Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Align-and-Repeat: One-Shot Visual Imitation Learning using Imagination

Author:
Murat Uzun

Supervisor:
Edward Johns

Submitted in partial fulfillment of the requirements for the MPhil degree in Computing of Imperial College London

Abstract

Visual imitation learning is a compelling framework that enables robotic agents to perform tasks using expert demonstrations. It is, however, currently too costly to collect high-quality expert behavior, as it requires manual kinesthetic teaching or tele-operation. Furthermore, as deep learning stays at the foundation of raw sensory input processing, it is possible to observe erroneous imitation behavior, especially when the operating conditions are not fully covered by the training data. In this work, we initially study the predictive uncertainty estimations in deep learning comparing the most commonly used methods in the literature from the robotic object manipulation context. Our empirical analysis exposes the involved challenges of uncertainty calibration and out-of-distribution robustness of neural network models. We then propose a practical method for one-shot visual imitation that enables robots to solve a number of everyday table-top manipulation tasks using a single expert demonstration. Our method does not assume any task-relevant prior knowledge and 3D object models. The expert demonstration, i.e., end-effector velocities, is collected together with the corresponding depth measurements using a calibrated wrist-cam. For each task, we divide the robot motion into two phases: the viewpoint alignment and the interaction. The proposed approach first learns to reconstruct an orthographic render of the workspace via an encoder-decoder type neural scene rendering model that is trained offline. Imagined full top-down render of the live scene, including the unexplored and occluded regions, is then used to guide the end-effector for task-relevant visual information collection. After sufficient visual exploration, the first frame of the live scene is aligned with the first frame of the demonstration scene via estimating a rigid transformation matrix, so that, we can simply repeat the expert actions. This allows executing complex behavior, i.e., 3D trajectory, without explicitly learning a policy. We conduct a number of manipulation experiments in simulation comparing our approach with similar baselines and illustrate its superior performance.

Statement of Originality

I hereby confirm that the content of this thesis is my own original work.

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.'

Acknowledgments

My sincere thanks goes to my supervisor Edward Johns. I am eternally indebted.

I also would like to acknowledge the support provided by Imperial College London via President's Scholarship Program, which made this research possible.

I would like to dedicate this thesis to my loving grandmothers Fethiye Uzun, Suzan Ulucenk, my great aunt Feriha Uzun, and my grandfather Ferit Uzun.

Contents

1 Introduction **1**

2 Background **5**
2.1 Deep Learning 5
2.1.1 Bayesian Foundations 8
2.1.2 Deep Learning as Inference 9
2.2 Markov Decision Process and Reinforcement Learning Framework . . 11
2.3 Imitation Learning 14

3 Related Work **16**
3.1 Core Vision 16
3.2 6D Pose Estimation 20
3.3 Active Perception 22
3.4 Robotic Grasping and Pick-and-Place 23
3.5 Imitation Learning 25
3.6 Sim2Real 27
3.7 Uncertainty Quantification 29

4 A Study on Policy Uncertainties **32**
4.1 Method 32
4.1.1 MC-dropout 32
4.1.2 Deep Ensembles 33
4.1.3 MIMO 34
4.2 Experiments 36
4.2.1 1D Regression Problem 36
4.2.2 Visual Imitation Problem 39

5 Align-and-Repeat **44**
5.1 Method 44
5.1.1 Demonstration Scene 45
5.1.2 Live Scene 45
5.2 Experiments 47
5.2.1 Setup 47
5.2.2 View-point Alignment 47
5.2.3 Guided Exploration 48
5.2.4 One-Shot Imitation 52

6 Conclusions **55**

Chapter 1

Introduction

Vision-based robotic object manipulation is a subject that has been studied extensively in the last decades. These systems are generically composed of a vision and a control module. Existing work has generally focused on: (1) training end-to-end visuomotor policies that maps the raw input images (possibly together with proprioceptive sensor readings) into actions, i.e. motor commands; (2) modular pipelines that consists of a perception, modeling and prediction, planning, and low-level control modules. The modular approaches enable us to leverage conventional computer vision, grasping, and configuration space planning algorithms.

End-to-end pipelines commonly utilizing the state-of-the-art deep reinforcement learning or imitation learning algorithms work better in contact-rich manipulation tasks, e.g., peg insertion which requires complex and high precision control policies. However, end-to-end pipelines lack the tractability, interpretability and require sample-inefficient training procedures mostly achieved with reward engineering. On the other hand, modular pipelines are easier to engineer and debug, more flexible, and more suitable for tasks with compound scenes containing various objects that need long-horizon planning. However, they suffer from the error propagation phenomena and may fail for the tasks that require below centimeter (cm) precision.

Demonstrating a behavior that solves the task at hand is usually simpler than scripting it. In robotics, visual imitation learning (VIL) can be described as the collection of algorithms that enables robots to acquire complex skills via utilizing expert demonstrations. Successful implementation of a VIL method enables us to have capable robotic agents without any task-specific programming. As powerful as this method sounds, there exists a number of challenges that should be cleared up before it is viable to deploy and ship robotic products that learn by imitation. Especially for domestic environments, it is required to develop algorithms with minimal 'teaching time' and engineering know-how.

Existing learning-based VIL approaches that simply use the raw visual inputs and provided expert actions, typically require a substantial amount of data, hence a prolonged amount of human-robot interaction time. Previous work mostly deals with this issue via (1) learning suitable prior policies via meta-learning [33, 58], (2) cat-

alyzing the real-world data collection and distribution [87, 143, 25], (3) utilizing simulation data with domain randomization [133, 8, 134, 59, 129], or (4) developing methods that carry self-supervision into action [12, 147]. Most of the recent body of research in VIL commonly depends on the feature extraction capabilities of black-box neural network models, mapping visual inputs to actions, without fully investigating the model behavior with the out-of-distribution (ood) cases. It is known that deep nets can learn highly discontinuous input-output mappings and could output counter-intuitive high-confidence false predictions [124, 36]. Furthermore, due to covariate-shift—a special case of ood robustness where input distribution changes and the output distribution that is conditioned on the input remains the same—, learned policies tend to struggle in open-set environments [121], and often require additional demonstrations [116] and fine-tuning to prevent silent failures.

In this work, first to better understand the predictive uncertainty estimations of deep net models and evaluate the most commonly used approaches in the literature, we compare existing uncertainty quantification (UQ) methods based on a 1D regression problem and across various image-to-action regression settings that are created in simulation, i.e., visual imitation learning. We show that the quality of the calibration highly depends on the utilized method and a number of hyper-parameters. Our results showcase the challenges involved and calls for further caution in safety-critical robotic deployments.

