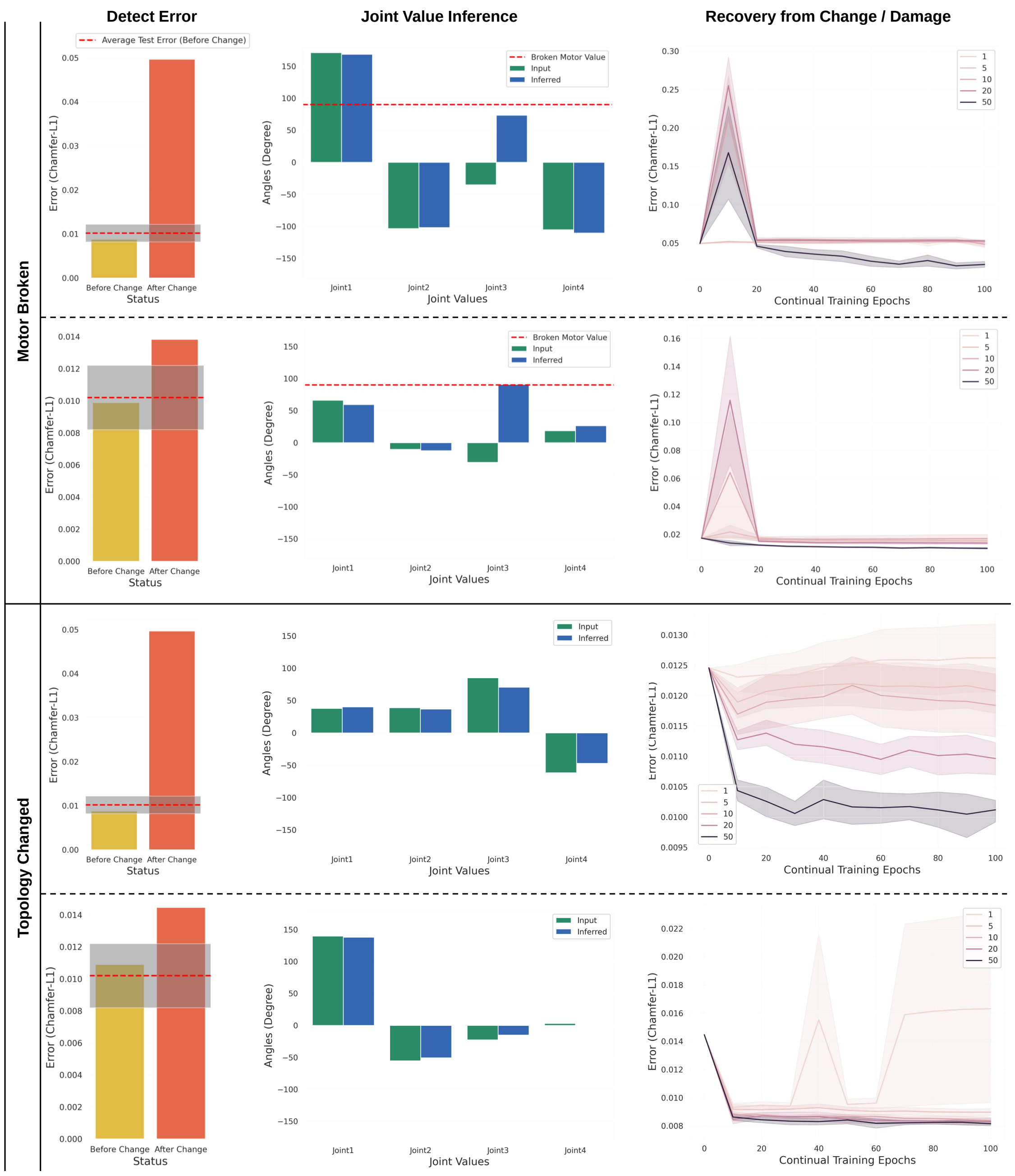


Fig. 3.8: **Resiliency Tests.** In the first column, the learned visual self-model can detect the change or damage through the large error gap. In the middle column, the learn3d visual self-model can identify the specific type of change through the mismatch between the input joint values and the inferred joint values. In the last column, we show how the visual self-model can update its internal belief to match the current robot morphology.

the detected changes with only a few new observations. We first collected a few more observations of the current robot through random movements. With the new observations, we used them as the training data to continue the training of our existing visual self-model. Fig.3.8 plots the intermediate model performances on the test instance at every 10 epochs. We found that our model required 50 examples to converge. Our visual self-model can quickly recover with the new training data after 100 epochs which took on average 8.13 minutes in the real world on a single GPU.

Another advantage of our visual self-model is its interpretability. In Fig.3.7(B), we can visualize the internal belief of the robot before and after the damage adaption. Through these visualizations, we can inspect what the robot's internal belief looks like and whether the robot has successfully updated its belief to match the current robot morphology. These visualizations can be queried in an online fashion with about 2.4 seconds on a single GPU.

Chapter 4: Robot Modeling the Behavior of Other Robots

Starting from this chapter, we go beyond the self-modeling of the robot embodiment and study how to generalize the self-modeling techniques to modeling of other robots. In this chapter, we will first introduce how robots can model the behavior of other robots [125]. We will then follow with the next chapter on how robots can model the visual perspective of other robots.

Behavior Modeling is an essential cognitive ability that underlies many aspects of human and animal social behavior [136], and an ability we would like to endow robots. Most studies of machine behavior modeling, however, rely on symbolic or selected parametric sensory inputs and built-in knowledge relevant to a given task. Here, we propose that an observer can model the behavior of an actor through visual processing alone, without any prior symbolic information and assumptions about relevant inputs. To test this hypothesis, we designed a non-verbal non-symbolic robotic experiment in which an observer must visualize future plans of an actor robot, based only on an image depicting the initial scene of the actor robot. We found that an AI-observer is able to visualize the future plans of the actor with 98.5% success across four different activities, even when the activity is not known a-priori. We hypothesize that such visual behavior modeling is an essential cognitive ability that will allow machines to understand and coordinate with surrounding agents, while sidestepping the notorious symbol grounding problem. Through a false-belief test, we suggest that this approach may be a precursor to the Theory of Mind, one of the distinguishing hallmarks of primate social cognition.

4.1 Background: Robot Theory of Mind

At about age three, a human child recognizes that other humans may have a differing world view than herself [137]. She will learn that a toy can be hidden from a caretaker who is not present in the room, or that other people may not share the same desires and plans as she does. The ability to recognize that different agents have different mental states, goals, and plans is often referred to as “Theory of Mind” (ToM). In children, the capacity for ToM can lead to playful activities such as “hide and seek”, as well as more sophisticated manipulations such as lying [138]. More broadly, ToM is recognized as a key distinguishing hallmark of human and primate cognition, and a factor that is essential for complex and adaptive social interactions such as cooperation, competition, empathy and deception.

The origins of ToM are difficult to ascertain, because cognitive processes leave no fossil record. In this work, we experimentally look for evidence that ToM would have been preceded by a much simpler precursor which we call “visual behavior modeling”. Researchers typically refer to the two agents engaged in Behavior Modeling or ToM as “actor” and “observer.” The actor behaves in some way based on its own perception of the world. The observer watches the actor and forms an understanding of the mental state held by the actor and/or the resulting actions that the actor intends to take. In the simplest case, the actor behaves deterministically, and the observer has full knowledge of the world external to the actor. In more complex cases, observers can have partial knowledge, and agents can be both actors and observers at the same time, simultaneously modeling each other, leading to infinite regress of nested co-adapting models. Here, we study only the simplest case, in both deterministic and stochastic conditions.

Theory of Behavior and ToM experiments are notoriously difficult to carry out because it is very challenging for a researcher to query the state of mind of the observer, in order to determine if the observer truly understands actor’s mental state and plans. In older children and adults, the state of mind of the observer can be queried directly by formulating a verbal question about the actor, such as “Tell me where Sally will look for the ball?” as in the classic “Sally and Anne” test [139, 140, 141].

In young children and primates, as well as in robots, the state of mind of the observer can only be indirectly inferred by inducing an action that reveals the observer’s understanding of the state of mind of the actor. For example, a researcher might ask the child to point at the box where Sally will look for the ball. A key challenge is that in order to be able to answer a researcher's question or to follow the researcher's instructions, the observer must already possess an advanced cognitive ability. For example, the observer must at least be able to understand symbolic questions, make decisions, and formulate responses. The ability to follow instructions, in itself, involves a fairly advanced degree of mental reasoning. This challenge makes it difficult to ascertain the degree to which ToM exists in simple life forms and in robots.

Meltzoff [142] noted this challenge and proposed the need for non-verbal tests to assess theory of mind. For example, one non-verbal assessment of ToM involves testing whether a child will mobilize to assist another person that appears to be struggling to open a door. Even though these tests are carried out without verbal communication, it is difficult and sometimes impossible for a researcher to conclude with absolute certainty whether the observers performed explicit reasoning while addressing the challenge, or acted instinctively out of some built-in conditional reflex.

Experiments in ToM of robots and AI agents suffer a similar challenge. In a recent set of impressive studies, such as Rabinowitz et al.[48] agents were able to successfully model the behavior of other agents on a grid world. Baker et al. [143] also demonstrated that complex tool manipulation and coordination strategies can emerge by having adversarial agents play hide and seek in a self-play manner. There are many other examples of AI agents that learn to play successfully against other agents (human and machine) in various video games [53], frequently outperforming their competitors with impressive gaps. However, in those studies, the players' inputs involve engineered features and discrete outputs, in the form of grounded symbols, objects, and actions, such as coordinates of agents and obstacles, and discrete actions such as move forward, backward, left or right. This raises the next fundamental question which is where these discrete relevant inputs and actions choices came from, and how were they grounded in the first place.

In some past studies [15], the observers' inputs included private internal actor states such as motor commands, that would normally be hidden from an external observer, and therefore could not be used legitimately in a ToM experiment, We focus here on observers that only have access to raw data captured externally using remote sensors, such as cameras and microphones.

Many AI agents use only visual inputs, but most do not explicitly model the behavior of the actor. Instead, they select from a finite repertoire of possible actions that will move them closer towards a goal. For example, Mnih et al [53] showed an impressive example of an agent that learns to play Atari games using only visual inputs, by producing discrete actions that maximize its score. Our own recent work [144] also shows that having agents play hide and seek with only first-person visual observation can lead to the recognition of the visibility of other agents and self-visibility to emerge, but the action space of the observer agent is still discrete. There is no direct evidence that either system models the plans of the other system. The question of visual behavior modeling is also interesting from an evolutionary point of view. There is no doubt the ToM that involves explicit modeling of actor plans confers an evolutionary advantage to the observer, and may have therefore evolved directly. Foreseeing the plans of an actor can help with many activities beyond competition, such as imitation learning, or cooperation. However, here we explore a precursor question: Can machines intelligently model behavior of other agents without access to explicit symbols, objects and actions?

Being able to model an actor without any explicit symbolic processing could shed light on the evolutionary origin of ToM, as well as offer an alternative path to train machines without the need for feature engineering, symbolic mechanisms, and sidestepping the inductive bias involved in deciding which symbols, objects, and actions need to be built into the system.

To test the hypothesis that longer term outcome of behavior can be predicted visually, we focus here on a more abstract notion termed "Theory of Behavior" (ToB). As shown in Fig. 1, we ask whether an observer can predict the conditional (and possibly stochastic) behavior of an actor based on visual observation of the actor alone, without access to the actor's internal states, prior knowledge about the behavior. We do not ask whether the observer can ascertain the explicit goal or objective of the actor. We avoid this question for two reasons: First, the explicit goal, if any, of the actor is a hidden variable that may or may not exist, and is subject to interpretation. Second, to process an explicit goal presumes preadapted ability to reason symbolically. In contrast, we hypothesize that the actor can develop the ability to visualize goal-oriented behavior even without

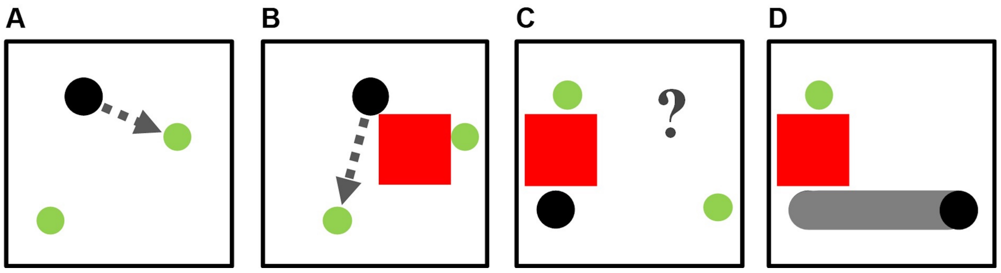


Fig. 4.1: **Visual Theory of Behavior** An actor robot (black circle) is programmed to move towards the nearest food (green circle) that it can see, and consume it. Sometimes (a), the nearest green circle is directly visible to the actor, but sometimes (b) the nearest green circle is occluded by an obstacle. When occluded, the actor will move towards the closest visible circle, if any. After watching the actor act in various situations, an observer-AI learns to envision what the actor robot will do in a new, unseen situation (c). The observer's prediction is delivered as a visualization of the actor robot's "trajectory smear" (d). This entire reasoning process is done visually, sidestepping the need for symbols, logic, or semantic reasoning.

having any internal notion of goals at all, and without using symbolic processing ability at all. If true, this suggests a visual precursor to theory of mind.

Our goal is to see if the observer can predict the long-term outcome of the actor's behavior, rather than simply the next frame in a video sequence, as do many frame-to-frame predictors [15, 6, 7, 8, 9, 10, 11, 12, 13, 14]. Prediction of next set of frames in a video sequence is both more difficult and less important to a robot that predicting the longer-term outcome of an actor's behavior. For example, a colleague that watches you rise out of your seat with an empty cup of coffee and head towards the coffee machine, cannot predict your exact movements frame by frame, but can likely predict that you will soon be back sitting at your desk with a refilled cup of coffee. We argue that this precise ability – to envision the outcome of a behavior - is the basis of TOM, not the frame-by-frame prediction that is difficult, yet seems to dominate prior work in visual prediction (probably because there is plenty of data for it).

4.2 Visual Behavior Modeling

4.2.1 Task Setup

In our experiment, we created a robotics system (Fig. 4.2) comprising a physical robot actor, and an observer agent that watches the actor from above. In order to evaluate the effectiveness of our proposed method, we followed the guidelines proposed by Shevlin and Halina [145] to design our evaluations from both a Machine Learning perspective and a Psychology perspective. During our experiments, we aimed to answer the following two questions: First, can an observer agent visually predict the future plans of the actor robot, without symbolic reasoning. Second, we are interested in understanding whether the observer agent gains some equivalent to perspective taking abilities.

To test whether the observer can model the behavior of the actor robot from purely visual input, we first pre-programmed (hard-coded) four types of behavior into the actor robot. Without being provided any symbolic information regarding which category of the current actor's behavior is, the observer machine is asked to generate its envisioned image of what the future scene will look like, including the trajectory and the consequences caused by the actor robot's behavior. The result is outputted by the observer as a predicted image frame.

The four types of pre-programmed behavior of the actor robot were: (a) Straight line behavior: the actor robot goes to the green circle in a straight line; (b) Elbow behavior: the actor robot goes to the green circle by first navigating towards an intermediate non-colinear point that is located midway and then moves to the green circle in a straight line form there; (c) Zig-zag behavior: the actor robot goes to the circle by navigating through two intermediated non-collinear control points positioned symmetrically on both sides of the straight line path to the goal; and (d) Single-food with obstacle behavior: this behavior is a more complex version of behavior (a) due to occlusion caused by an extra obstacle. In behavior (d), the actor robot goes to the green circle in a straight line if and only if the green circle is visible to it. However, if the green circle is not visible to the actor robot due to the occlusion, the actor robot will not move. When determining visibility, we

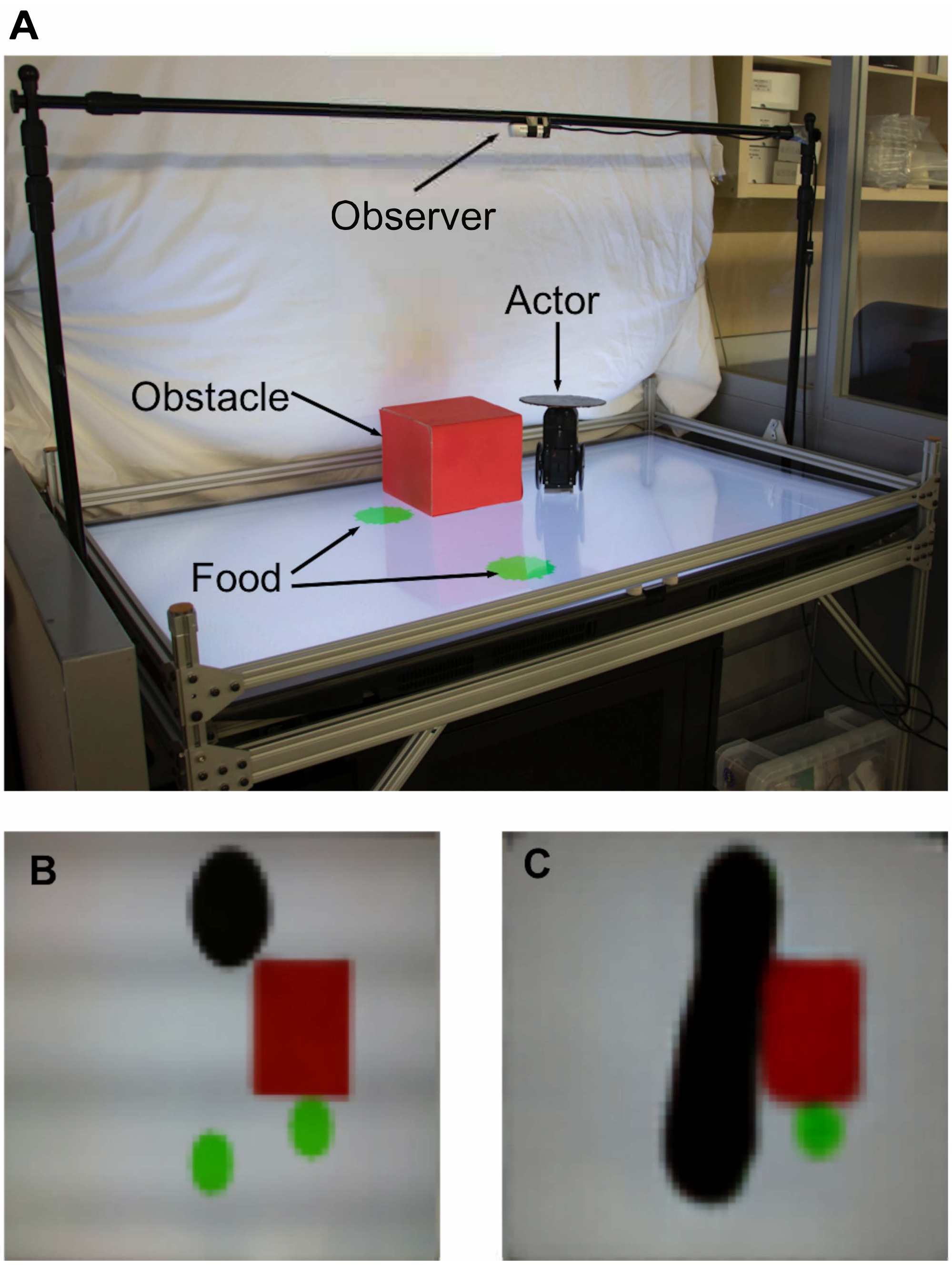


Fig. 4.2: **Experimental setup.** (A) Actor robot in playpen, showing observer, green food, and red obstacle. (B) A sample input image as seen by the observer, and (C)sample output image produced by the observer, which includes a prediction of the motion path of the actor.

use a line of sight as the line joining the center of the actor with the center of each food circle, and test computationally whether that line intersects with any obstacle. The actor does not require any physical sensor in this experiment.

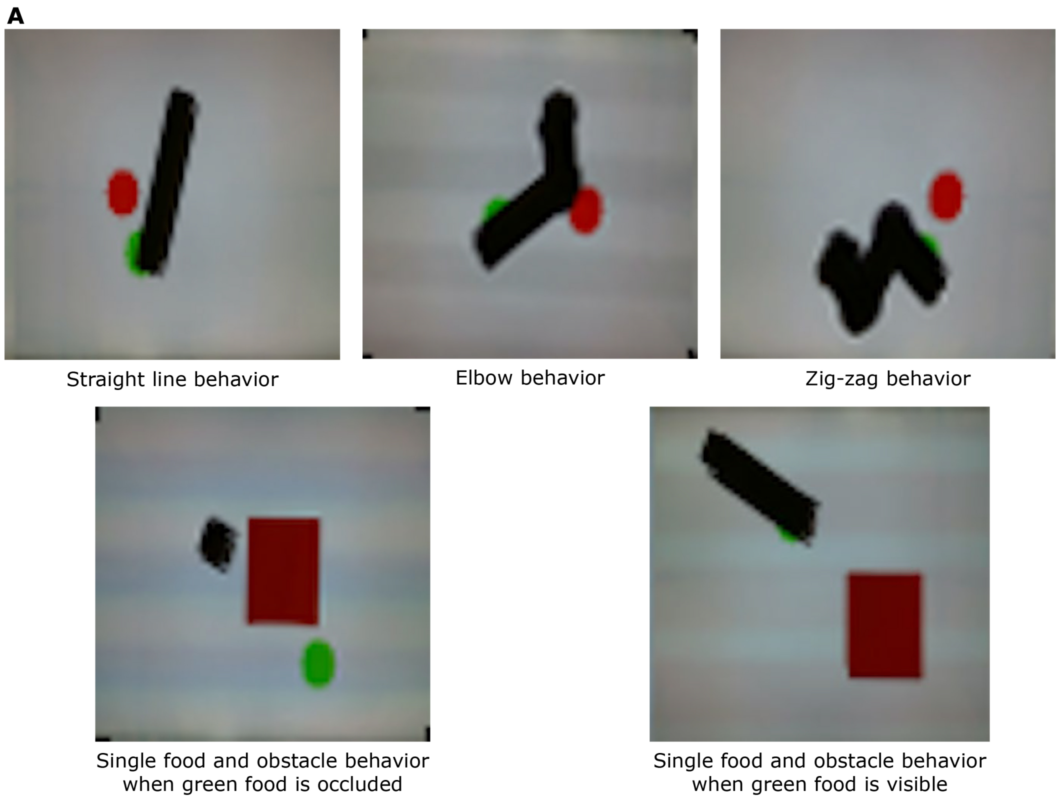
We show samples of these four behaviors in Fig. 4.3A, using actual camera shots of the smeared-motion of the robot as seen from the point of view of the observer camera. We use the term "smeared motion" to denote an integration of multiple video frames into a single image by recording the minimum intensity of each pixel across its time history. Thus, a dark robot moving along a trajectory will leave a dark path in its wake.

While some of the four actor's policies described above involve nested logical clauses, we show that the observer can envision the outcome of those actions without using any explicit logic reasoning, or any explicit notion of objects, obstacles, actions and goals.

The observer AI can only see the world external to the actor using an overhead camera that captures the actor, the green circles and the red obstacle (if any). Importantly, the only information available to the observer are raw camera input images. No labeling, segmentation, motor commands, trajectory coordinates or other derived information is provided to the observer other than the raw camera image itself. The observer is not told, for example, that green pixels are associated with goals, or that colors and shapes even matter. The key question is whether the observer will learn that whether and where the actor moves depends on what the actor can see from the point of view of the actor, not what the observer itself sees.

Importantly, the observer does not output a symbolic answer, such as which green circle will be consumed, which direction the actor will move, or whether or not the actor will move. Instead, the observer envisions the outcome of the actor's actions holistically by producing a single image of the resulting world, without any explicit knowledge of circles, obstacles, actions or trajectories and without any explicit symbolic reasoning.

The behavior of the actor robot was pre-programmed using hard-coded logic, based on its personal world view. The observer AI was represented as a deep neural network that receives the overhead image as input and produces a prediction of the resulting actor behavior as output. Each input scene is a 64×64 pixel image. The output envisioned by the observer is also a 64×64 pixel image that depicts the actor's behavior as a motion smear and removing consumed food circles, if any.


A
Straight line behavior
Elbow behavior
Zig-zag behavior
Single food and obstacle behavior when green food is occluded
Single food and obstacle behavior when green food is visible

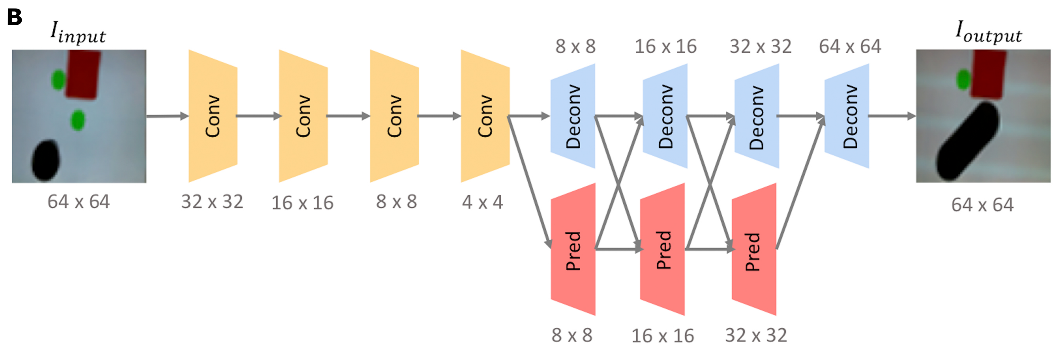


Fig. 4.3: **Visualization of the behaviors of the actor robots and Observer network architecture.** (A) We pre-programmed four types of behaviors for the actor robot. The images shown here are produced by integrating a sequence of frames from a video captured by the top-down camera. The robot path is shown in black, the rectangular obstacle (if any) in red and the goal circles in green or red. (B) The image prediction network is composed of several layers of convolutional units and deconvolutional units. At the deconvolutional stage, we utilize multi-scale prediction to maintain high resolution at the output image. Numbers indicate the dimension of output feature map after each module.

4.2.2 Network Design

The architecture of our network is a fully convolutional multi-scale vision prediction (image-to-image) network, comprising 12 fully convolutional layers with 609,984 neurons and 570K parameters. Such networks are often used for predicting the next frame in a video. However, instead of predicting the next frame in a series of frames, we required the network to predict the last frame based on an image created by integrating the first half of the frame sequence (for the first three policies of the actor robot) or just the first frame (for the fourth policy of the actor robot). This is akin to asking a person to predict "how the movie will end" based on the opening scene.

Specifically, our network comprises two main parts: an encoder network and a multi-scale decoder network. The encoder network is composed of several convolutional layers where each convolutional layer is followed by a batch normalization layer and a ReLU non-linear activation function. Such a unit is denoted as a whole as "Conv" block in Fig. 4.3B. It has been demonstrated [6, 126] that a multi-scale convolutional network is useful for refining coarse feature representations to higher resolution.

Inspired by these prior works, we used similar multi-scale prediction networks within the decoder network. Each input image is fed into a "Pred" block which is a convolutional layer followed by a transposed convolutional layer and a Sigmoid non-linear activation function to generate a sub-sampled predicted feature map.

This predicted feature map is then concatenated with the output from a "Deconv" block to be fed into the next stage. The output from each stage is sent into both the "Deconv" unit and the "Pred" unit to get the feature predictions of the next stage. By the end of the decoder network, the "Deconv" will generate the final output which has the same size as the input image. Similar to the "Conv" unit, each "Deconv" block in Fig. 4.3B is composed of a transposed convolutional layer and a Sigmoid non-linear activation function. We minimize a Mean Squared Error loss on pixel values for training our network.

4.3 Experiments

4.3.1 Modeling Results

The observer AI was trained using 2,400 input-output image pairs recorded experimentally, of which 20 percent were used for testing. The input image is a single frame with the projection of the first half of the trajectory of the actor robot, and the output image is the final frame with the project of the entire trajectory of the actor robot. Both the training and testing data were balanced across all four types of behaviors of the actor robot. Each original image pair was expanded into 10 additional cases using random rotation and symmetry transformations with 0.5 probability of each, in order to augment the training set and testing set. The augmentation was done after splitting into training and testing set so as to not cross-contaminate the training and testing datasets. The resulting performance reported in Fig. 4.4A was measured on a withheld test set of 1,200 image pairs for each of the four policies of the actor robot.

Fig. 4.4B shows qualitative examples of various raw input scenes as seen by the observer AI, and both successful and failed outcomes it envisioned. When the observer envisions that a green circle is consumed, it clears most of the green pixels. If the observer AI believes that the actor will move, the envisioned the trajectory of the actor robot as a motion smear. We emphasize again that the observer has no explicit notion of food, obstacles, actions, coordinated, position, policies, line of sight etc. It only sees an image of the world and envisions an image of the future. The observer did not know which policy was being employed by the actor.

The proposed visual Theory of Behavior was considered experimentally successful if and only if the scene envisioned by the observer had correctly erased the food consumed by the actor, and also correctly visualized the actor's motion trajectory. To evaluate whether the observer has successfully foreseen the high-level behavior of the actor robot, we automatically processed the images produced by the observer using classical machine vision techniques. We extracted both the largest contours of the ground truth and predicted trajectory, as well as the position of the food circles using their colors measured from the ground truth image. We calculated the shortest distance

A

Actor Behavior	Success Rate
Straight line behavior	99.90% (n=980)
Elbow behavior	98.94% (n=944)
Zia-zag behavior	96.70% (n=999)
Single food with obstacle behavior	98.52% (n=811)

B

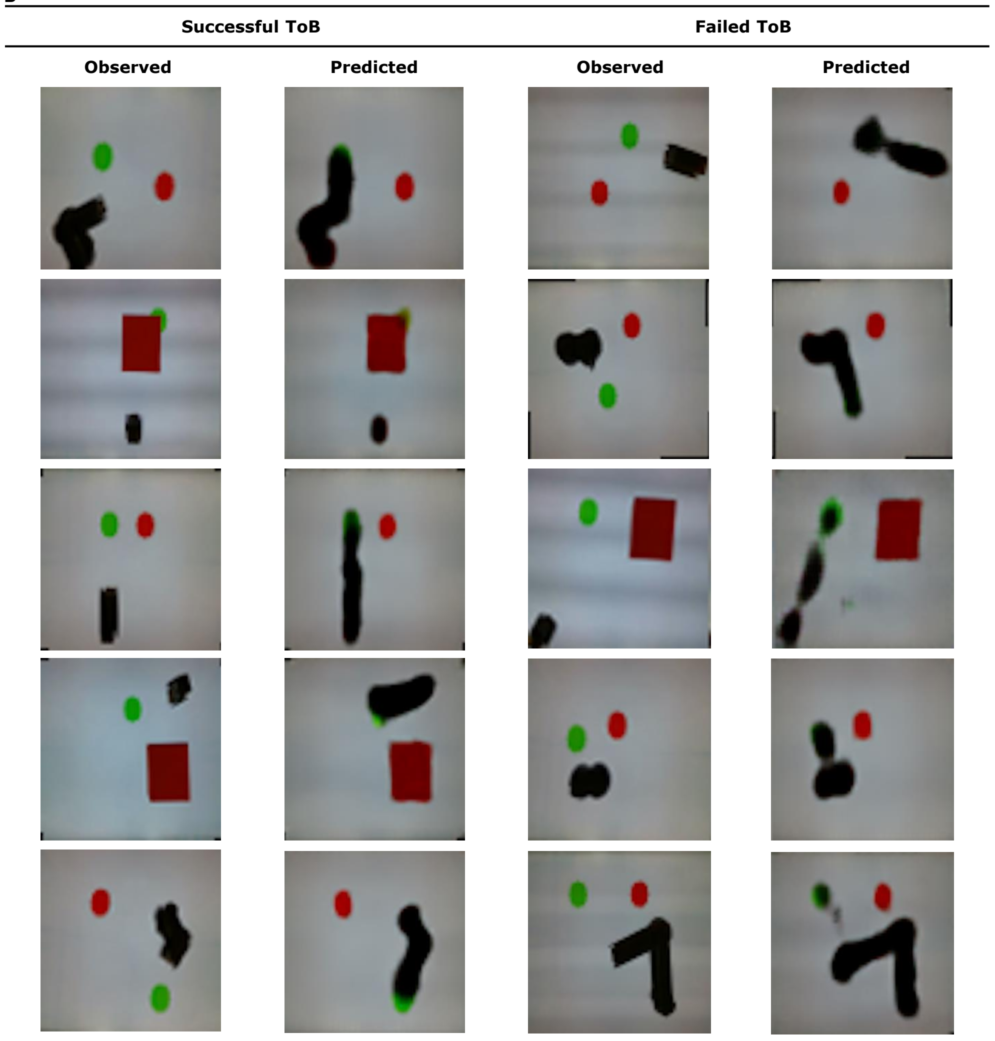


Fig. 4.4: **Success Rate of Observer AI and Physical scenes and outcomes envisioned by the observer.** (A) We report the successful rate of the observer AI for each type of behavior during testing. Noted that all the behaviors of the actor robot are seen during training together and no other information is given except the single image frame. Our observer AI achieves a 98.5% success rate on average across all four types of behaviors of the actor robot. (B) The first column and the third column show some sample scenarios involving the actor robot, one or two green circles, and a square obstacle. The second and the fourth column show the outcome as envisioned by the observer. The left two columns (Success) show successful vision, whereas the right two columns (Failure) show failed envision.

from all the points on the contours to the center of the green food (target), we define this value as D_{target}.

In 6.65% of the cases, the machine vision algorithm used to evaluate success or failure automatically was unable to reliably determine the contour or the position of the foods in the ground truth images, and therefore was unable to determine whether or not the behavior was "successful". These cases were largely caused by a particularly short length of the trajectory leading to situations where the robot already covers a substantial part of the food due to the angle of the camera view. These cases were automatically omitted from the statistical analysis.

For the remaining 93.35% of cases where success or failure can be determined reliably, we compute the success rate by checking if the envisioned actor location is within one robot-diameter from the real actor position. In other words, for each predicted output, if D_{target} is smaller or equal than the diameter of the Actor robot, we mark that prediction as a success. Otherwise, we mark it as a failure. Then the success rate is calculated by dividing the total number of successful predictions by the total number of predictions. Each individual policy is evaluated separately during testing. However, the category of policies is never used during training.

As shown in Fig. 4.4A, our observer AI is able to achieve a 98.52% accuracy on average across all four types of behaviors of the actor robot, without knowing in advance which policy will be used. Combing with the qualitative results shown in Fig. 4.4B, we show that our observer AI is able to understand and predict the behavior of the actor robot without any symbolic representations.

4.3.2 Handling Ambiguity in Visual Behavior Modeling

An interesting question is whether the observer can handle ambiguity. For example, ambiguity may arise from stochastic behavior of the actor or from sensory noise. To test this, instead of integrating the first half of the trajectories of the first three policies of the actor robot into one single input image, we only give the first image as the visual input for all four policies to the observer AI. Now the training input becomes just a single frame from the initial scene, and the output to be predicted is a single image that should integrate all future frames from the first frame to the last frame, across the entire yet-unseen trajectory of the actor robot.

Under this setting, there are multiple possible future behaviors of the actor robot when given only single starting frame. The question is not whether or not the prediction will be correct (as it is impossible), but rather how with the observer represent the ambiguity. Fig. 4.5 shows that the observer AI handle the uncertainty by generating blurry, foggy images.

We also wanted to investigate if the level of blurriness is reduced as the observer is given more frames from the activity of the actor, thereby reducing the uncertainty. We fed the observer input image that integrated increasingly longer series of frames up to the current time stamp. In Fig. 4.5, we show the sequence of input and output images subject to uncertainty.

4.3.3 False-Belief Test

To ensure that our proposed Visual Theory of Behavior is consistent with protocols commonly practiced in studies of Perspective Taking or Theory of Mind, we further designed our experiments to include a false belief test. In this case, the actor robot has one simple, pre-programmed (hard coded) behavior. The actor robot always moves towards the nearest green circle that it can see with a straight line of sight. When two circles are visible, the actor will move towards the closest circle. However, if the closest circle is hidden behind an obstacle, such as a red box, then the actor will not see the green circle and will therefore not move towards it. Instead the actor might move towards a different circle that is visible. Similarly, Fig. 4.6 shows the qualitative examples of the visual observations as well as predictions envisioned by the observer AI. The observer AI was trained using 600 input-output pairs which then are augmented to 6,000 image pairs in the same way as indicated above.

There are mainly two types of sub-behaviors under this single policy of the actor robot. We call it a “food visible” case when the actor robot perceives the physically closest food and hence moves towards it. Alternatively, we call it a “food obscured” case when the actor robot is unable to see the physically closest food due to the occlusion by an obstacle, and only consumes the visible food which is further from it.

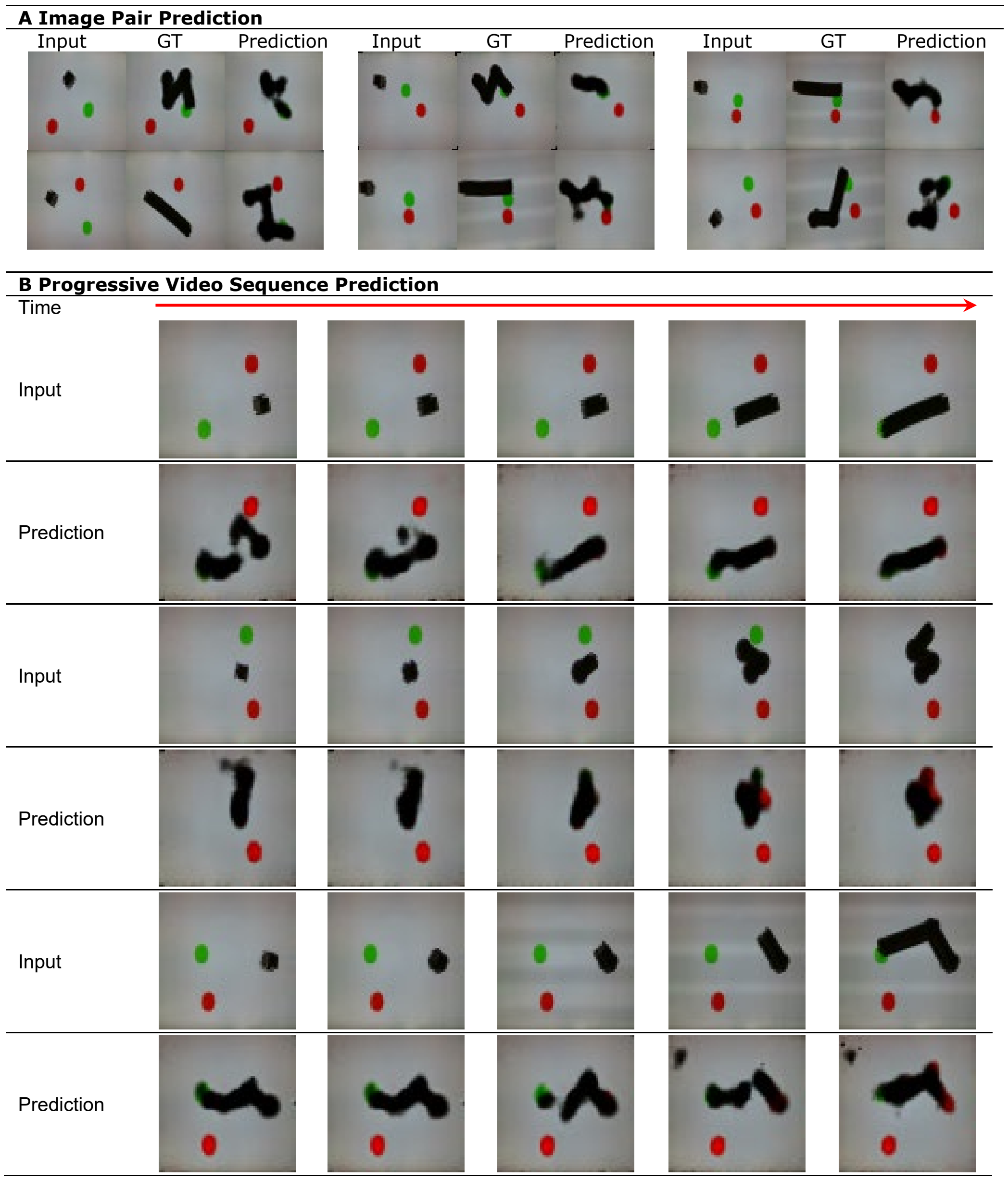


Fig. 4.5: **Handling ambiguity.** This observer AI is trained only on the first frame as input. Therefore, there are multiple possible future trajectories possible based on only this input frame. As shown in A. Image Pair Prediction, the observer handles this ambiguity by outputting one of the heuristic behaviors or as a blurry trajectory. Although our observer model is trained using start and end image pairs (with integration of all the past frames), the observer can be used in an online fashion during evaluation. B. Progressive Video Sequence Prediction shows the prediction results using our model across multiple time stamps.