Following the discussions on uncertainty quantification (UQ), we introduce a practical one-shot imitation learning approach that fuses classical vision concepts and machine learning in a more interpretable and controllable form. We simply divide the robot motion that solves the task into two phases: view-point alignment phase and interaction phase. This modularization is analogous to the coarse-to-fine approach that is introduced in [62]. Before the expert demonstration, our method first follows a pre-scripted planar trajectory to collect an adequate number of depth measurements into its buffer to be able to create a detailed orthographic render, i.e., heightmap, of the robot workspace. Given a table-top scene with a single object of interaction, then we collect expert actions i.e., end-effector velocities, which could be provided using kinesthetic teaching or teleoperation. This point marks the end of the human effort. At testing, the robot needs to align itself with the object so that the the relative pose between the two is the same as the start of the expert demonstration, so that the robot can replay the provided trajectory. To achieve this alignment, our method first creates a quantitatively precise orthographic render of the scene, where the exploration is guided by a neural-scene-rendering model – that predicts the top-down render of the workspace including the unexplored parts. This neural network model is trained offline via a dataset that is created based on the heightmap of the demonstration scene through a scripted randomization procedure. After having created orthographic renders of both the live and the demonstration scenes, we simply estimate the transformation that aligns the relative pose between the end-effector and the object in the live scene with the corresponding relative pose at the initial time step of the expert interaction – meaning that after we exe-

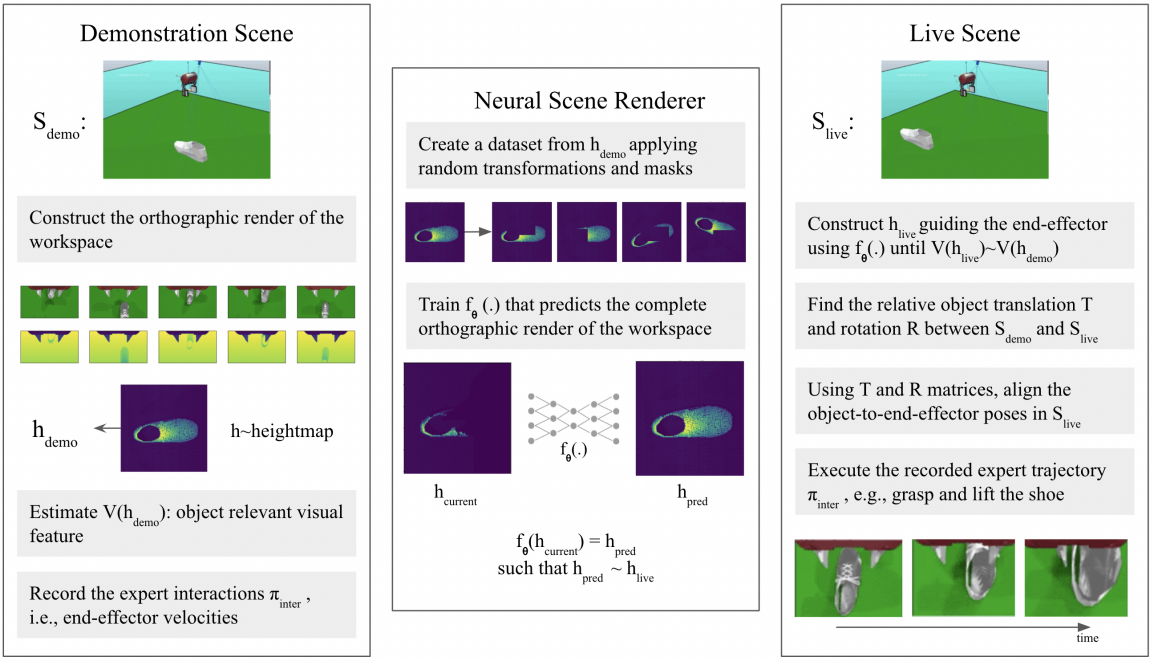


Figure 1.1: Overview of the align-and-repeat method.

cute the estimated transformation, our vision sensor that located at the wrist cam should capture the exact same observation that was previously captured at the initial time of the expert interaction. After the viewpoint alignment, we directly repeat the recorded expert actions, i.e. interaction policy, and complete the task. We show the effectiveness of the proposed approach in number of every-day manipulation tasks in simulation. Overview of the method is illustrated in Figure-1.1.

Overall, this thesis addresses two important open problems in robot learning for manipulation. First and foremost, robots should be useful to all of the public, independent of the users' robotics or software engineering know-how. In other words, any type of novel skill should be taught with a minimal amount of effort and human-robot interaction time. Most, if not all, of the proposed algorithms in robot learning research, require significant engineering (either in simulation or real world), prior task knowledge, experimentation design, etc. No one in a regular domestic setting would want to interact with a robot providing hours-long expert trajectories for imitation learning, re-start the scene for episodic reinforcement learning algorithms, or carefully watch the self-supervised training to ensure everything is alright during self-supervision. Second, the policies should be robust to previously unseen or even completely novel scenarios. This comes as a natural requirement considering the complexity and diversity of real-world scenes given that a silent failure could cause harm to the robot's environment and itself. In brief, to be able to achieve global-scale general-purpose robotic deployments, we need practical, effective, and robust solutions.

This document is structured as follows: We first provide the background on the used

technologies, methods, and frameworks, e.g., deep learning, Markov decision processes, Bayesian inference, etc. We review the relevant literature on robotic object manipulation, mostly focusing on the recent work. We then present our analysis on the predictive uncertainty estimations of neural network models, specifically studying a simple 1D toy regression problem, and a simulated visual imitation learning task. Then, we introduce our align-and-repeat approach—providing the details of the method and its performance against a number of baselines. Finally, in the conclusions section, we discuss the implications of the results, and scope for further research.

Chapter 2

Background

2.1 Deep Learning

Deep learning is a programming paradigm that is composed of numerous computational layers (of differentiable units called neurons) that allows learning useful representations of data [73]. Compared to traditional machine learning techniques that generally require careful feature engineering, deep learning has shown to be powerful in extracting complicated patterns from raw data by back-propagating errors [117] using gradient-based optimizers, e.g., stochastic gradient descent (SGD), Adam, Adagrad [68, 30], etc.

A single neuron - the main building block of neural nets - initially computes the weighted sum of the inputs and then applies a non-linear activation rule. There exist different types of deep architectures that serve different goals. Multilayer perceptrons (MLPs) are feed-forward networks that define a non-linear, parametrized, and continuous mappings from an input vector $\boldsymbol{x}$ to an output vector $\boldsymbol{y}$ through a forward pass, $\boldsymbol{y} = f_\theta(\boldsymbol{x})$. MLPs are composed of the input neurons, layers of hidden neurons, and the output neurons. Network parameters are calibrated by evaluating the model performance given the labeled training data (that ideally represent the noisy version of the real mapping function). An MLP can be trained to perform regression and classification tasks. An illustration of an MLP can be seen in Figure 2.1.

Neural networks are trained using a data set $\mathcal{D} = \{\boldsymbol{x}^n, \boldsymbol{y}^n\}$, minimizing a pre-defined cost function by altering the model parameters, i.e., weights and biases of each neuron. Cost function (or objective function) essentially measures how close the model outputs $f_\theta(\boldsymbol{x}^n)$ are to the target values $\boldsymbol{y}^n$. One of the most common cost function for regression problems is called 'Mean Square Error (MSE) Loss' (sometimes referred to as l_2 loss) which is defined as:

$$J_{l_2}(\boldsymbol{\theta}) = \frac{1}{N} \sum_n \sum_i \left(y_i^n - f_i(\boldsymbol{x}^n, \boldsymbol{\theta})\right)^2 \tag{2.1}$$

where $n = 1, \ldots, N$ represents the unique data points in the data set, N is the total number of items in training data, and subscript i implies the ith element of a vector,

e.g., $i = 1, \ldots, M$ assuming $y \in \mathbb{R}^{M \times 1}$. Loss minimization is achieved via evaluating the gradient of $J(\boldsymbol{\theta})$ using the chain-rule based back-propagation algorithm [117]. Note that, a small number of large outliers can effect the calculated MSE value significantly. Often, the cost function is modified including regularization terms, e.g., weight decay is given by

$$\lambda(\boldsymbol{\theta}) = \alpha \sum_{j} \theta_j^2 \tag{2.2}$$

where α is a hyper parameter widely referred as weight decay rate. Similarly, mean absolute loss (l_1 loss) – which is more robust towards the outliers in the data set – is given by:

$$J_{l_1}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n} \sum_{i} |y_i^n - f_i(\boldsymbol{x}^n, \boldsymbol{\theta})| \tag{2.3}$$

For binary classification problems, on the other hand, the binary cross-entropy loss (BCE) function is traditionally used. It is defined as

$$J_{xent}(\boldsymbol{\theta}) = -\sum_{n} [y^n \log(f_\theta(\boldsymbol{x}^n)) + (1 - y^n) \log(1 - f_\theta(\boldsymbol{x}^n))] \tag{2.4}$$

where $y^n = \{0, 1\}$. Note that, BCE loss can be easily used for multi-class classification problems using one-hot encoding. And often, the activation units of the output layer is highly interconnected with the chosen cost function.

Convolutional neural networks (CNNs) [74], are also a feed-forward type of architecture and mostly used with digital images. CNN architectures leverage the correlation among nearby pixels. In machine learning terms, convolution, refers to sliding a kernel/filter on the input matrix outputting the summation of the element-wise multiplications of the filter weights and the patch of the input matrix that the filter currently sees. Therefore, each output unit of the convolution operation is only affected by the input units that overlap with the kernel. Collection of these output

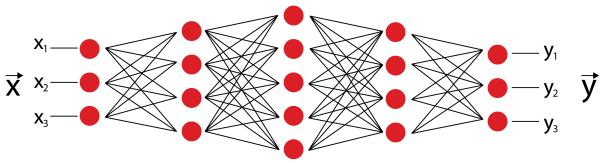


Figure 2.1: Illustration of an MLP architecture

units are called feature maps. Sliding a kernel is an attempt at finding local patterns. See Figure 2.2 for a schematic illustration of a CNN. CNN architectures played a significant role in deep learning revolution started by the success of AlexNET [70] in 2012 ImageNet object recognition challenge.

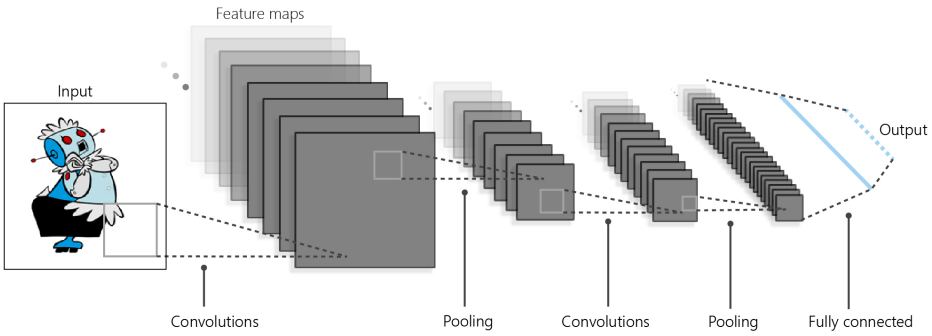


Figure 2.2: Illustration of a regular/shallow CNN architecture designed for image classification (adapted from [75]).

Another really well-studied neural network architecture is called Autoencoder (AE) which aims to learn the identity function to reconstruct its inputs through learning low-dimensional latent representations. It was initially proposed as a nonlinear generalization of principle component analysis (PCA) in [52]. This latent representation is sometimes referred to as code, $\boldsymbol{z}$. An AE is composed of two sequential networks, namely the encoder network f_θ that takes in the original high dimensional input and outputs the latent code, and the decoder network g_ϕ that reconstructs the input from the provided latent feature vector. Overall, a feed-forward operation outputs $\boldsymbol{x}' = g_\phi(f_\theta(\boldsymbol{x}))$ where we want $\boldsymbol{x}' \sim \boldsymbol{x}$. Thus we would want to penalize the network parameters if the reconstruction is dissimilar from the input. An illustration of an AE architecture is shown in Figure 2.3 where the dimension of the latent code $\boldsymbol{z}$ is smaller than the dimension of the input vector $\boldsymbol{x}$.

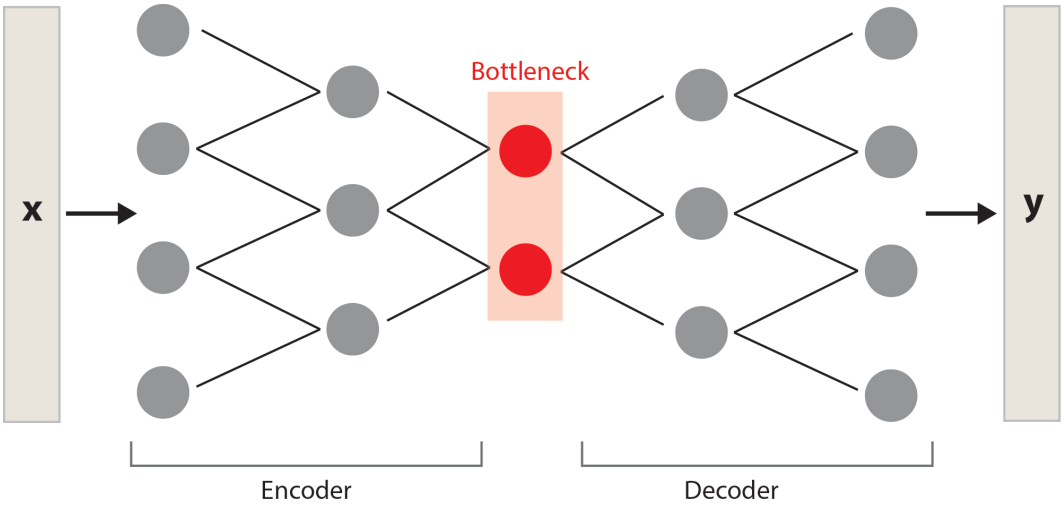


Figure 2.3: Illustration of an AE architecture

A typical loss function for an AE is as follows

$$J_{\theta,\phi} = \frac{1}{N}\sum_{n}[\boldsymbol{x}^n - f_\theta(g_\phi(\boldsymbol{x}^n))]^2 \tag{2.5}$$

Note that, it is possible to memorize all the data points independent of the dimension of the code/bottleneck with a large-enough capacity decoder. It is, however, possible to overcome this training data memorization circumstance via proper regularization schemes while still pushing the AE model to reconstruct its inputs. Some of these regularizations are implemented via enforcing sparsity constraints on the bottleneck [106, 86], requiring the code to be robust to small changes, and partially corrupting the input [136, 112], etc. Encoder-decoder type architectures are heavily used in image-to-image translation problems (such as image denoising) and segmentation tasks.