This experimental design can be seen as a false belief test. In the first step, we train the observer agent only with the "food visible" cases where the observer only sees the actor moves towards the

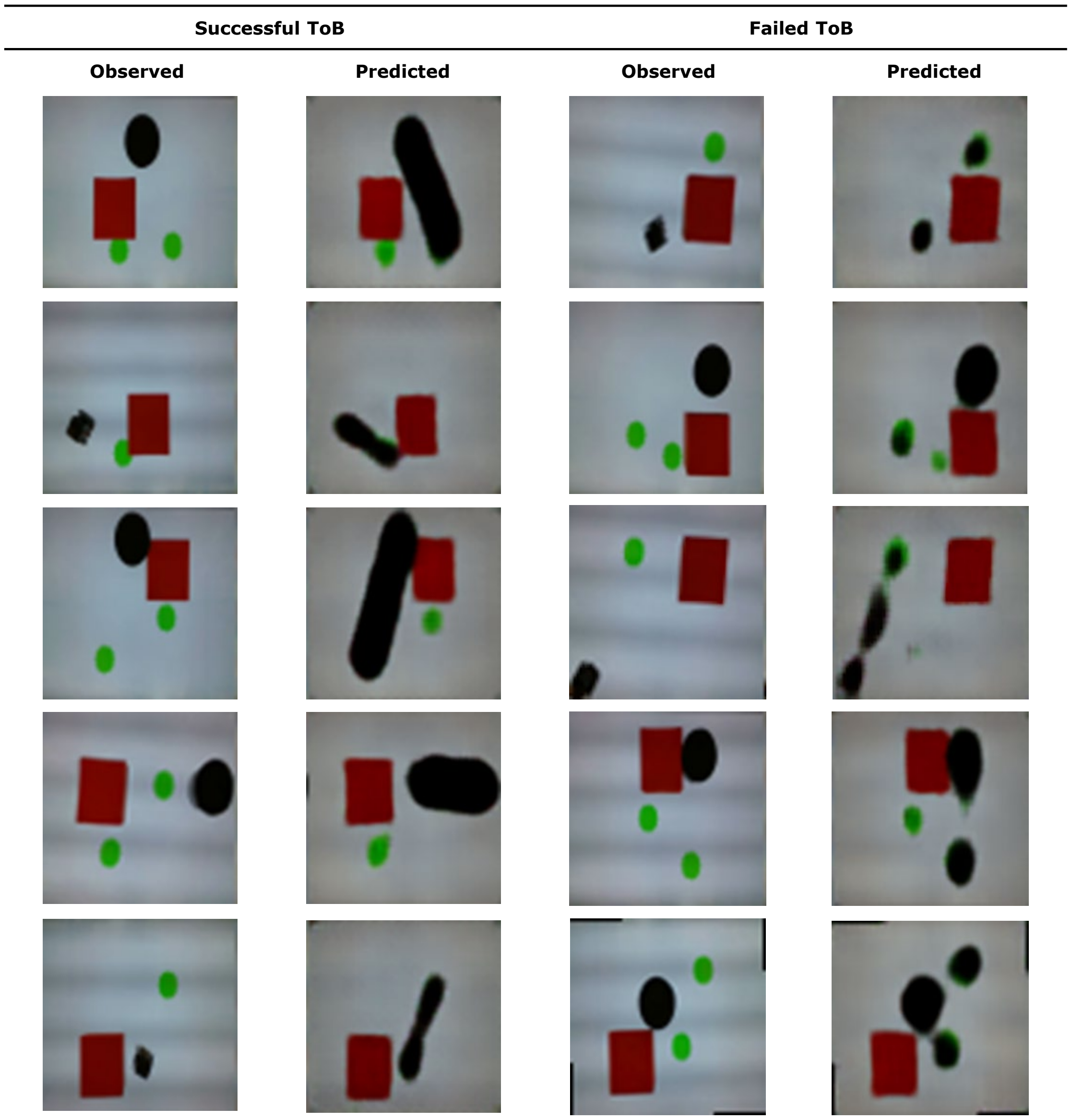


Fig. 4.6: **Physical scenes and outcomes envisioned by the observer.** The first column and the third column show some sample scenarios involving the actor robot, one or two green foods, and a square obstacle. The second and the fourth column show the outcome as envisioned by the observer. The left two columns (Success) show successful vision, whereas the right two columns (Failure) show failed or blurry envision.

closet food. However, the actual policy of the actor robot is to consume the closest food it can see. Hence, there should be a mismatch of understanding of the policy between the observer and the ground-truth policy of the actor. Indeed, our results in Fig. 7 shows that if we test the observe model with only "food visible" cases, the observer AI was able to correctly envision the scene with

Train / Test	**Success Rate**
Food Visible Train, Food Visible Test	97.30% (n=667)
Food Visible Train, Food Obscured Test	56.82% (n=88)
Half Food Visible Train, Half Food Obscured Train, Food Visible Test	98.21% (n=669)
Half Food Visible Train, Half Food Obscured Train, Food Obscured Test	100.00% (n=88)

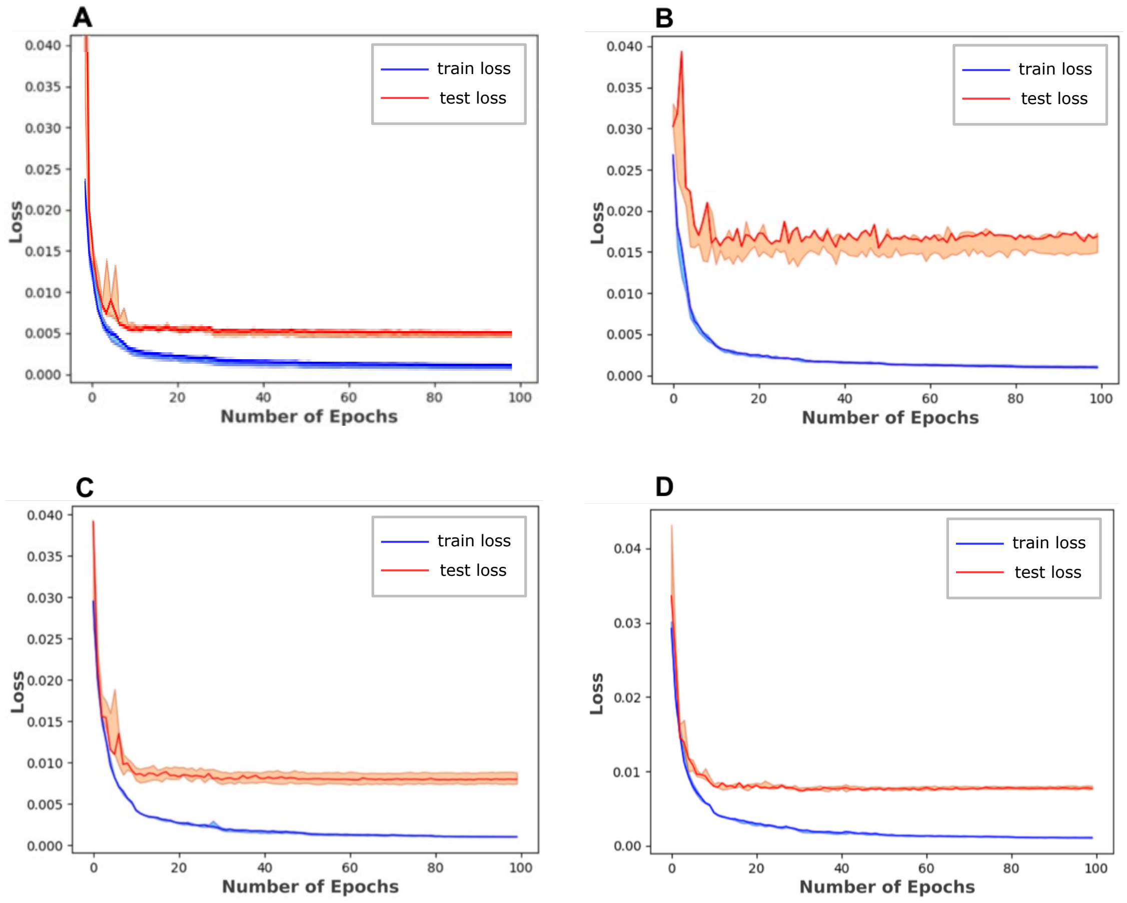


Fig. 4.7: **Training and testing of the observer and corresponding success rate.** We first gathered training and testing data by randomly placing the actor, two green food items, and the obstacle (Table, A). We also collected "obscured" test cases where we deliberately placed the closest food to where it is not visible to the actor (Table, B). Higher success rates were achieved by balancing the training data with half "visible" data and half "obscured" data (Table, C and D). Learning curves across all four above scenarios are shown. Error bars are presented to show experiment results across three different random seeds used in both data splitting and network training.

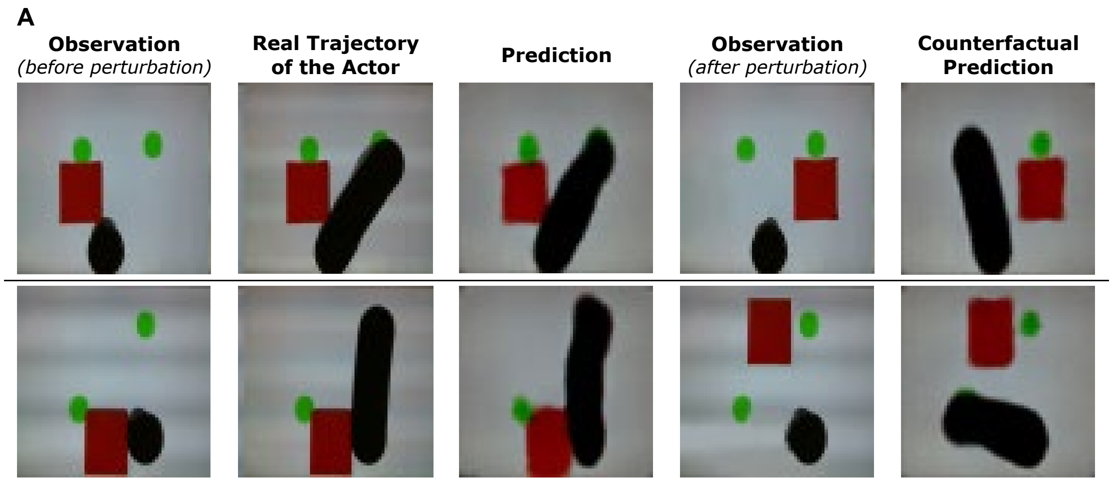
97.3% accuracy. However, if we test the observer model with only the "food obscured" cases, the observer AI's accuracy drops to 56.82%. This means that the observer trained only on food visible cases fails to notice the difference of the perspective between itself and the actor robot.

After we balanced the training dataset by including 50% "food visible" cases and 50% "food obscured" cases, the results change. We then test the trained observer model with "food obscured" testing set again. As shown in Fig. 4.7, the observer AI achieved 100% accuracy on all the "food obscured" testing set and even further improves the "food visible" testing accuracy to 98.21%. This statistically significant improvement in the performance can be interpreted to imply that it is necessary to teach perspective taking to the observer agent under our setting by exposing it to both food visible and food obscured examines.

Fig. 8A presents qualitative visualizations when we perform counterfactual perturbation to the observation of the observer agent, under "food obscured" cases. Specifically, by moving the obstacle from blocking the physically closest food to blocking the physically farther food, we observe that the observer changes its prediction accordingly, which further validates that the observer recognizes the perspective differences between itself and the actor robot.

To further investigate to what degree our observer network is able to model behavior of the actor robot under various unseen changes in the environment, we replicate the exact setting and data processing pipeline from the real world in simulation and perform a systematically evaluation. Specifically, we first vary the size of the robot, the obstacle and the foods in the "food visible" testing data and "food obscured" testing data respectively. We only vary one element at a time. We then gradually change the color of the food and the obstacle to each other. Finally, we add one additional food and one additional obstacle respectively during testing.

The results are summarized in Fig. 4.8B. Our observer model is still able to give accurate prediction when the sizes of the objects are changed, but performs worse when the color is changed by 40% and finally breaks when the color is changed by 60%. This suggests that our observer model does not heavily rely on size of the entities to model the behavior, but it remembers the color information to some extent to assist the prediction.


A
Observation (before perturbation)
Real Trajectory of the Actor
Prediction
Observation (after perturbation)
Counterfactual Prediction

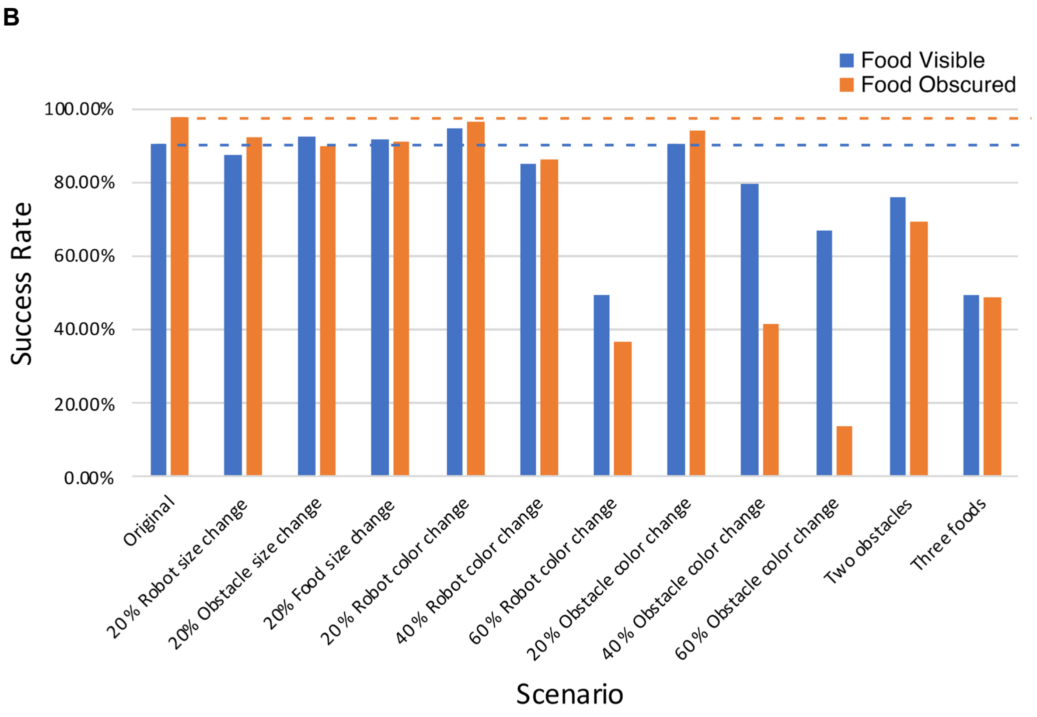


Fig. 4.8: **Counterfactual Perturbation and Prediction Sensitivity.** (A) The first column shows the original observation of the observer and the third column shows the prediction from the observer. The second column is the real trajectory of the actor robot after releasing it. We then move the obstacle to a different location from the first column. The resulted new observation of the observer is shown in the fourth column. The last column shows the counterfactual prediction after seeing this new observation. Both examples show the observer changes its prediction after the physically closest green dot becomes visible from being obscured by the obstacle. (B) Our model is able to give accurate prediction when the sizes of different objects as well as actor robot is changed. However, the observer model performs worse when the color is changed by 40%, and eventually breaks when the color is changed by 60%. The size change in these experiments is limited by the arena size (128 × 128 pixels).

Chapter 5: Robot Modeling the Visual Perspective of Other Robots

Imagine that you see your colleague trying to open the door, only to find that the door is blocked by the doorstop hidden on the other side. Without exchanging words, you rush to remove the doorstop, allowing the door to swing open. You took this action because you understood what your colleague was trying to do (open the door), and you understood that she could not see the doorstop (from her viewpoint). We refer to these intuitive abilities as Behavior Modeling and Visual Perspective Taking, respectively. Visual Perspective Taking [146, 147, 148, 149, 150, 151] is essential for many useful multi-agent skills such as understanding social dynamics and relationships, engaging in social interactions, and understanding intentions of others. A simple action such as getting out of the way of a busy person or assisting a person about to sit down, involve both of these abilities.

Like humans, robots that can perform VPT and use that ability to model others [125] have many advantages across a variety of applications such as social robotics, assistive and service robotics. However, even though current robots can perform a variety of tasks by themselves or move as a group, these robots often do not demonstrate the capability of performing perspective taking or behavior modeling. Several challenges still remain.

First, visual observations are high-dimensional with a large amount of information. Relying on manual feature extractions and hand-designed learning procedure [152] cannot scale in complex setups. Second, the behaviors among different robots are cross-dependent. Consequently, it is necessary to account for the mutual interaction [153] between related robots. Third, directly learning on physical robots requires strong data efficiency. Though recent methods [60, 143, 154, 155] demonstrate that similar cognitive abilities could emerge from reward-driven learning, they often require millions of learning steps. Lastly, learning to predict the long-term future is very challenging. Most existing visual prediction frameworks [156, 15, 157, 158, 159] follow an iterative paradigm to rollout the prediction, which is both time and memory consuming under long-horizon tasks.

In this chapter, we propose a self-supervised vision-based learning framework [66, 60] to take a small step towards realizing VPT and TOB in navigation robots involving future opponent modeling. We call our framework VPT-TOB. Our key idea is to explicitly predict the future perspective of the other robot as an image by conditioning on a top-down visual embedding of first-person RGB-D camera images and time-abstracted action embeddings. We further present a value prediction model that can leverage the future perspective prediction to evaluate action proposals. During test time, we ask the robot to propose future goals on the top-down map and generate potential action plans with its low-level controller. The robot can then use our VPT-TOB model to imagine future perspectives of the other robot and evaluates its goal proposal with the value prediction model. The robot can choose the safest position to navigate to.

Our experiments on a physical hide-and-seek task suggest that our method outperforms baseline algorithms that do not explicitly consider other agents' perspectives and behaviors, showing the importance of explicit visual perspective taking and behavior modeling. Our hider robot exhibits diverse behaviors during hiding and an interpretable decision-making process by providing how the robot is envisioning the future state and actions of another robot.

The primary contribution of this chapter is to demonstrate that a long-term vision prediction framework can be used to model the perspectives and behaviors of other robots. We present a value prediction module that can take the perspective prediction to evaluate action proposals during test time. We perform several experiments and ablation studies in both real and simulation environments to evaluate the effectiveness of our design decisions.

5.1 Visual Perspective Taking Network

In this chapter, we frame the Visual Perspective Taking and Behavior Modeling as an action-conditioned vision predictive model. Our method encourages the learning robot to explicitly contemplate the future visual perspective and behavior of the other robot by outputting where the other

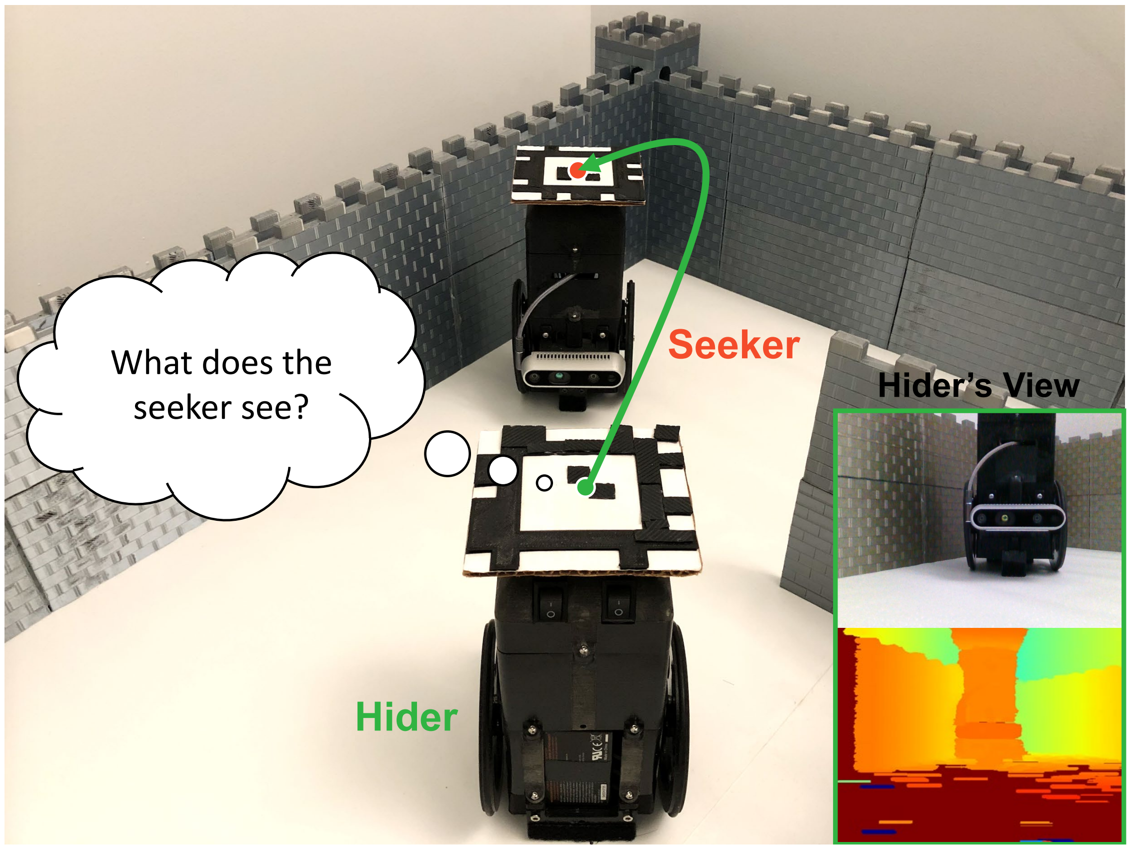


Fig. 5.1: Visual Perspective Taking (VPT) refers to the ability to estimate other agents' viewpoint from its own observation. With our VPT framework, the hider robot learns to predict the seeker robot future observation and plan its actions based on this prediction to avoid being captured.

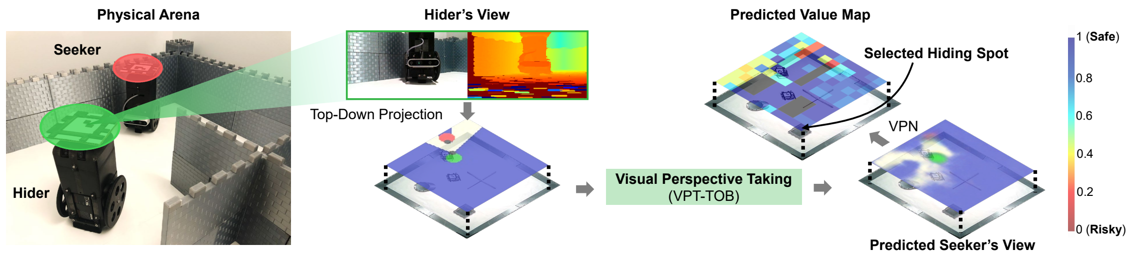


Fig. 5.2: **Method Overview:** our robot learns to infer the future view of the other robot with our VPT-TOB model based on its initial visual observation and future action embeddings (blue is invisible area). With these anticipations and a value prediction model, the robot can produce a value map indicating the safety level (color bar) and selects the best hiding spot.

agent will be, and what the other agent will see as an image. Our hider robot can be trained directly on the physical setup. Once trained, the robot can answer "what if" questions visually: what will the other agent do, and what will its view become, if I choose to perform this action for some time?

Task Setup To investigate the effectiveness of this framework in opponent modeling, we consider a hide-and-seek setting [60] where a hider robot needs to avoid being captured by another seeker robot while navigating around the environment with various obstacles. In order to predict the future perspective of the seeker, the hider robot needs to infer the geometric view from the seeker, reason about the environment spatial layout, and form a model of the seeker's policy. This task reflects several challenging aspects in real-world setup such as occlusions in robot perception and cross-dependent actions among multiple robots.

We 3D printed two-wheeled robots with size of 110mm by 130mm on a 1.2m by 1.2m plane with three obstacles. Similar to Wu et al. [160], for simple prototype, we place fiducial markers [161] on the robots and track them with a top-down camera. However, our setup aims at representing what could be possible to achieve with only the onboard RGB-D camera for SLAM. Therefore, the robots only perceive what could be captured with the first-person RGB-D camera with 86°FoV with noisy depth and color information and occlusions.

Both robots start by facing at each other at a random location in the environment and moves in the same speed (10 mm/step). The robots can move forward or backward and rotate with 10°as given primitives. We only focus on stationary case where the seeker robot follows a heuristic expert policy and leave the non-stationary scenario for future work as the first step towards grounding VPT to physical robots. If the hider is visible through its first-person camera, the seeker will navigate towards it with A* [162]; if the seeker loses the hider in its camera view, it will navigate to the last known position of the hider and then continuously explore the room until the hider shows again.

5.1.1 State Representation

We represent the robot state observation as a RGB image. The image representation depicts a top-down visibility map. This is similar to the representations in other mobile navigation and self-driving systems [163, 164, 165, 166, 160]. We project the depth information from first-person view to a 2D bird's eye view to acquire the top-down visibility map. Areas behind the object boundary are treated as free space. We colorize the observation map to reflect this. The hider robot is always aware of its own position, while the seeker position is only available when it is visible through a color filter from the first-person view. The robots are plotted as circles with different colors. We denote the final visual state representation of the hider and the seeker at timestamp t as $I_{H,t}$ and $I_{S,t}$.

5.1.2 Action Representation

We design the action representation with two key considerations in mind. First, we want to align the action representation with the state representation. Second, we need to preserve information about past action sequences. Thus, we propose to encode the robot's trajectory sequence as two 2-channel images with the same dimensions as our top-down visibility map. Say we are interested in the action sequence from t=0 up to a future time t=t_i. The first image is a visitation map $F_{t=0:t_i}$ denoting how many times each point on the map will be visited by the robot. This is achieved through the second alpha-channel where darker color represents more frequent visitation. The second image is a time-encoded trajectory map $T_{t=0:t_i}$ denoting the traversal order. Similar to F, the darker the color in T, the later the position will be traversed. Both F and T are necessary for a complete action representation to avoid the ambiguity brought by the trajectory intersection. An example of a training data pair is shown in Fig. 5.3.

5.1.3 Data Collection

We experiment with two types of exploration policies for the hider to collect the training data. The first is a random policy created by sampling from the given action primitives. However, this policy explores poorly thus resulting in the seeker easily catching the hider. The hider mostly stays alive for only 20 to 30 steps which biases the training data to be short-sighted. The second policy is human play data [167]. We ask a human subject to play as the hider in front of a display from the first-person view for 100 trajectories. Since the human subject has never played this version of the game before, the trajectories are diverse ranging from successful trials to failures, making it a good strategy to stay with.

During robot data collection, the hider robot randomly samples a human trajectory and uses the

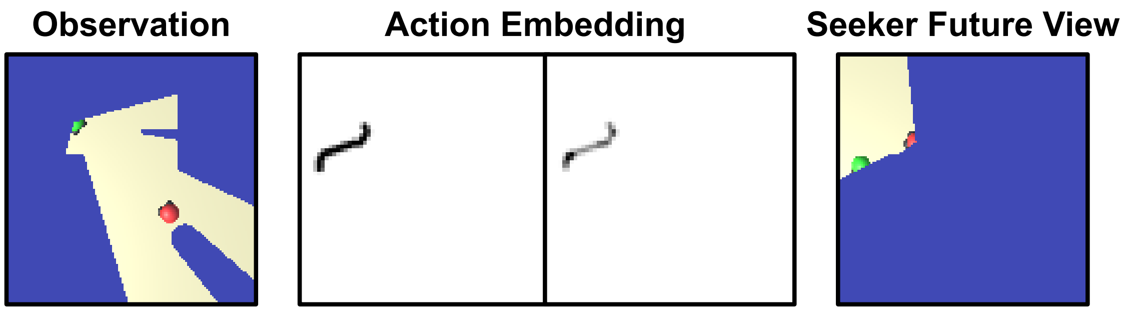


Fig. 5.3: An example of the training data pair for VPT-TOB.

selected trajectory as is but from a different initial condition. It is also possible to train an imitation policy on the human data, but we find the current strategy sufficient to give enough balanced data. A horizon will terminate when one of two conditions is triggered: 1) the seeker robot catches the hider robot, or 2) the total number of steps reaches a maximum value (100 in our setup).

5.1.4 VPT-TOB Network

The objective of the VPT-TOB network f_p is to anticipate the visual observation $\mathbf{I}_{\mathbf{S},\mathbf{t}=\mathbf{t_i}}$ of the seeker at future time step t_i given its initial observation $\mathbf{I}_{\mathbf{H},\mathbf{t}=\mathbf{0}}$ and action embedding ($\mathbf{F}_{\mathbf{t}=\mathbf{0}:\mathbf{t_i}}$, $\mathbf{T}_{\mathbf{t}=\mathbf{0}:\mathbf{t_i}}$). Intuitively, the hider robot needs to first anticipate where it will end up if it takes certain action sequences. Secondly, the robot needs to model the behavior of the other robot so that it learns how the seeker will react. Finally, the hider needs to project the view using its prediction of the seeker agent, which requires an understanding of scene geometry and view projection. During training, the hider has access to the seeker's view as a supervised signal. During testing, the hider has to predict the seeker's views.

Network Architecture Our VPT-TOB model is a fully convolutional encoder-decoder network [125]. The encoder is a 8-layer fully convolutional network which takes in the concatenated visual observation and action embedding as input with size 128×128. The decoder network comprises four transposed convolutional layers where each of them is followed by another 2D convolution and a transposed convolution for high-resolution output [126] with size 128×128.

Training VPT-TOB model can be supervised with a simple pixel-wise mean-squared error loss.

We use 1,000 real-world trajectories for training and 200 trajectories for validation and testing. We use Adam [127] optimizer with a learning rate of 0.001 and batch size 256. We decrease the learning rate by 10% at the 25% and 65% of the training progress. Formally, the loss function can be written as:

$$\mathcal{L}_{\text{VPT-TOB}} = \text{MSE}(f_p(\mathbf{I_{H,t=0}}, \mathbf{F_{t=0:t_i}}, \mathbf{T_{t=0:t_i}}), \mathbf{I_{S,t=t_i}})$$

5.1.5 Inference and Planning

At test time, the hider robot needs to propose a set of action plans and envision the possible outcomes for them with VPT-TOB to pick the best plan. One way to represent the plan is to specify a goal location and then perform visual MPC [15] with the learned opponent model to find the best path. However, in our task setup in which no human specified goal is available, a more reasonable way is to propose a spatial location in the room and plan with a low-level controller policy (*e.g.*, A*). This parameterization emphasizes a safer hiding location instead of a safer path to follow given a required goal.

Value Estimation Map We therefore aim to generate a value map to evaluate the risk level of the possible goal locations. A value map is a matrix where each element denotes the safety of choosing the corresponding position as the goal. The earlier the hider robot is caught towards that goal, the riskier that goal is. Formally, the value for each element on the value map is defined as the following accumulation form:

$$V_{i,t=0:T} = \sum_k v_{i,t=(kN-1)}, \; v = \begin{cases} 1, & \text{if hider is caught} \\ 0, & \text{otherwise} \end{cases}$$

where k is an integer from 0 to $\lfloor \frac{T}{N} \rfloor$ and T is the maximum time budget for this horizon and N is the step size. Under this formulation, we discretize the environment as a grid map (121 endpoints for our setup). From the current hider position, the planned path from A* can be embedded every N (N=10 for us) steps.

Value Prediction Network (VPN) Our value prediction network f_v takes in the visual prediction from VPT-TOB to output the binary v value for each step. We can then use the above formula to compute the accumulated value along all the steps. We just need a few forward passes on VPT-TOB and VPN to obtain the entire value map within 1.4 seconds on a single GPU.

Intuitively, VPN estimates whether the hider will be caught given the predicted future view of the seeker. Our VPN network consists of 3-convolutional layers followed by 2 fully-connected layers. We apply max-pooling after each convolution and a dropout layer right after the first fully-connected layer. The network is trained to minimize a cross-entropy loss. We augment the data by randomly rotating the images among $(90°, 180°, 270°)$. In total, we have $2,000$ images for training, and 400 images each for validation and evaluation. We train our model for 100 epochs with a batch size of 256 and a learning rate of 0.0005. We decrease the learning rate by 10% at 20% and 50% of the training progress. We use Adam as the optimizer. Formally, the loss function can be written as:

$$\mathcal{L}_{\text{VPN}} = -(v \log\Big(f_v(\mathbf{\hat{I}_{S,t=t_i}})\Big)) + (1-v)\log\Big(1 - f_v(\mathbf{\hat{I}_{S,t=t_i}})\Big)$$

5.2 Experiments

We first perform a series of experiments in our real-world setup. To evaluate different components in our algorithm, we further replicate our physical setup in a simulation and execute several ablation and baseline comparisons.

VPT-TOB and VPN A straightforward evaluation of the VPT-TOB model is to provide quantitative visualizations of the imagined frames. Since our goal with VPT-TOB is to capture the perspective information and opponent behavior for long-term planning, a perceptual error will not be ideal for our evaluation. Meanwhile, our VPN model needs to take in the predicted frame from VPT-TOB to infer whether the hider robot will be safe. Therefore, we can use the performance of VPN on the output of VPT-TOB as a quantitative metric for our VPT-TOB model.

Value Estimation Map To evaluate the overall performance of the entire planning pipeline, we

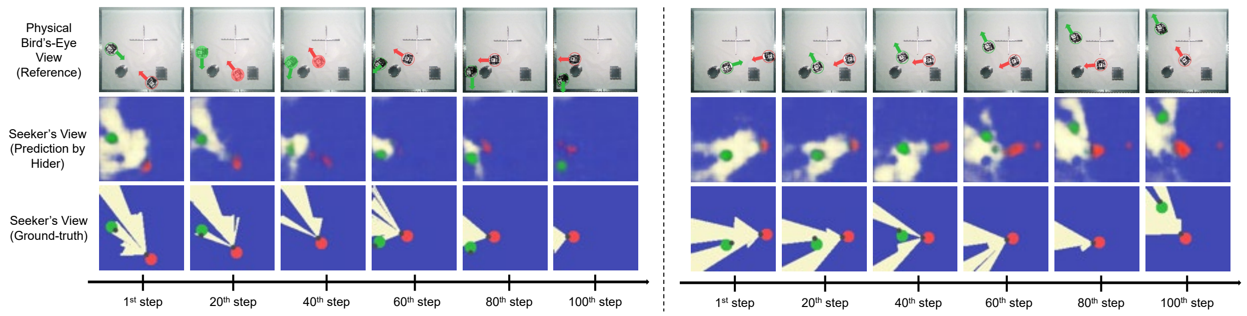

Physical Bird's-Eye View (Reference)
Seeker's View (Prediction by Hider)
Seeker's View (Ground-truth)
1st step
20th step
40th step
60th step
80th step
100th step

Fig. 5.4: **VPT-TOB predictions from physical robots:** we here show two example sequences. The predictions are shown as a function of time. Our hider robot accurately predicts the future perspectives of the seeker robot only by its own initial observation and a future potential action plan.

quantify the accuracy with the entire value map. We measure this in simulation which can provide us with ground-truth values conveniently by moving the hider robot to all possible goal locations and collecting the accumulated values. During test time, what matters for decision making is the relative value between different goal locations instead of the absolute value. To this end, our metric for the value map is a relative ranking accuracy. Say we choose g_i and g_j as a pair to evaluate. The relative ranking of this pair is correct if it matches the relative ranking from the ground-truth value map. We evaluated all pairs at least 3 units away. This metric will result in the same ranking among all comparisons as the last metric, but provide an overall evaluation.

5.2.1 Real-World Results

The prediction results from our VPT-TOB model is a sequence of images based on a single initial observation from the hider's view and the future action embedding. In Fig. 5.4, we show the predictions from the random hider robot every 20 steps up to 100 steps corresponding to about 25 seconds. Each predicted image presents where the hider thinks the seeker robot will be, and what the seeker's view will look like. Our model gives accurate predictions about the perspective and behavior of the opponent robot.

For test time planning, we randomly place the hider and seeker robot in the environment and ask the hider robot to plan and execute its goal selection from VPT-TOB and VPN. Fig. 5.5 shows the video frames where the hider robot navigates to its selected goal and successfully avoids the

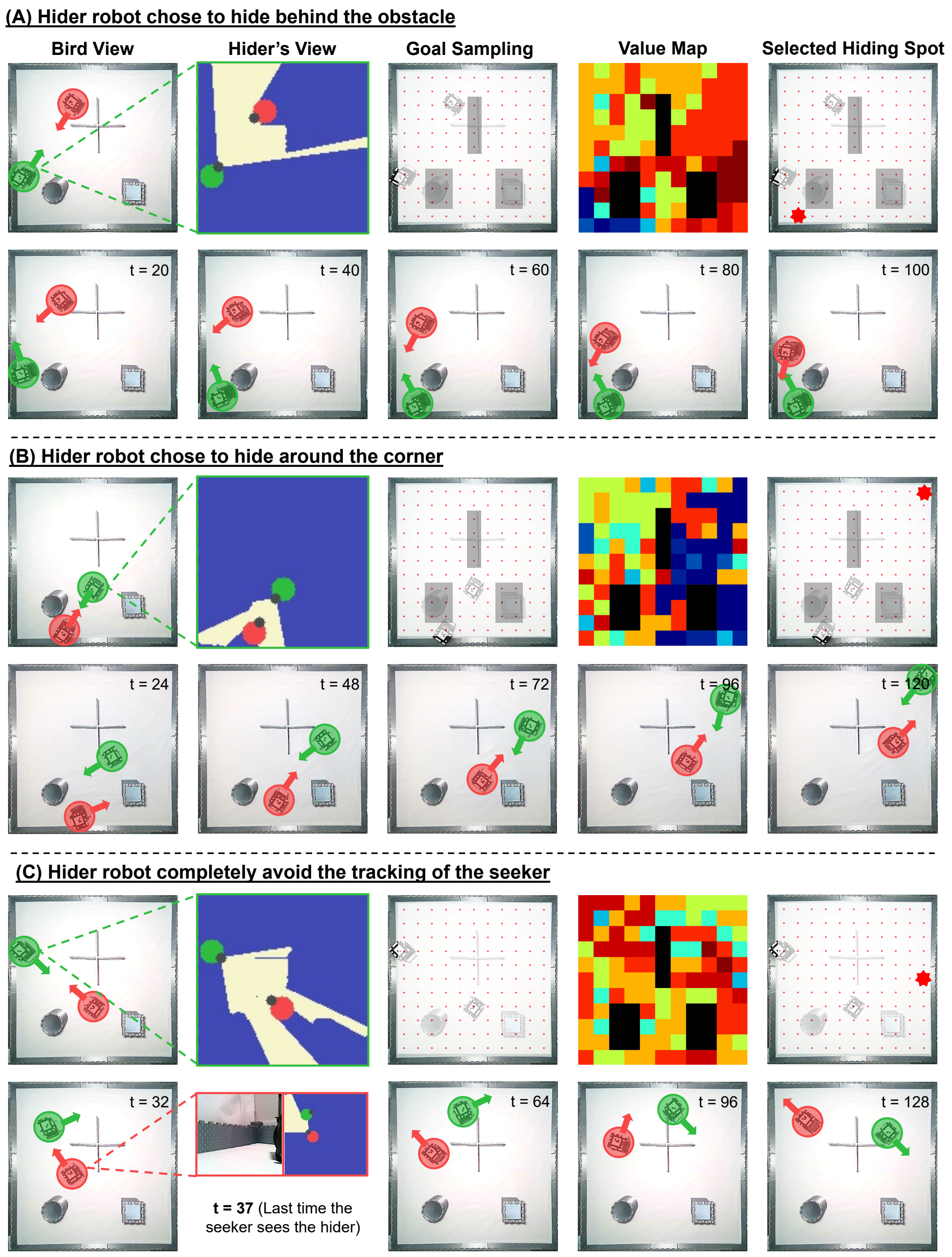

(A) Hider robot chose to hide behind the obstacle
Bird View
Hider's View
Goal Sampling
Value Map
Selected Hiding Spot
t = 20
t = 40
t = 60
t = 80
t = 100
(B) Hider robot chose to hide around the corner
t = 24
t = 48
t = 72
t = 96
t = 120
(C) Hider robot completely avoid the tracking of the seeker
t = 32
t = 37 (Last time the seeker sees the hider)
t = 64
t = 96
t = 128

Fig. 5.5: **Planning and hiding** Our hider robot produces a value map with our VPT-TOB and VPN networks. (Blue is safe and red is dangerous.) The hider then navigates to the farthest location with the highest value (star mark). Interestingly, in the last example, the hider completely got rid of the seeker from the 37$^{\text{th}}$ time step.

seeker only by relying on its initial value map estimation for the future 100 time steps. There is no extra decision making cost during execution. Interesting hiding behaviors emerged automatically from our setup as shown in Figure 5.5.

We further test the performance of our VPN model under the real-world setup by inputting the predictions from the VPT-TOB model. Our VPN model achieves an accuracy of 88.39%. As we will discuss in the next section, this is very similar to what we can achieve in the simulation, demonstrating the effectiveness of our method in the physical robot platform.

5.2.2 Simulation Analysis

Our simulation is built with Unity to evaluate different components in our approach. The simulation can run in parallel and hence provides an efficient and scalable testing environment. To provide extensive analysis, we use in total 4,500 trajectories as training and 500 trajectories each for validation and testing. We can also get the top-down observation directly by applying a mesh filter in Unity with similar result from our real robot sensors. We run the hider robot for 200 steps.

5.2.3 Baselines

Self-perspective predicts the hider's own future states conditioned on its initial observation and action plans other than explicitly modeling the other robot. This follows recent works on learning visual world models [15, 157, 168, 159, 169] with adapted architectures for fair comparison.

Coordinate-value directly predict whether the hider robot will be caught in the future from initial observation and action plans. The input consists of an initial state observation from the hider, a vector action embedding with spatial coordinate information, and a future coordinate value where the hider will end up using the given actions.

Vector-action is the same with VPT-TOB but replaces our action representation with a vector embedding as in [170]. This embedding is concatenated with the intermediate features to produce the final predicted image.

The self-perspective baseline aims to test whether modeling the opponent agent is necessary. Both the self-perspective and coordinate-value baselines serve to examine if explicit opponent modeling is more effective than relying on the network to learn the opponent model implicitly.

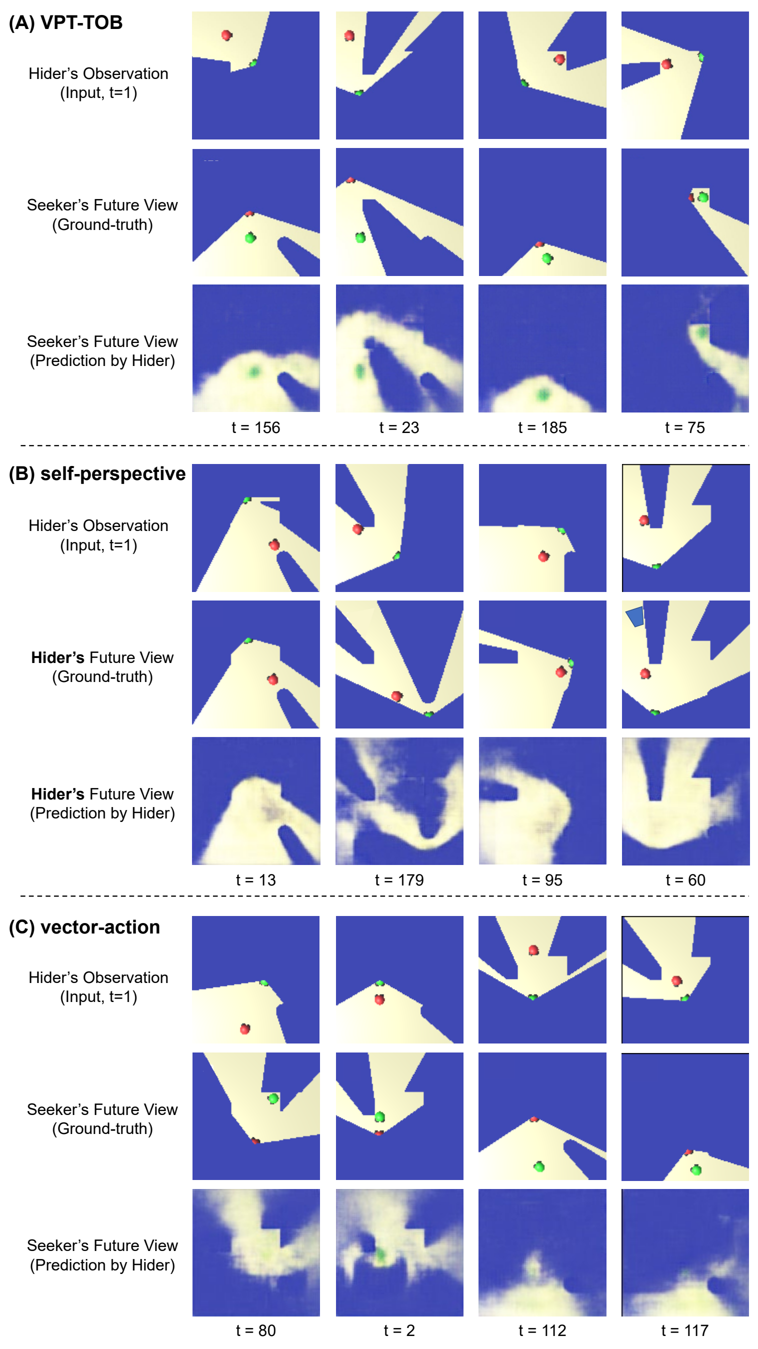

(A) VPT-TOB
Hider's Observation (Input, t=1)
Seeker's Future View (Ground-truth)
Seeker's Future View (Prediction by Hider)
t = 156
t = 23
t = 185
t = 75
(B) self-perspective
Hider's Observation (Input, t=1)
Hider's Future View (Ground-truth)
Hider's Future View (Prediction by Hider)
t = 13
t = 179
t = 95
t = 60
(C) vector-action
Hider's Observation (Input, t=1)
Seeker's Future View (Ground-truth)
Seeker's Future View (Prediction by Hider)
t = 80
t = 2
t = 112
t = 117

Fig. 5.6: Predictions from VPT-TOB and other baselines. Our method produces more accurate modeling of the other robot's future perspective, suggesting the advantage of explicit opponent modeling and our visual action embedding.

Vector-action is used to evaluate if our action representation can improve the quality of VPT-TOB model.

Through our simulation comparisons, we show the accuracy of VPN across all the applicable

Table 5.1: VPT-TOB and VPN Performance

Method	**Success Rate on Held-out Examples**
Self-perspective + VPN	76.61%
Coordinate-value	76.40%
Vector-action + VPN	82.98%
VPT-TOB (ours)	**88.45%**

methods in Tab. 5.1 in which the input of VPN comes from our VPT-TOB or other baseline models. We run all the methods on 1,000 unseen examples with uniform starting locations and horizon length. The result suggest the advantage of our method. Both VPT-TOB and vector-action outperforms self-perspective and coordinate-value, validating the necessity to explicitly model the opponent perspective and behavior. The gap between vector-action and VPT-TOB indicates that our action representation offers additional gains for learning an accurate visual opponent model.

Visualizations of the predicted frames substantiate our conclusion (Fig. 5.6). For better comparison, we also display bird's-eye view in the first row and ground-truth seeker's view in the last row. However, these images are only for visualization, not inference.

Fig. 5.7 shows the generated value maps from our approach comparing with the ground-truth value maps. Overall, our framework demonstrates an accurate estimation about the safety level across the room.

5.2.4 Generalization Across Training Time Horizon

With the relative ranking accuracy, we can quantify the performance of our approach from a higher-level perspective. Since our model should also learn temporal abstractions from our action representation, we extend the test horizon up to 175% of the maximum number of steps given in training and examine how the relative ranking accuracy changes. We achieved this test by simply projecting longer horizon action trajectories on our image-based action embedding.

Tab. 5.2 shows the performance. Our model is able to predict far beyond its training horizon with very small losses of the accuracy. This suggests that our method also learns to model the temporal dynamics while modeling the other agent.

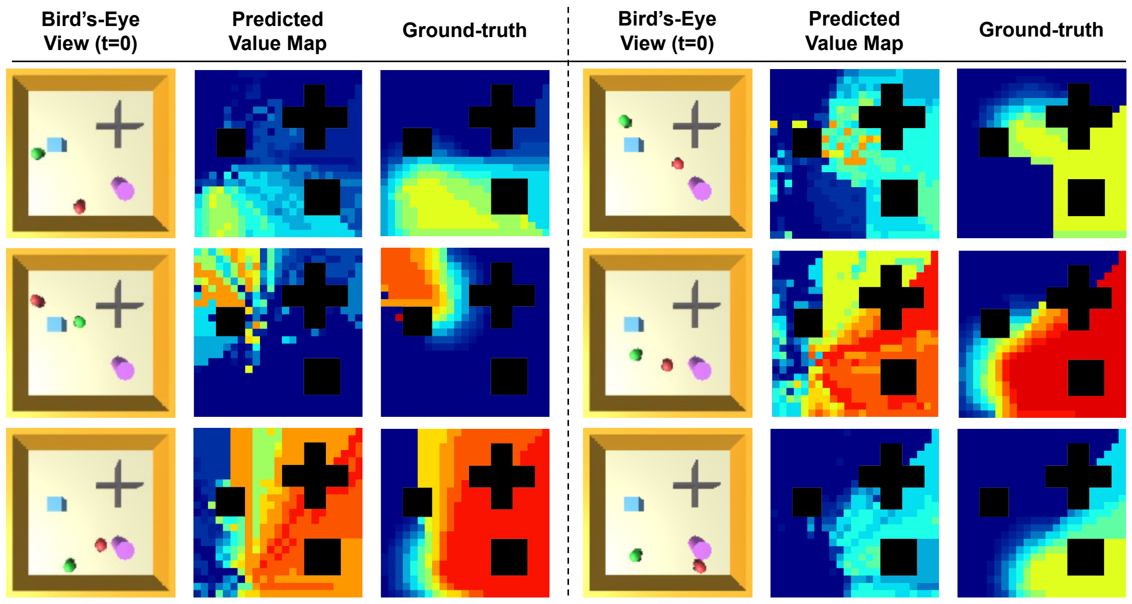


Fig. 5.7: Predicted value maps versus ground-truth value maps. Our method produces accurate value estimations and reflect the overall value patterns.

Table 5.2: Generalization on Prediction Horizon

Prediction Steps	**Relative Ranking Accuracy**
200 (max seen during training)	75.29 ± 12.23%
250	73.75 ± 10.35%
300	73.03 ± 10.40%
350	72.77 ± 10.82%

Chapter 6: Physical Environment Modeling through Visual Discovery

We have introduced how to build toward "generalist robots" by enabling robots to model their embodied self and their surrounding agents. With these self models and the models of other agents, robots can leverage this consistent knowledge to perform various future tasks across different domains. Another major aspect of robot learning tasks is to model their physical environments. To fully generalize on multiple environmental conditions, robots need to distill compact physical knowledge of the environment, such as the governing law of physics, from noisy and high-dimensional observations. In this chapter, we will introduce the essential step of distilling compact physical knowledge of dynamical systems purely from high-dimensional video recordings. That is, the discovery of fundamental variables hidden in experimental data [171].

All physical laws are described as mathematical relationships between state variables. These variables provide a complete and non-redundant description of the relevant system, while laws describe how these variables change with time. However, despite the prevalence of computing power and AI, the process of identifying the hidden state variables themselves has resisted automation. Most data-driven methods for modeling physical phenomena still rely on the assumption that the relevant state variables are already known. A longstanding question is whether it is possible to identify state variables from scratch, given only high-dimensional observational data. Here we propose a new principle for determining how many state variables an observed system is likely to have and what these variables might be, directly from video streams. We demonstrate the effectiveness of this approach using video recordings of a variety of physical dynamical systems, ranging from elastic double pendulums to fire flames. Without any prior knowledge of the underlying physics, our algorithm discovers the intrinsic dimension of the observed dynamics and identifies candidate sets of state variables. We suggest that this approach could catalyze the understanding, prediction, and control of increasingly complex systems.

6.1 Background

Mathematical relationships are known to describe nearly all physical laws in nature [172], and these mathematical expressions are almost always formulated as relationships between physical state variables that describe the physical system. This suggests that before any natural law can be discovered, the relevant state variables must first be identified [173, 174].

For example, it took civilizations millennia to formalize basic mechanical variables such as mass, momentum and acceleration. Only once these notions were formalized, could laws of mechanical motion be discovered. Laws of thermodynamics were discovered only after concepts such as temperature, pressure, energy and entropy were formalized. Laws of solid mechanics could only be discovered once variables such as stress and strain were formalized. Electromagnetism, fluid dynamics, quantum mechanics and so forth all required their own set of fundamental state variables to be defined, before they could be formalized into existence. Without the proper state variables, even a simple system may appear enigmatically complex.

The set of state variables for modeling any system is not only hidden, but it is also not unique (Fig. 6.1). In fact, even for well-studied systems in classical mechanics, such as swinging pendulum, many sets of possible state variables exist. For the pendulum, the state variables are typically the angle of the arm $q_1 = \theta$ and the angular velocity of the arm $q_2 = \dot{\theta}$. The angle and angular velocity are convenient choices because they can be directly measured. However, alternative sets of state variables, such as kinetic and potential energies of the arm, could also be used as state variables.

A key challenge, however, occurs when the system is new, unfamiliar or complex, and the relevant set of state variables is unknown. Although there are various techniques such as Dynamic Mode Decomposition (DMD) and Singular Value Decomposition (SVD) [175] developed to learn dynamical systems based on observations, none of these methods has the ability to process a video of a pendulum, for example, and without any further knowledge, output the double pendulum's four state variables. Such an ability, if had, could help scientist gain insight into the physics underlying

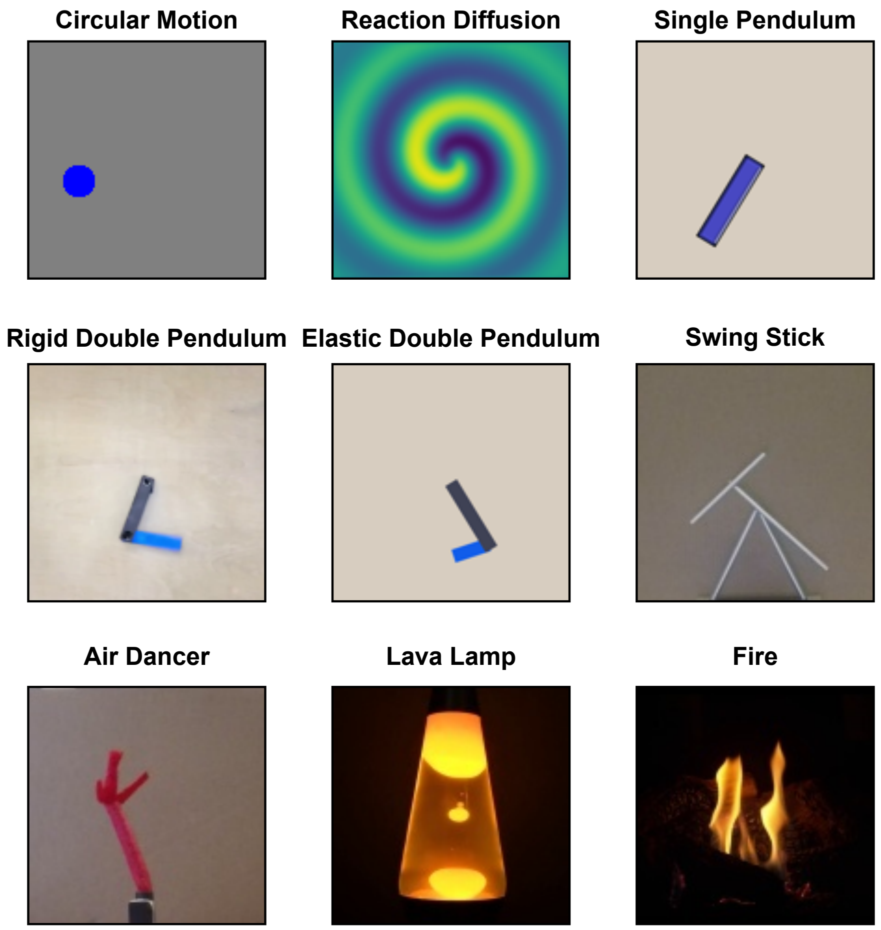


Fig. 6.1: **What state variables describe these dynamical systems?** Identifying state variables from raw observation data is a precursor step to discovering physical laws. The key challenge is to figure out how many variables will give a complete and non-redundant description of the system's states, what are the candidate variables, and how the variables are dependent on each other. Our work studies how to retrieve possible set of state variables from data distributions non-linearly embedded in the ambient space.

increasingly complex phenomena, especially when theory is not keeping pace with observations.

Data-analytics tools have impacted almost every aspect of scientific discovery [176, 177]: Machines can measure, collect, store and analyze vast amounts of data. New machine learning techniques can create predictive models, find analytical equations [178] and invariants [179], and even generate causal hypotheses along with new experiments to validate or refute these hypotheses [180, 181, 182]. Yet, a longstanding question is whether it is possible to automatically uncover the hidden state variables themselves. Finding such variables is still a laborious process requiring teams of human scientists toiling over decades.

The ability of human scientists to distill vast streams of raw observations into laws governing a concise set of relevant state variables has played a key role in many scientific discoveries. It is thus of great importance to have tools for automated scientific discovery that could help distill raw sensory perceptions into a compact set of state variables and their relationships.

Numerous machine learning tools have been demonstrated to model the dynamics of physical systems automatically, but most of them were already provided measurements of the relevant state variables in advance [183, 184, 185, 178, 186, 179, 187, 188, 189, 190, 191, 192, 193, 194][1]. For example, our own previous work on distilling natural laws [179] assumed an input stream corresponding to state variables such as angle and angular velocity of a pendulum arm. Brunton et al. [192] required access to spatial coordinates and their derivatives for modeling a Lorenz system, Udrescu and Tegmark [194] combined neural networks with known physical properties to solve equations from the Feynman Lecture on Physics, given provided variables, Mrowca et al. [195] required access to the position, velocity, mass, and material property of the particles that compose the objects being modeled and Champion et al. [196] predefined possible basis functions to constrain the training of an autoencoder for observation reconstruction.

The goal of this work is to find a way to overcome this key barrier to automated discovery – by explicitly identifying the intrinsic dimensionality of a system and the corresponding hidden state variables, purely from the visual information encoded in raw camera observations. A key challenge in identifying state variables is that they are often hidden and might be hard to measure directly. An even greater challenging aspect of state variable identification is that there might be a large number of potential descriptive variables that are related to the varying state of the system, but are neither compact nor complete in their description of the system.

For example, a camera observing a swinging pendulum with an imaging resolution of 128×128 pixels in three color channels, will measure 49,152 variables per frame. Yet this enormous set of measurement, while intuitively descriptive, is neither compact nor complete: In fact, we know that the state of a swinging pendulum can be described fully by only two variables: its angle and

[1]By State Variables, we refer to compact and complete sets of quantitative variables that fully describe the observed dynamical system evolving with time.

angular velocity. Moreover these two state variables cannot be measured from a single video frame alone. In other words, a single frame, despite the large number of measurements, is insufficient to describe the full state of a pendulum.

The questions that we aim to answer are: Given a series of video frames of a swinging pendulum that contain the full and accurate motion trajectories, for example, is there a way to know that only two variables are required to describe its dynamics in full? And is there an automated process to reduce the vast deluge of irrelevant and superfluous pixel information into representations in terms of the two state variables? Naturally, we would like this process to work across a variety of physical systems and phenomena.

The starting point of our approach is to model the system dynamics directly from video representations via a neural network with bottleneck latent embeddings [197, 198, 199]. If the network is able to make accurate future predictions, the network should internally encapsulate a relationship connecting relevant current states with future states. Our paramount challenge is to distill and extract the hidden state variables from the network encoding.

Our key idea involves two major stages. First, after training the dynamics predictive neural network, we calculate the minimum number of independent variables needed to describe the dynamical systems, known as its *intrinsic dimension*, with geometric manifold learning algorithms. This initial stage produces accurate intrinsic dimension estimations on a variety of systems from the model's bottleneck latent embeddings which are already reduced by hundreds of times compared to the raw image space.

Armed with the intrinsic dimension obtained in the first stage, in the second stage, we design a latent reconstruction neural network to further identify the governing state variables with the exact dimension as the intrinsic dimension. We term these identified state variables *Neural State Variables*. Through both quantitative and qualitative experiments, we demonstrated that Neural State Variables can accurately capture the overall system dynamics.

Beyond our two-stage approach to reveal the system intrinsic dimension and the possible set of state variables, another major contribution of our work is to leverage the discovered Neural

State Variables both as an intermediate representation and evaluation metric for stable long-term future predictions of system behaviors. Due to the special reduced-dimension property of Neural State Variables, they can provide very stable long-term predictions, while higher dimensional auto-encoders often yield blurred or plain background predictions if iterated just a few steps into the future.

Finally, we present a hybrid prediction scheme which achieves both accurate and stable long-term predictions. Furthermore, we derive a quantitative evaluation metric for long-term prediction stability with Neural State Variables by approximating the true system dynamics using the most compact latent space. We also demonstrate that Neural State Variables can offer a robust representation space for modeling system dynamics under various visual perturbations.

6.2 Discovery of the Intrinsic Dimension

6.2.1 Modeling Dynamical Systems from Videos

The dynamics of a physical system defines the rule that governs how the current system states will evolve into their successive states in the future. Mathematically, provided the ambient space $\mathcal{X}$ and the state space $\mathcal{S} \subset \mathcal{X}$, one can formulate the dynamical system as

$$\boldsymbol{X}_{t+dt} = F(\boldsymbol{X}_t), \quad t = 0, dt, 2dt, 3dt, \ldots, \tag{6.1}$$

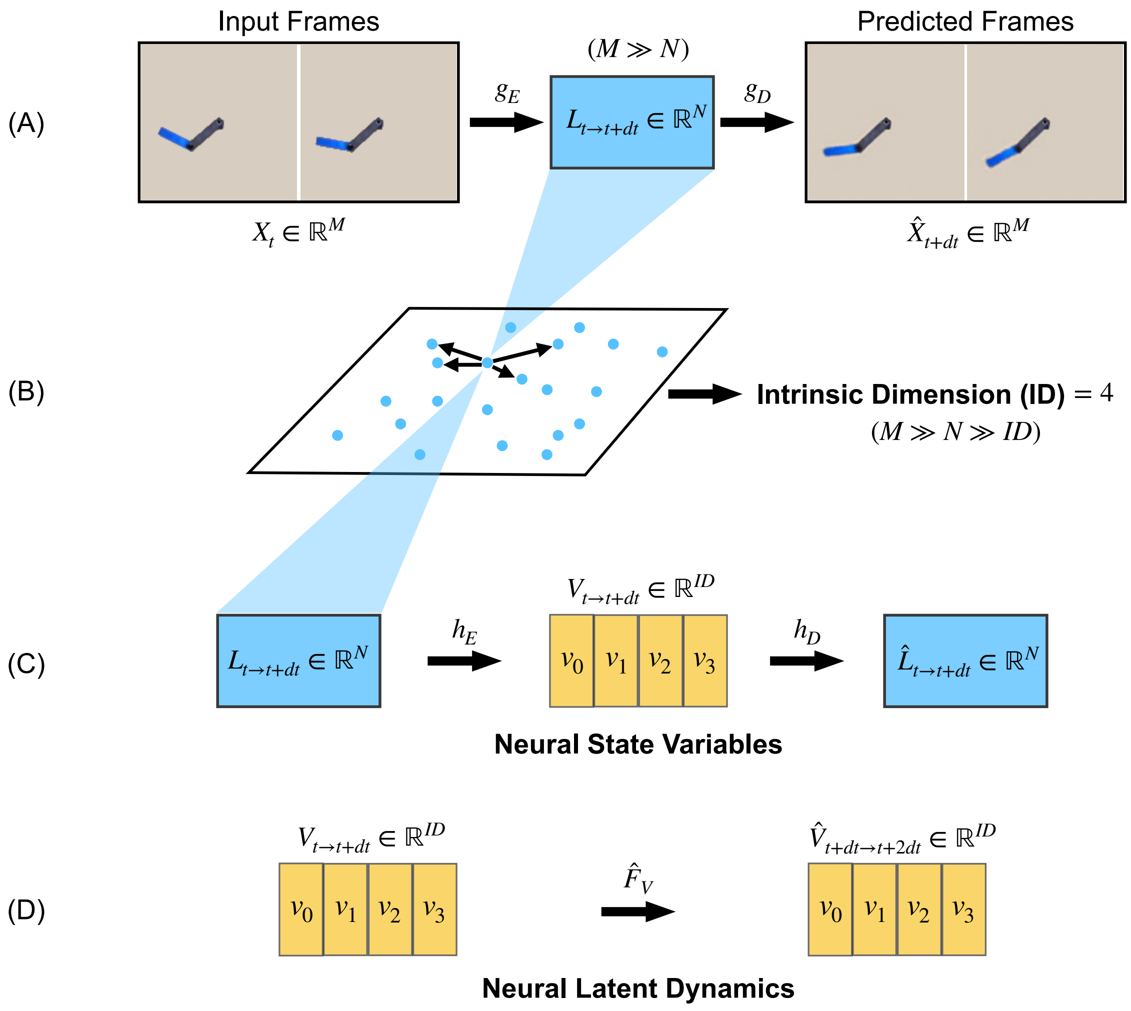
where $\boldsymbol{X}_t \in \mathcal{S}$ is the system's current state at time t, dt is the discrete time increment. $F : \mathcal{S} \to \mathcal{S}$ describes the state evolution from $\boldsymbol{X}_t$ to the system's successive state $\boldsymbol{X}_{t+dt}$ at time $t + dt$. Throughout this chapter we will consider the system as discrete in time. Any continuous in time dynamical system can also be discretized to formulation (6.1) with an appropriate sampling interval dt.

The first step towards modeling a dynamical system is to choose the representation of system states. Previous studies assume that the states are given as the direct measurements of a set of pre-defined state variables, such as the position and velocity of a rigid body object. However, defining which state variables to measure requires expert prior knowledge of the system. For an unfamiliar physical system, we do not know in advance what quantities to measure. Moreover, most state variables are not directly measurable, as they correspond to properties that are physically unobservable in a non-intrusive manner or cannot be uniquely determined without prior knowledge. One example is the measurement of the position of the pendulum arms is very challenging without installing tracking markers which can change the system dynamics.

In this work, we chose video frames as the state representation. Using the notations above, $\mathcal{X}$ is the high dimensional image space. This choice comes with several advantages. First, video recordings do not require prior knowledge of the inner working processes of the observed dynamical system. Second, video cameras collect a rich stream of physics signals, without requiring expensive and specialized equipment. If we can apply our method to data collected by video cameras, then this approach could potentially operate with other types of sensor arrays.

Our goal is to learn a the most compact space that implicitly captures the entire system dynamics, using only high dimensional visual data as input. To achieve this, we formulate a self-supervised learning problem to leverage the natural supervision from future video streams. Our model is based on an auto-encoder neural network to map the high-dimensional visual observations to a relative low-dimensional embedding which will then be projected to a high-dimensional image space again to predict the future video frames. If the model can successfully produce accurate future predictions, the low-dimensional bottleneck has to capture sufficient information on the system dynamics.

Formally, our framework comprises five major components as shown in Fig. 6.2(A): A pair of input image frames $\boldsymbol{X_t}$, an encoder network g_{E}, a latent embedding vector $\boldsymbol{L_{t \to t+dt}}$, a decoder network g_{D}, and a pair of output future frames $\boldsymbol{X_{t+dt}}$. For the dynamical systems studied in our chapter, both the input and output image pairs are two consecutive frames with the dimension $128 \times 128 \times 3$ RGB channels. The pairs of frames are concatenated to form single input and output image. The encoder g_{E} and decoder g_{D} are fully convolutional networks where the decoder also comes with residual connections and multi-scale predictions to enhance higher resolution



Input Frames
Predicted Frames
(A)
$(M \gg N)$
g_E
$L_{t\to t+dt} \in \mathbb{R}^N$
g_D
$X_t \in \mathbb{R}^M$
$\hat{X}_{t+dt} \in \mathbb{R}^M$
(B)
Intrinsic Dimension (ID) = 4
$(M \gg N \gg ID)$
(C)
$L_{t\to t+dt} \in \mathbb{R}^N$
h_E
$V_{t\to t+dt} \in \mathbb{R}^{ID}$
v_0 v_1 v_2 v_3
h_D
$\hat{L}_{t\to t+dt} \in \mathbb{R}^N$
Neural State Variables
(D)
$V_{t\to t+dt} \in \mathbb{R}^{ID}$
$\hat{F}_V$
$\hat{V}_{t+dt\to t+2dt} \in \mathbb{R}^{ID}$
v_0 v_1 v_2 v_3
Neural Latent Dynamics

Fig. 6.2: **Two-stage modeling of dynamical systems.** (A) and (B) **First stage: intrinsic dimension estimation.** We first modelled the dynamical systems via the evolution from $\boldsymbol{X_t}$ to $\boldsymbol{X_{t+dt}}$ with a fully convolutional encoder-decoder network directly from video observations. The dimension of the latent vectors $\boldsymbol{L_{t\to t+dt}}$ is often much lower than the dimension of the input vectors, but much higher than the intrinsic dimension of the system. To identify the intrinsic dimension of the system, we applied geometric manifold learning algorithms on this set of relative high dimension latent vectors. (C) **Second stage: discover Neural State Variables.** We applied another encoder-decoder network on top of the above latent vectors to automatically determine the Neural State Variables by limiting the latent dimension of this network with the identified intrinsic dimension. Our two-step approach can produce Neural State Variables with the exact dimension of the system intrinsic dimension. (D) Once we determine the Neural State Variables, we can leverage the system dynamics in the space of Neural State Variables as an indicator of dynamics stability. Therefore, we learn a neural latent dynamics to predict the the Neural State Variables at the next time step from the current Neural State Variables.

predictions. The network first outputs $\boldsymbol{L}_{t\to t+dt} = g_{\mathrm{E}}(\boldsymbol{X_t})$ and then generates the future frames $\boldsymbol{\hat{X}_{t+dt}}$.

To train the encode and decoder networks, we use simple L2 loss function without other constraints:

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{X}} \left[\| g_{\mathrm{D}}(g_{\mathrm{E}}(\boldsymbol{X_t})) - \boldsymbol{X_{t+dt}} \|_2^2 \right].$$

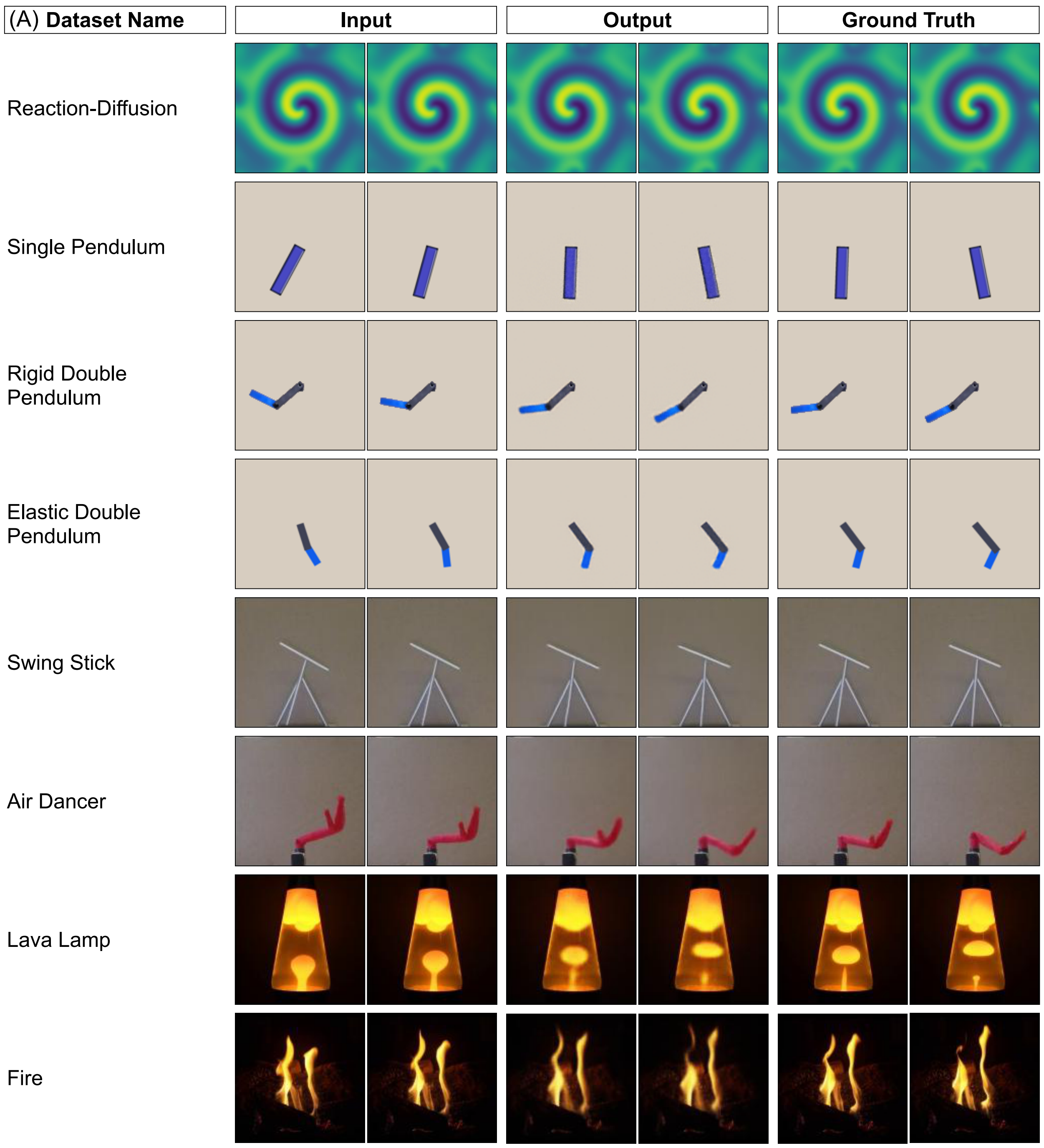
The learned mapping $\hat{F} = g_{\mathrm{D}} \circ g_{\mathrm{E}}$ provides a numerical approximation of the system's evolution mapping F through the latent embedding.

One critical but largely ignored design decision is the dimension LD of the latent embedding $\boldsymbol{L} \in \mathbb{R}^{\mathrm{LD}}$. In Machine Learning, LD is often treated as a hyperparameter selected using an "educated guess" because it is not immediately clear what the best value of LD should be. However, this dimensionality is especially important for physical dynamics modeling. When LD is large, the latent embedding can hold large numbers of useful bits of information about the system dynamics. However large embedding vectors overfit the data are have limited capacity for longer range prediction. More importantly, large latent spaces hide and obfuscate the compact set of state variables we are after. When LD is too small, the network may under-fit the data.

Therefore, we aim to come as close as possible to the exact number of state variables. In the next section, we will carefully analyze these issues and introduce our novel solutions to systematically discover the most compact space of the embedding vector without imposing optimization challenges.

To study the generality of the proposed approach, we compiled a dataset comprising videos of nine physical dynamical systems from various experimental domains (Fig. 6.1), ranging from simple periodic motion (circular motion, single pendulum), chaotic kinematics (rigid double pendulum, elastic double pendulum, swing stick), nonlinear wave (reaction-diffusion system), multi-phase flow (lava lamp), to aeroelasticity (air dancer) and combustion (flame dynamics). Each dataset comes with raw video recordings of the dynamical systems. For certain dataset where measurements of known physical quantities are easy to obtain, we also provide ground-truth estimations for evaluation purposes only.

In Fig. 6.3(A), we show comparisons between the predicted video frames and the ground-truth recordings. Our model was able to produce accurate video predictions. Our model also substan-



(B) **Elastic Double Pendulum Evaluations**

Method	θ_1 (deg)	θ_2 (deg)	z (m)	$\dot{\theta}_1$ (deg/s)	$\dot{\theta}_2$ (deg/s)	$\dot{z}$ (m/s)	Total energy (J)
Copy data	9.03 (±0.04)	19.93 (±0.10)	0.02 (±0.00)	186.49 (±1.12)	454.52 (±3.04)	0.87 (±0.01)	0.05 (±0.00)
Linear extrapolation	3.30 (±0.02)	7.91 (±0.05)	0.02 (±0.00)	186.49 (±1.12)	454.52 (±3.04)	0.87 (±0.01)	0.09 (±0.00)
Our model	0.78 (±0.01)	2.65 (±0.02)	0.00 (±0.00)	62.59 (±0.45)	180.82 (±1.22)	0.27 (±0.00)	0.06 (±0.00)

tially outperforms linear extrapolation and copying input data baselines in Fig. 6.3(B). For dataset with ground-truth physical quantities such as elastic double pendulum, our system was able to predict the physical variables accurately compared to the ground truth. Overall, the evaluation results suggest that the model successfully captured a nontrivial understanding of the system dynamics.

6.2.2 Intrinsic Dimension Estimation

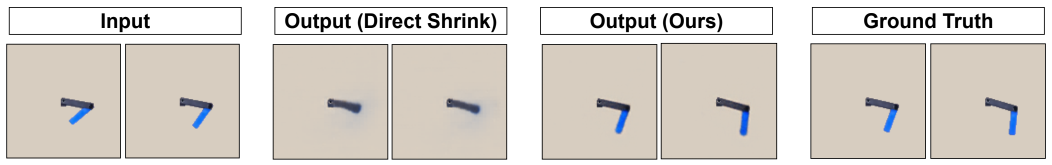
Intrinsic Dimension (ID) has served as a fundamental concept in many advances in physical sciences. In general, the intrinsic dimension refers to the minimum number of independent variables needed to fully describe the state of a dynamical system. The intrinsic dimension is independent of specific representations of the system or choice of a particular set of state variables. In a more quantitative way, the intrinsic dimension could be equivalently defined as the topological dimension of the state space $\mathcal{S}$ as a manifold in the ambient space $\mathcal{X}$ [200, 201, 202].

A common assumption when analyzing a physical system is that the intrinsic dimension is known a priory. An even stronger assumption is that the corresponding state variables themselves are given. Yet these assumptions do not hold for unknown or partially known systems. In order to uncover the underlying dynamics of a wide range of systems and make future predictions of their future behaviors, we need to automatically identify the intrinsic dimension of the systems and extract the corresponding state variables from observed data, which is often high dimensional and noisy.

A naive approach using an auto-encoder predictive framework is to keep reducing the size of the latent embedding vector through trial and error until the output is no longer valid. However, this approach does not yield satisfactory results because the output deteriorates long before the

Fig. 6.3 *(preceding page)*: **Prediction visualizations and physics evaluations** (A) Visualizations of our basic prediction results. (B) For systems where physical variables happen to be available, we obtained the physical variables from both the predicted frames and the ground truth frames. We then performed physics evaluations on these systems. We show the results of elastic double pendulum here. The elastic double pendulum dataset has 60 fps. Our prediction model outperforms both the copy data and linear extrapolation baselines suggesting that our model captures nontrivial understanding of the system's second order dynamics.

(A)



(B)

System	ID from Raw Images	ID from Latent Vectors	Ground Truth
Circular motion	3.67 (±0.20)	2.19 (±0.05)	2
Reaction diffusion	2.20 (±0.13)	2.16 (±0.14)	2
Single pendulum	3.65 (±0.08)	2.05 (±0.02)	2
Rigid double pendulum	7.08 (±0.15)	4.71 (±0.03)	4
Swing stick	11.47 (±0.38)	4.89 (±0.33)	4
Elastic double pendulum	7.55 (±0.15)	5.34 (±0.20)	6
Air dancer	8.09 (±0.19)	7.57 (±0.13)	NA
Lava lamp	7.99 (±0.41)	7.89 (±0.96)	NA
Fire	20.10 (±0.50)	24.70 (±2.02)	NA

(C)

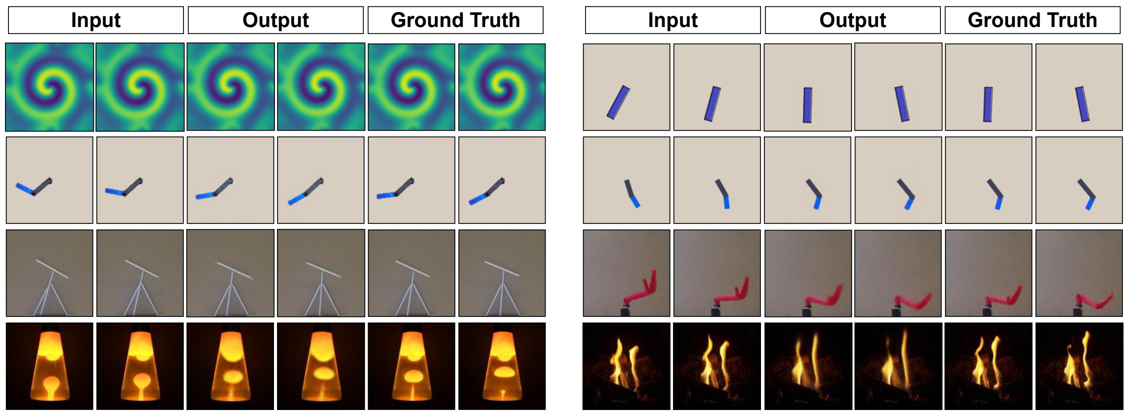


Fig. 6.4: **Intrinsic Dimension (ID) and Neural State Variables** (A) Keep reducing the size of the latent embedding on the original auto-encoder to find ID is not feasible due to optimization difficulties. The network could not converge to a satisfactory solution. With our two-stage method to retrieve the system intrinsic dimension and further discovered Neural State Variables, we can bypass this limitation to produce accurate future predictions. (B) Our method estimates ID without prior knowledge about the systems' state variables. The estimated ID value are rounded to the nearest even integer as position and velocity variables are in pairs. For systems with known IDs, our calculations give accurate results. For unknown systems, the ranking of the ID also makes sense. Our method outperforms direct estimations from raw images. (C) More results on one-step prediction with our discovered Neural State Variables.

minimal set of state variables is reached. As shown in Fig. 6.4(A), the model predictions broke down when we directly shrank the size of the latent space to the intrinsic dimension.

Inspired by traditional manifold learning methods that utilize geometric structures of the embedding vectors (such as their nearest distances), we propose a solution that can automatically discover the intrinsic dimension of a dynamical system from the latent vectors. Our approach only needs a one-time network training step. Specifically, we applied the Levina-Bickel's algorithm [203] on the latent embedding space. The algorithm considers latent vectors $\{\boldsymbol{L}^{(1)}, \boldsymbol{L}^{(2)}, \cdots, \boldsymbol{L}^{(N)}\}$ collected from the trained dynamics predictive model as N data points on a manifold of dimension ID in the latent embedding space. A key geometric observation is that the number of data points within distance r from any given data point $\boldsymbol{L}^{(i)}$ is proportional to r^{ID} when r is small. Based on the observation, the Levina-Bickel's algorithm derives the local ID estimator near $\boldsymbol{L}^{(i)}$ as $\frac{1}{k-2}\sum_{j=1}^{k-1}\log\frac{T_k(\boldsymbol{L}^{(i)})}{T_j(\boldsymbol{L}^{(i)})}$, where $T_k(\boldsymbol{L}^{(i)})$ is the Euclidean distance between $\boldsymbol{L}^{(i)}$ and its k^{th} nearest neighbor in $\{\boldsymbol{L}^{(1)}, \boldsymbol{L}^{(2)}, \cdots, \boldsymbol{L}^{(N)}\}$. The global ID estimator is then calculated as:

$$\mathrm{ID}_{\text{L-B}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{k-2}\sum_{j=1}^{k-1}\log\frac{T_k(\boldsymbol{L}^{(i)})}{T_j(\boldsymbol{L}^{(i)})}.$$

Fig. 6.4(B) shows the estimations across all the systems in our holdout dataset along with baseline comparisons from raw image observations and partial ground-truths. Our method demonstrates highly accurate estimations of the intrinsic dimension of all known systems. Although we cannot account for the ground-truth intrinsic dimension of other systems, we do see that our experiments presented a reasonable and intuitive relative ranking among all listed systems.

We also compared the performance of Levina-Bickel's algorithm with other popular intrinsic dimensionality estimation algorithms including MiND_ML, MiND_KL, Hein, and CD [202, 204, 205, 206, 207] by following the original implementations [206, 205]. Though all the algorithms demonstrated promising performance, we found that the Levina-Bickel algorithm gives the most robust and reliable estimation.

6.3 Neural State Variables

6.3.1 Discovery of the Neural State Variables

As we have discussed above, the minimum set of independent state variables $\boldsymbol{V}$ used to describe the dynamical system has the dimension known as the intrinsic dimension, namely $\boldsymbol{V} \in \mathbb{R}^{\mathrm{ID}}$. To simplify the terminology, we refer them as **State Variables** directly throughout the rest of this chapter.

Now that we have identified the number of state variables, we need to find the actual variables themselves (bearing in mind that the set is not unique). We propose a two-stage framework to retrieve possible State Variables from raw video data with both dynamics predictive and latent reconstruction neural networks. We term our subset of State Variables as **Neural State Variables**. Hence, with Neural State Variables $\boldsymbol{V}$, the dynamical system can be expressed as the evolution of the trajectory $\{\boldsymbol{V}_{0\rightarrow dt}, \boldsymbol{V}_{dt\rightarrow 2dt}, ...\}$.

Our key idea is to break down the identification process into two stages as illustrated in Fig. 6.2(A)-(C). The first stage is to train a dynamics predictive model and identify the intrinsic dimension ID as indicated in the previous section. This stage yields a relative low-dimensional latent embedding $\boldsymbol{L} \in \mathbb{R}^{\mathrm{LD}}$ where LD is still much larger than ID. As shown in Fig. 6.3, the network can converge to output accurate future frames, indicating that the latent embedding captures sufficient information about the complete system dynamics.

The second stage operates directly on the latent embedding to further distill the Neural State Variables. Directly reducing the latent vector using the first step did not converge (Fig. 6.4(A)). Therefore, we trained a second auto-encoder network that takes in the pre-trained latent embedding and outputs the reconstruction of the input. The special property of this network is that the size of the latent embedding equals to the intrinsic dimension obtained from the first step. With a minimum reconstruction error, we can identify this latent embedding vectors as the Neural State Variables. Specifically, the network can be expressed as follows: $\boldsymbol{V}_{t\rightarrow t+dt} = h_{\mathrm{E}}(\boldsymbol{L}_{t\rightarrow t+dt})$ and $\hat{\boldsymbol{L}}_{t\rightarrow t+dt} = h_{\mathrm{D}}(\boldsymbol{V}_{t\rightarrow t+dt})$ where h_{E} and h_{D} refers to the encoder and decoder network of the latent reconstruction model. We train the latent reconstruction model with the L2 loss: $\mathcal{L} = \mathbb{E}_{\boldsymbol{L}}\left[\|h_{\mathrm{D}}(h_{\mathrm{E}}(\boldsymbol{L}_{t\rightarrow t+dt})) - \boldsymbol{L}_{t\rightarrow t+dt}\|_2^2\right]$.

Overall, our two-stage method bypasses the optimization challenges and avoides the risk of under-fitting the observed data. In Fig. 6.4(C), we qualitatively demonstrate the effectiveness of our approach. For all the systems in our dataset, our framework is able to predict accurate future frames from super compact variables with dimension ID (e.g., ID = 4 for rigid double pendulum and ID = 6 for elastic double pendulum). In the next section, we will provide quantitative evaluations and demonstrate its key usage to predict the long-term evolution of these dynamical systems.

6.3.2 Neural State Variables for Stable Long-Term Prediction

Forecasting the long-term future behaviors of unknown physical systems by learning to model their dynamics is critical for numerous real-world tasks. With a dynamics predictive model giving the one-step prediction, we can perform model rollout to feed each step's prediction as the input to predict the next state. However, there are two main challenges to obtaining satisfactory long-term predictions:

- **One-Step Prediction Accuracy** The learned dynamics may not be accurate since prediction errors are iteratively introduced at every prediction step. This issue mainly attributes to the one-step prediction accuracy.
- **Long-term Prediction Stability** Due to error accumulation, the predicted sequences may not be able to maintain the ground truth state space: one repeated observation from past studies is that the long-term predicted sequences become blur, heavily distorted, or plain background within only a few rollouts. We also observed similar phenomena in our experiments as shown in Fig. 6.4(A). This is a very important issue to resolve because if objects deform or entirely disappear without following the system dynamics, it would be impossible to follow the system evolution faithfully. Here, we first define this phenomena as long-term prediction stability. We then present quantitative analysis for various prediction schemes.

Finally, we will present our solution to approach the stability challenge with Neural State Variables.

Long-term prediction stability refers to the deviation between the predicted sequences generated from the learned dynamics and the ground truth state space governed by the system dynamics. Given a metric $M_{\mathcal{S}}(\cdot)$ that measures the deviation from a predicted state to the true state space $\mathcal{S}$, and a prediction sequence $\{\hat{\boldsymbol{X}}_0, \hat{\boldsymbol{X}}_{dt}, \cdots\}$ from any initial state $\hat{\boldsymbol{X}}_0$, we can quantify the stability of a prediction scheme as the growth rate of $M_{\mathcal{S}}(\hat{\boldsymbol{X}}_t)$ as a function of t.