2.1.1 Bayesian Foundations

A measure of plausibility of the joint parameter vector θ given the data $\mathcal{D}$ can be expressed using the Bayes' theorem as [84]

$$p(\theta|\mathcal{D},\mathcal{H}) = \frac{p(\mathcal{D}|\theta,\mathcal{H})p(\theta|\mathcal{H})}{p(\mathcal{D}|\mathcal{H})} \tag{2.6}$$

where $p(\theta|\mathcal{H})$ represents the prior belief in the plausibility (probability density function) of θ which is generally based on intuition, $p(\theta|\mathcal{D},\mathcal{H})$ represents the posterior PDF after the newly acquired information, $p(\mathcal{D}|\mathcal{H})$ is called evidence (also referred to as marginal likelihood), and finally $p(\mathcal{D}|\theta,\mathcal{H})$ is termed the likelihood which depicts a compatibility between the data and the model parameters. It is regarded as a weighting function that assigns different weights to different model parameter

sets according to the observed data, and the adopted model assumptions. Finally, $\mathcal{H}$ simply depicts the full hypothesis space, e.g, a neural network architecture.

2.1.2 Deep Learning as Inference

The neural network learning process can also be interpreted from a probabilistic point of view [84]. Using the Bayesian framework, each parameter of a neural network can be represented with a prior distribution that is later updated given training data. As the exact computation of the marginal posteriors of the parameters is intractable, Monte Carlo methods or approximate variational methods are commonly utilized. Variational inference casts the posterior inference as an optimization problem. MCMC methods, on the other hand, approximate the predictive posterior, i.e., assuming single-input single-output models $p(y|x, D) = \int p(y|x, \theta)p(\theta|D)d\theta$, as $p(y|x, D) \sim 1/S \sum_{s=1}^{S} p(y|x, \theta^s)p(\theta^s)$. Deep learning literature has adopted a number of MCMC methods where the ideas came from statistical physics. MCMC methods require many samples, and typically demand a number of tricks to make them work well [139].

For example, a regression problem can be framed as follows for a Gaussian noise model. Say $y = f_\theta(\mathbf{x}) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \sigma)$, then our parametric model defines a conditional distribution of y given $\mathbf{x}^n$ as

$$p(y|\mathbf{x}^n; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - f_\theta(\mathbf{x}^n))^2}{2\sigma^2}\right) \tag{2.7}$$

Likelihood of the parameters given the data set $\mathcal{D}$ is given by

$$p(\mathcal{D}|\theta) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \prod_{n=1}^{N} \left(\exp\{-\frac{(y^n - f_\theta(\mathbf{x}^n))^2}{2\sigma^2}\}\right) \tag{2.8}$$

Given a set of training examples $\mathcal{D} = \{\mathbf{x}^n, y^n\}_{n=1}^N$, neural network parameters θ can be learnt by maximizing the likelihood of the parameters – equivalently by minimizing the negative log-likelihood of the parameters. Maximum likelihood estimate (MLE) of the parameters θ^{MLE} is given by:

$$
\begin{aligned}
\theta^{MLE} &= \underset{\theta}{\operatorname{argmax}} \left(\log p(\mathcal{D}|\theta)\right) \\
&= \underset{\theta}{\operatorname{argmin}} \left(-\log p(\mathcal{D}|\theta)\right) \\
&= \underset{\theta}{\operatorname{argmin}} \left(-\sum_n \log p(y^n|\mathbf{x}^n, \theta)\right) \\
&= \underset{\theta}{\operatorname{argmin}} \left(\frac{N}{2}\log 2\pi\sigma^2 + \sum_n^N \frac{(y^n - f_\theta(\mathbf{x}^n))^2}{2\sigma^2}\right) \\
&= \underset{\theta}{\operatorname{argmin}} \left(\text{constant.} + \frac{2\sigma^2}{N} J_{l_2}(\theta)\right)
\end{aligned}
\tag{2.9}
$$

As can be seen in Equation 2.9, minimizing the negative log likelihood (or maximizing the likelihood) is equivalent to minimizing the MSE loss. In brief, choosing a parametric model that defines the distribution $p_\theta(\mathbf{y}|\mathbf{x})$ automatically retrieves the cost function as [45]

$$
J(\theta) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{data}} \log p_\theta(\mathbf{y}|\mathbf{x}) \tag{2.10}
$$

Likewise, regularization term, e.g. weight decay as shown in Equation 2.2, can be interpreted as a Gaussian prior probability $p(\theta|\alpha) \sim \mathcal{N}(\mathbf{0}, 1/2\alpha\mathbf{I}^2)$. Using the Bayes' Rule (Equation 2.6), minimizing a neural network objective function with a regularizer can be equivalently described as finding the single best point estimate (MAP) of the parameter vector θ.

Similarly, multi-class classification problems - where the target y indicates one of M possible classes - can be framed using a softmax model with M output neurons, as follows:

$$
P(y = m|\mathbf{x}) = \frac{\exp\left\{f_{\theta m}(\mathbf{x})\right\}}{\sum_{k=0}^{M} \exp\left\{f_{\theta k}(\mathbf{x})\right\}} \tag{2.11}
$$

Computed posterior distributions of the parameters can then be used to evaluate the predictive uncertainty of the neural network model given a new input $\mathbf{x}^*$ as follows.

$$
\begin{aligned}
p(\mathbf{y}^*|\mathbf{x}^*, D) &= \int p(\mathbf{y}^*, \theta|\mathbf{x}^*, \mathcal{D}) d\theta \\
&= \int p(\mathbf{y}^*|\mathbf{x}^*, \theta) p(\theta|\mathcal{D}) d\theta
\end{aligned}
\tag{2.12}
$$

A general recipe for the Beyasian deep learning approach can be summarized as follows: We first specify the likelihood function, i.e., the output distribution and the neural network architecture. We then choose an approximate inference procedure,

e.g., variational inference, MCMC, or ensembles. And finally, at test time, we average the predictions either using samples from the posteriors, or analytically.

Aleatoric and epistemic uncertainty constitute two different classes of uncertainty [67]. Epistemic uncertainty (also sometimes referred as model uncertainty) refers to the missing piece of knowledge that could be acquired, for example, with more data or model refinement. Epistemic uncertainty grows as we move further away from the training distribution or as we decrease the size of the training dataset. Aleatoric uncertainty, on the other hand, refers to the inherent noise of the process, or the inevitable corruption of the input data that cannot be eliminated with additional information, e.g. sensor noise, label discrepancies. Aleatoric uncertainty of the model prediction is not affected by the out-of-distribution inputs. Epistemic and aleatoric uncertainties are represented respectively defining model weights and model outputs via probability distributions.

It should be noted that, deep learning models are known to output high confidence false predictions [124]. To be able to use these models for risk-sensitive real-life applications it is important to estimate the predictive uncertainties of these models. In other words, it is critical to know what our model does not know, e.g., we should be able to detect if a model is falsely overconfident. Good uncertainty estimates simply quantify when we can trust the model's predictions. Well calibration, in the classification setting, means to get confidence predictions that are closely correlated with the expected accuracy. For example 100 predictions with confidence of 0.8, should output the correct class 80% of the time. Deep model predictions can be calibrated yet inaccurate. There has been a great deal of recent effort to develop Bayesian learning tools for deep models as the Bayesian school of thought provides a structured way to calculate posterior distributions via assigning priors on the model parameters. Characterization of posterior distributions of the weight space get more challenging as the dimensionality of the weight space increases.

2.2 Markov Decision Process and Reinforcement Learning Framework

A Markov Decision Process (MDP) is a stochastic control process in which a decision maker, i.e., an agent, a policy, interacts with a random and fully visible environment. It is defined by a tuple $(S, \mu, A, P^a_{ss'}, \gamma, R)$ where S and A respectively represent the set of possible world states (state-space), and the set of possible actions that could be selected or executed on each time step (action-space). A state provides the complete information of the world, whereas an observation, $\mathbf{o}$, could contain only a partial description. An action could be discrete or continuous depending on the environment, e.g., in an Atari setting all the available actions are discrete whereas in real-world robotics actions and controls are continuous. μ depicts the initial state distribution over S. $P^a_{ss'}$ gives the distribution over the next state, s', given the current state $s \in S$ and the action $a \in A$ (also known as the transition operator and represents

the environment dynamics). $R : S \times A \to \mathbb{R}$ is the reward function, and depending on the environment, it could depend on the current state-action pair and the next state. $\gamma \in [0, 1]$ is called the discount factor that represents the value of the future rewards with respect to the immediate one. An illustration of an MDP process is shown in Figure 3.5.

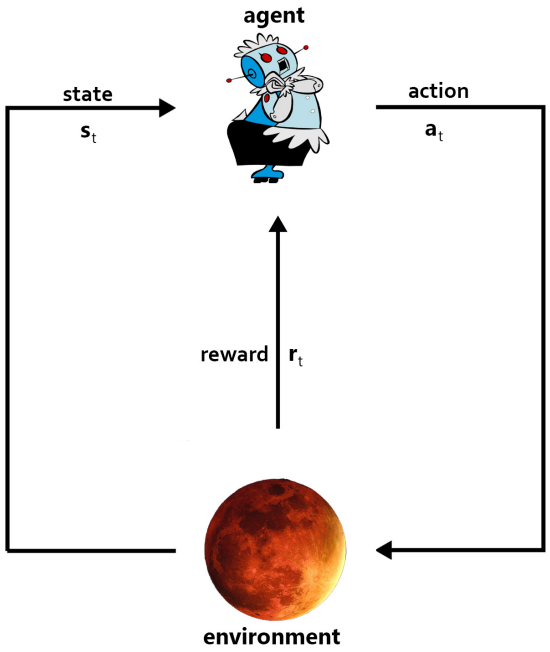


Figure 2.4: Illustration of a Markov Decision Process – the setting that an RL agent operates in.

In an MDP, the agent starts from the initial state s_0 that is sampled from the distribution μ and at each time-step t it has to select an action a_t. As a result of the selected action, the agent interacts with the environment and lands on a state s_{t+1} which is drawn from $P^a_{ss'}$ according to the environment dynamics. By repeatedly selecting from available actions, the agent follows a trajectory $\tau = (s_0, a_0, s_1, a_1, ...)$ and collects rewards, e.g., $R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ...$. Note that rewards could be stochastic, and could be represented in one of the following functional forms: current state $R(s_t)$; current state and action $R(s_t, a_t)$; or current state, action and the consecutive state $R(s_t, a_t, s_{t+1})$. In essence, the reward function specifies the target objective, i.e., the intention, of the agent [95].

The reinforcement learning (RL) problem can be posed as selecting/finding a policy $\pi : S \to A$ such that the expected return (sum of expected rewards) is maximized [125, 122]. The policy can be deterministic, $\mathbf{a}_t = \pi(\mathbf{s}_t)$, or stochastic - described by a conditional distribution $\pi(\mathbf{a}_t|\mathbf{s}_t)$. Note that, because of the Markov property (the future is independent of the past given the present), the optimal policy that outputs the optimal action sequence is only a function of the current state $\mathbf{s}_t$. If the agent's observation is not the same with the state of the environment, e.g., noisy location

estimations through vision sensors, then the process is called partially observable Markov decision process (POMDP). An illustration of POMDP is shown in Figure 2.5

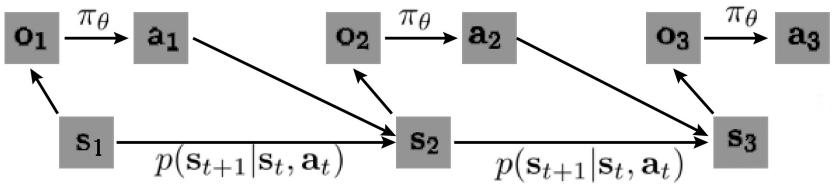


Figure 2.5: Graph representation of a POMDP.

Markov property can be formally defined as follows. A state $\mathbf{s}_t$ is a Markov state if and only if:

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t) = p(\mathbf{s}_{t+1}|\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, ...\mathbf{s}_t) \tag{2.13}$$

It should be noted that in RL setting there is only a reward signal and no supervisor is present. Agent's previous actions influence the reward it receives, and the feedback is not instantaneous.

If we have a deterministic policy $\pi(\mathbf{s})$, then for a given initial state $\mathbf{s}_0$ we would have a trajectory $\tau = \pi(\mathbf{s}_0)$ as

$$\tau = [s_0, a_0, s_1, a_1, \ldots, s_T, a_T] \tag{2.14}$$

and the distribution of the trajectory can be expressed as

$$p(\tau) = p(\mathbf{s}_0) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \pi(\mathbf{s}_t)) \tag{2.15}$$

If a stochastic policy is used, e.g., Gaussian policies and categorical policies, on the other hand, action is drawn from the conditional probability distribution as

$$\mathbf{a} \sim \pi(.|\mathbf{s}) \tag{2.16}$$

and then the distribution of the trajectory τ can be expressed as

$$p(\tau) = p(\mathbf{s}_0) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \pi(\mathbf{s}_t))\pi(\mathbf{a}_t|\mathbf{s}_t)) \tag{2.17}$$

A policy can be parametrized by θ as $\pi_\theta(\mathbf{s}_t)$ or $\pi_\theta(.|\mathbf{s}_t)$. The goal of the reinforcement problem can be then formally defined as

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{2.18}$$

In the robotics setting, observations can be camera images and sensor readings, actions can be joint torques and rewards can be specified according to the task at hand, e.g., if we want to navigate the end effector to a target location in the operation space, our reward function could be defined inversely proportional to the Euclidean distance between the end effector and the target location.