One challenge is to define at what point is the predicted image so degraded that it does not count as a prediction at all. We define an image quality test as follows (used only for evaluation): For systems for which we have prior knowledge about their conventional state variables, and we can extract these physical variables from the corresponding videos through classic computer vision techniques (e.g., color and contour extraction), $M_{\mathcal{S}}^{\text{phys}}(\cdot)$ can be readily defined as a binary value indicating whether the same set of physical variables can still be distilled from a predicted state $\hat{\boldsymbol{X}}$ as its corresponding ground truth state. Consequently, if the predicted frame is heavily blurred or distorted, we will not be able to distill meaningful physical variables. Thus, $M_{\mathcal{S}}^{\text{phys}}(\hat{\boldsymbol{X}})$ will be one. Otherwise $M_{\mathcal{S}}^{\text{phys}}(\hat{\boldsymbol{X}})$ will be zero.

Moreover, to more generally capture the long-term predictive stability of various prediction schemes, $M_{\mathcal{S}}^{\text{phys}}(\cdot)$ should be evaluated on prediction sequences with multiple initial states. Therefore, we further define $M_{\mathcal{S}}^{\text{phys}}(\cdot)$ as the Reject Ratio to indicate how many predicted frames at each time step from different initial states will fail to pass the physical variables extraction test.

With the above test, we can quantitatively compare the stability of various long term prediction schemes. These schemes are based on iterative model rollouts but they differ in the size of intermediate variables. When the model rollouts are through high dimensional latent vectors (8192 variables or 64 variables), the iterative scheme is given by $\hat{\boldsymbol{X}}_{t+dt} = g_{\text{D}} \circ g_{\text{E}}(\hat{\boldsymbol{X}}_t), \quad t = 0, dt, \ldots,$ where g_{E} and g_{D} represent the first auto-encoder that transforms input frames to the predicted frames via latent embeddings. When the model rollouts are through Neural State Variables, the iterative scheme is given by $\hat{\boldsymbol{X}}_{t+dt} = g_{\text{D}} \circ h_{\text{D}} \circ h_{\text{E}} \circ g_{\text{E}}(\hat{\boldsymbol{X}}_t), \quad t = 0, dt, \ldots,$ where h_{E} and h_{D}

represent the second auto-encoder that reconstructs latent embeddings via Neural State Variables. The original latent embeddings are computed from input frames with g_{E}, and the reconstructed latent embeddings will be sent to g_{D} to produce the final predicted frames.

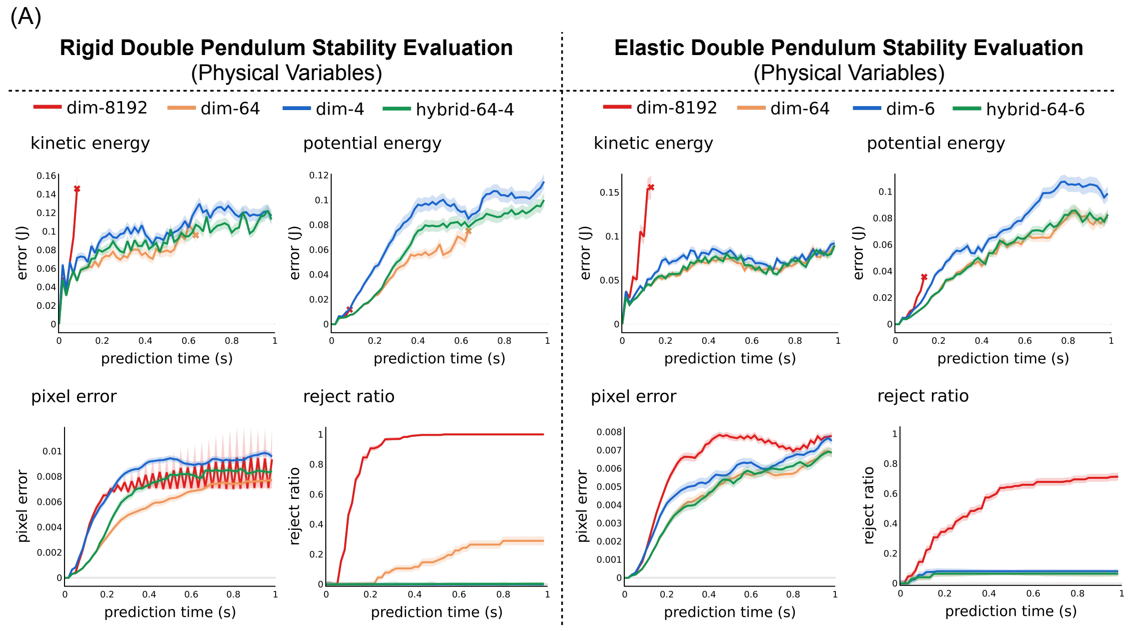
Fig. 6.5(A) shows the stability results on rigid double pendulum and elastic double pendulum where we can extract the physical variables from videos. The 8192-dim and 64-dim scheme cannot give stable long-term future predictions. In our experiments, we noticed that both schemes can provide stable predictions when the system intrinsic dimension is smaller or equal than 2.

Inspired by lessons learned from computational physics, an effective fix to the unstable long-term prediction is to construct a prediction scheme where the predicted states will be projected into a small neighborhood of the state space. Here Neural State Variables serve as a strong candidate solution. This is because Neural State Variables have the same dimension as the intrinsic dimension. This fact prevents predictions from falling off the system manifold into new dimensions.

The blue curves in Fig. 6.5(A) illustrate the stability introduced by using Neural State Variables as intermediate representations for long-term predictions. Neural State Variables provide the most stable predictions across all the systems. However, since Neural State Variables were obtained by performing reconstruction on a relative high-dimensional latent embedding, they have an inferior performance on one-step prediction accuracy.

To combine the best of two worlds, we propose a hybrid scheme as our final solution: using Neural State Variables as stabilizers while performing long-term predictions with their corresponding high-dimensional latent embeddings. Formally, the hybrid scheme follows an $N + 1$ pattern where for every N steps performed with the high-dimensional latent vectors, a one step prediction is followed with the Neural State Variables. As shown by the green curves in Fig. 6.5(A) and (B), our hybrid scheme offers stable and accurate long-term predictions. In Fig. 6.5(A) and (B), the hybrid scheme was implemented with specific values of N between 3 and 6. We also conducted experiments with different choices of N and found that the outcomes were not sensitive to the particular values of N.

Another important note is that the use of pixel error as the evaluation metric, though easy



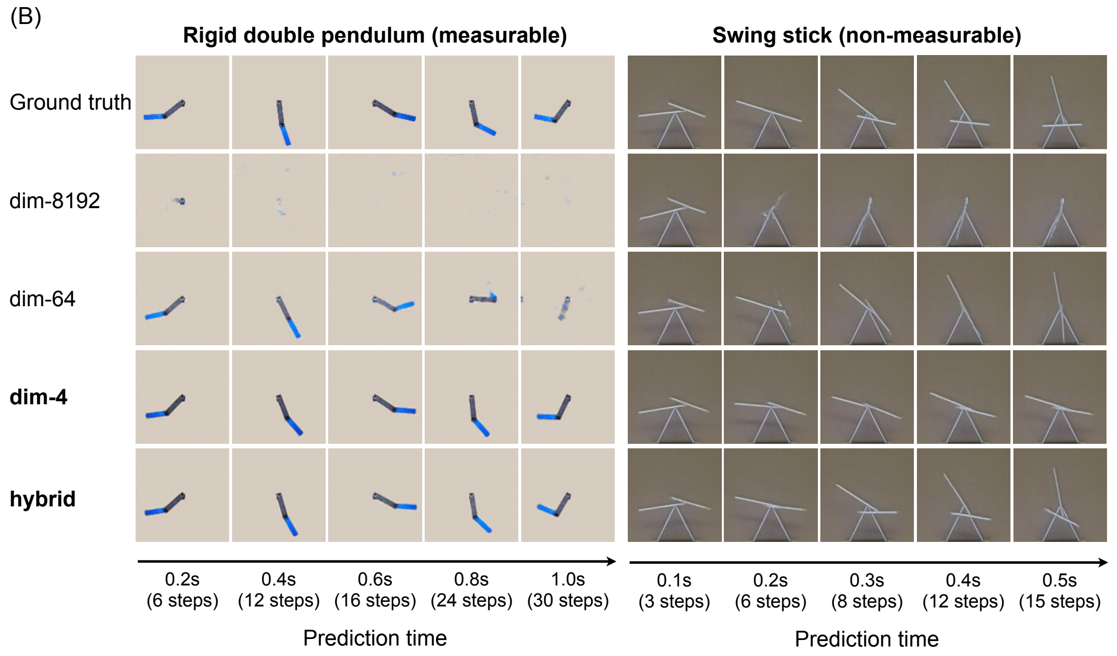


Fig. 6.5: **Long-term Prediction Stability** (A) For systems where we could extract physical variables such as rigid and elastic double pendulum, using Neural State Variables as intermediate representations gives the most stable long-term future predictions. With our hybrid scheme, we can achieve relatively stable and accurate predictions. Pixel error cannot replace physics-based evaluation for measuring the long-term prediction stability. (B) Predictions through Neural State Variables can maintain the ground truth state space for long-term predictions. The hybrid scheme offers the most stable and accurate predictions.

to compute, can be misleading for the evaluation of long-term predictions when the predictions quickly become unstable. As quantitatively and qualitatively demonstrated in Fig. 6.5, pixel errors remain roughly the same after the predicted images become plain backgrounds. These pixel errors are even smaller than the pixel errors computed from a slightly inaccurate but clear prediction. This observation further emphasizes the significance of designing an appropriate $M_{\mathcal{S}}(\cdot)$ metric.

6.3.3 Neural State Variables for Dynamics Stability Indicators

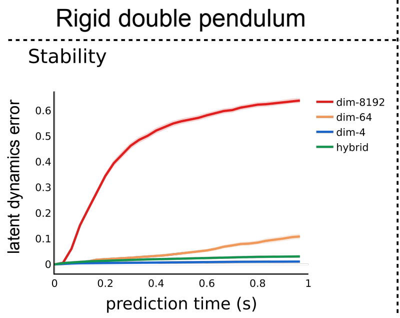
So far, for evaluating long-term prediction stability, we have been assuming that we can extract the physical variables from the system states during evaluation. Yet, it is common that, in most of the video representations in our dataset, we know neither which variables to extract nor how to extract them directly from videos. As noted above, pixel errors are also not reliable. In this case, a very challenging but important problem is how we can evaluate the long-term prediction stability from videos. Resolving this problem can potentially open up the door to quantitatively evaluate prediction stability of various schemes for many complex and unknown systems, all directly from videos.

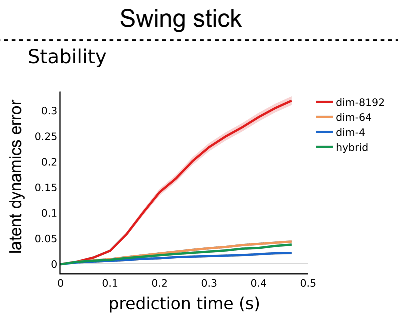
Following the framework in the last section, the key is the design of the metric $M_{\mathcal{S}}(\cdot)$. Here we propose a solution based on Neural State Variables, namely $M_{\mathcal{S}}^{\text{neur}}$. Specifically, $M_{\mathcal{S}}^{\text{neur}}$ is a metric on a pair of states $(\hat{\boldsymbol{X}}_t, \hat{\boldsymbol{X}}_{t+dt})$.

$$M_{\mathcal{S}}^{\text{neur}}(\hat{\boldsymbol{X}}_t, \hat{\boldsymbol{X}}_{t+dt}) = \left\| h_{\text{E}} \circ g_{\text{E}}(\hat{\boldsymbol{X}}_{t+dt}) - \hat{F}_V(h_{\text{E}} \circ g_{\text{E}}(\hat{\boldsymbol{X}}_t)) \right\|,$$

where $\hat{F}_V$ is a neural network trained to approximate the latent dynamics on the space of Neural State Variables $\hat{\boldsymbol{V}}_{t+dt\to t+2dt} \leftarrow \hat{F}_V(\boldsymbol{V}_{t\to t+dt})$, $h_{\text{E}} \circ g_{\text{E}}(\hat{\boldsymbol{X}}_t) = \hat{\boldsymbol{V}}_{t\to t+dt}$ and $h_{\text{E}} \circ g_{\text{E}}(\hat{\boldsymbol{X}}_{t+dt}) = \hat{\boldsymbol{V}}_{t+dt\to t+2dt}$ are Neural States in $\mathbb{R}^{\text{ID}}$, $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^{\text{ID}}$.

Intuitively, $M_{\mathcal{S}}^{\text{neur}}$ measures how far the predicted dynamics, reflected by the given predicted sequence, deviates from the reduced system dynamics projected onto the space of Neural State Variables. As shown in the previous sections, all latent embeddings with higher or equal to the intrinsic dimension may provide accurate short-term approximation of system dynamics. However, we chose the space of Neural State Variables as the reference because it has the same dimension





Latent dynamics error vs. Reject ratio

System	Correlation
Single pendulum	97.33%
Rigid double pendulum	97.45%
Elastic double pendulum	98.76%

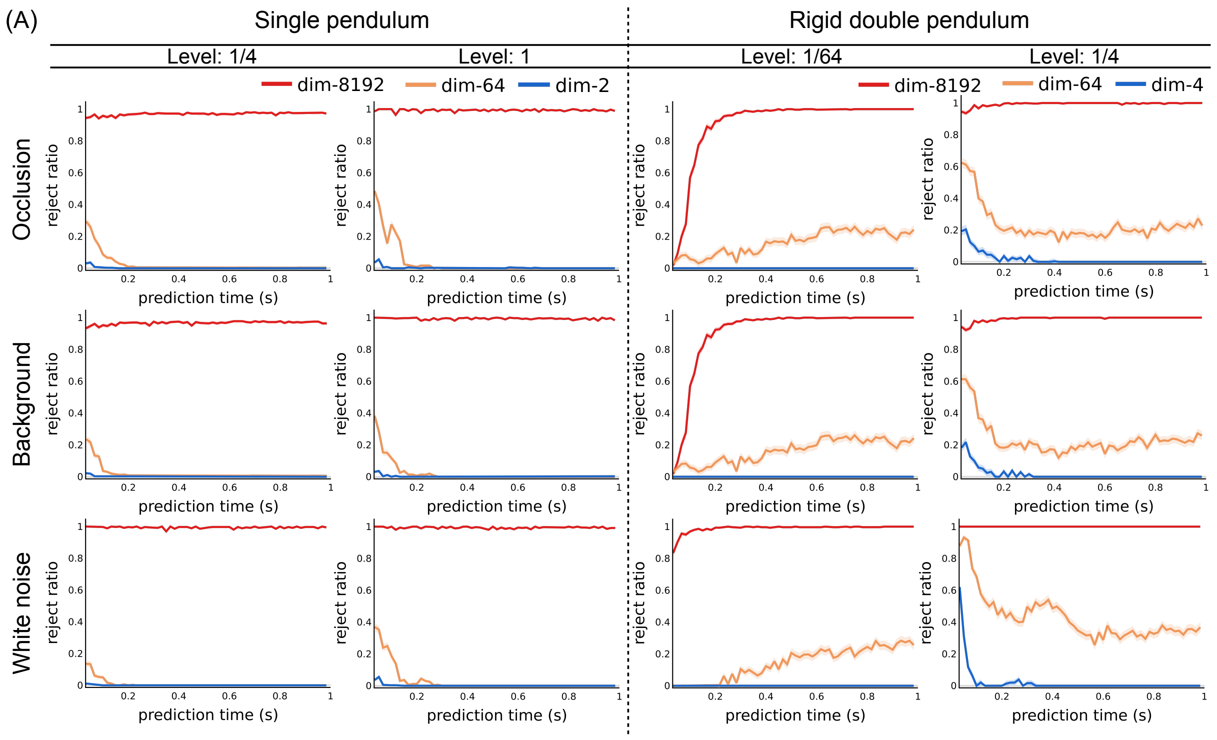
Fig. 6.6: **Neural State Variables for Dynamics Stability Indicators** The latent dynamics on the space of Neural State Variables, namely neural latent dynamics, can be used as an effective indicator to quantify the stability of long-term future predictions. The strong correlation between the latent dynamics error and the physics reject ratio implies that our latent dynamics error can be used as an alternative metric to measure long-term prediction stability.

as ID. First, as mentioned above, Neural State Variables project the predicted states in the small neighborhood of the ground truth states. Moreover, the Euclidean distance serve as a good metric to measure dynamics deviation in this case, while other higher dimensions may suffer the curse of dimensionality when designing the distance metric. Overall, M_S^{neur} is an ideal alternative candidate to M_S^{phys}.

Similar to M_S^{phys}, the final M_S^{neur} is computed across multiple prediction sequences with various initial states. We show the evaluation results with our stability metric based on Neural State Variables in Fig. 6.6. M_S^{neur} produces patterns that highly match with M_S^{phys} for the systems where we know how to extract physical variables. This can also be seen in the correlation plot in Fig. 6.6 where we computed the Pearson correlation coefficient between reject ratio of all models (dim-8192, dim-64, dim-ID, hybrid) at all prediction steps and the respective latent dynamics errors. For unknown systems, we observed the same trend where high-dimensional latent embeddings schemes are often not stable. In conclusion, our M_S^{phys} metric can help us measure the long-term prediction stability directly from videos without additional prior knowledge of the system.

6.3.4 Neural State Variables for Robust Long-Term Prediction

Another critical factor when modeling system dynamics from videos is the robustness against visual perturbations. Therefore we applied several visual perturbations on the input video frames



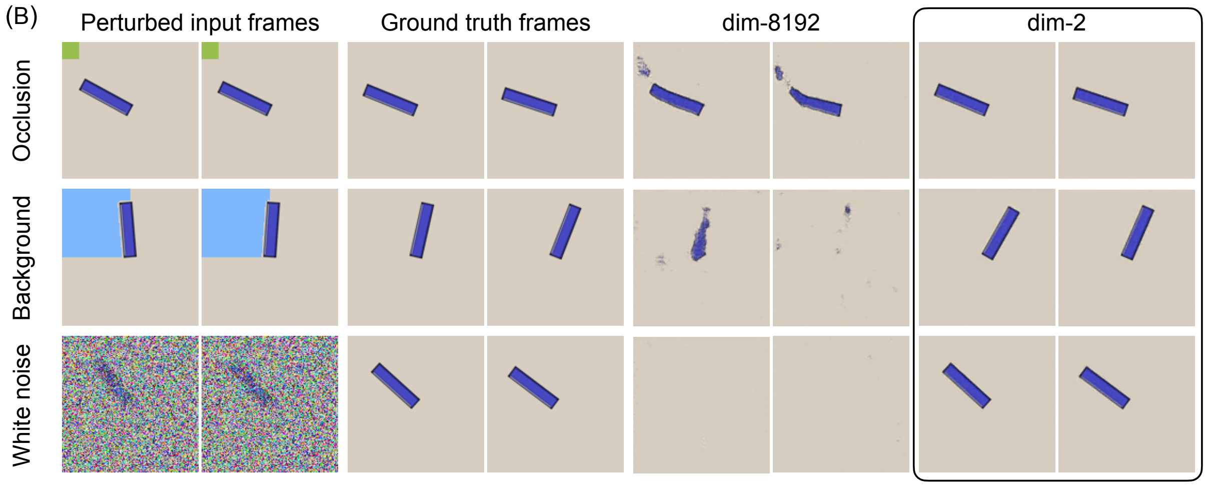


Fig. 6.7: **Neural State Variables for robust long-term prediction** (A) The physical reject ratio can effectively measure long-term prediction robustness from perturbed initial data. For both systems and all combinations of perturbation types and perturbation levels, the model rollouts through Neural State Variables provide the most robust predictions. (B) From perturbed initial frames, the dynamics predictive model with high dimensional latent vectors produces blurred images or pure background with only one model rollout, while the model rollouts through Neural State Variables still give reasonable predictions.

during test time and evaluated the performances of different models.

Specifically, we performed three types of perturbations. The first type is to simulate camera occlusions by covering certain portion of the input frames with a randomly generated color square. The area of the square indicates the level of the perturbation. For example, $\frac{1}{64}$ means the area of the square is $\frac{1}{64}$ times of the area of one input frame.

The second type is to simulate background color change by covering certain portion of the input frame background with a randomly generated color square. The main difference between this perturbation and the first one is that the color square will not cover the object. The level definition is the same with the first perturbation type.

Lastly, to simulate possible sensor noises, we added random Gaussian noise on the input frames. The Gaussian noise has a zero mean and different level of standard deviations. For example, $\frac{1}{64}$ means the Gaussian noise has a standard deviation of $\frac{1}{64} \times 255$ where 255 is the highest pixel value in the input frames.

We show the test-time results using the physical reject ratio metric in Fig. 6.7(A). The quantitative results clearly demonstrate the strong robustness of models on the Neural State Variables space across all level of perturbations. The models with very high dimensional latent space quickly produce unstable predictions.The models with a relative lower dimension but still higher than ID can sometimes give stable predictions again after several unstable rollouts. However, even though the predictions can become stable again, it requires much more number of prediction steps. We also show qualitative visualizations in Fig. 6.7(B).

6.3.5 Analysis

We hypothesize that Neural State Variables contain rich physical meanings that align with the conventional definition of the physical State Variables. In this section, we verify this hypothesis through both quantitative regression experiments and qualitative visualizations.

We trained a small neural network with five layers of MLPs to regress conventional physical variables including positions, velocities, and energies from learned Neural State Variables. Our results are shown in Fig. 6.8(A). Using 30% of labeled data, the learned Neural State Variables can be used to accurately regress the physical variables. We then compared the regression errors

(A) Single pendulum (30% training data)

Latent Variables	θ (deg)	$\dot{\theta}$ (deg/s)	Energy (J)
dim-2 PCA of dim-8192 Latents	19.99 (±0.96)	134.68 (±7.76)	0.94 (±0.12)
dim-2 PCA of dim-64 Latents	63.23 (±3.42)	252.00 (±9.60)	1.22 (±0.07)
dim-2 Latents	6.15 (±0.64)	43.45 (±3.84)	0.35 (±0.02)

Rigid double pendulum (30% training data)

Latent Variables	θ_1 (deg)	θ_2 (deg)	$\dot{\theta}_1$ (deg/s)	$\dot{\theta}_2$ (deg/s)	Energy (J)
dim-4 PCA of dim-8192 Latents	29.23 (±1.15)	45.74 (±0.47)	163.67 (±2.77)	380.42 (±2.98)	0.11 (±0.00)
dim-4 PCA of dim-64 Latents	16.87 (±0.91)	16.89 (±0.32)	196.52 (±4.74)	322.27 (±13.35)	0.10 (±0.00)
dim-4 Latents	10.32 (±0.59)	10.17 (±0.51)	88.01 (±3.86)	126.88 (±7.66)	0.06 (±0.00)

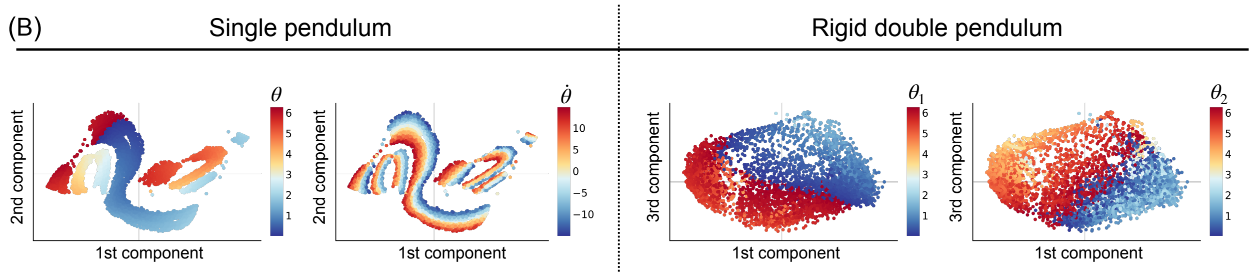


Fig. 6.8: **PCA regression and Neural State Variables visualizations** (A) Neural State Variables capture much richer information about the system dynamics than state variables obtained through PCA from other high dimensional latent embedding vectors. (B) We also visualize the Neural State Variables after applying PCA on them. The colors represent the value of different physical variables. Examples shown here suggest interesting symmetrical structures encoded in the Neural State Variables.

with those from the first few principal components of high dimensional latent vectors from our dynamics predictive model. Using the same number of state variables, which equals ID, and the same labeled data, the regression errors using principal components of high dimensional latent vectors are much larger than those using Neural State Variables, especially for velocity variables. Therefore, state variables obtained through PCA, or equivalently through linear neural networks, can hardly capture the dynamics of the system.

Visualizations colored by the value of physical variables in Fig. 6.8(B) can further demonstrate the physical meaning of the Neural State Variables. We observe that indeed the physical variables are captured in the set of Neural State Variables chosen by our modeling system. The charts also reveal the inherent symmetrical nature of these variables.

Chapter 7: Physical Environment Modeling through Acoustic Vibration

The majority of robotic systems nowadays heavily rely on camera sensors to perceive the information from the environment to model the physics and dynamics. However, environmental conditions are not always ideal for cameras. Although robots frequently need to interact with containers, objects inside containers often become occluded from cameras, making state estimation from vision impractical. In unconstrained settings, environments can also lack ideal lighting conditions, with limited visual signals to indicate the location and state of objects. Moreover, camera systems require extensive calibration for 3D state estimation, which is often fragile during contact and collisions.

In this chapter, we will demonstrate how to leverage *acoustic vibrations* for robots to sense and model their physical environment [208]. In particular, we will focus on the task of object state estimation. Unlike vision, sound remains robust during occlusions or poor illuminations. For example, when an object is dropped inside a container, it may not be visible to any camera, but the collision between the object and the bin will cause a sound that can be easily picked up by a contact microphone. The exact incidental vibration will depend on the location and pose of the objects inside the bin. We demonstrate how to use this audio signal to reconstruct the objects' states inside containers.

We introduce The Boombox, a smart container that uses its vibration of itself to reconstruct an image of its contents. The box is no larger than a cubic square foot. Unlike most containers, the box uses contact microphones to detect its own vibration. Exploiting the link between acoustic and visual structure, we show that a convolutional network can use these vibrations to predict the visual scene inside the container within centimeters, even under total occlusion and poor illumination. Figure 7.1 illustrates our box and one reconstruction from the sound.

Interacting with bins and containers is a fundamental task in robotics, making state estimation

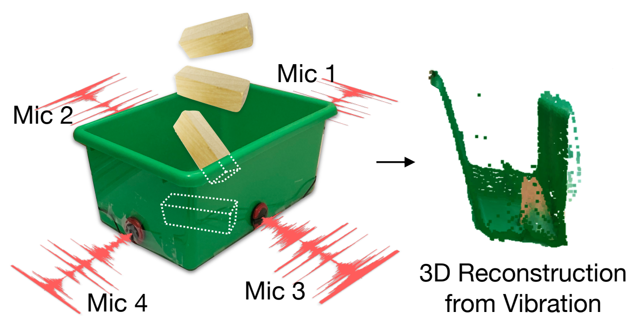


Fig. 7.1: **The Boombox.** We introduce a "smart" container that is able to reconstruct an image of its inside contents. The approach works even when the camera cannot see into the container. The box has four contact microphones on each face. When objects interact with the box, they cause incidental acoustic vibrations. From these vibrations, we learn to predict the visual scene inside the box.

of the objects inside the bin critical. While robots often use cameras for state estimation, the visual modality is not always ideal due to occlusions and poor illumination. We introduce The Boombox, a container that uses sound to estimate the state of the contents inside a box. Based on the observation that the collision between objects and their containers will cause an acoustic vibration, we present a convolutional network for learning to reconstruct visual scenes. Although we use low-cost and low-power contact microphones to detect the vibrations, our results show that learning from multimodal data enables state estimation from affordable audio sensors. Due to the many ways that robots use containers, we believe the box will have a number of applications in robotics.

7.1 The Boombox

7.1.1 Detecting Vibrations

The Boombox, shown in Figure 7.2A, is a plastic storage container that is 15.5cm $\times$ 26cm $\times$ 13cm (width $\times$ length $\times$ height) with an open top. The box is a standard object that one can buy at any local hardware store. When an object collides with the box, a small acoustic vibration will be produced in both the air and the solid box itself. We have attached contact microphones on each wall of the plastic cuboid storage bin in order to detect this vibration. Unlike air microphones, contact microphones are insensitive to the vibrations in the air (which human ears hear as sound). Instead, they detect the vibration of solid objects.

The microphones are attached on the outer side of the walls, resulting in four audio channels.

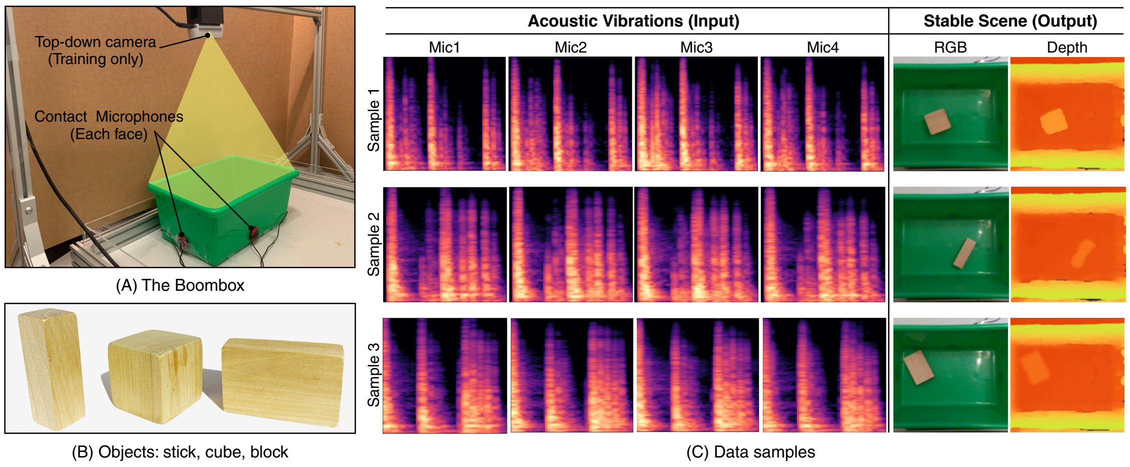


Fig. 7.2: **The Boombox Overview.** (A) The Boombox can sense the object through four contact microphones on each side of a storage container. A top-down RGB-D camera is used to collect the final stabilized scene after the object movements. (B) We drop three wooden objects with different shapes. (C) Input and output data visualizations.

We arrange the microphones roughly at the horizontal center of each wall and close to the bottom. As our approach will not require calibration, the microphone displacements can be approximate. We used TraderPlus piezo contact microphones, which are very affordable (no more than $5 each).[1]

7.1.2 Vibration Characteristics

When objects collide with the box, the contact microphones will capture the resulting vibrations. Figure 7.3 shows an example of the vibration captured from two of the microphones. We aim to recover the visual structure from this signal. As these vibrations are independent of the visual conditions, they allow perception despite occlusion and poor illumination. There is rich structure in the raw acoustic signal. For example, the human auditory system uses inter-aural timing difference (ITD), which is the time difference of arrival between both ears. Humans also locate sounds with inter-aural level difference (ILD), which is the amplitude level difference between both ears [209].

However, in our settings, extracting these characteristics is challenging. In practice, objects will bounce around in the container before arriving at their stable position, as shown in Figure 7.4.

[1]We found that these microphones gave sufficiently clear signals while being more affordable than available directional microphone arrays. Each microphone was connected to a laptop through audio jack to USB converter. We use GarageBand software to record all four microphones together to synchronize the recordings.

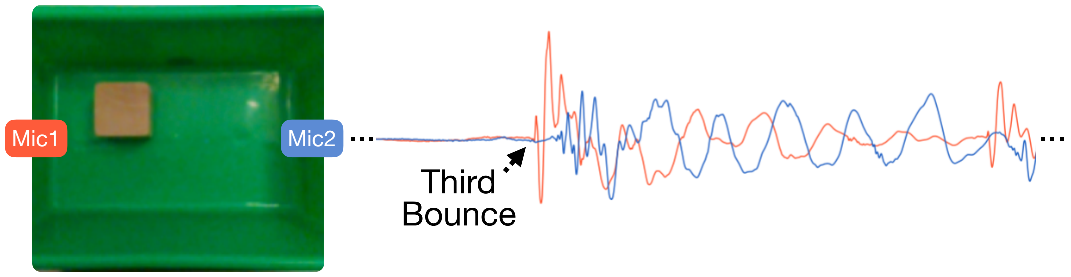


Fig. 7.3: **Vibration Characteristics.** We visualize a vibration captured by two microphones in the box. Several distinctive characteristics need to be combined over time in order to accurately reconstruct an image with the right position, orientation, and shape of objects.

Each bounce will produce another, potentially interfering vibration. In our attempts to analytically use this signal, we found that the third bounce has the best signal for the time difference of arrival, but as can be seen from Figure 7.3, even on the third bounce the time difference of arrival is unclear in the actual waveform. There are a multitude of factors that make analytical approaches not robust to our real-world signals. Firstly, we are working with a moving signal, whereas time difference of arrival calculations work best on stationary signals due to the fact that it compares the time taken for a signal to travel from a fixed location. This makes it very difficult to analytically segment the signal into chunks of roughly the same location. Secondly, there are echos that make non-learning based methods difficult to identify phase shifts as the environment is a small container. Finally, the fact that the microphones are close together means that the time difference of arrival is encompassed in few samples, thus making it susceptible to noise.

Instead of hand crafting features, we will train a model to identify the fraction of the signal that is most robust for final localization. Our model will learn to identify the useful features from the signals to reconstruct a 3D scene, which includes the shape, orientation, and position of the contents.

7.1.3 Multimodal Training Dataset

Since vision and sound are naturally synchronized, we will use vision as self-supervision to learn robust characteristic features of the acoustic signal. We collected a multimodal training dataset by dropping objects into the box and capturing resulting images and vibrations. We position

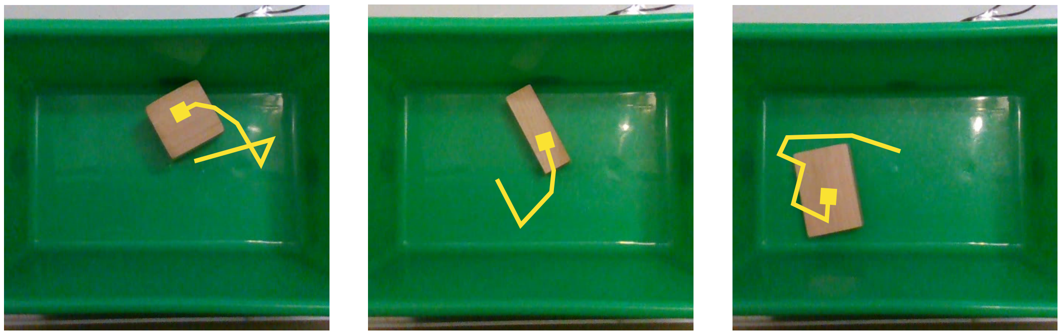


Fig. 7.4: **Chaotic Trajectories.** We show the chaotic trajectories of objects as they bounce around the box until becoming stable. The moving sound source and multiple bounces also create potentially interfering vibrations, complicating the real-world audio signal.

an Intel RealSense D435i camera that looks inside the bin to capture both RGB and depth images, which we only use during training.[2] We use three wooden blocks with different shapes to create our dataset. The blocks have the same color and materials, and we show these objects in Figure 7.2B. We hold the object above the bin, and freely drop it. After dropping, the objects bounce around in the box a few times before settling into a resting position. We record the full process from all the microphones and the top-down camera. Overall, our collection process results in diverse falling trajectories across all shapes with a total of 1,575 sequences. Figure 7.2C shows an overview of the dataset. After learning, our approach will be able to reconstruct the 3D visual scene from the box's vibration alone.

7.2 Predicting the Visual Scene from Vibration

To estimate the state inside the container, we will learn to reconstruct the visual modality from the audio modality. We present a convolutional network that translates vibrations into images.

7.2.1 Model

We will fit a model that reconstructs the visual contents from the vibrations. Let A_i be a spectrogram of the vibration captured by microphone i such that $i \in \{1, 2, 3, 4\}$. Our model will predict the image $\hat{X}_{\text{RGB}} = f_{\text{RGB}}(A; \theta)$ where f is a neural network parameterized by θ. The

[2]The camera is 42cm away from the bottom of the bin to capture clear top-down images.

network will learn to predict the image of a top-down view into the container. We additionally have a corresponding network to produce a depth image $\hat{X}_{\text{depth}} = f_{\text{depth}}(A;\theta)$.

Reconstructing a pixel requires the model to have access to the full spectrogram. However, we also want to take advantage of the spatio-temporal structure of the signal. We therefore use a fully convolutional encoder and decoder architecture. The network transforms a spectrogram representation (time $\times$ frequency) into an C-channel embedding with width and height being 1$\times$1, such that the receptive field of every dimension reaches every magnitude in the input and every pixel in the output. Unlike image-to-image translation problems [210, 211, 212], our task requires translation across modalities.

We use a multi-scale decoder network [6, 213, 125]. Specifically, each decoder layer consists of two branches. One branch is a transposed convolutional layer to up-sample the intermediate feature. The other branch passes the input feature first to a convolutional layer and then a transposed convolution so that the output for the second branch matches the size of the first branch. We then concatenate the output from these two branches along the feature dimension as the input feature for the next decoder layer. We perform the same operation for each decoder layer except the last layer where only one transposed convolution layer is needed to predict the final output image.

We use a spectrogram to represent audio signals. We apply a Fourier Transform before converting the generated spectrogram to Mel scale. Since we have four microphones, audio clips are concatenated together along a third dimension in addition to the original time and frequency dimension.

7.2.2 Learning

In most cases, we care about predicting the resting position of the object. We therefore train the network f to predict the final stable image. For RGB image predictions, we train the network to minimize the expected squared error between the image X_{RGB} and the predictions from audio A:

$$\mathcal{L}_{\text{RGB}} = \mathbb{E}_{A,X}\left[\|f_{\text{RGB}}(A;\theta) - X_{\text{RGB}}\|_2^2\right] \tag{7.1}$$

In order to reconstruct shape, we also train the network to predict a depth image from the audio input. We train the model to minimize the expected L1 distance:

$$\mathcal{L}_{\text{depth}} = \mathbb{E}_{A,X}\left[\|f_{\text{depth}}(A;\phi) - X_{\text{depth}}\|_1\right] \tag{7.2}$$

Since ground truth depth often has outliers and substantial noise, we use an L1 loss [214]. We use stochastic gradient descent to estimate the network parameters θ and ϕ. After learning, the model predicts both the RGB image and the depth image from just the vibration. The visual modality is only supervising representations for the audio modality, allowing reconstructions when cameras are not viable, such during occlusions or low illumination.

7.2.3 Implementation Details

Our network takes in the input size of $128 \times 128 \times 4$ where the last dimension denotes the number of microphones. The output is a $128\times128\times3$ RGB image or a $128\times128\times1$ depth image. We use the same network architecture for both the RGB and depth output representations except the feature dimension in the last layer for different modalities. All network details are listed in the Appendix. Our networks are configured in PyTorch and PyTorch-Lightning. We optimized all the networks for 500 epochs with Adam [127] optimizer and batch size of 32 on a single NVIDIA RTX 2080 Ti GPU. The learning rates starts from 0.001 and decrease by 50% at epoch 20, 50, and 100.

7.3 Experiments

Our experiments analyze the hardware and software at reconstructing an image of the contents from audio. In this section, we first quantitatively evaluate the performance. We then show qualitative results for the reconstructions. Finally, we visualize the learned representations.

Since the physics behind our dataset is chaotic, every time an object is dropped into the container, we obtain a unique example with a different resting position and orientation. We randomly split the dataset into a training set (80%), a validation set (10%), and a testing set (10%). All of our results are evaluated with three random seeds for training and evaluation with various splits of the dataset. We report the mean and the standard error of the mean for all outcomes.

7.3.1 Evaluation Metrics

We use two evaluation metrics for our final scene reconstruction that focus on the object state.

IoU measures how well the model reconstructs both shape and location. Since the model predicts an image, we subtract the background to convert the predicted image into a segmentation mask. Similarly, we performed the same operation on the ground-truth image. IoU metric then computes intersection over union with the two binary masks.

Localization score evaluates whether the model produces an image with the block in the right spatial position. This metric is especially useful for object picking tasks with a suction gripper where the spatial location of the block matters. With the binary masks obtained in the above process, we can fit a bounding box with minimum area around the object region. We denote the distance between the center of the predicted bounding box and the center of the ground-truth bounding box as d, and the length of the diagonal line of ground-truth box as l. We report the fraction of times the predicted location is less than half the diagonal: $\frac{1}{N}\sum_{i=1}^{N}[d_i \leq l/2]$.

7.3.2 Baselines

Time Difference of Arrival (TDoA). We compare against an analytical prediction of the location. A standard practice is to localize sound sources by estimating the time difference of arrival across an array of microphones. In our case, the microphones *surround* the sound source. There are several ways to estimate the time difference of arrival, and we use the Generalized Cross Correlation with Phase Transform (GCC-PHAT), which is a established, textbook approach [215]. Once we have our time difference of arrival estimate, we find the location in the box that would yield a time difference of arrival that is closest to our estimate.

Random Sampling. To evaluate if the learned models simply memorize the training data, we

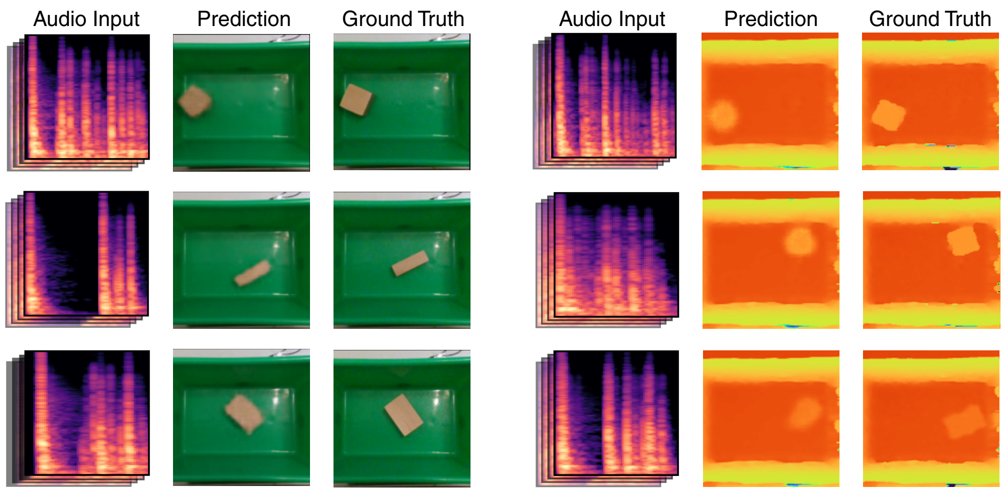


Fig. 7.5: **Model Predictions with Mixed Shapes.** From left to right on each column, we visualize the audio input, the predicted scene, and the ground-truth images. Our model can produce accurate predictions for object shape, position and orientation.

compared our method against a random sampling procedure. This baseline makes a prediction by randomly sampling an image from the training set and using it as the prediction.

Average Bounding Box. The average bounding box baseline measures to what extent the model learns the dataset bias. We extracted object bounding boxes from all the training data through background subtraction and rectangle fitting to obtain the average center location, box sizes and box orientation. This baseline uses the average bounding box as the prediction for all the test samples.

Nearest Neighbor. To evaluate the generalization performance from training data distribution, we construct a nearest neighbor baseline. For each test input audio, we use the resulted image from the training data with most similar audio as the prediction. The similarity is measured by a L2 distance.

7.3.3 Reconstruction with Mixed Shapes

We first analyze how well The Boombox reconstructs its contents when the shape is not known a priori. We train a single model with all the object shapes. The training data for each shape are simply combined together so that the training, validation and testing data are naturally well-balanced with respect to the shapes. This setting is challenging because the model needs to learn

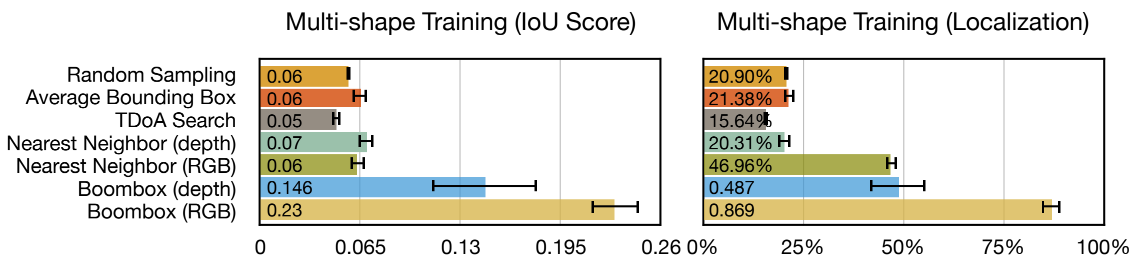


Fig. 7.6: **Reconstruction with Mixed Shapes.** Our model outperforms baseline methods at localization and shape prediction.

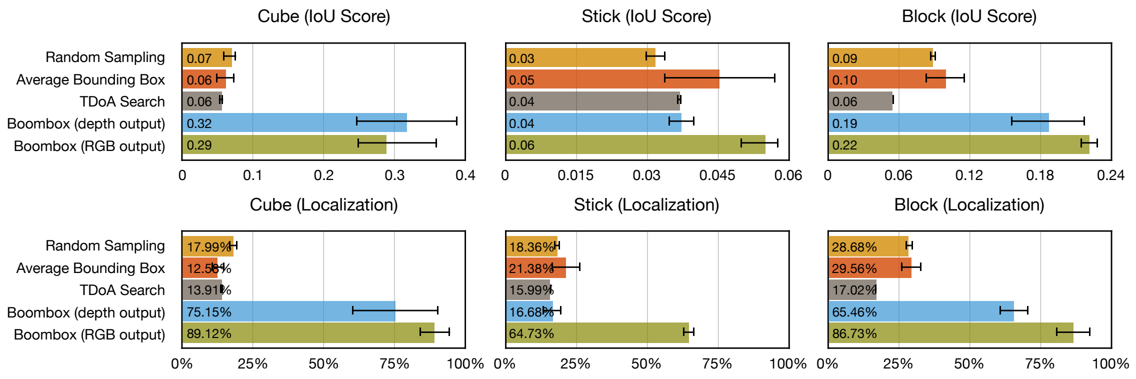


Fig. 7.7: **Reconstruction with Known Shape.** We show the performance of each individual model trained with one of the three objects. We report both the mean and the standard error of the mean from three random seeds. Our approach enables robust features to be learned to predict the location and shape of the dropped objects.

audio features for multiple shapes at once.

Figure 7.6 shows the convolutional networks are able to learn robust features to localize both the position and orientation, even when shapes are mixed. Our method outperforms TDoA often by significant margins, suggesting that our learning-based model is learning robust acoustic features for localization. Due to the realistic complexity of the audio signal, the hand-crafted features are hard to reliably estimate. Our model outperforms both the random sampling and average bounding box baseline, indicating that our model learns the natural correspondence between acoustic signals and visual scene rather than memorizing the training data distribution. We show qualitative predictions for both RGB and depth images in Figure 7.5.

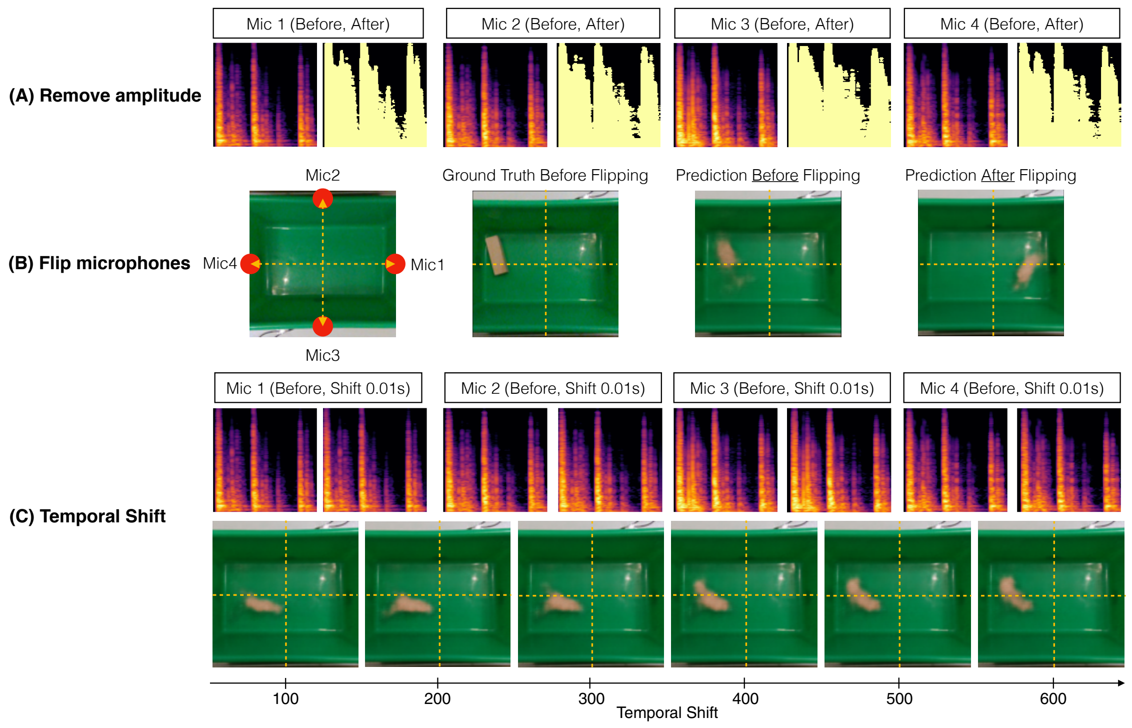


Fig. 7.8: **Visualization of Ablation Studies.** We visualize the impact of different ablations on the model. **A**) By thresholding the spectrograms, we remove the amplitude from the input. **B**) We experimented with flipping the microphones only at testing time. The model's predictions show a corresponding flip as well in the predicted images. **C**) We also experimented with shifting the relative time difference between the microphones, introducing an artificial delay in the microphones only at testing time. A shift in time causes a shift in space in the model's predictions. The corruptions are consistent with a block falling in that location.

7.3.4 Reconstruction with Known Shape

We next analyze how well the model performs when the object shape is known, but the position and orientation is not. We train separate models for each shape of the object independently. Figure 7.7 shows The Boombox is able to reconstruct both the position and orientation of the shapes. The convolutional network obtains the best performance for most shapes on both evaluation metrics. These results highlight the relative difficulty at reconstructing different shapes from sound. By comparing the model performance across various shapes, the model trained on cubes achieves the best performance while the model trained on blocks performs slightly worse. The most difficult shape is the stick.

When the training data combines all shapes, the model should share features between shapes,

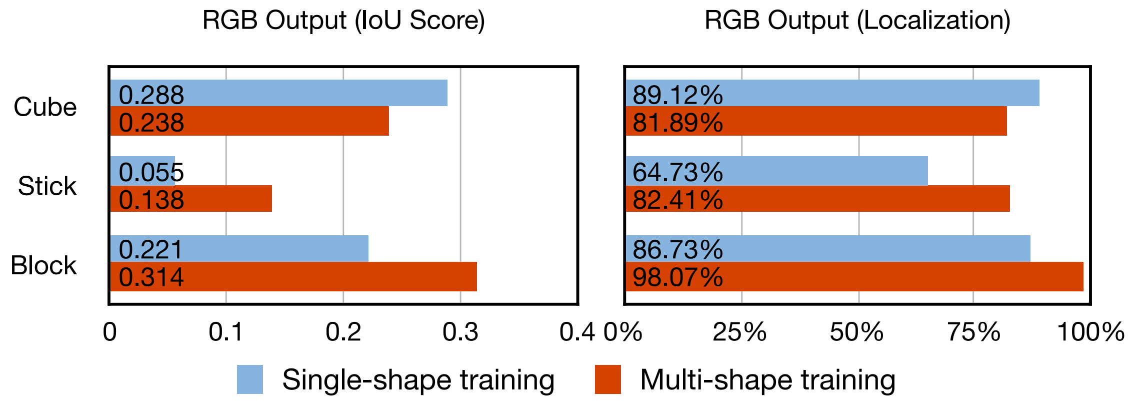


Fig. 7.9: **Shape Transfer.** Performance improves by training with multiple shapes.

thus improving performance. To validate this, we compare performance on the multi-shape versus models trained with a single known shape. Figure 7.9 shows that the performance on the block and stick shapes are improved by a large margin. We notice that the performance of the cube drops due to the confusion between shapes. When the cube confuses with the stick or the block, because of the smaller surface area of these two shapes, the cube performance slightly degrades.

7.3.5 Ablations and Analysis

To better understand what features the model has learned specifically, we perform several ablation studies, shown in Figure 7.8 and Figure 7.10.

Flip microphones. The microphones' layout should matter for our learned model to localize the objects. When we flipped the microphone location, due to the symmetric nature of the hardware setup, the predictions should also be flipped accordingly. To study this, we flipped the input of Mic1 and Mic4 as well as the input of Mic2 and Mic3 in the testing set, shown in Figure 7.8. Our results in Figure 7.8B shows that our model indeed produces a flipped scene. The performance in Figure 7.10 nearly drops to zero, suggesting that the model implicitly learned the relative microphone locations to assist its final prediction.

Remove amplitude. The relative amplitude between microphones can indicate the relative position of the sound source to different microphones. We removed the amplitude information by thresholding the spectrograms, shown in Figure 7.8. We retrained the network due to potential

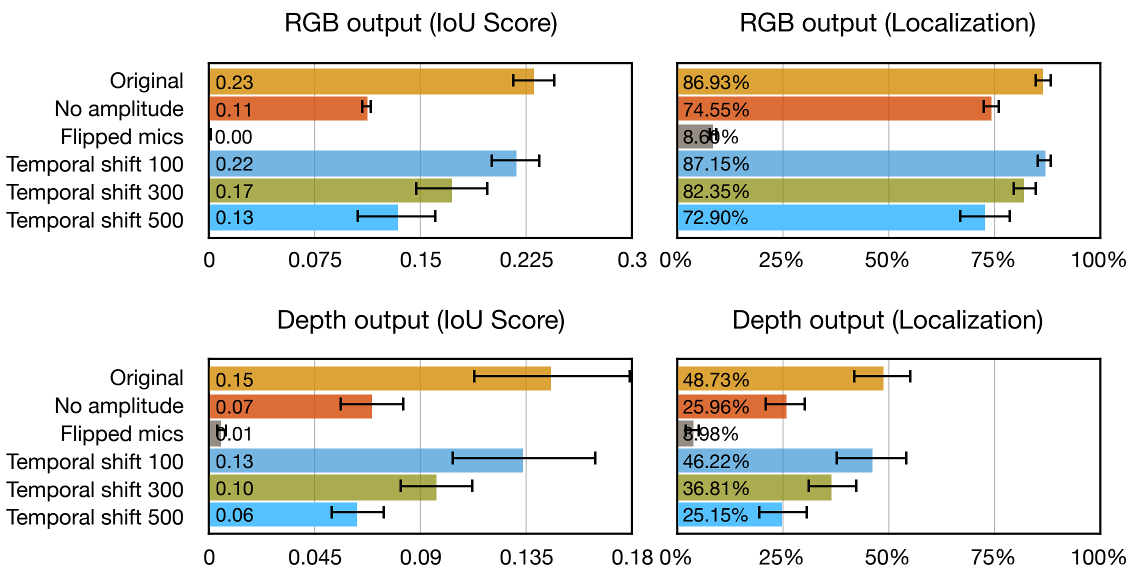


Fig. 7.10: **Quantitative ablation studies.** We experiment with different perturbations to our input data to understand the model predictions.

distribution shift. As expected, even though the time and frequency information are preserved, the model performs much worse (Figure 7.10), suggesting that our model additionally learns to use amplitude for the predictions.

Temporal shift. We are interested to see if our model learns to capture features about the time difference of arrival between microphones. If so, when we shift the audio temporally, the prediction should also shift spatially. We experimented with various degrees of temporal shifts on the original spectrograms. For example, shifting 500 samples corresponds to shifting about 0.01s (500 / 44,000). By shifting the Mic1's spectrogram forward and Mic4's spectrogram backward with zero padding to maintain the same amount of time, and preforming similar operation on Mic2 and Mic3 respectively, we should expect that the predicted object position shifts towards the left-up direction. In Figure 7.8, we can clearly observe this trend as temporal shift increases. Shifting the signal in time decreases the model's performance, demonstrating that the model has picked up on the time difference of arrival.

Feature Visualization: We finally visualize the latent features in between our encoder and decoder network with t-SNE[216], shown in Figure 7.11. We colorize the points based on ground truth position and orientation. The magnitude distance from the center of the image is represented

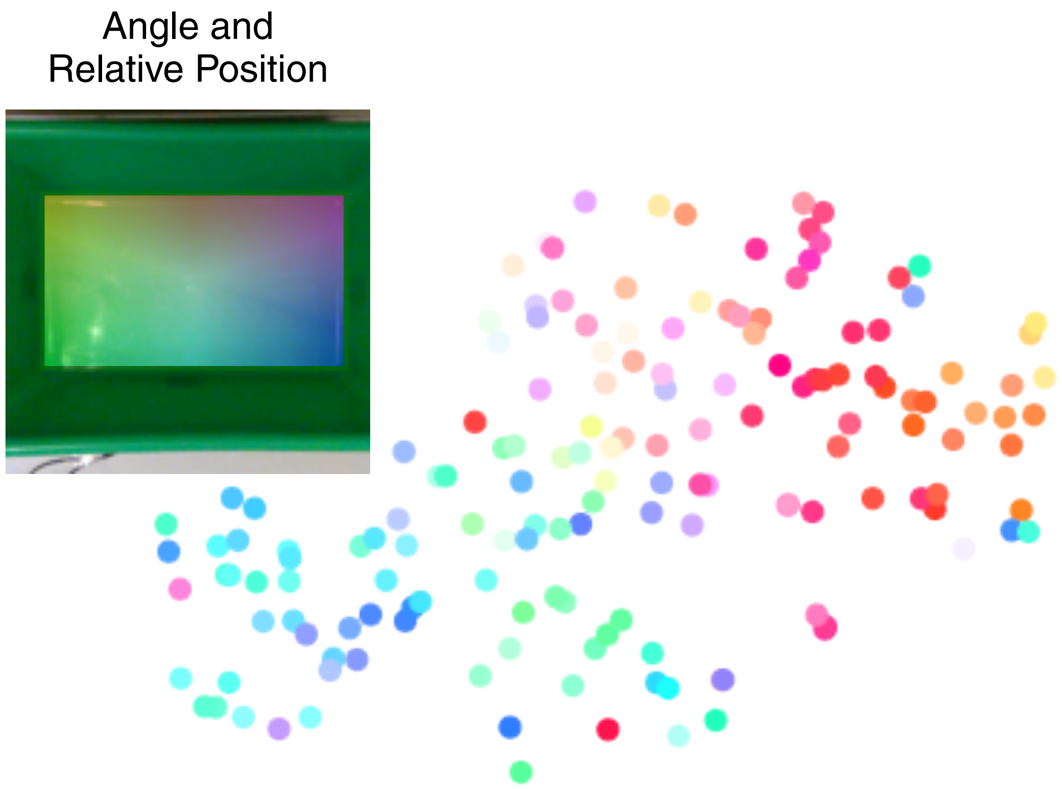


Fig. 7.11: **Low-dimensional embedding.** We visualize the learned features in the encoder with t-SNE.

by saturation, and the angle from the horizontal axis is represented by hue. We find that there is often clear clustering of the embeddings by their position and orientation, showing that the model is robustly discriminating the location of the impact from sound alone. Moreover, the gradual transitions between colors suggest the features are able to smoothly interpolate spatially.

Conclusion

This thesis aims to study the fundamental aspects towards building "generalist robots" through world modeling. I believe that if robots can carry over and update consistent knowledge of the world throughout their learning and functioning time, they do not need to learn each new task and new environment from scratch. Such continual and efficient learning capability is clearly a critical property of future generation of robots.

In particular, this thesis approaches the challenging and abstract world modeling problem by decomposing the world modeling problem into robot self-modeling, robot modeling of other agents, and robot modeling their physical environment. This thesis has demonstrated that this choice of decomposition not only learns accurate predictive models on each of the three component themselves, it also encourages a different perspective on robot learning research by focusing on obtaining disentangled representations of the world as task-related component and task-agnostic component. As shown in this thesis, this key idea have produced powerful results in numerous robotic applications which were considered highly challenging or even impossible before.

Looking ahead, there are still many challenges to build generalist robots. First, building generalist robots require multi-modal robots. The world comes with rich information of physics, dynamics and contexts, which no single sensor can completely capture. Yet, majority of robotics applications rely primarily on vision sensor. However, there are critical information hidden in other modalities for interaction tasks such as sound, tactile feedback and temperature. Such information is not only relevant for robots, as understanding multimodal information is extremely important for humans as well. By building multi-modal robots, we can even better understand human behaviors

[217] and teach robots novel concepts by talking to them [218]. Therefore, one immediate future direction is to learn to distill integrated knowledge from different sensory modalities and build robust multi-modal perception system.

Scaling the world models to complex and constantly changing environments on physical robots remains an important yet relatively underexplored problem. This requires to build computational models that are situational aware. For example, a robot may encounter sudden dynamic payload or broken hardware, or similar changes to its partner's body. To be fully functional and reliable, the robot needs to quickly identify these changes or unforeseen situations and adapt to them with minimal interruptions.

Another major aspect is to study human-robot and robot-robot systems since humans will be surrounded by different kinds of robots in the future. Enabling smooth, safe and robust "integrated intelligence" system requires considerable improvements to the current systems and algorithms. Critical research questions involve better understanding of high-level human behaviors such as abstract goals, intentions, desires, and behaviors from both non-verbal behaviors and high-level language instructions, more efficient and effective communication methods between collaborative robots, and more interpretable robot behaviors for other robots and humans.

By considering robotic research from traditional lab setting to complex open-world scenarios through physical world modeling, I believe that future generalist robots can work seamlessly alongside with humans as an integral part of the society.

References

[1] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[2] R. Kwiatkowski and H. Lipson, "Task-agnostic self-modeling machines," *Science Robotics*, vol. 4, no. 26, eaau9354, 2019.

[3] A. Sanchez-Gonzalez *et al.*, "Graph networks as learnable physics engines for inference and control," in *International Conference on Machine Learning*, PMLR, 2018, pp. 4470–4479.

[4] B. Chen, Y. Hu, L. Li, S. Cummings, and H. Lipson, "Smile like you mean it: Driving animatronic robotic face with learned models," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2739–2746.

[5] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar, "Manipulation for self-identification, and self-identification for better manipulation," *Science Robotics*, vol. 6, no. 54, eabe1321, 2021.

[6] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *arXiv preprint arXiv:1511.05440*, 2015.

[7] N. Kalchbrenner *et al.*, "Video pixel networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1771–1779.

[8] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Advances in neural information processing systems*, vol. 29, 2016.

[9] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, "Learning to generate long-term future via hierarchical prediction," in *international conference on machine learning*, PMLR, 2017, pp. 3560–3569.

[10] C. Vondrick and A. Torralba, "Generating the future with adversarial transformers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1020–1028.

[11] T. Xue, J. Wu, K. Bouman, and B. Freeman, "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks," *Advances in neural information processing systems*, vol. 29, 2016.

[12] J. Yang, A. Kannan, D. Batra, and D. Parikh, “Lr-gan: Layered recursive generative adversarial networks for image generation,” *arXiv preprint arXiv:1703.01560*, 2017.

[13] C. Finn, I. Goodfellow, and S. Levine, “Unsupervised learning for physical interaction through video prediction,” *Advances in neural information processing systems*, vol. 29, 2016.

[14] J. Van Amersfoort, A. Kannan, M. Ranzato, A. Szlam, D. Tran, and S. Chintala, “Transformation based models of video sequences,” *arXiv preprint arXiv:1701.08435*, 2017.

[15] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2786–2793.

[16] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *European conference on computer vision*, Springer, 2016, pp. 69–84.

[17] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.

[18] X. Wang and A. Gupta, “Unsupervised learning of visual representations using videos,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2794–2802.

[19] G. Larsson, M. Maire, and G. Shakhnarovich, “Colorization as a proxy task for visual understanding,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6874–6883.

[20] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy, “Tracking emerges by colorizing videos,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 391–408.

[21] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman, “Learning and using the arrow of time,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8052–8060.

[22] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang, “Self-supervised spatiotemporal learning via video clip order prediction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 334–10 343.

[23] R. D. Hjelm *et al.*, “Learning deep representations by mutual information estimation and maximization,” *arXiv preprint arXiv:1808.06670*, 2018.

[24] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, “Unsupervised representation learning by sorting sequences,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 667–676.

[25] I. Misra, C. L. Zitnick, and M. Hebert, “Shuffle and learn: Unsupervised learning using temporal order verification,” in *European Conference on Computer Vision*, Springer, 2016, pp. 527–544.

[26] B. Fernando, H. Bilen, E. Gavves, and S. Gould, “Self-supervised video representation learning with odd-one-out networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3636–3645.

[27] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[28] P. Agrawal, J. Carreira, and J. Malik, “Learning to see by moving,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 37–45.

[29] D. Jayaraman and K. Grauman, “Learning image representations tied to ego-motion,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1413–1421.

[30] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, “View synthesis by appearance flow,” in *European conference on computer vision*, Springer, 2016, pp. 286–301.

[31] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[32] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki, “Self-supervised learning of motion capture,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5236–5246.

[33] X. Wang, A. Jabri, and A. A. Efros, “Learning correspondence from the cycle-consistency of time,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2566–2576.

[34] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, “Temporal cycle-consistency learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1801–1810.

[35] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang, “Joint-task self-supervised learning for temporal correspondence,” *arXiv preprint arXiv:1909.11895*, 2019.

[36] Z. Lai and W. Xie, “Self-supervised learning for video correspondence flow,” *arXiv preprint arXiv:1905.00875*, 2019.

[37] C. Sun, F. Baradel, K. Murphy, and C. Schmid, “Contrastive bidirectional transformer for temporal representation learning,” *arXiv preprint arXiv:1906.05743*, 2019.

[38] I. Rahwan *et al.*, “Machine behaviour,” *Nature*, vol. 568, no. 7753, pp. 477–486, 2019.

[39] C. L. Baker, J. Jara-Ettinger, R. Saxe, and J. B. Tenenbaum, “Rational quantitative attribution of beliefs, desires and percepts in human mentalizing,” *Nature Human Behaviour*, vol. 1, no. 4, pp. 1–10, 2017.

[40] B. Scassellati, “Theory of mind for a humanoid robot,” *Autonomous Robots*, vol. 12, no. 1, pp. 13–24, 2002.

[41] W. G. Kennedy, M. D. Bugajska, A. M. Harrison, and J. G. Trafton, ““like-me” simulation as an effective and cognitively plausible basis for social robotics,” *International Journal of Social Robotics*, vol. 1, no. 2, pp. 181–194, 2009.

[42] J. Gray and C. Breazeal, “Manipulating mental states through physical action,” *International Journal of Social Robotics*, vol. 6, no. 3, pp. 315–327, 2014.

[43] R. Yokoya, T. Ogata, J. Tani, K. Komatani, and H. G. Okuno, “Discovery of other individuals by projecting a self-model through imitation,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 1009–1014.

[44] E. S. Spelke and K. D. Kinzler, “Core knowledge,” *Developmental science*, vol. 10, no. 1, pp. 89–96, 2007.

[45] Y. Takahashi, Y. Tamura, M. Asada, and M. Negrello, “Emulation and behavior understanding through shared values,” *Robotics and Autonomous Systems*, vol. 58, no. 7, pp. 855–865, 2010.

[46] J Kiley Hamlin, T. Ullman, J. Tenenbaum, N. Goodman, and C. Baker, “The mentalistic basis of core social cognition: Experiments in preverbal infants and a computational model,” *Developmental science*, vol. 16, no. 2, pp. 209–226, 2013.

[47] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, “Modeling others using oneself in multiagent reinforcement learning,” in *International conference on machine learning*, PMLR, 2018, pp. 4257–4266.

[48] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. A. Eslami, and M. Botvinick, “Machine theory of mind,” in *International conference on machine learning*, PMLR, 2018, pp. 4218–4227.

[49] K.-J. Kim and H. Lipson, "Towards a simple robotic theory of mind," in *Proceedings of the 9th workshop on performance metrics for intelligent systems*, 2009, pp. 131–138.

[50] M. Ramirez and H. Geffner, "Goal recognition over pomdps: Inferring the intention of a pomdp agent," in *Twenty-second international joint conference on artificial intelligence*, 2011.

[51] O. Evans, A. Stuhlmüller, and N. Goodman, "Learning the preferences of ignorant, inconsistent agents," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[52] A. F. Winfield, "Experiments in artificial theory of mind: From safety to story-telling," *Frontiers in Robotics and AI*, vol. 5, p. 75, 2018.

[53] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[54] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 3389–3396.

[55] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 3357–3364.

[56] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[57] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

[58] D. Kalashnikov *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[59] P. Mirowski *et al.*, "Learning to navigate in cities without a map," in *Advances in Neural Information Processing Systems*, 2018, pp. 2419–2430.

[60] B. Chen, S. Song, H. Lipson, and C. Vondrick, "Visual hide and seek," in *Artificial Life Conference Proceedings*, MIT Press, 2020, pp. 645–655.

[61] Y. Liang, B. Chen, and S. Song, "Sscnav: Confidence-aware semantic scene completion for visual semantic navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 194–13 200.

[62] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," *arXiv preprint arXiv:1710.03748*, 2017.

[63] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[64] S. Liu, G. Lever, J. Merel, S. Tunyasuvunakool, N. Heess, and T. Graepel, "Emergent coordination through competition," *arXiv preprint arXiv:1902.07151*, 2019.

[65] M. Jaderberg *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019. eprint: `https://science.sciencemag.org/content/364/6443/859.full.pdf`.

[66] B. Chen, Y. Hu, R. Kwiatkowski, S. Song, and H. Lipson, "Visual perspective taking for opponent behavior modeling," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 678–13 685.

[67] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[68] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun, "MINOS: Multimodal indoor simulator for navigation in complex environments," *arXiv:1712.03931*, 2017.

[69] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2054–2063.

[70] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.

[71] Manolis Savva* *et al.*, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[72] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[73] A. Giusti *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2015.

[74] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of memory, active perception, and action in minecraft," *arXiv preprint arXiv:1605.09128*, 2016.

[75] D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, and R. E. Schapire, "Exploratory gradient boosting for reinforcement learning in complex domains," *arXiv preprint arXiv:1603.04119*, 2016.

[76] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N Siddharth, and P. H. Torr, "Playing doom with slam-augmented deep reinforcement learning," *arXiv preprint arXiv:1612.00380*, 2016.

[77] S. Daftry, J. A. Bagnell, and M. Hebert, "Learning transferable policies for monocular reactive mav control," in *International Symposium on Experimental Robotics*, Springer, 2016, pp. 3–11.

[78] P. Mirowski *et al.*, "Learning to navigate in complex environments," 2017.

[79] S. Brahmbhatt and J. Hays, "Deepnav: Learning to navigate large cities," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5193–5202.

[80] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 2371–2378.

[81] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2616–2625.

[82] Y. Zhu *et al.*, "Visual semantic planning using deep successor representations," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 483–492.

[83] S. Gupta, D. Fouhey, S. Levine, and J. Malik, "Unifying map and landmark based representations for visual navigation," *arXiv preprint arXiv:1712.08125*, 2017.

[84] U. Jain *et al.*, "Two body problem: Collaborative visual task completion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6689–6699.

[85] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[86] L. Bu, R. Babu, B. De Schutter, *et al.*, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[87] K. Tuyls and G. Weiss, “Multiagent learning: Basics, challenges, and prospects,” *Ai Magazine*, vol. 33, no. 3, pp. 41–41, 2012.

[88] Y. Shoham, R. Powers, and T. Grenager, “If multi-agent learning is the answer, what is the question?” *Artificial Intelligence*, vol. 171, no. 7, pp. 365–377, 2007.

[89] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, “Is multiagent deep reinforcement learning the answer or the question? a brief survey,” *arXiv preprint arXiv:1810.05587*, 2018.

[90] D. Gandhi, A. Gupta, and L. Pinto, “Swoosh! rattle! thump!–actions that sound,” *arXiv preprint arXiv:2007.01851*, 2020.

[91] C. Matl, Y. Narang, D. Fox, R. Bajcsy, and F. Ramos, “Stressd: Sim-to-real from sound for stochastic dynamics,” *arXiv preprint arXiv:2011.03136*, 2020.

[92] F. Bu and C.-M. Huang, “Object permanence through audio-visual representations,” *arXiv preprint arXiv:2010.09948*, 2020.

[93] G. G. Gallup Jr, “Self-awareness and the emergence of mind in primates,” *American Journal of Primatology*, vol. 2, no. 3, pp. 237–248, 1982.

[94] P. Rochat, “Five levels of self-awareness as they unfold early in life,” *Consciousness and cognition*, vol. 12, no. 4, pp. 717–731, 2003.

[95] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, pp. 2149–2154.

[96] E. Todorov, T. Erez, and Y. 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.

[97] F. Faure *et al.*, “Sofa: A multi-model framework for interactive physical simulation,” in *Soft tissue biomechanical modeling for computer assisted surgery*, Springer, 2012, pp. 283–321.

[98] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” 2016.

[99] J. Lee *et al.*, “Dart: Dynamic animation and robotics toolkit,” *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.

[100] R. Tedrake and the Drake Development Team, *Drake: Model-based design and verification for robotics*, 2019.

[101] Z. Faraj *et al.*, “Facially expressive humanoid robotic face,” *HardwareX*, vol. 9, e00117, 2021.

[102] R. Plutchik, “Emotions: A general psychoevolutionary theory,” *Approaches to emotion*, vol. 1984, pp. 197–219, 1984.

[103] N. Reissland, “Neonatal imitation in the first hour of life: Observations in rural nepal.,” *Developmental Psychology*, vol. 24, no. 4, p. 464, 1988.

[104] A. N. Meltzoff and M. K. Moore, “Imitation in newborn infants: Exploring the range of gestures imitated and the underlying mechanisms.,” *Developmental psychology*, vol. 25, no. 6, p. 954, 1989.

[105] R. Van Baaren, L. Janssen, T. L. Chartrand, and A. Dijksterhuis, “Where is the love? the social aspects of mimicry,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1528, pp. 2381–2389, 2009.

[106] J. Piaget, *Play, dreams and imitation in childhood*. Routledge, 2013, vol. 25.

[107] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and autonomous systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[108] M. Blow, K. Dautenhahn, A. Appleby, C. L. Nehaniv, and D. Lee, “The art of designing robot faces: Dimensions for human-robot interaction,” in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 331–332.

[109] C. Breazeal, A. Takanishi, and T. Kobayashi, *Social robots that interact with people.* 2008.

[110] S. Saunderson and G. Nejat, “How robots influence humans: A survey of nonverbal communication in social human–robot interaction,” *International Journal of Social Robotics*, vol. 11, no. 4, pp. 575–608, 2019.

[111] J.-H. Oh, D. Hanson, W.-S. Kim, Y. Han, J.-Y. Kim, and I.-W. Park, “Design of android type humanoid robot albert hubo,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 1428–1433.

[112] T. Hashimoto, S. Hitramatsu, T. Tsuji, and H. Kobayashi, “Development of the face robot saya for rich facial expressions,” in *2006 SICE-ICASE International Joint Conference*, IEEE, 2006, pp. 5423–5428.

[113] T. Hashimoto, S. Hiramatsu, and H. Kobayashi, “Dynamic display of facial expressions on the face robot made by using a life mask,” in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2008, pp. 521–526.

[114] H. S. Ahn, D.-W. Lee, D. Choi, D.-Y. Lee, M. Hur, and H. Lee, “Designing of android head system by applying facial muscle mechanism of humans,” in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, IEEE, 2012, pp. 799–804.

[115] D. Loza, S. Marcos Pablos, E. Zalama Casanova, J. Gómez García-Bermejo, and J. L. González, “Application of the facs in the design and construction of a mechatronic head with realistic appearance,” 2013.

[116] C.-Y. Lin, C.-C. Huang, and L.-C. Cheng, “An expressional simplified mechanism in anthropomorphic face robot design,” *Robotica*, vol. 34, no. 3, p. 652, 2016.

[117] W. T. Asheber, C.-Y. Lin, and S. H. Yen, “Humanoid head face mechanism with expandable facial expressions,” *International Journal of Advanced Robotic Systems*, vol. 13, no. 1, p. 29, 2016.

[118] N. Liu and F. Ren, “Emotion classification using a cnn_lstm-based model for smooth emotional synchronization of the humanoid robot ren-xin,” *PloS one*, vol. 14, no. 5, e0215216, 2019.

[119] H.-J. Hyung, H. U. Yoon, D. Choi, D.-Y. Lee, and D.-W. Lee, “Optimizing android facial expressions using genetic algorithms,” *Applied Sciences*, vol. 9, no. 16, p. 3379, 2019.

[120] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, “Openpose: Realtime multi-person 2d pose estimation using part affinity fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[121] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand keypoint detection in single images using multiview bootstrapping,” in *CVPR*, 2017.

[122] R. Liu *et al.*, “An intriguing failing of convolutional neural networks and the coordconv solution,” in *Advances in Neural Information Processing Systems*, 2018, pp. 9605–9616.

[123] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, “Everybody dance now,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5933–5942.

[124] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[125] B. Chen, C. Vondrick, and H. Lipson, “Visual behavior modelling for robotic theory of mind,” *Scientific Reports*, vol. 11, no. 1, pp. 1–14, 2021.

[126] A. Dosovitskiy *et al.*, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[127] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[128] M. Pantic, M. Valstar, R. Rademaker, and L. Maat, "Web-based database for facial expression analysis," in *2005 IEEE international conference on multimedia and Expo*, IEEE, 2005, 5–pp.

[129] B. Chen, R. Kwiatkowski, C. Vondrick, and H. Lipson, "Full-body visual self-modeling of robot morphologies," *arXiv preprint arXiv:2111.06389*, 2021.

[130] H. E. Gardner, *Frames of mind: The theory of multiple intelligences*. Hachette Uk, 2011.

[131] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[132] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[133] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[134] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[135] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[136] J. B. Watson, "Psychology as the behaviorist views it.," *Psychological review*, vol. 20, no. 2, p. 158, 1913.

[137] H. M. Wellman and D. Liu, "Scaling of theory-of-mind tasks," *Child development*, vol. 75, no. 2, pp. 523–541, 2004.

[138] X. P. Ding, H. M. Wellman, Y. Wang, G. Fu, and K. Lee, "Theory-of-mind training causes honest young children to lie," *Psychological Science*, vol. 26, no. 11, pp. 1812–1821, 2015.

[139] H. Wimmer and J. Perner, "Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception," *Cognition*, vol. 13, no. 1, pp. 103–128, 1983.

[140] S. Baron-Cohen, A. M. Leslie, and U. Frith, "Does the autistic child have a "theory of mind"?" *Cognition*, vol. 21, no. 1, pp. 37–46, 1985.

[141] A. M. Leslie and U. Frith, "Autistic children's understanding of seeing, knowing and believing," *British Journal of Developmental Psychology*, vol. 6, no. 4, pp. 315–324, 1988.

[142] A. N. Meltzoff, "Understanding the intentions of others: Re-enactment of intended acts by 18-month-old children.," *Developmental psychology*, vol. 31, no. 5, p. 838, 1995.

[143] B. Baker *et al.*, "Emergent tool use from multi-agent autocurricula," in *International Conference on Learning Representations*, 2020.

[144] B. Chen, S. Song, H. Lipson, and C. Vondrick, "Visual hide and seek," *arXiv preprint arXiv:1910.07882*, 2019.

[145] H. Shevlin and M. Halina, "Apply rich psychological terms in ai with care," *Nature Machine Intelligence*, vol. 1, no. 4, pp. 165–167, 2019.

[146] J. Piaget, *Child's Conception of Space: Selected Works vol 4*. Routledge, 2013, vol. 4.

[147] J. H. Flavell, "The development of knowledge about visual perception.," in *Nebraska symposium on motivation*, University of Nebraska Press, 1977.

[148] J. H. Flavell, E. F. Flavell, F. L. Green, and S. A. Wilcox, "Young children's knowledge about visual perception: Effect of observer's distance from target on perceptual clarity of target.," *Developmental Psychology*, vol. 16, no. 1, p. 10, 1980.

[149] N. Newcombe and J. Huttenlocher, "Children's early ability to solve perspective-taking problems.," *Developmental psychology*, vol. 28, no. 4, p. 635, 1992.

[150] H. Moll and A. N. Meltzoff, "How does it look? level 2 perspective-taking at 36 months of age," *Child development*, vol. 82, no. 2, pp. 661–673, 2011.

[151] A. Pearson, D. Ropar, A. F. d. Hamilton, *et al.*, "A review of visual perspective taking in autism spectrum disorder," *Frontiers in human neuroscience*, vol. 7, p. 652, 2013.

[152] J. G. Trafton *et al.*, "Children and robots learning to play hide and seek," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 242–249.

[153] S. V. Albrecht and P. Stone, "Autonomous agents modelling other agents: A comprehensive survey and open problems," *Artificial Intelligence*, vol. 258, pp. 66–95, 2018.

[154] L. Weihs *et al.*, "Artificial agents learn flexible visual representations by playing a hiding game," *arXiv preprint arXiv:1912.08195*, 2019.

[155] A. Labash, J. Aru, T. Matiisen, A. Tampuu, and R. Vicente, “Perspective taking in deep reinforcement learning agents,” *Frontiers in Computational Neuroscience*, vol. 14, 2020.

[156] F. Leibfried, N. Kushman, and K. Hofmann, “A deep learning approach for joint video frame and reward prediction in atari games,” *arXiv preprint arXiv:1611.07078*, 2016.

[157] S. Racanière *et al.*, “Imagination-augmented agents for deep reinforcement learning,” in *Advances in neural information processing systems*, 2017, pp. 5690–5701.

[158] S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed, “Recurrent environment simulators,” *arXiv preprint arXiv:1704.02254*, 2017.

[159] D. Hafner *et al.*, “Learning latent dynamics for planning from pixels,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 2555–2565.

[160] J. Wu *et al.*, “Spatial action maps for mobile manipulation,” *arXiv preprint arXiv:2004.09141*, 2020.

[161] E. Olson, “Apriltag: A robust and flexible visual fiducial system,” in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3400–3407.

[162] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[163] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, “Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation,” in *Conference on Robot Learning*, PMLR, 2017, pp. 185–194.

[164] X. Chen, A. Ghadirzadeh, J. Folkesson, M. Björkman, and P. Jensfelt, “Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3110–3116.

[165] P. Shah, M. Fiser, A. Faust, J. C. Kew, and D. Hakkani-Tur, “Follownet: Robot navigation by following natural language directions with deep reinforcement learning,” *arXiv preprint arXiv:1805.06150*, 2018.

[166] T. Bruls, H. Porav, L. Kunze, and P. Newman, “The right (angled) perspective: Improving the understanding of road scenes using boosted inverse perspective mapping,” in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 302–309.

[167] C. Lynch *et al.*, “Learning latent plans from play,” in *Conference on Robot Learning*, 2020, pp. 1113–1132.

[168] A. Byravan, F. Lceb, F. Meier, and D. Fox, "Se3-pose-nets: Structured deep dynamics models for visuomotor control," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

[169] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[170] L. Buesing *et al.*, "Learning and querying fast generative models for reinforcement learning," *arXiv preprint arXiv:1802.03006*, 2018.

[171] B. Chen, K. Huang, S. Raghupathi, I. Chandratreya, Q. Du, and H. Lipson, "Discovering state variables hidden in experimental data," *arXiv preprint arXiv:2112.10755*, 2021.

[172] P. W. Anderson, "More is different," *Science*, vol. 177, no. 4047, pp. 393–396, 1972.

[173] J. M. T. Thompson and H. B. Stewart, *Nonlinear dynamics and chaos*. John Wiley & Sons, 2002.

[174] M. W. Hirsch, S. Smale, and R. L. Devaney, *Differential equations, dynamical systems, and an introduction to chaos*. Academic press, 2012.

[175] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[176] J. Evans and A. Rzhetsky, "Machine science," *Science*, vol. 329, no. 5990, pp. 399–400, 2010.

[177] S. Fortunato *et al.*, "Science of science," *Science*, vol. 359, no. 6379, 2018.

[178] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 9943–9948, 2007.

[179] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *science*, vol. 324, no. 5923, pp. 81–85, 2009.

[180] R. D. King, S. H. Muggleton, A. Srinivasan, and M. Sternberg, "Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming," *Proceedings of the National Academy of Sciences*, vol. 93, no. 1, pp. 438–442, 1996.

[181] D. Waltz and B. G. Buchanan, "Automating science," *Science*, vol. 324, no. 5923, pp. 43–44, 2009.

[182] R. D. King *et al.*, "The robot scientist adam," *Computer*, vol. 42, no. 8, pp. 46–54, 2009.

[183] J. P. Crutchfield and B. McNamara, "Equations of motion from a data series," *Complex systems*, vol. 1, pp. 417–452, 1987.

[184] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidid, O. Runborg, C. Theodoropoulos, *et al.*, "Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis," *Communications in Mathematical Sciences*, vol. 1, no. 4, pp. 715–762, 2003.

[185] C. Yao and E. M. Bollt, "Modeling and nonlinear parameter estimation with kronecker product representation for coupled oscillators and spatiotemporal systems," *Physica D: Nonlinear Phenomena*, vol. 227, no. 1, pp. 78–99, 2007.

[186] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of fluid mechanics*, vol. 641, pp. 115–127, 2009.

[187] M. D. Schmidt *et al.*, "Automated refinement and inference of analytical models for metabolic networks," *Physical biology*, vol. 8, no. 5, p. 055 011, 2011.

[188] G. Sugihara *et al.*, "Detecting causality in complex ecosystems," *science*, vol. 338, no. 6106, pp. 496–500, 2012.

[189] H. Ye *et al.*, "Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling," *Proceedings of the National Academy of Sciences*, vol. 112, no. 13, E1569–E1576, 2015.

[190] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models," *Nature communications*, vol. 6, no. 1, pp. 1–8, 2015.

[191] P. Benner, S. Gugercin, and K. Willcox, "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM review*, vol. 57, no. 4, pp. 483–531, 2015.

[192] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[193] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances*, vol. 3, no. 4, e1602614, 2017.

[194] S.-M. Udrescu and M. Tegmark, "Ai feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, eaay2631, 2020.

[195] D. Mrowca *et al.*, "Flexible neural representation for physics prediction," *arXiv preprint arXiv:1806.08047*, 2018.

[196] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22 445–22 451, 2019.

[197] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural networks*, vol. 2, no. 1, pp. 53–58, 1989.

[198] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy," *Advances in neural information processing systems*, vol. 6, pp. 3–10, 1994.

[199] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International conference on artificial neural networks*, Springer, 2011, pp. 52–59.

[200] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[201] F. Camastra and A. Staiano, "Intrinsic dimension estimation: Advances and open problems," *Information Sciences*, vol. 328, pp. 26–41, 2016.

[202] P. Campadelli, E. Casiraghi, C. Ceruti, and A. Rozza, "Intrinsic dimension estimation: Relevant techniques and a benchmark framework," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[203] E. Levina and P. J. Bickel, "Maximum likelihood estimation of intrinsic dimension," in *Advances in neural information processing systems*, 2005, pp. 777–784.