2.3 Imitation Learning

In robotics, demonstrating a behavior that solves the task at hand is usually simpler than scripting/hard-coding it which demands a particular know-how. The field that studies this problem is best known as imitation learning which can be approached from two broad angles: (1) Behavioral cloning (BC) and (2) Inverse Reinforcement Learning (IRL). In behavioral cloning our goal is to correctly estimate the optimal policy π^* from expert demonstrations, whereas in IRL (sometimes also referred as inverse optimal control) we estimate the reward function via the available data and use it to learn the optimal policy. The typical imitation learning setting is illustrated in Figure 2.6

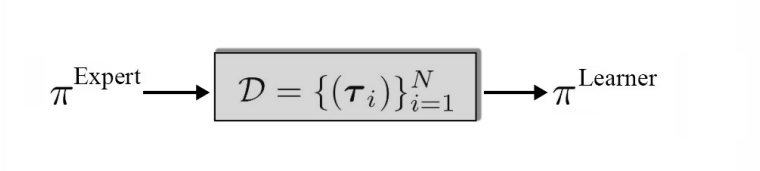


Figure 2.6: Illustration of a typical imitation learning setting.

The vanilla BC algorithm can be seen in Algorithm 1. It is a supervised learning problem where trajectory τ_i ($i = 1 \dots N$) are considered to be the samples from the optimal policy π^*, and a mapping from states to actions is learned using a plausible mathematical model class for the policy, e.g., neural network, SVM, decision tree, etc. Note that in BC, different from the regular supervised learning setting assumptions (e.g., independent and identically distributed data samples - i.i.d.), the distribution of the encountered states depends largely on the previously executed actions of the policy π_θ. This error aggregation and distributional shift problem has been widely studied e.g., [116] - we refer the reader to Background and Relevant Work Section for further details.

Algorithm 1: Vanilla Behavioral Cloning

Collect expert trajectories, $D = \{\tau_i\}_1^N$, $\tau_i = [s_0, a_0, s_1, \dots]$
Select a model class for the policy, π_θ
Define an cost function, $J(D, \pi_\theta)$
Minimize cost J w.r.t. the parameters θ
return π_θ

In BC, the agent directly mimics the expert, directly assuming its optimal and without knowing the intention of the trajectories. Potentially, discovering what the expert is trying to achieve may empower the agent to behave more optimal and robust than the expert. The aim of IRL is to identify the reward function, i.e., purpose, of the expert via the provided expert demonstrations. The problem of IRL can be mathematically formalized as, finding R^* such that [3]:

$$E\left[\sum_{t=0}^{\infty}\gamma^t R^*(\mathbf{s}_t)|\pi^*\right] \geq E\left[\sum_{t=0}^{\infty}\gamma^t R^*(\mathbf{s}_t)|\pi\right] \ \forall\pi \tag{2.19}$$

The problem of IRL is ill-posed – a policy can be optimal for numerous reward functions – and necessitating further assumptions to obtain a unique solution. Solution procedures sequentially update the reward function and policy predictions.

Chapter 3

Related Work

In this section, we review the previous studies in the area of robotic object manipulation and also provide relevant background information. Due to its practical value, there exists a vast literature on the subject matter yet it still remains to be an unsolved problem. We categorize the related work into the following categories: studies on core computer vision techniques that could be utilized for the downstream manipulation tasks mostly through employing motion primitives, i.e., classification, object detection, instance segmentation, 6D pose estimation, active perception; robotic grasping and pick-and-place systems that execute visuomotor policies; imitation learning; sim2real; uncertainty quantification methods for deep networks from the perspective of optimal/self-aware controller. It should be highlighted that, for some of the recent works, categorizing a study into one of the aforementioned categories is challenging as the research community tries to find ways to address the data availability problem in robotics by implementing an amalgam of these techniques.

3.1 Core Vision

A robot, by definition, is a machine that operates in a real world doing mechanical and routine tasks with human-like skills (ideally) either on command or automatically. A regular robot control pipeline is composed of a perception, modelling, planning, and low-level control modules - all depending on incomplete information. Deep learning has proved to be the champion in extracting useful visual features especially in constrained closed-set scenarios for number of tasks, e.g., object classification, object detection, semantic segmentation, etc. Robotic vision is regarded to be more safety critical, since different from computer vision, predictions of a deep learning-based perception system ultimately trigger actions which could expose the environment of the robot to catastrophic consequences[121].

With the availability of large data sets, in particular the ImageNet Large-Scale Visual Recognition Challenge [26] (ILSVRC - 1.4 M images, 1000 classes) and tools for par-

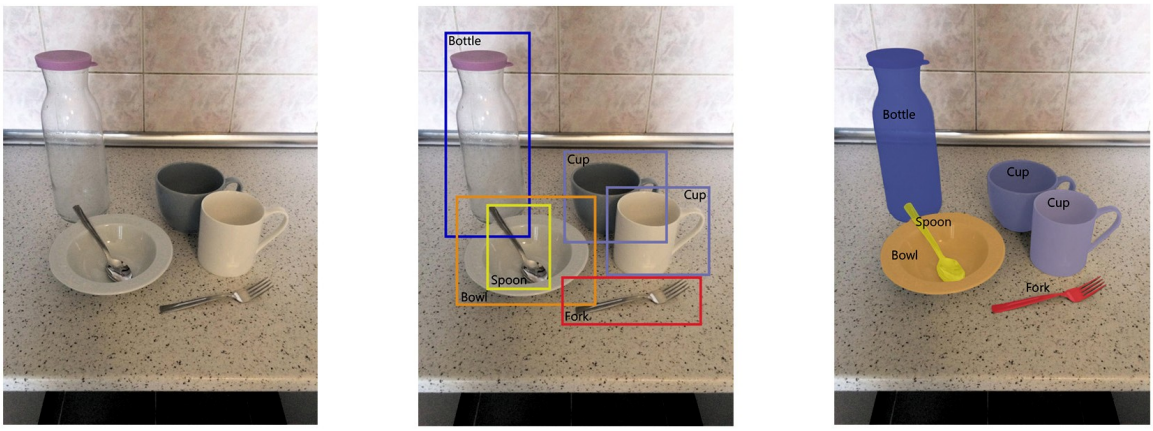
allel processing, the computer vision field has been revolutionized with the advent of deep learning.

The first major success of deep learning in image classification was achieved in 2012 via AlexNet [70] – a convolutional neural net (CNN) architecture – surpassing the previous state-of-the-art old-school/feature based methods by a very large margin (i.e., 10%) in ILSVRC where two different accuracy (percentage of the correct predictions) based evaluation metrics were used, namely Top-1 and Top-5. In 2014, crafting different architectural details, e.g., 1x1 convolutions, network depth, etc., ILSVRC classification performance increased considerably by the introduction of GoogLeNet and VGGNet [119, 123] – following the meme 'we need to go deeper' [2]. Note that, VGGNet arhitecture is composed of 3x3 convolutional kernels only - empirically demonstrating that sufficient receptive field size can be achieved simply via sequentially stacking convolutional layers. Then in 2015, thanks to ILSVRC yet again, another major breakthrough happened with the innovation of residual connections [48] - which enables training really deep architectures with hundreds of layers, e.g., 152, facilitating gradients to propagate towards the deeper units.

In general, acquiring labeled data is a hard and costly task. To avoid overfitting, which generally originates from small data set sizes and high capacity deep networks, a technique called 'transfer learning' is utilized. It can be employed (1) across different tasks, e.g., classification to object detection/segmentation: all the object detection/segmentation models uses a ImageNet pre-trained backbone architecture - assuming that the initial layers of the network is able to capture useful visual features that are required for the task at hand; (2) across domains, e.g., robotics simulation to real-world robotics. Transfer learning has attracted a great deal of research interest because of its significance. Today most of the significant pre-trained backbones are publicly available. The main hypothesis is that the neural network layers hierarchically extract features starting from the more general low-level features that can be shared across different tasks.

It should be noted that, CNNs were popular back in 1990s [75], yet due to insufficient computation resources – limited GPU training – and the lack of some key architectural elements, e.g. ReLU activation functions, dropout idea, and weight initialization techniques, etc., they were considered to be outdated for almost two decades up until the consequential success of AlexNet [93, 53, 43].

Deep learning revolution had a substantial impact on the object detection and semantic segmentation front as well. Figure 3.1 illustrates the core computer vision tasks with a kitchen table-top scene, i.e., classification, object detection, and semantic segmentation. Previously, most of the approaches were utilizing hand-crafted yet quite powerful features, e.g., HOG, SIFT, ORB filters [24, 83]. With the return of CNNs, a great performance leap has occurred, and the state-of-the-art performance in object detection and segmentation benchmarks, e.g., MS-COCO [79], and the PASCAL VOC [32], have been advancing since.



(a) image classification (b) object localization (c) instance segmentation

Figure 3.1: Illustration of core computer vision tasks. Left: Multi-class classification networks output a soft-max vector representing the probabilistic scores of the detected classes for objects that are in the scene. Middle: Object localization networks output rectangular bounding boxes with respective class labels and confidence scores. Right: Instance segmentation networks output pixel-wise semantic masks for each object class differentiating different instances within the same class.

There are two main approaches in object detection, namely one-stage detectors and (roughly) two-stage detectors. Two-stage detectors were successfully introduced in [42] with a method called R-CNN (starting the line of work later called as the R-CNN family). These detectors are composed of two modules, one for region proposal, and one for classification (and bounding box offsetting). They first identify good candidate bounding boxes or regions, then classify the content of the candidate regions while refining the bounding box predictions. The R-CNN approach is relatively slow during testing as (1) it performs a selective search to create 2000 region proposals, and (2) it feeds all the proposed regions through a CNN (47 secs per image on a GPU!), further it requires the separate training of each module. Considering such downsides, [41] proposed a model iteration called Fast R-CNN, introducing the RoI pooling layer that evaluates all the region proposals in one forward pass, and estimates class labels and bounding box regressions using the RoI feature vector – sharing some of the computation, hence speeding up the the previous model. Following up on this, the next iteration of the model (Faster R-CNN) brought forth more speed-up via replacing the selective search module with a region proposal CNN (RPN) [110]. Further improvements came with the name 'Mask R-CNN' extending the Faster R-CNN framework with an object instance segmentation branch, outperforming other detection and segmentation models of the time [47]. Mask R-CNN is still used as a backbone in many applications and as a comparative benchmark in segmentation related tasks. It is worth noting that two stage detectors require tedious and alternating training procedures for their modules. One-stage detectors

(sometimes referred to as one-shot detectors), on the other hand, are mostly trained with ease in an end-to-end fashion. They do not contain a different module for object region proposal. They instead directly regress a predefined number of bounding boxes for each predefined region on the 2D input image and perform non-maximum suppression. They are simpler to implement and can operate fast enough for real-time applications, e.g., autonomous driving. Noteworthy initial implementations include SSD [80] and YOLO [107]. Further iterations of these models became better, faster, stronger [108, 109] utilizing comparatively recent ideas like, batch norm, anchor boxes, skip connections, etc. It should be noted that the main issue with one stage detectors is that the most of the candidate bounding box regions belong to the background class and accumulated loss of the many background class examples overwhelms the loss of rare useful examples. This issue is mostly dealt with hard-negative mining and careful loss function tuning/weighting. RetinaNet work, for example, introduces focal loss to tackle the aforementioned issue [78].

Semantic object segmentation, which can be defined as pixel-wise image classification/masking aiming to subdivide the image into a set of object categories, have been studied extensively over the past few years. One of the first deep learning based segmentation works was proposed by [82] pioneering fully convolutional network (FCN) architecture (getting rig of the full-connected layers as it is possible to do the same mathematical operation with 1x1 convolutions). The FCN architecture was not, however, optimal for transmitting the contextual cues. [114] introduced encoder-decoder like architecture called U-Net, demonstrating that with a properly designed architecture that carries contextual information through skip connections and heavy data augmentation, it is possible to train a high-performance segmentation network even with a very small training data set, e.g., the training data set in this work was composed of 30 medical images with 512x512 resolution. An illustration of a simple encoder-decoder architecture that could be used for segmentation is shown in Figure 3.2. Similarly, V-Net work proposed a Dice coefficient based loss function in order to decrease the adverse effects of background-foreground imbalance [91]. Later on, extending the Faster R-CNN architecture, Mask-R CNN [47] and its variants, such as [55] where a new partial supervision method is demonstrated, were introduced. Semantic image segmentation is an ever changing/improving field yielding better models every other day [21, 22].

Core computer vision research plays a crucial role in robotic object manipulation. We want robots to semantically understand and locate what they see via their sensors, and plan accordingly. Especially for unstructured and cluttered environments, an ideal robot should be able to operate with previously unseen objects. This idea was explored in [141], where the authors proposed a class agnostic segmentation pipeline (and a new synthetic data set) that was specifically tuned for segmenting random tabletop objects. Notably, the proposed framework was trained only using a large scale non-photorealistic RGB-D data, and still outperformed state-of-the-art unseen object instance segmentation methods, showcasing their method's capability in real-world on OCID and OSD data sets[1, 111].

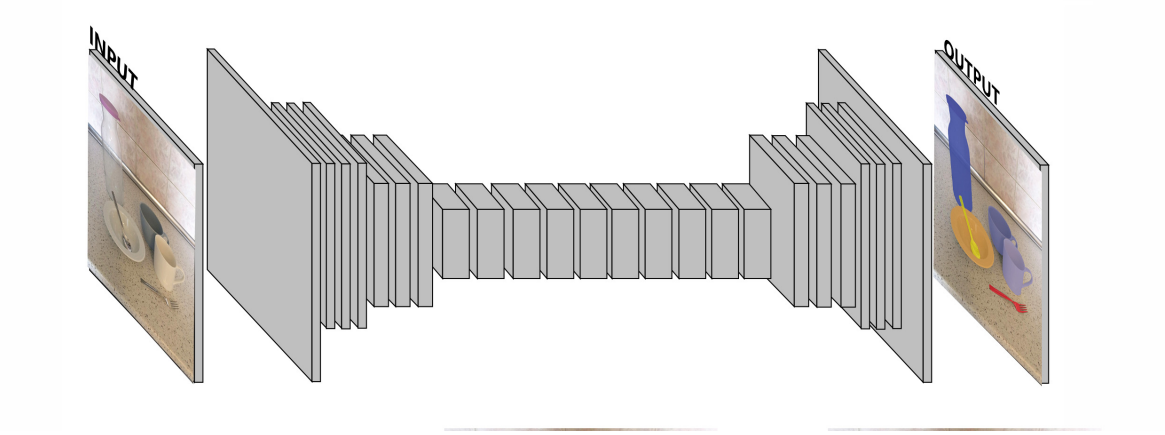


Figure 3.2: Illustration of a simple encoder-decoder based segmentation network. Given the input image of size $3 \times H \times W$, the model outputs a tensor of size $C \times H \times W$ where K represents the number of different object classes. Each channel of the output tensor predicts a segmentation mask of one of the K classes.