[204] A. Rozza, G. Lombardi, C. Ceruti, E. Casiraghi, and P. Campadelli, "Novel high intrinsic dimensionality estimators," *Machine learning*, vol. 89, no. 1, pp. 37–65, 2012.

[205] C. Ceruti, S. Bassis, A. Rozza, G. Lombardi, E. Casiraghi, and P. Campadelli, "Danco: An intrinsic dimensionality estimator exploiting angle and norm concentration," *Pattern recognition*, vol. 47, no. 8, pp. 2569–2581, 2014.

[206] M. Hein and J.-Y. Audibert, "Intrinsic dimensionality estimation of submanifolds in rd," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 289–296.

[207] P. Grassberger and I. Procaccia, "Measuring the strangeness of strange attractors," in *The theory of chaotic attractors*, Springer, 2004, pp. 170–189.

[208] B. Chen, M. Chiquier, H. Lipson, and C. Vondrick, "The boombox: Visual reconstruction from acoustic vibrations," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, 2022, pp. 1067–1077.

[209] J. W. Strutt, "On our perception of sound direction," *Philosophical Magazine*, vol. 13, no. 74, pp. 214–32, 1907.

[210] J.-Y. Zhu, T. Park, P. Isola, and A. 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.

[211] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[212] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.

[213] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[214] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 4796–4803.

[215] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE transactions on acoustics, speech, and signal processing*, vol. 24, no. 4, pp. 320–327, 1976.

[216] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[217] D. Epstein, B. Chen, and C. Vondrick, "Oops! predicting unintentional action in video," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 919–929.

[218] B. Chen, Y. Li, S. Raghupathi, and H. Lipson, "Beyond categorical label representations for image classification," *arXiv preprint arXiv:2104.02226*, 2021.

[219] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

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

[221] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. arXiv: `1409.1556 [cs.CV]`.

[222] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: `1512.03385 [cs.CV]`.

[223] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, 2016. arXiv: `1608.06993 [cs.CV]`.

[224] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[225] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[226] A. Kuznetsova *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv:1811.00982*, 2018.

[227] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," *Lecture Notes in Computer Science*, 387–402, 2013.

[228] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv e-prints*, arXiv:1312.6199, arXiv:1312.6199, 2013. arXiv: `1312.6199 [cs.CV]`.

[229] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[230] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[231] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial intelligence review*, vol. 18, no. 2, pp. 77–95, 2002.

[232] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018.

[233] T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, J. Kautz, and B. Catanzaro, "Few-shot video-to-video synthesis," *arXiv preprint arXiv:1910.12713*, 2019.

[234] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[235] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[236] C. Finn, P. Abbeel, and S. Levine, *Model-agnostic meta-learning for fast adaptation of deep networks*, 2017. arXiv: `1703.03400 [cs.LG]`.

[237] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *ICLR*, 2017.

[238] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-learning with memory-augmented neural networks,” in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML’16, New York, NY, USA: JMLR.org, 2016, 1842–1850.

[239] ——, *One-shot learning with memory-augmented neural networks*, 2016. arXiv: `1605.06065 [cs.LG]`.

[240] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, 2014. arXiv: `1412.6572 [stat.ML]`.

[241] A. Kurakin, I. Goodfellow, and S. Bengio, *Adversarial examples in the physical world*, 2016. arXiv: `1607.02533 [cs.CV]`.

[242] C. Szegedy *et al.*, *Intriguing properties of neural networks*, 2013. arXiv: `1312.6199 [cs.CV]`.

[243] U. Shaham, Y. Yamada, and S. Negahban, “Understanding adversarial training: Increasing local stability of supervised models through robust optimization,” *Neurocomputing*, vol. 307, pp. 195 –204, 2018.

[244] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, 2017. arXiv: `1706.06083 [stat.ML]`.

[245] P. Samangouei, M. Kabkab, and R. Chellappa, *Defense-gan: Protecting classifiers against adversarial attacks using generative models*, 2018. arXiv: `1805.06605 [cs.CV]`.

[246] D. Meng and H. Chen, *Magnet: A two-pronged defense against adversarial examples*, 2017. arXiv: `1705.09064 [cs.CR]`.

[247] J. Lu, T. Issaranon, and D. Forsyth, *Safetynet: Detecting and rejecting adversarial examples robustly*, 2017. arXiv: `1704.00103 [cs.CV]`.

[248] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, *On detecting adversarial perturbations*, 2017. arXiv: `1702.04267 [stat.ML]`.

[249] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, *Pixeldefend: Leveraging generative models to understand and defend against adversarial examples*, 2017. arXiv: `1710.10766 [cs.LG]`.

[250] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, *Defense against adversarial attacks using high-level representation guided denoiser*, 2017. arXiv: `1712.02976 [cs.CV]`.

[251] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, *Blocking transferability of adversarial examples in black-box learning systems*, 2017. arXiv: `1703.04318 [cs.LG]`.

[252] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, *Distillation as a defense to adversarial perturbations against deep neural networks*, 2015. arXiv: `1511.04508 [cs.CR]`.

[253] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. arXiv: `1503.02531 [stat.ML]`.

[254] N. Papernot and P. McDaniel, *Extending defensive distillation*, 2017. arXiv: `1705.05264 [cs.LG]`.

[255] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017.

[256] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[257] M. Goibert and E. Dohmatob, "Adversarial robustness via label-smoothing," 2020.

[258] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv e-prints*, arXiv:1412.6572, arXiv:1412.6572, 2014. arXiv: `1412.6572 [stat.ML]`.

[259] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, and M. Cobos, *Dcase 2019: Cnn depth analysis with different channel inputs for acoustic scene classification*, 2019. arXiv: `1906.04591 [cs.SD]`.

[260] J.-X. Zhang, Z.-H. Ling, L.-J. Liu, Y. Jiang, and L.-R. Dai, "Sequence-to-sequence acoustic modeling for voice conversion," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 3, 631–644, 2019.

[261] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[262] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[263] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[264] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[265] A. Sorokin and D. Forsyth, "Utility data annotation with amazon mechanical turk," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2008, pp. 1–8.

[266] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[267] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

Appendix A: Full-Body Visual Self-Modeling of Robot Morphologies

Background on Computational Representation of Visual Self-Model

In this supplementary document, we will first introduce the background of Signed Distance Functions (SDF) as a method to represent 3D shapes. We will then describe how we can train a neural network to represent the 3D shape with SDF representation as a continuous and generative model. Finally, we will describe our key design decisions to enable learning both the robot morphology and kinematics such that a consistent visual self-model can acquire generalizable robot body and kinematics representations for various future tasks.

Signed Distance Function as 3D Shape Representation

Signed Distance Function (SDF) has been commonly used to represent the geometry of 3D shapes. In our case, the object we would like to represent is the robot body. Specifically, SDF maps a given 3D coordinate $\boldsymbol{X} = (x, y, z)$ to a signed value. The magnitude of this value reflects the closest distance from this 3D point to the surface of the robot body, and the sign of this value indicates whether the point is inside or outside the surface boundary. Here, a negative sign means the point is located inside the surface boundary and a positive sign means the point is located outside the surface boundary. Therefore, when SDF equals to zero, the given 3D point is on the surface of the robot body. In other words, the robot morphology can be represented as the zero-level iso-surface or zero-level set of the SDF. Mathematically, the function can be defined as follows:

$$SDF : \mathbb{R}^3 \mapsto \mathbb{R}, \boldsymbol{X} \to SDF(\boldsymbol{X})$$

The surface of the robot morphology is thus represented by the iso-surface $SDF(\cdot) = 0$. By querying the SDF values on 3D points within the region of interests, we can recover the 3D surface boundary of the robot body. We can further reconstruct the 3D mesh of the robot morphology with Marching Cubes algorithm, for example, to visualize the entire shape. Depending on the sampling resolution of the input coordinates $\boldsymbol{X}$, we can also control the resolution of the reconstructed mesh. Therefore, if we choose to represent the coordinate values as continuous functions, the SDF also provides a continuous representation of the robot morphology.

Differentiable SDF with Deep Neural Networks

Once we have created the SDF representation of the target robot morphology, inspired by the recent work of using neural networks to learn a differentiable function mapping from 3D coordinates to the SDF values, we can use a neural network to learn such mapping from pairs of data created as above. Formally, we aim to learn the parameters θ of a function f_θ such that $f_\theta(\boldsymbol{X})$ closely approximates $SDF(\boldsymbol{X})$. The network architecture are MLP layers following the previous work. To train the network end to end, there are mainly two methods. The first method is to perform regression with respect to the ground-truth SDF values. Such method can be trained with distance-based loss function such as L1 loss. The second method is to treat the SDF prediction problem as an Eikonal boundary problem to constrain the norm of the spatial gradients, the zero-level SDF values and the normals of the on-surface points, as well as SDF values of the off-surface points. One recent advancement of such neural network design is to replace the ReLU activation functions between the MLP layers as periodic activation functions for finer details in the final predictions. We therefore utilize sine activation functions and follow the second method to train the network. In our experiments, with the same activation functions, the performances of both methods are similar. Overall, the learned SDF representation is a continuous generative model for 3D objects.

Query-Based Visual Self-Model of Full-Body Robot Morphologies

We can only represent a single robot morphology by following the above steps. However, a visual self-model of the robot should represent the full-body morphology with kinematics awareness, since kinematic awareness conditioned on robot actions is central for leveraging the learned self-model for various motion planning and control tasks. More importantly, we expect the visual self-model to generalize strongly on unseen motor states of the robot beyond the training set. As discussed in our main texts, another critical consideration when constructing our visual self-model is to enable query-based inputs. Query-based inputs make it possible for us to query the pose-conditioned body occupancy only around the 3D regions that we care about for the future tasks at hand. Based on these two considerations, implicit neural representations, such as the above differentiable SDF formulations, serve as great foundations for designing our final visual self-model. We will now describe other major decisions to realize our visual self-model.

First, we use two branches of neural networks to process the two query-based inputs: a spatial coordinate $\boldsymbol{X} = (x, y, z)$ and a set of motor angles $\boldsymbol{A}$. Such two branches can process the two input components separately so that the spatial coordinate queries and the robot motor states can be processed independently before they are fused together for the final SDF value predictions. Following the functionalities of these two branches, we name the first branch as coordinate network C and the second branch as kinematic network K. This design decision allows the network to learn rich kinematic information in the kinematic branch without confusing with the information from the coordinate network branch, hence producing useful features for future motion planning and control tasks as shown in the Results section. After concatenating the features from the coordinate network and the kinematic network, we send the final feature vector to another few layers of MLPs F to predict the corresponding SDF values. We discussed the final loss function to train our network in the Materials and Methods section. The formulation of our final network can be expressed as:

$$F \circ (K, C) : (\mathbb{R}^3, \mathbb{R}^N) \mapsto \mathbb{R}, (\boldsymbol{X}, \boldsymbol{A}) \to F(C(\boldsymbol{X}), K(\boldsymbol{A}))$$

where N is the dimension of the motor state vector.

By formulating the query-based visual self-model with the above implicit neural representations, the learned model can generalize on unseen motor angles to infer the full-body robot morphologies. As thoroughly described in the 3D Self-Aware Motion Planning and Damage Identification and Recovery sections, we can further design gradient-based optimization algorithms on top of our acquired visual self-model for fast motion planning and control as well as damage assessment tasks.

Appendix B: Visual Behavior Modelling for Robotic Theory of Mind

B.1 Experimental setup

We performed all of our experiments on the hardware setup shown in Fig. **??**. In order to automatically generate and display the target food and other foods on the background, we used a flat panel display. This display has a screen size of 57.5 inches by 33 inches and 2160P (4K) resolution. We then connected the display to a computer to send auto-generated background maps.

The Observer machine is an Alienware-15 laptop connected to a top-view camera to acquire the scene videos as seen from above. The camera used was an Intel RealSense Camera (D415) with up to 1280 * 720 resolution. Higher resolutions tended to capture refresh patterns.

We set up a camera support structure (shown in black over the display) and a display support structure (shown in silver below the display) respectively to attach both the top-view camera and the display together. The Actor Robot navigated around on top of the display surface. The robot was controlled by a master computer through WiFi network. To protect the surface of the display while running the robot, we place a clear acrylic sheet on top of the display.

We now describe the steps that we used to conduct our experiments. First, we define each trajectory as the full motion path from the moment the robot starts to execute its internal policy until it finishes.

For each such trajectory, we place the actor robot at a randomly-generated position on the background display. Then the display shows a background image that has various configurations, depending on the type of the designed environments. If there is an obstacle in the pre-defined environment, a randomly generated obstacle position will be displayed and we place a red cardboard box at the corresponding location, to serve as a real obstacle.

In order to avoid the possibility of the observer determining the robot orientation from the

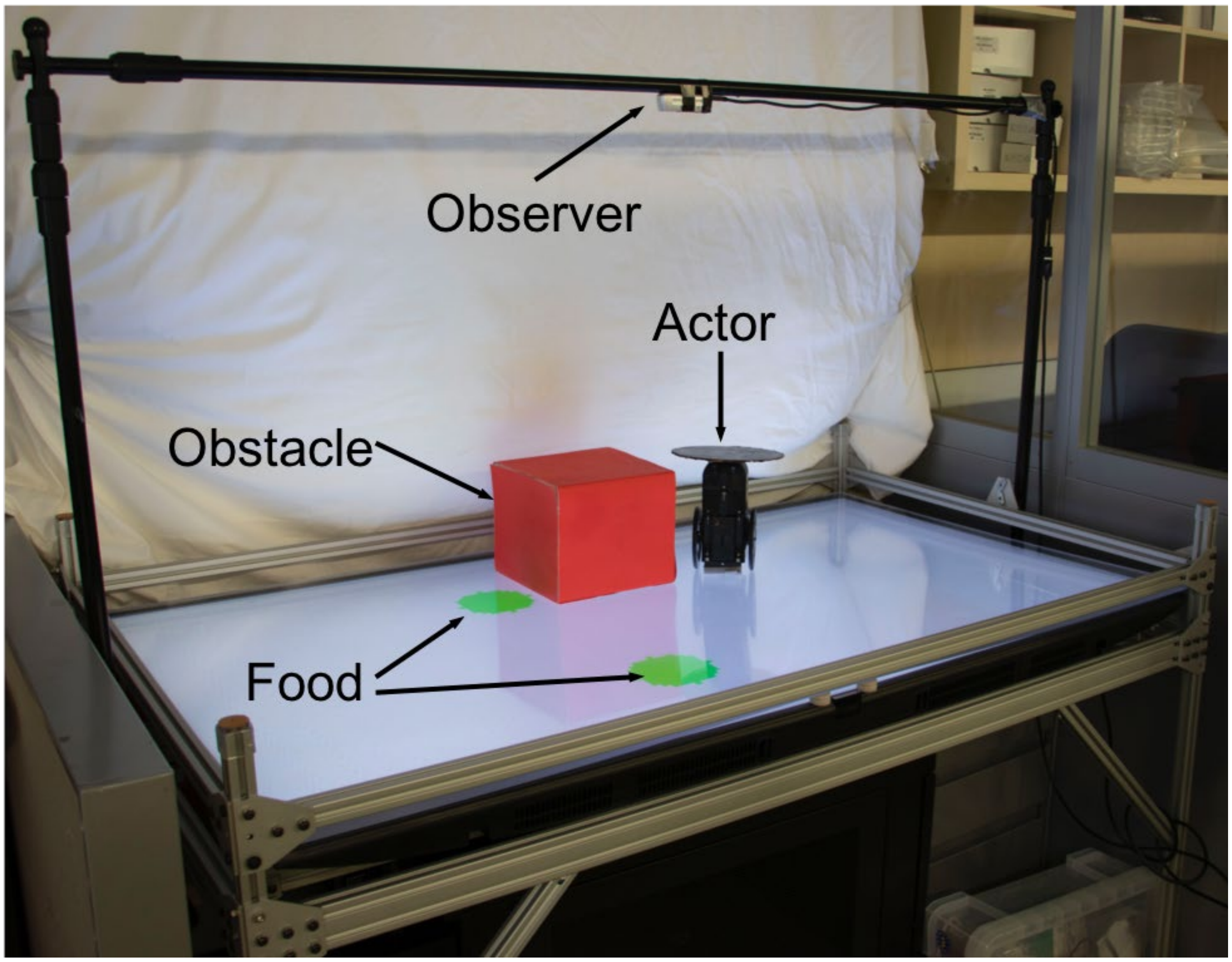


Fig. B.1: **Experiment platform setup.** We built an experimental platform for data collection and evaluations. An Actor robot navigated on top of a TV display where different green dots or red dots are displayed. An Observer watches the scene through a bird-view camera.

direction its wheels are pointing, we covered the robot with a black cylinder. This is why the robot looks like a black circle in the videos. We show some of examples of the initial state at the beginning of each policy in Fig.B.2.

After the initial setup, the Actor will start to execute its internal policy, while the Observer starts to record the video for the current trajectory through the bird-view camera until the Actor finishes. We then repeat the above steps with different positions of the foods and the obstacles, if applicable, to collect our dataset.

B.2 Robot details

We designed a 3D printed robot for our experiments. There are several practical challenges to address in this experiment. First, robot must be small enough to have room to maneuver on the display. Second, the battery of the robot requires sufficient power capacity to enable extensive data collection continuously. Such scalability is extremely important for data-driven approaches.

To overcome these challenges, we present our design of the Actor robot. The robot is a two wheeled robot with fully 3D-printed body and off-the-shelf electronic components. We show different views and dimensional information of the robot in Fig. B.3. We will also open source both the hardware design (CAD files for 3D printing, links for available parts inside the robot and assemble manuals) and software packages for this robot upon the paper is accepted. In Fig. B.4, we show the key parameters of the Actor robot for references.

B.3 Video Processing

With the intention of creating a pure visual theory of mind model, we preprocessed the recorded videos into two compound images. The steps involved in the video collection are described in the previous section. We now discuss the video processing steps. Fig. B.5 provides an overview of the entire processing pipeline. In essence, a compound image is created by taking the minimum value of each pixel over an entire video. This compresses a video into a single image. We assume standard RGB encoding,

We denote the video frames $I_{i,0}$, $I_{i,1}$, . . . , $I_{i,T}$ as corresponding to the frames of the i^{th} trajectory where T is the total number of frames in the current trajectory. We provide the Observer one single initial frame image as the input and expect the Observer to predict a single image that describes both the future trajectories of the Actor and the goal of the Actor.

To do so, we stacked all the frames from $I_{i,0}$ to $I_{i,T}$ by saving the smallest pixel values across of these frames to form one single output image. The input is just one first initial frame image. This pair of input and output image is further used to train our visual theory of mind model in a supervised way. We show some example input frames and output images in Fig. B.6.

Intuitively, stacking images like this is trying to project all the past waypoints of the Actor onto one single image without losing the background environment information. Hence, the output images have the whole trajectory of the Actor robot among the entire process. It is worth to note that we do not specifically track the position of the robots nor hand-crafted any image features to achieve this.

We collected 600 real world videos. We pre-processed these videos using the aforementioned method which gives us 600 input and output pairs of images. Furthermore, we augment 500 of them with 0.5 probability of being flipped right to left, and 0.5 probability of being flipped up and down, to get 5,000 training image pairs. Using the same augmentation pipeline, we augment 100 of the images to get 1,000 testing data pairs.

B.4 Deep Learning Architecture

We designed a multi-scale fully convolutional encoder-decoder network to serve as our image prediction network. An overview of our network architecture is shown in Fig. B.7. Our network takes one single RGB image as the input and generates one single RGB image as output. Both the size of the input I_{input} and the size of the output I_{output} are $64 \times 64 \times 3$. The network mainly has two parts, an encoder network and a multi-scale decoder network. The encoder network is composed of several convolutional layers where each convolutional layer is followed by a batch normalization layer and a ReLU non-linear activation function. Such a unit is denoted as a whole as “Conv” block in Fig. B.7.

It has been demonstrated that a multi-scale convolutional network is useful for refining coarse feature representations to higher resolution. Inspired by these prior works, we used similar multi-scale prediction networks within the decoder network. Each input feature is fed into a “Pred” block which is a convolutional layer followed by a transposed convolutional layer and a Sigmoid non-linear activation function to generate a sub-sampled predicted feature map. This predicted feature map is then concatenated with the output from a “Deconv” block to be fed into the next stage. The output from each stage is sent into both the “Deconv” unit and the “Pred” unit to get the feature predictions of the next stage.

By the end of the decoder network, the “Deconv” will generate the final output which has the same size as the input image. Similar to the “Conv” unit, each “Deconv” block in Fig. B.7 is composed of a transposed convolutional layer and a Sigmoid non-linear activation function. We show all the parameter settings of our architecture in Fig. B.8. We optimize our network to minimize a Mean Square Error loss. We train our network with Adam optimizer with an initial learning rate 0.01 and batch size 128 for 100 epochs. The learning rate decays by 90% at epoch 10, 30, 50, and 80.

Fig. B.9 shows more sample input, target output and predicted output images after training the network. Qualitatively, our predictions generally match the target outputs and can successfully model the policy of the Actor robot and its internal hidden goal. We also show the learning curve of the “hardest” policy across multiple runs in Fig. **??**.

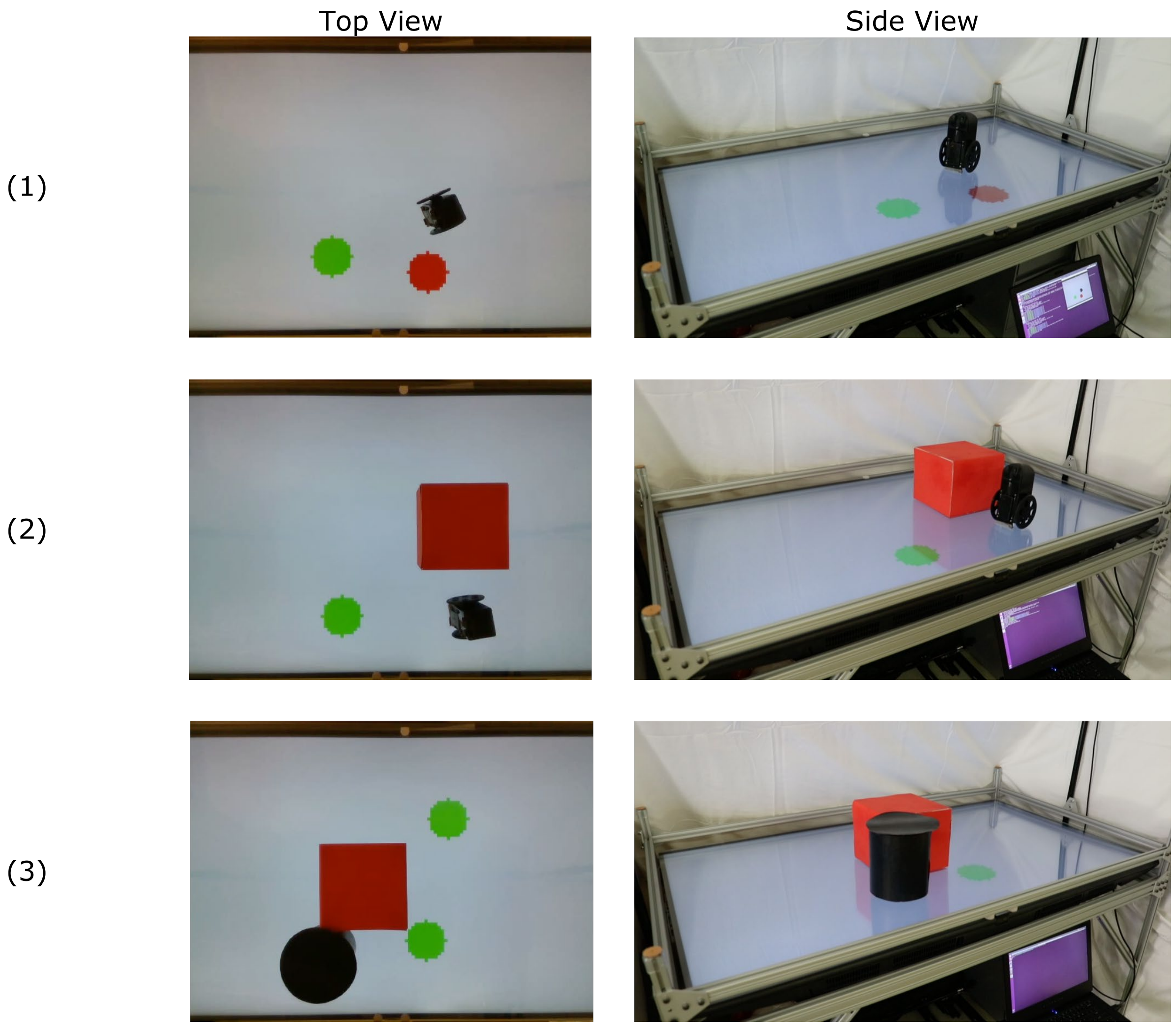


Fig. B.2: **Examples of initial setup for each actor policy.** Each row shows a pair of images taken from different angles while the actor robot performs various policies. The left column is taken from the top view camera and the right column is taken from a side angle to show a “1st person view”. (1) The Actor robot always navigates towards the green food and ignore the red food. (2) The Actor robot always go to the visible green food. If there is no green food visible (e.g., the green food is occluded by some obstacle), the Actor robot will stay in put. (3) The Actor robot always pursues the closest food it can perceive.

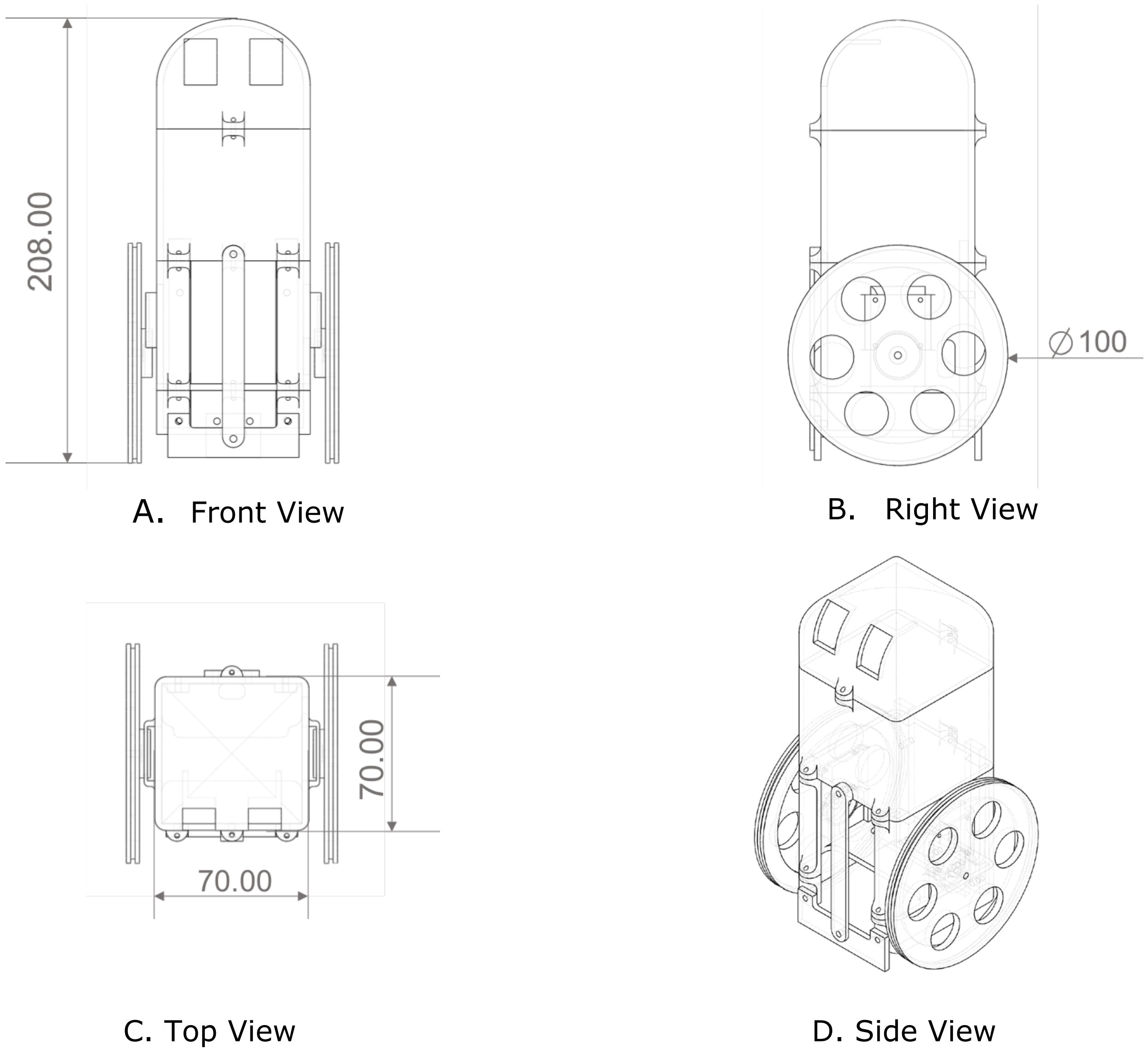


Fig. B.3: **CAD model of the Actor robot.** We use this CAD file to 3D print the whole body of our Actor robot. Different views (unit: mm) have been shown in each figure along with important dimensional parameters.

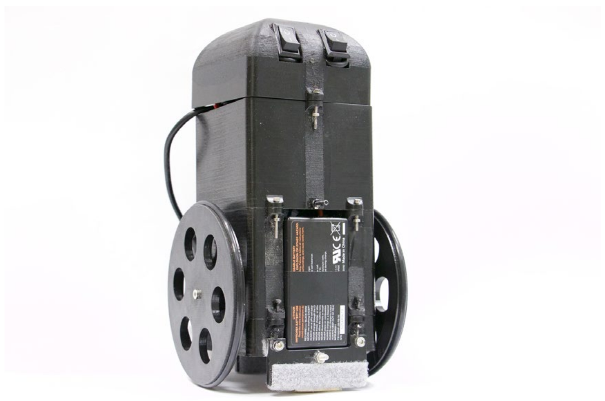
	Specifications	
	Basic Capacities	Navigation, Remote control, easy to attach various sensors such as camera, microphone etc.
	Size	See Fig. S3.
	Weight	654 g
	Speed Range	0 ~ 30 cm / s
	Accuracy (Straight Line):	Approximately 1.5 cm off after running with 15.5 cm / s for 1 second (before calibration)

Fig. B.4: **Key parameters of the Actor robot.**

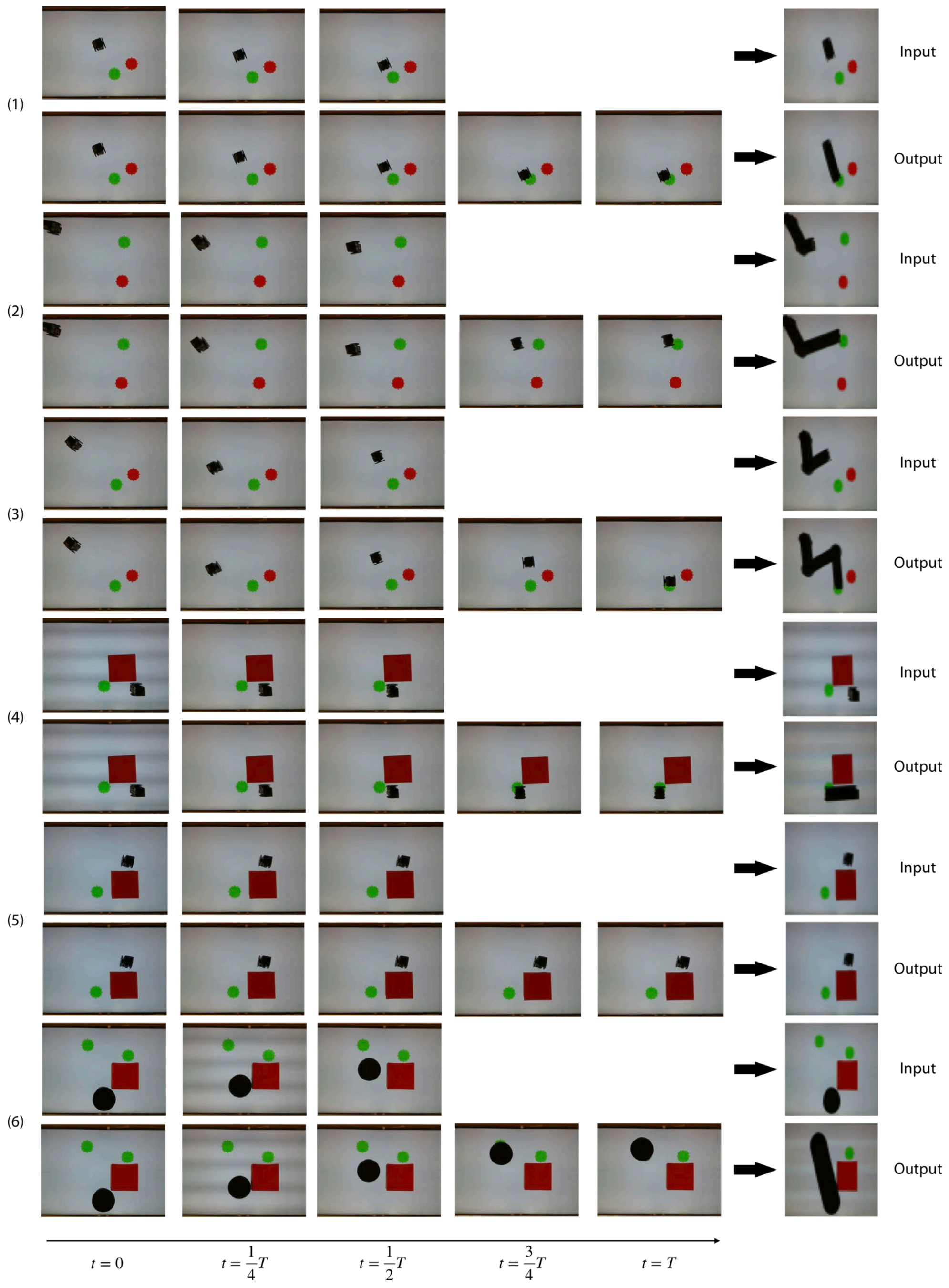


Fig. B.5: **Examples of data preprocessing pipeline of straight line policy, "elbow" line policy, "zig-zag" line policy and one single food with obstacle policy.** Every two rows correspond to one preprocessing pipeline. In the 1st row, all the frames in the first half of the video are stacked together to generate one single input image. In the 2nd row, all frames on the current video are stacked together to form one single target image. Here we only show three frames and five frames for each row for illustration purposes. In reality, we process all of the obtained video frames. As we can see in the last column, each single image contains all the information of the current environment and the encoded past trajectory of the Actor robot.

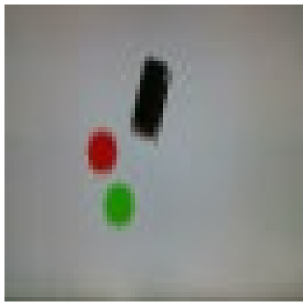
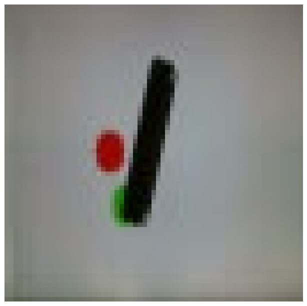
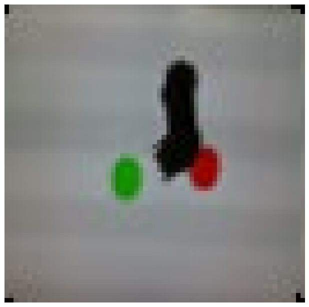
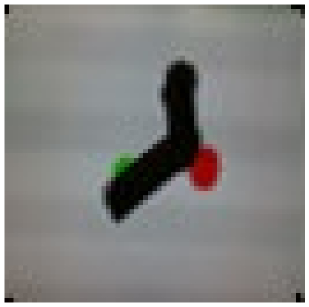
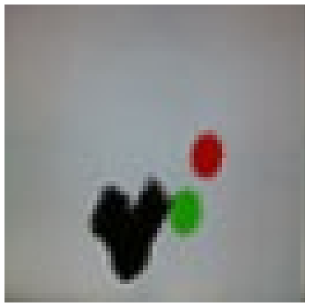
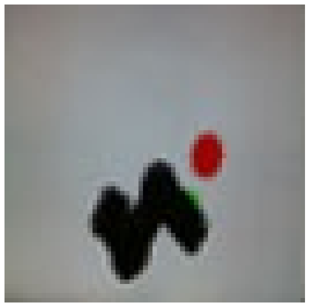
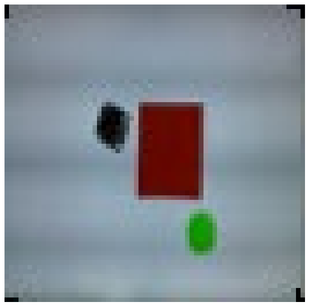
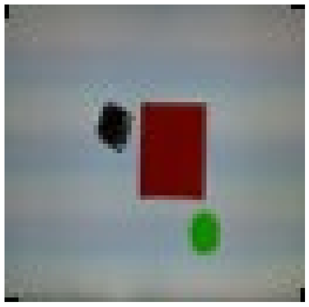
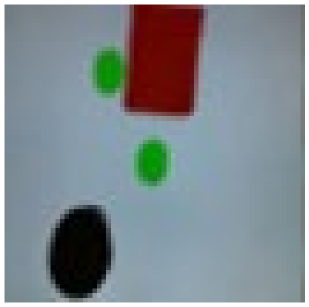
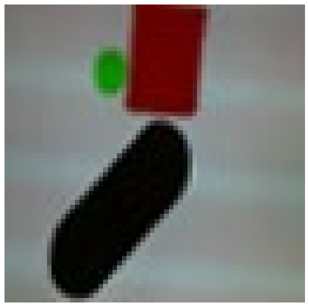
Policy Description	Input	Target Output
Straight Line Policy: The Actor always goes to the green food in straight line.		
Elbow Policy: The Actor always goes to the green food by first going to a control point and then navigating to the green food in straight lines.		
Zig-Zag Policy: The Actor always goes to the green food by first going to two control points subsequently and then navigating to the green food in straight line.		
One Food Obstacle Policy: The Actor always goes to the green food if it is visible to the robot. Otherwise, the Actor robot will not move. (The red cube is the obstacle and the robot is not able to see through it.)		
Two Foods Obstacle Policy: The Actor always goes to the closest green food it can see in a straight line. (The red cube is the obstacle and the robot is not able to see through it.)		

Fig. B.6: **The descriptions of the Actor policy.**

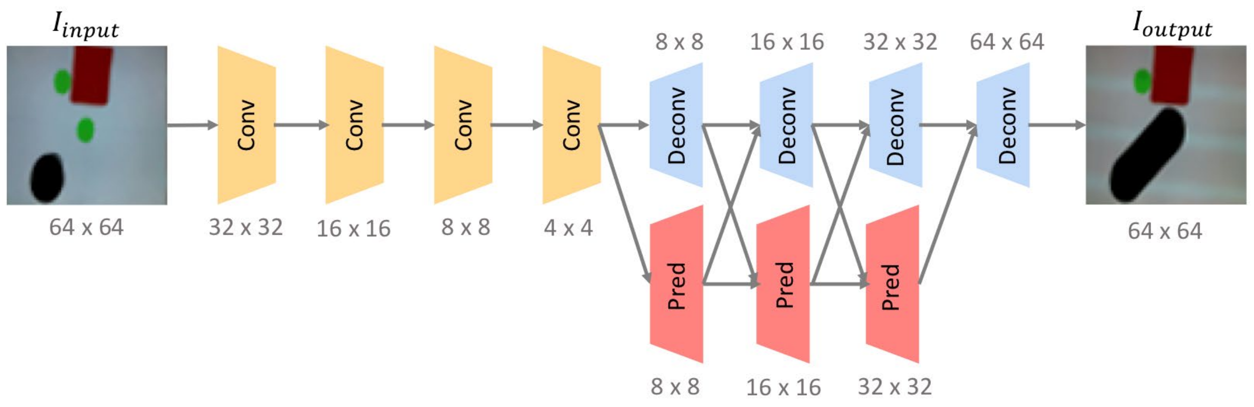


Fig. B.7: **Image Prediction Network Architecture.** Our image prediction network is composed of several layers of convolutional layers and deconvolutional layers. At the deconvolutional stage, we utilize multi-scale prediction to maintain high resolution in our output image.

Layer	**Kernel Size**	**Num Outputs**	**Stride**	**Padding**	**Dilation**	**Activation**
Conv1	4 × 4	32	2	1	1	ReLU
Conv2	4 × 4	32	2	1	1	ReLU
Conv3	4 × 4	64	2	1	1	ReLU
Conv4	4 × 4	128	2	1	1	ReLU
Deconv4	4 × 4	64	2	1	1	Sigmoid
Deconv3	4 × 4	32	2	1	1	ReLU
Deconv2	4 × 4	16	2	1	1	ReLU
Deconv1	4 × 4	3	2	1	1	ReLU
Pred3Conv	3 × 3	3	1	1	1	N/A
Pred2Conv	3 × 3	3	1	1	1	N/A
Pred1Conv	3 × 3	3	1	1	1	N/A
Pred3Deconv	4 × 4	3	2	1	1	Sigmoid
Pred2Deconv	4 × 4	3	2	1	1	Sigmoid
Pred1Deconv	4 × 4	3	2	1	1	Sigmoid

Fig. B.8: **Parameters of the Image Prediction Network.** The name of the layer corresponds to the unit in the architecture and the number indicates the sequence of the current layer. We choose to use Sigmoid as our final activation function due to the normalization.

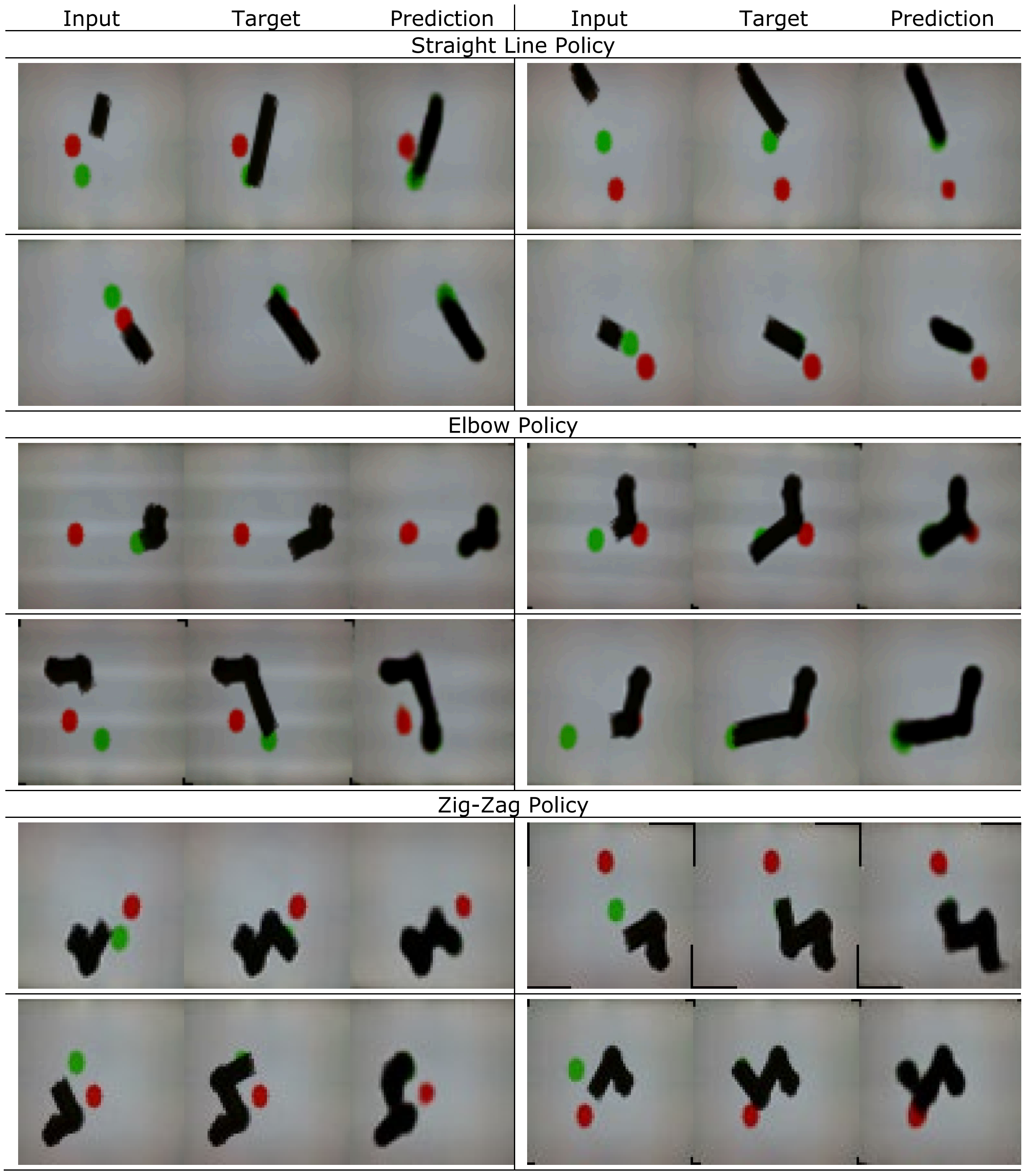


Fig. B.9: **Parameters of the Image Prediction Network.** We show some examples of the prediction results from our Observer network. Each group of three images represent one example. The 1st image in each group is the input image shown to the Observer network and the second image in each group is the target output of the Observer network. We get the second image by recording the real motion of the Actor robot. The third image is predicted from our Observer network and it should match the second image if the Observer is able to successfully infer the future trajectories and the goal of the Actor robot.

Appendix C: The Boombox: Visual Reconstruction from Acoustic Vibrations

C.1 Data Processing Details

There are three modalities (RGB, depth, and audio) in our collected data. Our goal for data processing is to obtain high-quality data samples for coherent dynamics and scene representation.

For each video recording, we aligned the RGB and depth streams with the camera parameters and extracted the last frame to represent the final scene after the object became stable. We applied edge-preserving and hole-filling spatial filters on the depth images to get less noisy depth images. We then cropped and resized all the image frames to remove the unnecessary backgrounds. In order to quantitatively evaluate our predictions, we further applied background subtractions leading to binary object masks.

Audio clips were recorded continuously across per data collection trail which included multiple sequences. Therefore, the first step requires segmenting the audio clips for each dropping sequence. We first calculate the audio segments for each microphone with energy threshold. As a means to synchronize among all four microphones, we use the earliest and latest sound arrival timestamps from four audio segments on the same sequence to finalize the segmentation windows.

C.2 Network Architectures

We list all the specific parameters of the encoder and decoder network in Table C.1 and Table C.2. All layers are accompanied with a batch normalization layer and a specified activation function. We will also open source all the hardware and software design along with the collected dataset.

Layer	Kernel Size	#Filters	Stride	Padding	Dilation	Activation
Conv1	4x4	32	2	1	1	ReLU
Conv2	4x4	32	2	1	1	ReLU
Conv3	4x4	64	2	1	1	ReLU
Conv4	4x4	128	2	1	1	ReLU
Conv5	4x4	128	2	1	1	ReLU
Conv6	4x4	128	2	1	1	ReLU
Conv7	4x4	128	2	1	1	ReLU

Table C.1: **Encoder Network Parameters:** we list all the specific parameters for the encoder network. In addition to the above convolutional layers, after each "Conv" layer, we attach another convolutional layer with the same number of filters as the current convolutional layer but with 4×4 kernel and 3 as stride.

Layer	Kernel Size	#Filters	Stride	Padding	Dilation	Activation
Deconv7	4x4	128	2	1	1	ReLU
Deconv6	4x4	128	2	1	1	ReLU
Deconv5	4x4	128	2	1	1	ReLU
Deconv4	4x4	64	2	1	1	ReLU
Deconv3	4x4	32	2	1	1	ReLU
Deconv2	4x4	16	2	1	1	ReLU
Deconv1	4x4	3/1	2	1	1	Sigmoid

Table C.2: **Decoder Network Parameters:** we list all the specific parameters for the decoder network. Except for the last layer, along with every transposed convolutional layer, the input of each layer is also first passed through a 2D convolutional layer with kernel size 3×3 and stride 1 and then a transposed 2D convolutional layer with kernel size 3×3 and stride 2. The output of this branch will then be concatenated with each "Deconv" layer along the feature dimension as the input of the next "Deconv" layer.

Appendix D: Automated Discovery of Fundamental Variables

Data Collection and Processing

In this section, we describe data collection processes for all datasets. The rigid double pendulum, swing stick, air dancer, fire and lava lamp datasets were collected from real-world physical experiments, while the circular motion, reaction diffusion, single pendulum, and elastic double pendulum datasets were collected from simulations.

Rigid double pendulum.

The physical parameters of our rigid double pendulum system is shown in Fig. D.1. The system used in our investigation is a two colored chaotic pendulum from 3D scientific: the first arm is black and the second arm is blue. Using the pivot attachment that came with the pendulum, the pendulum is installed against a brown-beige wall in the laboratory. There are 4 bearings on the pendulum. Three of them are fixed in place and one is left loose to reduce friction. We used an iPhone7 to record videos at 720 p and 240 fps.

Length of the first arm: 20.5 cm	Double Pendulum
Length of the second arm: 17.9 cm	
Mass of the first arm: 0.262 kg	
Mass of the second arm: 0.110 kg	
Depth of the arms: 3.8 cm	

Fig. D.1: Physical parameters of the rigid double pendulum system

We collected a total of 100 videos, with an approximate length of 15 seconds for each video. We used 80 of these videos for training and validation, and 20 of them for testing. For better video quality, we trimmed each video to 11s in order to avoid the movement at the beginning and the end of recording caused by humans and small changes in brightness or illumination caused by the camera. Another reason is that the dynamics towards the late part of the recordings are more predictable due to the lack of energy and the loss of momentum. Afterwards, we sub-sampled the video to construct a video dataset with 60 fps to produce sufficient visual difference between subsequent frames in a prediction triplet. To feed the video frames into our visual predictive models, the images are resized to 128×128.

Since we are interested in evaluating the results of prediction from the double pendulum system, we further equalized the background of the pendulum system with a simple color filtering so that our vision algorithms can detect the position and orientation of the pendulum arms with another color filtering during the evaluation process. We performed this additional step only to the double pendulum, for the sake of evaluation alone, while other systems do not involve this extra preprocessing step.

Swing stick.

The physical parameters of the swing sticks can be found in Fig. D.2. The system being used is from Geelong Shope, made out of a high-quality base, aluminum sticks, high quality bearings and black rubber feet. We used a GoPro Hero 5 Black camera to capture the motion of the system on a 240 frame per second and 720 pixel setting. The GoPro was mounted on a tripod directly at the height of the swing stick table. For each video sequence, we held the stick to a random position and then applied a force on the arm to cause motions that can last for a longer period of time.

We collected a total of 23 videos of approximately 150 seconds for each video sequence. We used 18 of these videos for training and validation and the remaining 5 for testing. Similar to the rigid double pendulum dataset, we trimmed the videos to obtain high-quality data, resulting in 140 seconds for each video. We further sub-sampled the video to 60 fps and resized the image frames

Mass without batteries: 248 g	Swing Stick
Mass with batteries: 340 g	
Span of the two sticks: 34 cm	
Height of the system: 37 cm	
Width of the base: 8 cm	
Length of the baes: 22.5 cm	
Length of the longer stick: 24 cm	
Length of the shorter stick: 18 cm	
Hinge point for the first arm (from the longer side): 17.5 cm	
Hinge point for the second arm (from longer side): 10.1 cm	

Fig. D.2: Physical parameters of the swing stick system

to 128 × 128 before sending the frames to our dynamics predictive model.

Air dancer.

The physics parameters of the air dancer system can be found in Fig. D.3. Initially, the air dancer ran on a 9 V capped battery that could directly be placed in the compartment. However, this led to undesirable complications in our study. First, the 9 V voltage was too high for the motor as the rotation of the fan prevented the dancer from showing sustained chaotic behavior. It would often stand upright after a very short duration. Second, the battery life for the system is short. Over the time of recording the dataset, the voltage of the battery quickly dropped.

To resolve these complications, we cut the battery connections from the dancer and plugged it into a variable dc power supply with 6.70 V, and grounding the negative connection. This produced

Span of the dancer: 19.2 cm	Air Dancer
Height of the dancer: 30.0 cm	
Overall height: 41.0 cm	
Width of the base: 7.0 cm	
Diameter of the fan: 3.0 cm	

Fig. D.3: Physical parameters of the air dancer system

appropriate airflow through the blower to enforce the repetitive high-low pressure phenomenon. We recorded the video with a GoPro Hero5 Black. In total, we collected 27 videos, each of approximate length 150 seconds. We used 22 videos for training and validation and 5 videos for testing. Following the previous steps, all the videos were trimmed to 140 seconds to obtain the final dataset. The image frames were resized to 128×128 with 60 fps.

Lava lamp.

We downloaded a real world recording of lava lamp system from YouTube. The video lasts about 4 hours with 2.5 fps.

Fire.

We downloaded a real world recording of fire system from YouTube. The video lasts about 3,603 seconds with 24 fps.

Circular Motion.

We simulated a circle moving along a circular path with a constant speed to construct the circular motion dataset. We fixed the center and the radius of the circular path as well as the radius of the circle. For each sequence, we randomly sampled the circle's initial position and constant speed. In total, we collected 1,100 sequences with 60fps. We used 880 of these sequences for training and validation, and 220 of them for testing.

Reaction diffusion.

We simulated the dynamics of a planar spiral wave to construct our reaction diffusion dataset. The dynamics of the system is driven by the following reaction-diffusion PDEs:

$$u_t = (1 - (u^2 + v^2))u + \beta(u^2 + v^2)v + d(u_{xx} + u_{yy}),$$
$$v_t = -\beta(u^2 + v^2)u + (1 - (u^2 + v^2))v + d(v_{xx} + v_{yy}),$$

with parameters $d = 0.1$ and $\beta = 1$. We ran only one simulation by solving the PDEs, following the original implementations. We then constructed the dataset by rendering the scalar field $u(x, y, t)$ at each time t as a 128×128 image with time t sampled at 5 fps. The long sequence of frames sampled from the simulation was divided into 100 shorter sequences. We used 80 of these sequences for training and validation, and 20 of them for testing.

Single pendulum.

We will present the physics equations of the single pendulum system in the "Physics Equations of Pendulum Systems" section. We set the pendulum mass as $m = 1$kg and the pendulum length as $L = 0.5$m. For each sequence, we randomly sampled the initial position and velocity of the pendulum arm. In total, we collected 1,200 sequences with 60fps. We used 960 of these sequences for training and validation, and 240 of them for testing.

Elastic double pendulum.

We will present the physics equations of the elastic double pendulum system in the "Physics Equations of Pendulum Systems" section. We set the physical parameters of this system to be the same as the physical parameters of the rigid double pendulum except that the first arm was replaced by a massless spring with a elasticity constant $k = 40\text{kg}/\text{s}^2$. For each sequence, we randomly sampled the initial angle and angular velocity of the spring, the initial length and stretch velocity of the spring, and the initial angle and angular velocity of the pendulum arm. In total, we collected 1,200 sequences with 60fps. We used 960 of these sequences for training and validation, and 240 of them for testing.

Physics Equations of Pendulum Systems

In this section, we provide more information on the physical state variables and equations of the three pendulum systems: the single pendulum, the rigid double pendulum, and the elastic double pendulum. See Fig. D.4 for a graphical illustration of those systems.

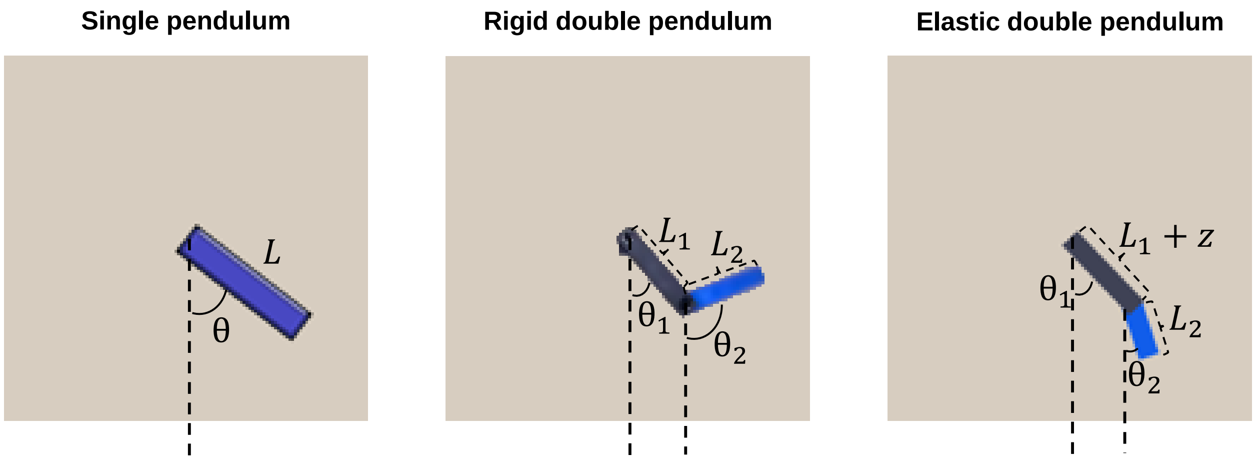


Fig. D.4: Illustration of our three pendulum systems

Single pendulum

Denote m the mass and L the length of the pendulum. The pendulum's momentum of inertia is $I = \frac{1}{3}mL^2$. We specify the system state by the pendulum's angular position θ and its respective angular velocity $\dot{\theta}$. The system's kinetic energy is

$$T = \frac{1}{2}I\dot{\theta}^2 = \frac{1}{6}mL^2\dot{\theta}^2.$$

Taking the configuration that the pendulum arm is horizontal as the zero point, the system's potential energy is

$$V = -\frac{1}{2}mgL\cos\theta.$$

Therefore, the system's Lagrangian is

$$L = T - V = \frac{1}{6}mL^2\dot{\theta}^2 + \frac{1}{2}mgL\cos\theta,$$

which gives the the equation of motion of the system:

$$\ddot{\theta} = -\frac{3g}{2L}\sin\theta.$$

The total energy of the system is:

$$E = T + V = \frac{1}{6}mL^2\dot{\theta}^2 - \frac{1}{2}mgL\cos\theta.$$

E is also the Hamiltonian of the system.

Rigid double pendulum

Denote m_1 and m_2 the masses of the two arms of the double pendulum, L_1 and W_1 the length and width of the first arm, and L_2 and W_2 the length and width of the second arm. The momenta of inertia of the two arms are:

$$I_1 = \frac{1}{12}(L_1^2 + W_1^2), \quad I_2 = \frac{1}{12}(L_2^2 + W_2^2).$$

We specify the system state by the two arms' angular positions θ_1 and θ_2, and their respective angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$.

The kinetic energy of the system is the sum of the two arms' translational and rotational kinetic energies, which is given by:

$$T = \frac{1}{2}\left(\frac{1}{4}m_1L_1^2 + m_2L_1^2 + I_1\right)\dot{\theta}_1^2 + \frac{1}{2}\left(\frac{1}{4}m_2L_2^2 + I_2\right)\dot{\theta}_2^2 + \frac{1}{2}m_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2),$$

The potential energy of the system is the sum of the two arms' gravitational potential energies. Taking the configuration that both arms are horizontal as the zero point, the potential energy of the system is given by:

$$V = -\left(\frac{1}{2}m_1 + m_2\right)gL_1\cos\theta_1 - \frac{1}{2}m_2gL_2\cos\theta_2,$$

The total energy of the system is the sum of kinetic and potential energies, which is given by:

$$\begin{aligned} E =& \frac{1}{2}\left(\frac{1}{4}m_1L_1^2 + m_2L_1^2 + I_1\right)\dot{\theta}_1^2 + \frac{1}{2}\left(\frac{1}{4}m_2L_2^2 + I_2\right)\dot{\theta}_2^2 + \frac{1}{2}m_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2) \\ &- \left(\frac{1}{2}m_1 + m_2\right)gL_1\cos\theta_1 - \frac{1}{2}m_2gL_2\cos\theta_2. \end{aligned}$$

E is also the Hamiltonian of the system.

Elastic double pendulum

The elastic double pendulum is composed of a massless spring and a rigid pendulum. We denote m the mass of the pendulum. In addition, we denote L_1 the original length and z the stretch of the spring, and L_2 and W_2 the length and width of the pendulum. The pendulum's momentum of inertia is $I = \frac{1}{12}(L_2^2 + W_2^2)$.

We specify the system state by the spring's angular position θ_1, the pendulum's angular position θ_2, the spring stretch z, and their respective velocities $\dot{\theta}_1$, $\dot{\theta}_2$, $\dot{z}$.

The kinetic energy of the system is the sum of the pendulum's translational and rotational kinetic energies, which is given by

$$\begin{aligned}T =&\frac{1}{2}m\left[(L_1+z)^2\dot{\theta}_1^2+\frac{1}{4}L_2^2\dot{\theta}_2^2+\dot{z}^2+(L_1+z)L_2\cos(\theta_1-\theta_2)\dot{\theta}_1\dot{\theta}_2+L_2\sin(\theta_1-\theta_2)\dot{\theta}_2\dot{z}\right]\\&+\frac{1}{2}I\dot{\theta}_2^2.\end{aligned}$$

The potential energy of the system is the sum of the gravitational potential energy of the pendulum and the elastic potential energy of the spring. Taking the configuration that both the spring and the pendulum are horizontal as the zero point of the pendulum's gravitational potential energy, the system's potential energy is given by:

$$V = mg\left[-(L_1+z)\cos\theta_1-\frac{1}{2}L_2\cos\theta_2\right]+\frac{1}{2}kz^2.$$

Therefore, the system's Lagrangian $L = T - V$ is given by

$$\begin{aligned}L =&\frac{1}{2}m\left[(L_1+z)^2\dot{\theta}_1^2+\frac{1}{4}L_2^2\dot{\theta}_2^2+\dot{z}^2+(L_1+z)L_2\cos(\theta_1-\theta_2)\dot{\theta}_1\dot{\theta}_2+L_2\sin(\theta_1-\theta_2)\dot{\theta}_2\dot{z}\right]\\&+\frac{1}{2}I\dot{\theta}_2^2+mg\left[(L_1+z)\cos\theta_1+\frac{1}{2}L_2\cos\theta_2\right]-\frac{1}{2}kz^2.\end{aligned}$$

Then we can derive the system's equations of motion from the above Lagrangian, resulting in a 3×3 system of ODEs:

$$A\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{z}\end{bmatrix}=b,$$

where

$$A=\begin{bmatrix}(L_1+z)^2 & \frac{1}{2}(L_1+z)L_2\cos(\theta_1-\theta_2) & 0\\ \frac{1}{2}(L_1+z)L_2\cos(\theta_1-\theta_2) & \frac{1}{4}L_2^2+\frac{I}{m} & \frac{1}{2}L_2\sin(\theta_1-\theta_2)\\ 0 & \frac{1}{2}L_2\sin(\theta_1-\theta_2) & 1\end{bmatrix},$$

and

$$b = \begin{bmatrix} -\frac{1}{2}(L_1 + z)L_2 \sin(\theta_1 - \theta_2)(\dot{\theta}_2^2) - 2(L_1 + z)\dot{\theta}_1\dot{z} - g(L_1 + z)\sin\theta_1 \\ \frac{1}{2}(L_1 + z)L_2 \sin(\theta_1 - \theta_2)(\dot{\theta}_1^2) - L_2\cos(\theta_1 - \theta_2)\dot{\theta}_1\dot{z} - \frac{1}{2}gL_2\sin\theta_2 \\ (L_1 + z)\dot{\theta}_1^2 + \frac{1}{2}L_2\cos(\theta_1 - \theta_2)\dot{\theta}_2^2 + g\cos\theta_1 - \frac{k}{m}z \end{bmatrix}.$$

It is straightforward to verify that the matrix A is symmetric and positive definite. Therefore the ODE system is always solvable.

The system's total energy is the sum of its kinetic and potential energies, which is given by:

$$\begin{aligned} E =& \frac{1}{2}m\left[(L_1 + z)^2\dot{\theta}_1^2 + \frac{1}{4}L_2^2\dot{\theta}_2^2 + \dot{z}^2 + (L_1 + z)L_2\cos(\theta_1 - \theta_2)\dot{\theta}_1\dot{\theta}_2 + L_2\sin(\theta_1 - \theta_2)\dot{\theta}_2\dot{z}\right] \\ &+ \frac{1}{2}I\dot{\theta}_2^2 - mg\left[(L_1 + z)\cos\theta_1 + \frac{1}{2}L_2\cos\theta_2\right] + \frac{1}{2}kz^2. \end{aligned}$$

E is also the Hamiltonian of the system.

Model Details

In this section, we provide details about our models for dynamics prediction, latent reconstruction, and latent dynamics prediction with Neural State Variables.

Dynamics predictive model architecture

The dynamics predictive model is an auto-encoder with specific parameters listed in Tab. D.1. All convolutional or transposed convolutional layers are accompanied with a batch normalization layer and a specified activation function. For the encoder network, after each "Conv" layer as shown in Tab. D.1, we attach another convolutional layer with the same number of filters as the current convolutional layer but with 3×3 kernel and 1 as stride. For the decoder network, along with each "Deconv" layer as shown in Tab. D.1 except for the last one, the input is also passed through a transposed convolutional layer with kernel size 4×4, 2 as stride, and a Sigmoid activation function. The output of this branch will then be concatenated with each "Deconv" layer along the feature dimension as the input of the next "Deconv" layer.

Layer	Kernel Size	#Filters	Stride	Padding	Activation
Conv1	4×4	32	2	1	ReLU
Conv2	4×4	32	2	1	ReLU
Conv3	4×4	64	2	1	ReLU
Conv4	4×4	128	2	1	ReLU
Conv5	3×4	128	(1,2)	1	ReLU
Deconv5	3×4	64	(1,2)	1	ReLU
Deconv4	4×4	64	2	1	ReLU
Deconv3	4×4	32	2	1	ReLU
Deconv2	4×4	16	2	1	ReLU
Deconv1	4×4	3	2	1	Sigmoid

Table D.1: Dynamics predictive model architecture

Latent reconstruction model architecture

The latent reconstruction model is also an auto-encoder with specific parameters listed in Tab. D.2. Each layer is a linear layer accompanied with a sine activation function, and ID refers to the system's intrinsic dimension. The intermediate latent vectors whose dimension is ID are identified as Neural State Variables.

Encoder Layer	#Filters	Activation	Decoder Layer	#Filters	Activation
Layer1	128	Sine	Layer5	32	Sine
Layer2	64	Sine	Layer6	64	Sine
Layer3	32	Sine	Layer7	128	Sine
Layer4	ID	Sine	Layer8	64	Sine

Table D.2: Latent reconstruction model architecture

Neural latent dynamics model architecture

The neural latent dynamics model is a simple six-layer MLP, the widths of first five layers are (32, 64, 64, 64, 32, ID) where ID is the dimension of Neural State Variables, and each layer is accompanied with the ReLU activation function.

More Physics Evaluation Results for Dynamics Predictive Model

In this section, we show physics evaluation results for the high-dimensional dynamics predictive model on the single pendulum and rigid double pendulum systems. As shown in Tab. D.3 and Tab. D.4, our dynamics predictive model outperforms both the copy data and linear extrapolation baselines on both systems.

Method	θ (deg)	$\dot{\theta}$ (deg/s)	Energy (J)
Copy data	10.09 (±0.04)	37.44 (±0.14)	0.10 (±0.00)
Linear extrapolation	0.67 (±0.00)	37.44 (±0.14)	0.28 (±0.00)
Our model	0.22 (±0.00)	18.97 (±0.12)	0.15 (±0.00)

Table D.3: Physics evaluation results on the single pendulum system

Method	θ_1 (deg)	θ_2 (deg)	$\dot{\theta}_1$ (deg/s)	$\dot{\theta}_2$ (deg/s)	Energy (J)
Copy data	8.29 (±0.04)	17.96 (±0.10)	111.64 (±0.78)	170.75 (±0.96)	0.06 (±0.00)
Linear extrapolation	2.39 (±0.02)	3.12 (±0.02)	111.64 (±0.78)	170.75 (±0.96)	0.08 (±0.00)
Our model	0.89 (±0.01)	1.55 (±0.01)	66.67 (±0.53)	113.94 (±0.84)	0.06 (±0.00)

Table D.4: Physics evaluation results on the rigid double pendulum system

More Results from Intrinsic Dimension Estimation

There are several intrinsic dimension estimation algorithms in addition to the Levina-Bickel's algorithm. In this section we provide the intrinsic dimension estimation results from another four widely used algorithms: MiND_ML, MiND_KL, Hein, and CD using the same latent vectors.

Those results together with results from the Levina-Bickel's algorithm and ground truth values are shown in Tab. D.5. As shown in the results, Levina-Bickel's algorithm provides robust and accurate estimations across all six systems.

System	Ground Truth	Levina-Bickel	MiND_ML	MiND_KL	Hein	CD
Circular motion	2	2.19 (±0.05)	2.66 (±0.07)	3.00 (±0.00)	6.33 (±0.47)	5.23 (±0.41)
Reaction diffusion	2	2.16 (±0.14)	2.34 (±0.08)	3.00 (±0.00)	3.00 (±1.41)	3.08 (±1.04)
Single pendulum	2	2.05 (±0.02)	2.05 (±0.01)	2.00 (±0.00)	2.00 (±0.00)	2.03 (±0.04)
Rigid double pendulum	4	4.71 (±0.03)	4.69 (±0.03)	5.00 (±0.00)	3.33 (±0.47)	3.63 (±0.10)
Swing stick	4	4.89 (±0.33)	4.38 (±0.10)	4.80 (±0.40)	3.33 (±0.47)	3.15 (±0.10)
Elastic double pendulum	6	5.34 (±0.20)	5.15 (±0.13)	5.13 (±0.34)	4.00 (±0.00)	4.26 (±0.12)

Table D.5: Intrinsic Dimension Estimation Results from Different Algorithms.

More Physics Evaluation Results for Hybrid Schemes

In this section, we provide physics evaluation results testing the hybrid schemes for long term predictions on the rigid double pendulum system. The hybrid scheme follows a $N+1$ pattern where every N steps performed with the high-dimensional latent vectors are followed by a one step prediction using Neural State Variables. We implemented the hybrid $N+1$ scheme with $N = 3, 4, 5, 6$ on the double pendulum system and compared physics evaluation results from the generated long term predictions. As shown in Fig. D.5, the hybrid scheme always produces reasonable long-term predictions, and performance is not sensitive to the choice of N.

More Theoretical Analysis on Long-Term Prediction Stability

In this section, we provide more theoretical analysis on long term prediction stability. We will fix the notations that $\mathcal{S} \subset \mathbb{R}^D$ is the system's state space where $\mathbb{R}^D$ is the high-dimensional image space, ID $= \dim\mathcal{S}$ is the system's intrinsic dimension, and LD is the latent space dimension of a given dynamics predictive model. The dimensions are assumed to satisfy ID $\ll$ LD $\ll D$. The ground truth dynamics is given by $\boldsymbol{X}_{t+1} = F(\boldsymbol{X}_t)$ where $F : \mathcal{S} \to \mathcal{S}$ is the system evolution mapping.

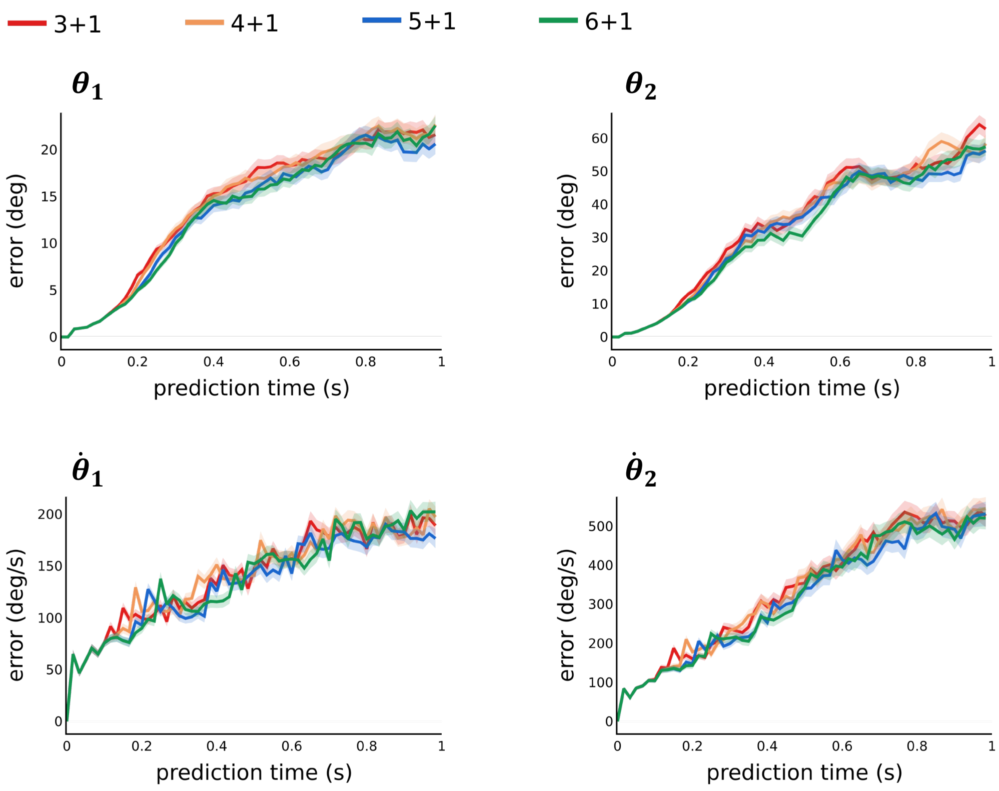


Fig. D.5: Physics evaluation with hybrid schemes on the double pendulum system

For the theoretical analysis purpose, let us define:

$$M_{\mathcal{S}}(\hat{\boldsymbol{X}}) \triangleq \operatorname{dist}(\hat{\boldsymbol{X}}, \mathcal{S}) = \inf_{\boldsymbol{X} \in \mathcal{S}} \left\| \hat{\boldsymbol{X}} - \boldsymbol{X} \right\|,$$

as the metric measuring the deviation from any predicted state $\hat{\boldsymbol{X}}$ to the state space $\mathcal{S}$, where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^D$.

Now let us consider the long term predictions generated from model rollouts through Neural State Variables. Starting from any initial state $\boldsymbol{X}_0 \in \mathcal{S}$, the model rollouts produce:

$$\boldsymbol{X}_0 = \hat{\boldsymbol{X}}_0 \rightarrow \hat{\boldsymbol{V}}_0 \rightarrow \hat{\boldsymbol{X}}_{dt} \rightarrow \hat{\boldsymbol{V}}_{dt} \rightarrow \hat{\boldsymbol{X}}_{2dt} \rightarrow \hat{\boldsymbol{V}}_{2dt} \rightarrow \hat{\boldsymbol{X}}_{3dt} \rightarrow \cdots,$$

where $\hat{\boldsymbol{X}}_t \in \mathbb{R}^D$, $t = 0, dt, 2dt, \cdots$ are the predicted frames, and $\hat{\boldsymbol{V}}_t \in \mathbb{R}^{\mathrm{ID}}$, $t = 0, dt, 2dt, \cdots$ are the corresponding Neural State Variables. We denote $\phi_{\mathrm{E}} = h_{\mathrm{E}} \circ g_{\mathrm{E}}$ that maps every $\hat{\boldsymbol{X}}_t$ to $\hat{\boldsymbol{V}}_t$ and $\phi_{\mathrm{D}} = g_{\mathrm{D}} \circ h_{\mathrm{D}}$ that maps every $\hat{\boldsymbol{V}}_t$ to $\hat{\boldsymbol{X}}_{t+dt}$ for $t = 0, dt, 2dt, \cdots$. Both ϕ_{E} and ϕ_{D} are compositions of neural networks that are trained to minimize the one-step prediction error. Therefore, $\hat{F} = \phi_{\mathrm{D}} \circ \phi_{\mathrm{E}}$ provides an approximation of the ground truth evolution mapping F through Neural State Variables. We will make the following assumption regarding the one-step dynamics approximation error.

Assumption 1. $M_{\mathcal{S}}(\hat{\boldsymbol{X}}) \leq \varepsilon \quad \forall \hat{\boldsymbol{X}} = \hat{F}(\boldsymbol{X}),\ \boldsymbol{X} \in \mathcal{S}$ for some $\varepsilon > 0$.

We note that all possible ground truth and predicted states fall into the set

$$\mathcal{Z} \triangleq \bigcup_{n=0}^{\infty} \hat{F}^{(n)}(\mathcal{S}),$$

where $\hat{F}^{(0)}(\mathcal{S}) = \mathcal{S}$ and

$$\hat{F}^{(n)}(\mathcal{S}) = \{\hat{\boldsymbol{X}} = \underbrace{\hat{F} \circ \hat{F} \cdots \circ \hat{F}}_{n \text{ times}}(\boldsymbol{X}) :\ \boldsymbol{X} \in \mathcal{S}\}, \quad n = 1, 2, \cdots$$

By definition $\mathcal{S} \subset \mathcal{Z} \subset \mathbb{R}^D$, which yields $\phi_{\mathrm{E}}(\mathcal{S}) \subset \phi_{\mathrm{E}}(\mathcal{Z}) \subset \mathbb{R}^{\mathrm{ID}}$. Here we make the second assumption saying that $\phi_{\mathrm{E}}(\mathcal{Z})$ is no larger than $\phi_{\mathrm{E}}(\mathcal{S})$ even if $\mathcal{Z}$ might be strictly larger than $\mathcal{S}$.

Assumption 2. $\phi_{\mathrm{E}}(\mathcal{S}) = \phi_{\mathrm{E}}(\mathcal{Z})$.

This assumption essentially relies on the dimensionality constraint. The state space $\mathcal{S}$ has dimension ID, while $\mathcal{Z}$ may have a higher dimension than ID. By projecting to the space of Neural State Variables whose dimension is ID, the extra dimensions of $\mathcal{Z}$ will be eliminated.

By Assumption 2 and the definition of $\mathcal{Z}$, we have:

$$\hat{F}(\mathcal{S}) = \phi_{\mathrm{D}} \circ \phi_{\mathrm{E}}(\mathcal{S}) = \phi_{\mathrm{D}} \circ \phi_{\mathrm{E}}(\mathcal{Z}) = \hat{F}(\mathcal{Z}) = \mathcal{Z}.$$

It together with Assumption 1 gives the following conclusion.

Proposition 1. Under Assumptions 1 and 2, we have $M_{\mathcal{S}}(\hat{\boldsymbol{X}}) \leq \varepsilon \quad \forall \hat{\boldsymbol{X}} \in \mathcal{Z}$.

As a direct corollary, for any long term prediction sequence $\{\hat{\boldsymbol{X}}_0, \hat{\boldsymbol{X}}_{dt}, \hat{\boldsymbol{X}}_{2dt}, \cdots\}$ from any initial state $\hat{\boldsymbol{X}}_0 = \boldsymbol{X}_0 \in \mathcal{S}$, we have $M_{\mathcal{S}}(\hat{\boldsymbol{X}}_t) \leq \varepsilon$ for all $t = 0, dt, 2dt, \cdots$. That is, the prediction sequence always stays in a fixed neighborhood of the ground truth state space $\mathcal{S}$ and the growth of

$M_{\mathcal{S}}(\hat{\boldsymbol{X}}_t)$ is well-controlled, which guarantees the stability of long term predictions generated with Neural State Variables. For the model rollouts through latent vectors whose dimension LD $\gg$ ID, similar arguments yield that $g_{\mathrm{E}}(\mathcal{S}) \subset g_{\mathrm{E}}(\mathcal{Z}) \subset \mathbb{R}^{\mathrm{LD}}$ and $g_{\mathrm{E}}(\mathcal{Z})$ could have a dimension higher than ID. Therefore, the long term predictions could escape from the state space $\mathcal{S}$ through the extra dimensions. By shrinking the latent dimension to the system's intrinsic dimension, such instability formation can be avoided.

More Results on Neural State Variables for Dynamic Stability Indicators

In this section, we show long term stability evaluation results using dynamic stability indicators with Neural State Variables. In Fig. D.6, the stability plots for six systems are shown. See the supplementary video for the corresponding visual predictions.

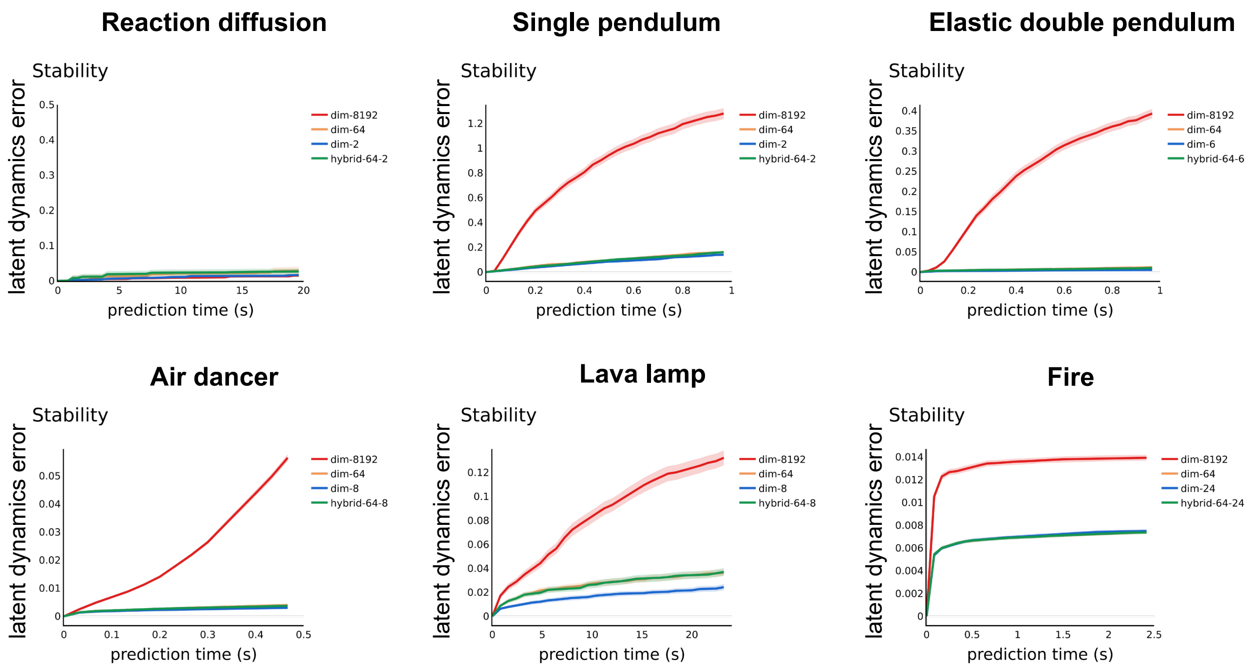


Fig. D.6: Stability evaluation with Neural State Variables

More Analysis on Neural State Variables

In this section, we show more results to demonstrate the rich physics information contained in our learned Neural State Variables. First, we give the quantitative latent regression results on the elastic double pendulum system in Tab. D.6. It can be observed that the learned Neural State Variables capture much richer information about the system dynamics than the variables obtained through PCA from high dimensional latent embedding vectors.

We also show visualizations of the Neural State Variables after applying PCA on them. See Fig. D.7.

Latent Variables	θ_1 (deg)	θ_2 (deg)	z (m)	$\dot{\theta}_1$ (deg/s)	$\dot{\theta}_2$ (deg/s)	$\dot{z}$ (m/s)	Total energy (J)
dim-6 PCA of dim-8192 Latents	23.86 (±1.14)	43.03 (±0.72)	0.02 (±0.00)	149.36 (±3.20)	397.20 (±9.50)	0.52 (±0.01)	0.08 (±0.00)
dim-6 PCA of dim-64 Latents	14.10 (±0.88)	27.96 (±0.85)	0.01 (±0.00)	170.68 (±6.37)	396.70 (±2.90)	0.59 (±0.01)	0.07 (±0.00)
dim-6 Latents	7.85 (±0.54)	14.11 (±0.50)	0.01 (±0.00)	70.03 (±1.41)	163.13 (±4.10)	0.27 (±0.01)	0.05 (±0.00)

Table D.6: Latent regression results on the elastic double pendulum system

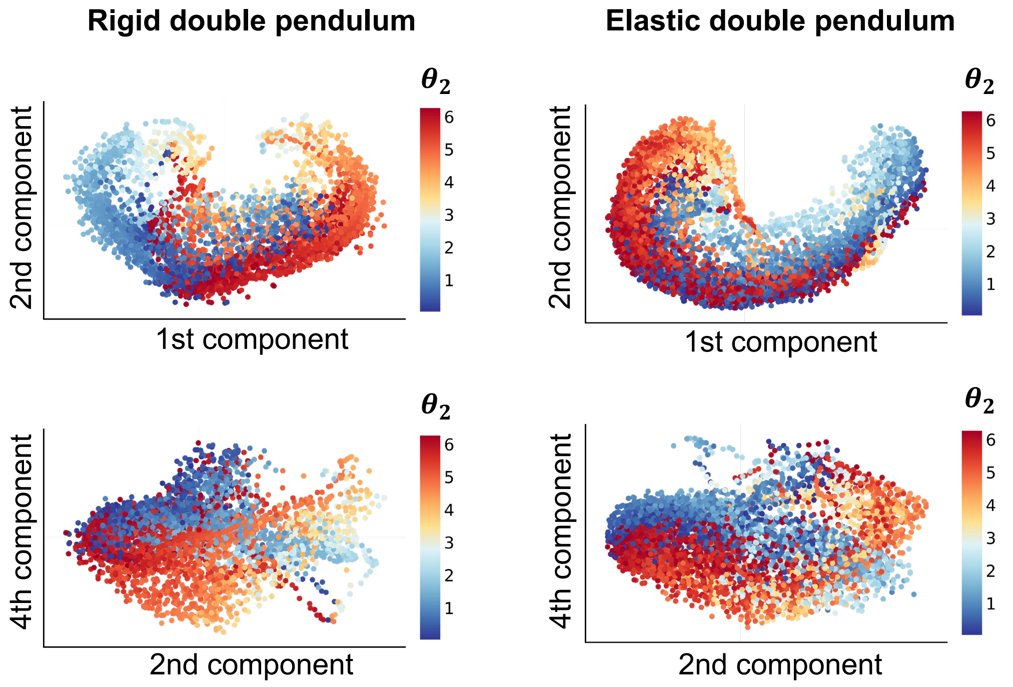


Fig. D.7: Visualization of Neural State Variables

Appendix E: Beyond Categorical Label Representations for Image Classification

E.1 Background and Motivation

Image classification is a well-established task in machine learning. The standard approach takes an input image and predicts a categorical distribution over the given classes. The most popular method to train these neural network is through a cross-entropy loss with backpropagation. Deep convolutional neural networks [219, 220, 221, 222, 223] have achieved extraordinary performance on this task, while some even surpass human level performance. However, is this a solved problem? The state-of-the-art performance commonly relies on large amounts of training data [224, 225, 226], and there exist many examples of networks with good performance that fail on images with imperceptible adversarial perturbations [227, 228, 229].

Much progress has been made in domains such as few-shot learning and meta-learning to improve the data efficiency of neural networks. There is also a large body of research addressing the challenge of adversarial defense. Most efforts have focused on improving optimization methods, weight initialization, architecture design, and data preprocessing. In this work, we find that simply replacing standard categorical labels with high dimensional, high entropy variants (e.g. an audio spectrogram pronouncing the name of the class) can lead to interesting properties such as improved robustness and efficiency, without a loss of accuracy.

Our research is inspired by key observations from human learning. Humans appear to learn to recognize new objects from few examples, and are not easily fooled by the types of adversarial perturbations applied to current neural networks. There could be many reasons for the discrepancy between how humans and machines learn. One significant aspect is that humans do not output categorical probabilities on all known categories. A child shown a picture of a dog and asked "what is this a picture of?" will directly speak out the answer — "dog." Similarly, a child being trained by a parent is shown a picture and then provided the associated label in the form of speech. These observations raise the question: Are we supervising neural networks on the best modality?

In this paper, we take one step closer to understanding the role of label representations inside the training pipeline of deep neural networks. However, while useful properties emerge by utilizing various label representations, we do not attempt to achieve state-of-the-art performance over these metrics. Rather, we hope to provide a novel research perspective on the standard setup. Therefore, our study is not mutually exclusive with previous research on improving adversarial robustness and data efficiency.

An overview of our approach is shown in Figure E.1. We first follow the above natural observation and modify the existing image classifiers to "speak out" the predictions instead of outputting a categorical distribution. Our initial experiments show surprising results: that neural networks trained with speech labels learn more robust features against adversarial attacks, and are more data-efficient when only less than 20% of training data is available.

Furthermore, we hypothesize that the improvements from the speech label representation come from its property as a specific type of high-dimensional object. To test our hypothesis, we performed a large-scale systematic study with various other high-dimensional label representations including constant matrices, speech spectrograms, shuffled speech spectrograms, composition of Gaussians, and high dimensional and low dimensional uniform random vectors. Our experimental results show that high-dimensional label representations with high entropy generally lead to robust and data efficient network training. We believe that our findings suggest a significant role of label representations which has been largely unexplored when considering the training of deep neural networks.

Our contributions are three fold. First, we introduce a new paradigm for the image classification task by using speech as the supervised signal. We demonstrate that speech models can achieve comparable performance to traditional models that rely on categorical outputs. Second, we quantitatively show that high-dimensional label representations with high entropy (e.g. audio spec-

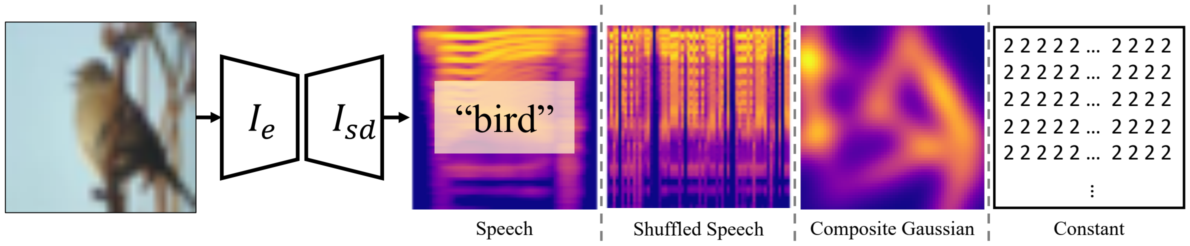


Fig. E.1: Label Representations beyond Categorical Probabilities: We study the role of label representation in training neural networks for image classification. We find that high-dimensional labels with high entropy lead to more robust and data-efficient feature learning.

trograms and composition of Gaussians) produce more robust and data-efficient neural networks, while high-dimensional labels with low entropy (e.g. constant matrices) and low-dimensional labels with high entropy do not have these benefits and may even lead to worse performance. Finally, we present a set of quantitative and qualitative analyses to systematically study and understand the learned feature representations of our networks. Our visualizations suggest that speech labels encourage learning more discriminative features.

E.2 Related Works

Data Efficiency and Robustness Data efficiency has been a widely studied problem within the context of few-shot learning and meta-learning [230, 231, 232, 233]. Researchers have made exciting progress on improving methods of optimization [234, 235], weight initialization [236, 237], and architecture design [238, 239].

There is also a large body of research addressing the challenge of adversarial defense. Adversarial training is perhaps the most common measure against adversarial attacks [240, 241, 242, 243, 244]. Recent works try to tackle the problem by leveraging GANs [245], detecting adversarial examples [246, 247, 248], and denoising or reconstruction [249, 250].

Most of these techniques for improving data efficiency and adversarial robustness study the problem from the perspective of model, optimizer and data. Relatively little research has been conducted on the labels themselves or more specifically, their representation. [251] augmented categorical labels with a new NULL class to allow the model to classify and reject perturbed examples. [252] utilizes model distillation [253] for adversarial defense. [254] further augment the label used to train the distilled model with the predictive uncertainty from the original model. Nevertheless, the method of [251] requires adversarial examples to train on while that of defensive distillation has been shown to be vulnerable to substitute model black-box attacks [255].

Label Smoothing The closest approach to our work is Label Smoothing (LS) [256]. Here we highlight key differences between our approach and LS. First, LS applies to discriminative outputs where both correct and incorrect class information are presented during training, while our output is generative and only correct class information is presented. That is, our outputs are not a distribution over classes. Although LS has been shown to improve adversarial robustness [257], it has not been shown to be effective for low-data learning. As we will show in our experiments, LS does not help when the amount of training data is limited, while our label representations lead to significant improvements. Therefore, our high-dimensional, high-entropy labels provide benefits beyond those provided by label smoothing.

E.3 Beyond Accuracy: Emergence of Robustness and Efficiency

It is well-known that deep neural networks with similar accuracy on the same task may perform very differently under different evaluation scenarios. Additionally, real-world applications rely on more considerations than just accuracy. Robustness and data efficiency are two practical challenges for deep neural networks. We test the emergence of those properties under various label representations.

E.3.1 Robustness under Adversarial Attacks

We evaluate the robustness of all the trained models using the fast gradient sign method (FGSM) [258] and the iterative method [241] across multiple widely used convolutional networks. We choose these attacks because their adversarial images I_{adv} are usually indistinguishable from the original images I or do not significantly affect human evaluation, but they can be very challenging for neural networks to correctly classify. When a loss function J is involved in generating the adversarial images, we use the cross-entropy loss for the text model and the smooth L1 loss for the speech model.

FGSM is a fast one-step attack that generates an adversarial image by adding a small adversarial perturbation to the original image. The perturbation is based on the gradient of the loss with respect to the original image. The maximum magnitude of the perturbation is maintained by ϵ:

$$||I - I_{\text{adv}}||_{\infty} \leq \epsilon. \tag{E.1}$$

We test both untargeted and targeted versions of FGSM. The untargeted attacks increase the loss between the predicted class and the true class Y_{true}:

$$I_{\text{adv}} = I + \epsilon \cdot \text{Sign}(\nabla_I J(I, Y_{\text{true}})), \tag{E.2}$$

whereas the targeted attacks decrease the loss between the predicted class and a target class Y_{target}:

$$I_{\text{adv}} = I - \epsilon \cdot \text{Sign}(\nabla_I J(I, Y_{\text{target}})). \tag{E.3}$$

We choose a random incorrect class as the target class for each input image, and the same target classes are used to test different models. All I_{adv} are normalized after the perturbation.

Iterative Method As an extension to FGSM, the iterative method applies multiple steps of gradient-based updates. In our experiments, we initialize the adversarial image I_{adv} to be the original image I so that $I^0_{\text{adv}} = I$. Then we apply FGSM for 5 times with a small step size $\alpha = \epsilon/5$. The untargeted update for each iteration becomes

$$I^{N+1}_{\text{adv}} = \text{Clip}_{I,\epsilon}\left\{I^N_{\text{adv}} + \alpha \cdot \text{Sign}(\nabla_I J(I^N_{\text{adv}}, Y_{\text{true}}))\right\}, \tag{E.4}$$

and the targeted update becomes

$$I^{N+1}_{\text{adv}} = \text{Clip}_{I,\epsilon}\left\{I^N_{\text{adv}} - \alpha \cdot \text{Sign}(\nabla_I J(I^N_{\text{adv}}, Y_{\text{target}}))\right\}, \tag{E.5}$$

where $\text{Clip}_{I,\epsilon}$ denotes clipping the total perturbation $I_{\text{adv}}^{N} - I$ in the range of $[-\epsilon, \epsilon]$. We use the same targeted classes from FGSM for the evaluation on the iterative method.

E.3.2 Learning Efficiency with Limited Data

We take the most straightforward approach to evaluate data efficiency. We start with only 1% of the original training data. We always use the full amount of testing data. We then gradually increase the amount of training data to 2%, 4%, 8%, 10%, and 20% of the original to perform extensive multi-scale evaluation.

E.4 Experimental Setup

Dataset We evaluate our models on the CIFAR-10 and CIFAR-100 datasets [224]. We use the same training, validation and testing data split ($45,000/5,000/10,000$) for all of our experiments. We also keep the same random seed for data preprocessing and augmentation. Therefore, we present apple to apple comparisons for all label representations.

Speech Label Generation We generate the speech labels shown in Figure E.1 by following the standard practice as in recent works [259, 260]:

- We first generate the English speech audio automatically with a text-to-speech (TTS)[1] system from the text labels in the corresponding dataset. Therefore, all the speech labels are pronounced consistently by the same API with the same parameters for controlled experiments. We leave the exploration of different languages and intonations as future work.

- We save each audio file in the WAVE format with the 16-bit pulse-code modulation encoding, and trim the silent edges from all audio files.

- Since different speech signals may last for various lengths, we preprocess[2] each speech label to generate a Log Mel spectrogram to maintain the same dimension. We use a sampling rate

[1]https://cloud.google.com/text-to-speech/
[2]https://librosa.github.io/

of $22,050$ Hz, 64 Mel frequency bands, and a hop length of 256. Another advantage of this preprocessing into spectrograms is that we can then utilize convolutional neural networks as the speech decoder to reconstruct our speech labels. Meanwhile, we convert the amplitudes to the decibel scale.

- Finally, the spectrograms are shaped into a $N \times N$ matrix with values ranging from -80 to 0, where N is double the dimension of the image input. Our resulting speech spectrogram can be viewed as a 2D image.

Given the first step in this procedure, note that for a given class (e.g. "bird") there is only one corresponding spectrogram. Therefore, the improved robustness we observe is not a result of any label-augmentation.

Other Labels For a deeper understanding of which properties of speech labels introduce different feature learning characteristics, we replicate all the experiments using the following high-dimensional label variants. We also show visualizations for all the label variants in Figure E.1.

- *shuffled-speech* We cut the speech spectrogram image into 64 slices along the time dimension, shuffle the order of them, and then combine them back together as an image. Although the new image cannot be converted back to a meaningful audio, it preserves the frequency information from the original audio. More importantly, this variant does not change the entropy or dimensionality of the original speech label.

- *constant-matrix* In contrast, the constant matrix represents another extreme of high-dimensional label representation, with all elements having the same value and zero entropy. The constant-matrix label has the same dimension as the speech label, and values are evenly spaced with a range of 80 (which is also the range of the speech labels).

- *Gaussian-composition* We obtain this representation by plotting a composition of 2D Gaussians directly as images. Each composition is obtained by adding 10 Gaussians with uniformly sampled positions and orientations.

- *random/uniform-matrix* We also adopt a matrix with same dimensions as the above high-dimensional labels by randomly sampling from a uniform distribution. We additionally construct a uniform random vector with low dimensionality to inspect whether dimensionality matters. Throughout the paper, we will use random matrix and uniform matrix interchangeably to refer to the same representation.

- *BERT embedding* We obtain the BERT embedding from the last hidden state of a pre-trained BERT model [261, 262]. We remove outlines larger than twice the standard deviation and normalize the matrix to the same range of our other high-dimensional labels. The BERT embedding results in a 64×64 matrix.

- *GloVe embedding* Similarly, we directly use the pretrained GloVe [263] word vectors.[3] This results in a 50-dimensional label vector.

Models We take three widely used convolutional networks as our base model: VGG19 [221], ResNet-32 and ResNet-110 [222]. Here we define that a categorical classification model has two parts: an image encoder I_e and a category (text) decoder I_{td}. Traditionally, I_{td} represents fully connected layers following various forms of convolutional backbones I_e. Finally, a softmax function is applied to the output from the last layer of I_{td} to convert the predictions to a categorical probability distribution. The prediction is retrieved from the class with the highest probability.

Similarly, we define that the models for our high-dimensional labels consists of two parts: an image encoder and a label decoder I_{ld}. Throughout the paper, we use the same image encoder but replace the category decoder I_{td} with one dense layer and several transpose convolutional layers as I_{ld}. All the layers inside I_{ld} are equipped with batch normalization [264] and leaky ReLU with a 0.01 negative slope.

Overall, the majority of parameters of the network comes from the image encoder which is shared across both categorical labels and other label representations. The decoder for high-dimensional labels increases the number of parameters by a very limited amount (see Appendix

[3] https://nlp.stanford.edu/projects/glove/

for all the numbers). Thus, our experiments are well-controlled with respect to the number of parameters.

Learning We train the categorical model to minimize the traditional cross-entropy objective function. We minimize Equation E.6 for other models with high-dimensional labels. We use the smooth L1 loss (Huber loss) L_s shown in Equation E.7. Here, y_i is the predicted label matrix while s_i stands for the ground-truth matrix.

$$\min_{\theta_s} \sum_i \mathcal{L}_s(y_i, s_i) \tag{E.6}$$

$$\mathcal{L}_s(y_i, s_i) = \begin{cases} \frac{1}{2}(y_i - s_i)^2, & \text{if } |y_i - s_i| \leq 1 \\ |y_i - s_i|, & \text{otherwise} \end{cases} \tag{E.7}$$

We optimize all the networks using stochastic gradient descent (SGD) with back-propagation, and we select the best model based on the accuracy on the validation set.

Evaluation For categorical models, a prediction is considered correct if the class with the highest probability indicates the same category as the target class. For high-dimensional labels, we provide two types of measurements. Given the model output, we select the ground-truth label that minimizes the distance to the output. We refer to this as the "nearest neighbor" (NN) choice. The other criteria is to check whether the smooth L1 loss is below a certain threshold. We use Amazon Mechanical Turk [265] to validate that generated speech below our threshold is correctly identifiable by humans. In our experiments, we find 3.5 is a reasonable threshold. The human evaluations on the speech label demonstrate that our metric captures both the numerical performance and the level of interpretability of the generated speech output. Note that we mainly rely on the NN method for evaluation and only refer to the threshold method to demonstrate the qualitative results from the speech label.

E.5 Results and Discussion

E.5.1 Do All the Models Learn to Classify Images?

We report the classification accuracy for all the labels in Table E.1. Speech labels, shuffled-speech and Gaussian-composition labels achieve comparable accuracy with the traditional categorical labels, while the constant-matrix performs slightly worse than these representations. This suggests that the constant-matrix label is harder to train with. We verify this observation by visualizing the training curves for all label representations on CIFAR-100 with the ResNet-110 image encoder in Figure E.2. The training curves show that the constant-matrix model takes longer to converge than the others, and converges to a higher loss.

Labels	CIFAR-10 Accuracy (%)			CIFAR-100 Accuracy (%)		
	VGG19	ResNet32	ResNet110	VGG19	ResNet32	ResNet110
Category	92.82 ± 0.08	92.34 ± 0.25	93.23 ± 0.29	70.98 ± 0.10	68.05 ± 0.72	70.03 ± 0.49
Speech Threshold	91.97 ± 0.17	91.90 ± 0.04	92.44 ± 0.11	69.13 ± 0.75	61.08 ± 0.27	67.88 ± 0.16
Speech NN	92.12 ± 0.18	92.34 ± 0.01	92.73 ± 0.08	70.27 ± 0.61	64.74 ± 0.36	69.51 ± 0.25
Shuffle Threshold	92.27 ± 0.16	91.49 ± 0.22	92.44 ± 0.05	67.04 ± 0.41	51.95 ± 0.85	63.00 ± 1.45
Shuffle NN	92.64 ± 0.19	92.72 ± 0.20	92.92 ± 0.24	70.88 ± 0.31	64.23 ± 0.80	69.41 ± 0.66
Composition Threshold	91.53 ± 0.24	91.49 ± 0.22	92.36 ± 0.02	68.06 ± 0.28	60.54 ± 0.54	67.17 ± 0.40
Composition NN	91.94 ± 0.22	92.39 ± 0.17	93.07 ± 0.03	70.20 ± 0.23	66.72 ± 0.44	70.62 ± 0.20
Constant Threshold	88.27 ± 0.63	88.50 ± 0.25	89.27 ± 0.15	62.99 ± 0.11	55.70 ± 0.36	58.78 ± 0.18
Constant NN	88.33 ± 0.65	88.61 ± 0.27	89.37 ± 0.13	34.29 ± 2.40	19.00 ± 1.19	24.46 ± 3.70

Table E.1: Classification accuracy on CIFAR-10 and CIFAR-100 for all label representations. Speech labels, shuffled speech labels, and composition of Gaussian labels all achieve comparable accuracies with categorical labels. Constant matrix labels perform slightly worse than the others.

E.5.2 Feature Robustness

In order to evaluate the robustness of the models, we take all the trained models (Table E.1) as the starting points for adversarial attacks with the FGSM and the iterative method. We apply an ϵ from 0 to 0.3 with a 0.05 increment to the normalized images by following the original FGSM setup [258] and test the model accuracy for each ϵ value. We only run attacks on the original correctly classified images and mark the original misclassified images as incorrect when we compute the

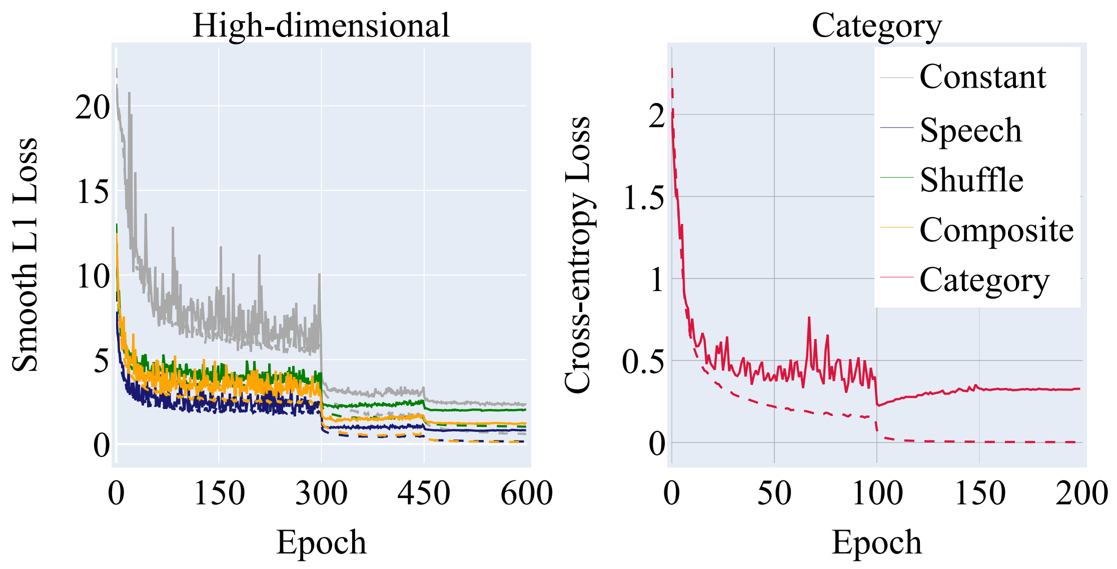


Fig. E.2: Training and validation losses on CIFAR-10 dataset with ResNet-110 image encoder for models with the speech / shuffled speech / composition of Gaussians / constant matrix labels (left) and categorical labels (right). All of the models are trained to converge. The model trained with constant matrix labels converges slower than models trained with other high dimensional labels.

accuracy for all ϵ values. We provide the accuracy computed on the subset of test images that are initially correctly classified in the Appendix, though the ranking among different models remains the same.

Figure E.3 shows the test accuracy under various attacks. Although the accuracy of all models decreases as the attack becomes stronger (larger ϵ), the models with speech labels, shuffled speech labels, and composition of Gaussian labels perform consistently much better than the models with traditional categorical labels across all three image encoders, for all types of attacks, and on both CIFAR datasets. Uniform random matrix labels perform similarly well in this setting (see the Appendix for details). Interestingly, models with constant matrix labels perform worse than all other models with high-dimensional labels, suggesting that there are some inherent properties that enhance model robustness beyond simply high dimensionality.

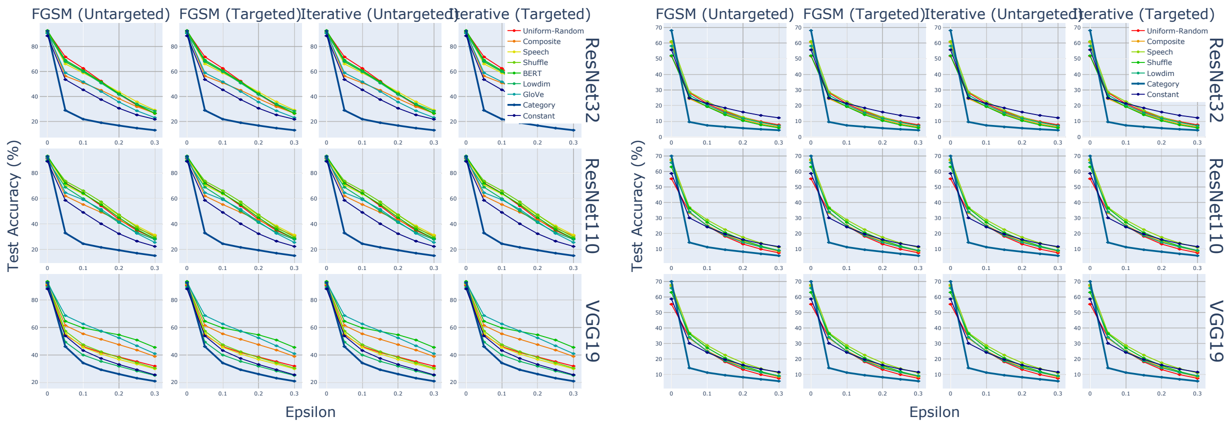


Fig. E.3: Test accuracy under adversarial attacks on CIFAR-10 (left four columns) and CIFAR-100 (right four columns). The accuracy evaluated by the threshold and the nearest neighbor is plotted in solid and dotted lines respectively. We show the results of targeted and untargeted FGSM and iterative method on three image encoders with three random seeds. The horizontal axis indicates the strength of different attacks.

E.5.3 Feature Effectiveness

With the CIFAR-10 dataset, we train models with various label representations using 1%, 2%, 4%, 8%, 10%, and 20% of the training data. For each amount of data, we train with the VGG19, ResNet-32, and ResNet-110 image encoders with five different random seeds. To conduct controlled experiments, we use the exact same training procedures and hyperparameters as the full-data experiments, so that the only difference is the amount of training data. All models are evaluated on the same validation set and test set. Figure E.4 reports the test accuracy. Similar to the results from the robustness evaluation, speech labels, shuffled speech labels, composition of Gaussian labels, and uniform random labels achieve higher accuracies than the models with categorical labels for both VGG19 and ResNet-110, and comparable results for ResNet-32. The results demonstrate that the speech models are able to learn more generalizable and effective features with less data. This property is extremely valuable when the amount of training data is limited.

Additionally, our results suggest that label-smoothing does not provide further benefits when the amount of training data is limited, as discussed above. Lastly, the performance of the models trained on constant matrix labels is consistent with that in the robustness experiment: it performs worse than all other high-dimensional labels. We provide further analysis in the next section.

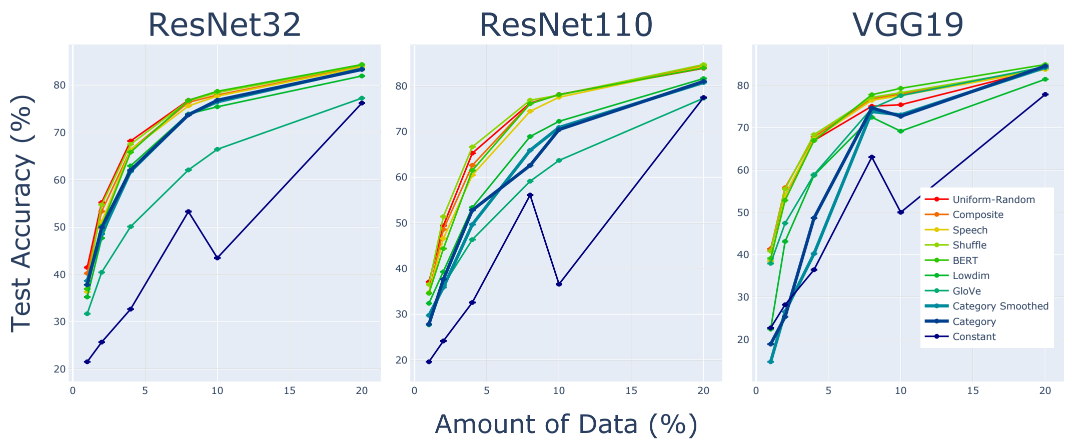


Fig. E.4: Test accuracy when limited training data is available. Accuracy is computed using the nearest-neighbor method

E.5.4 What is Special about Audio Labels?

Our experiments for robustness and data efficiency suggest that high-dimensional labels hold some interesting inherent property beyond just high-dimensionality that encourage learning of more robust and effective features. We hypothesize that high-dimensional label representations with high entropy provide stronger learning signals which give rise to better feature representations.

To verify our hypothesis, we measure several standard statistics over various label representations, shown in Table E.2. Specifically, we measure the normalized L1 and L2 distance between pairs of labels for each representation. We further measure the entropy for each individual label.

Interestingly, although the Manhattan and Euclidean distance between pairs of labels do not show any particularly useful patterns, the average entropy of the speech labels, the shuffled speech labels, and composition of Gaussian labels are all higher than that of the constant matrix and original categorical labels. The ranking of the entropy between these two groups exactly matches the performance in our robustness and data efficiency experiments shown in Figure E.3 and Figure E.4. This correlation suggests that high dimensional labels with high entropy may have positive

Label Types	CIFAR-10			CIFAR-100		
	Entropy	L1 Distance	L2 Distance	**Entropy**	L1 Distance	L2 Distance
Category	0.47 ± 0.00	2.00 ± 0.00	1.41 ± 0.00	0.08 ± 0.00	2.00 ± 0.00	1.41 ± 0.00
Constant	0.00 ± 0.00	26.07 ± 15.72	26.07 ± 15.72	0.00 ± 0.00	21.76 ± 15.16	21.76 ± 15.16
Speech	**11.35** ± 0.35	23.80 ± 5.16	12.95 ± 2.70	**11.37** ± 0.32	21.45 ± 5.53	13.15 ± 2.19
Shuffle	**11.35** ± 0.35	35.29 ± 2.18	18.87 ± 1.53	**11.37** ± 0.32	34.40 ± 2.59	17.81 ± 1.24
Composite	**12.00** ± 0.00	24.13 ± 3.36	19.41 ± 1.47	**12.00** ± 0.00	25.75 ± 4.72	20.60 ± 2.96
BERT	**11.17** ± 0.00	5.71 ± 0.94	2.06 ± 0.24	**11.17** ± 0.00	7.89 ± 2.84	2.63 ± 0.70
GloVe	**5.64** ± 0.00	7.35 ± 1.69	1.30 ± 0.31	**5.64** ± 0.00	5.62 ± 0.90	0.99 ± 0.16

Table E.2: Different basic statistics of all types of label representations. Labels that encourage more robust and effective feature learning also have higher entropy than other label forms.

impacts on robustness and data-efficient training.

We further validate that the benefits come from both high dimensionality and high entropy by training a model with low-dimensional and high-entropy labels. We generated these labels by sampling from a uniform distribution, following the same procedure as the uniform-random matrix label described previously. We found that while models trained with this label perform comparably to ones trained with high-dimensional high-entropy labels in terms of adversarial robustness (see Appendix), high-dimensional and high-entropy label models outperform the low-dimensional high-entropy model in terms of data efficiency, as shown by the curve "Low-dim" in Figure E.4. We find a similar result for categorical models trained with label smoothing, which has been previously shown to improve adversarial robustness [257]. In fact, high dimensionality is a prerequisite for high entropy because the maximum entropy is limited by the dimensionality of the label.

Note that the model trained with label smoothing uses the standard cross-entropy loss, meanwhile the low-dimensional high-entropy model is trained with the Huber loss. As a result, we argue that the loss is not responsible for the improved performance of models trained with high-dimensional, high-entropy labels.

E.5.5 Visualizations

Throughout the training process, we visualized the learned features immediately after the image encoder layer of ResNet-110 with t-SNE [266], both for audio and categorical models on CIFAR-10 test set. The results are shown in Figure E.5. We observe that the embedding of the learned features evolve as the training progresses. Compared with the feature embedding of the categorical model, the embedding of the speech models shows the formation of clusters at earlier stages of training. More separated clusters are also obtained towards convergence. We provide further visualizations and Grad-CAM interpretations in the Appendix.

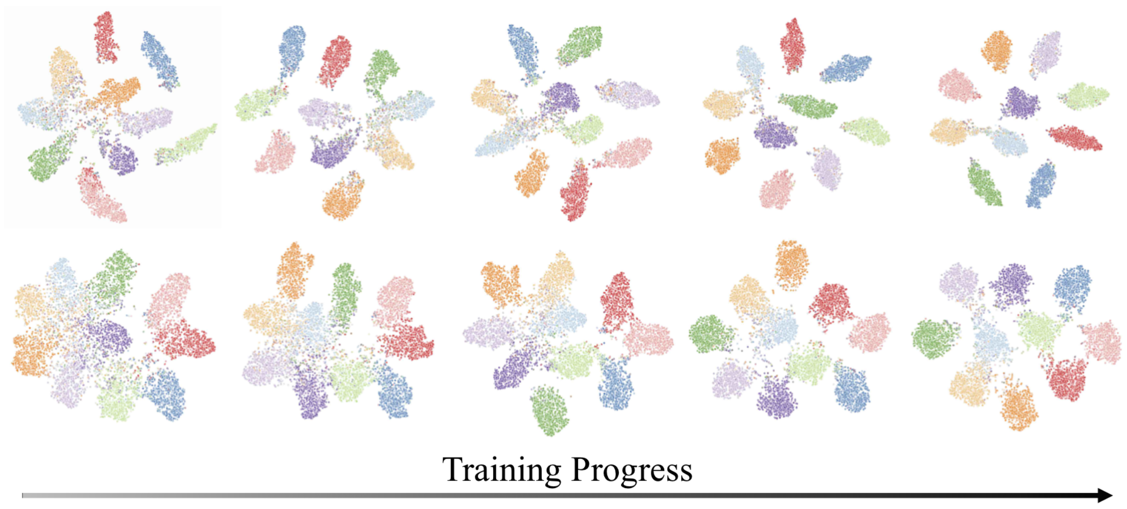


Fig. E.5: T-SNE progression for speech (top row) and categorical (bottom row) models with ResNet-110 image encoder. From left to right, the plot shows 10%, 30%, 50%, 70%, and 100% progress in training. The speech model develops distinctive clusters at an earlier stage and has better separated clusters overall.

E.6 Threshold Validation

We deployed a large-scale study on Amazon Mechanical Turk[4] to validate our choice of 3.5 as a classification threshold for the speech model.

In particular, we asked workers to listen to the outputs of the speech model and choose from the set of classes (with a "None" class for unintelligible output) the one which best fits the output. We

[4]https://www.mturk.com

assigned 3 workers to evaluate each output from the VGG19 speech model on CIFAR-10 test set. We chose a restrictive selection criteria to ensure maximum quality of responses. Only workers with a $\geq 99\%$ approval rating and at least $10,000$ approvals were selected.

To measure agreement between humans and our speech model, for each sample in the test set we determine the decision made by the model using our pre-selected threshold (loss < 3.5 is correct, while loss ≥ 3.5 is incorrect). Then we compare these decisions to those of the human workers. When we count each of the three workers independently, we find that humans agree with the model 99.4% of the time. When we take a majority vote (2/3 humans agreeing) we find that humans agree with the model 99.8% of the time. We conclude that 3.5 is a reasonable threshold for evaluating the model.

E.7 Subset Robustness

Additional robustness evaluation is computed using the subset of the test images that are initially correctly classified by the models without any adversarial attacks (Figure E.6). All test accuracy starts at 100% and decreases with stronger attacks. The strength of the attacks is described by the value of epsilon.

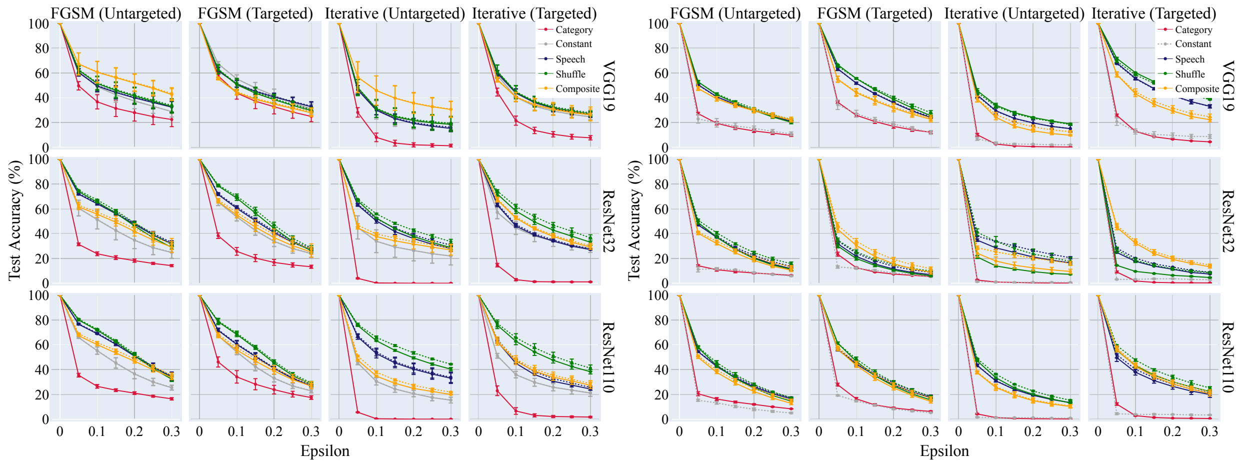


Fig. E.6: Test accuracy under adversarial attacks on CIFAR-10 (left four columns) and CIFAR-100 (right four columns) for the initially correct subset of the test images. The accuracy evaluated by the threshold and the nearest neighbor is plotted in solid and dotted lines respectively. The general trend from the subset is similar to that from the full test set.

E.8 Additional Results on Robustness Evaluation

We here include the full results on robustness evaluation on CIFAR-10 dataset in Figure E.7.

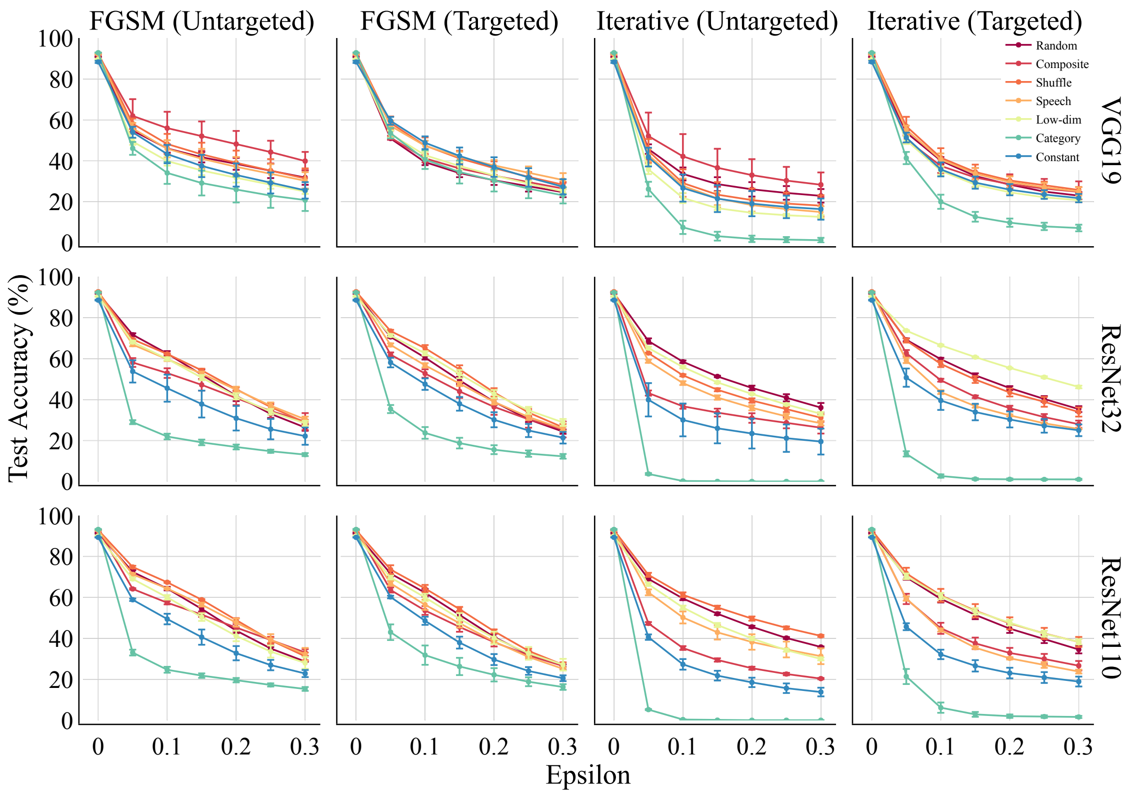


Fig. E.7: Full results of the robustness evaluation on CIFAR-10

E.9 Hyperparameters

E.9.1 Implementation Details

We train the categorical models for 200 epochs with a starting learning rate of 0.1, and decay the learning rate by 0.1 at epoch 100 and 150. The high-dimensional models are trained for 600 epochs with the same initial learning rate, and we drop the learning rate by 0.1 at epoch 300 and 450. All models are trained with a batch size of 128 using the SGD optimizer with 0.9 momentum and 0.0001 weight decay. One exception is when train categorical models with the VGG19 image encoder, we use a larger weight decay, 0.0005. We implement our models using PyTorch [267]. All experiments are performed on a single GeForce RTX 2080 Ti GPU. The limited-data experiments are conducted using the same settings as the full data experiments.

E.9.2 Architecture Parameters

We provide the parameter counts for the all models in Table E.3 and Table E.4. The majority of the parameters come from the image encoders. High-dimensional models have slightly more parameters than categorical models due to the high-dimensional label decoder (Table E.5).

Model	CIFAR-10		
	VGG19	ResNet32	ResNet110
Category	2×10^7	4.67×10^5	1.73×10^6
High-dimensional	2.01×10^7	5.80×10^5	$1,84 \times 10^6$

Table E.3: Total number of parameters of the category and high-dimensional models for CIFAR-10 dataset

Model	CIFAR-100		
	VGG19	ResNet32	ResNet110
Category	2.01×10^7	4.73×10^5	1.74×10^5
High-dimensional	2.02×10^7	5.80×10^5	1.84×10^5

Table E.4: Total number of parameters of the category and high-dimensional models for CIFAR-100 dataset

E.10 Dataset

To demonstrate the effectiveness of our proposed method, we evaluate our models on the CIFAR-10 and CIFAR-100 datasets [224]. For each dataset, we train different models on the same training set and evaluate the models on the same validation set using the same random seeds for fair comparisons. To preprocess the training images, we randomly crop them with a padding size 4 and perform random horizontal flips. All CIFAR images are normalized with mean $(0.4914, 0.4822, 0.4465)$ and standard deviation $(0.2023, 0.1994, 0.2010)$ of the training set.

Layer	Input	Output	Kernel	Stride	Padding
Dense	I_e out	64	-	-	-
ConvTranspose 2D	$64 \times 1 \times 1$	$64 \times 4 \times 4$	4×4	1×1	0
ConvTranspose 2D	$64 \times 4 \times 4$	$32 \times 8 \times 8$	4×4	2×2	1×1
ConvTranspose 2D	$32 \times 8 \times 8$	$16 \times 16 \times 16$	4×4	2×2	1×1
ConvTranspose 2D	$16 \times 16 \times 16$	$8 \times 32 \times 32$	4×4	2×2	1×1
ConvTranspose 2D	$8 \times 32 \times 32$	$1 \times 64 \times 64$	4×4	2×2	1×1

Table E.5: The architecture of the high-dimensional label decoder I_{ld}. The input dimension of the first dense layer is the dimension of the output of the image encoder I_e. The output of the last ConvTranspose2d layer is the target label.

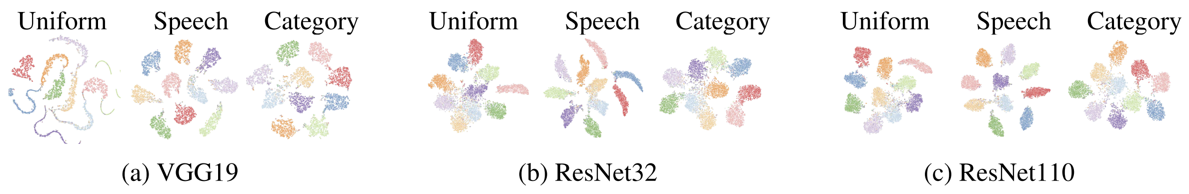


Fig. E.8: T-SNE of the best uniform (left), speech (middle), and categorical (right) models trained with the same random seed. Speech models show best separated clusters across all three models.

CIFAR-10 consists of 60, 000 images of size 32×32 uniformly distributed across 10 classes. The dataset comes with 50, 000 training images and 10, 000 test images. We use a 45, 000/5, 000 training/validation split.

CIFAR-100 also comprises 60, 000 images of size 32×32, but it has 100 classes each containing 600 images. The dataset is split into 50, 000 training images and 10, 000 test images. We randomly select 5, 000 images from the training images to form the validation set.

E.11 Visualizations

In addition to the progressive t-SNE plots we presented in the main context, we plot the embedding of all three types of image encoders of models trained on speech labels, categorical labels, and constant matrix labels in Figure E.8. We only show the results from the models with highest accuracy. Similarly, we observe that speech models have better separation of clusters. The feature embedding of the constant model is worse than that of the speech model, further confirming that the speech representation is unique in addition to its dimensionality.

Grad-CAM We visualize activations from beginning, intermediate, and final layers in the image encoders for both speech and categorical models. We see that the input activations for the speech model conform more tightly than those of the categorical model to the central object of each image at all three stages. This may suggest that the speech model learns more discriminative features than the categorical model. These features are also visually more easily understandable to humans.

E.12 Conclusion

We introduce a novel paradigm for the traditional image classification task by replacing categorical labels with high-dimensional, high-entropy matrices such as speech spectrograms. We demonstrate comparable accuracy on the original task with our speech labels, however, models trained with our speech labels achieve superior performance under various adversarial attacks and are able to learn in a more data efficient manner with only a small percentage of training data. We further study the inherent properties of high dimensional label representations that potentially introduce the advantages. Through a large scale systematic study on various label representations, we suggest that high-entropy, high-dimensional labels generally lead to more robust and data efficient training. Our work provides novel insights for the role of label representation in training deep neural networks.