3.2 6D Pose Estimation

6D pose of a rigid object refers to its 3D location and orientation with respect to a selected coordinate frame which defines a unique configuration in 3D space. Many real-world applications, such as augmented reality, virtual reality, autonomous driving, and robotic object manipulation, demand accurate 6D poses of the objects. Over the last couple years, the availability of computational resources, deep pre-trained models and benchmark data sets such as LINEMOD [51], OCCLUSION [19], YCB-Video [140], and T-LESS [54] have accelerated the research progress. Recent methods estimate the poses of object instances from RGB images, RGB-D data, and point clouds. An illustration of object poses in mapped 3D bounding boxes in 2D image frame can be seen in Figure 3.3.

Capability to estimate the 6D poses of multiple, unstructured and potentially cluttered objects is particularly useful for a robot to understand its surroundings in order to plan the sequence of actions to complete the task at hand. For example, 6D position of an object can be used to plan a grasp via pre-defined motion primitives given the known class of the object. Current challenges in 6D pose estimation include illumination changes, camera calibration/dependency, appearance difference between the object in the live scene vs the object's textured CAD model. Traditional approaches addresses this problem via (1) matching extracted good local features of an object from an input image (scale-invariant, affine-invariant, illumination-invariant, etc.) with their correspondences on a respective 3D model – requiring rich texture objects, (2) using a set of templates for the object of interest sampled from different viewpoints.

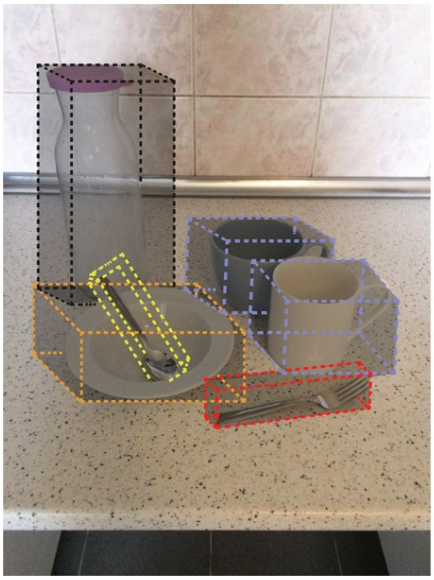


Figure 3.3: The pose of a rigid object determines the unique configuration of it in 3D space. In 6D pose estimation we mostly aim to calculate rigid transformation from object coordinate frame to the camera coordinate frame. Having a calibrated camera and the object models enables really accurate instance pose estimation.

More recent studies – following the general trend – tackle this challenging problem using deep CNN architectures. Notably, [140] introduced the model called PoseCNN that predicts 6D poses of the object instances decoupling translation and rotation estimation branches as follows: 3D translation of an object is estimated by localizing its center in the input RGB image and predicting its distance from the camera by predicting a unit vector from each pixel towards the center; 3D rotation of the object is estimated by regressing a quaternion representation. They also propose a loss function called 'Shape-Match Loss' that partially solves the training problems with symmetric objects. [126] presented a relatively simple one-shot architecture (sometimes referred to as YOLO-6D) that can operate in real-time. Their approach initially determines the 2D bounding boxes (projected vertices of the 3D bounding boxes) of the objects in the scene, then estimates the 6D pose using the Perspective-n-Point (PnP) algorithm [76]. Similarly, [64] developed a method using an SSD architecture as a backbone with minor modifications. Their method initially provides unpolished discrete approximations of the poses and then refines them. [132] proposed a one-shot model that was only trained with a mix of photo-realistic and domain-randomized data – showing that it is possible to achieve state-of-the-art results in 6D pose estimation only with synthetic data. [100] presented Pix2pose framework that regresses pixel-wise 3D coordinates of an object only from RGB images - utilizing generative advesarial training with 3D models without textures. These predicted 3D coordinates are then used to form 2D-3D correspondences to directly compute the poses with the PnP algorithm with ransac iterations. Following up on the previous work, [145] demonstrated a 6D pose inference pipeline that was composed of two blocks, namely the correspondence block and the pose block. Correspondence block

includes one encoder (resnet12) and 3 decoder heads – the ID mask, U and V channels. The correspondence maps are then used to map each 2D point to a coordinate on an actual 3D model. Recently, [71] introduced a high-performing framework, called 'Cosy-Pose', for recovering the 6D poses of multiple known objects using a set of input RGB images from unknown camera viewpoints. Their approach simply creates a single consistent scene via matching 6D object pose hypotheses which are drawn using the input image set. Cosy-Pose outperformed the previous methods by large margin on the YCB-Video and T-LESS benchmarks.

Availability of the reliable and cheap 3D sensors promoted the line of research that utilizes RGB and depth features in conjunction. For example, [138] proposed a heterogeneous architecture that extracts the pose information of the known objects from the per-pixel embedding of both RGB and depth values. [127] introduced a new discrete-continuous formulation for rotation regressions to resolve the symmetric object problem. They uniformly sample rotation anchors and predict a deviation from each anchor to target together with uncertainty scores of each prediction - performing better than the state of the art methods. Very recently, [99] have provided a pose estimation framework unseen objects. Their approach takes a set of reference RGB images of a target object with known poses, and build a 3D representation of the object. Using this 3D representation, the network is then able to render arbitrary views of the object. To estimate the pose of the unknown object, they simply search for the best pose where the rendered image matches the input image.

3.3 Active Perception

The notion of perceiving visual information through interaction is generally referred to as 'interactive Perception' [16] where the embodied agent executes a sequence of actions and monitors the outcome. Motion provides an additional signal that can be utilized in robotic vision tasks. Temporal correlation of the subsequent video frames can be used to estimate the optical flow field that describes the pixel displacements between subsequent frames. The idea of clustering the pixels with similar motion characteristics to get segmentation masks has been studied in [96, 57, 130, 18]. In some cases interactive perception can be utilized in self-supervised learning or labeled data generation pipelines. For example, in [15], the authors presented a self-supervised/autonomous method to acquire a segmentation mask of the object-in-gripper without requiring any camera calibration or object/robot model using the extracted masks of the robot hand via optical flow – utilizing the work by [56]. Similarly in [35], the authors introduce another automated method to get the segmentation annotations of the grasped objects by segmenting the manipulator from the foreground using a self-recognition network that was trained using the known forward kinematics of the robot arm and a calibrated depth camera.

3.4 Robotic Grasping and Pick-and-Place

Robotic grasping (and pick-and-place in general) is one of the most extensively studied areas of robotic manipulation. Most of the recent developments focus on top-down planar grasps using the representational power of deep networks. A complete survey is beyond the scope of this work. Instead, we review some of the notable and highly relevant works to set a general scene of the current trends, mostly focusing on data-driven approaches.

With the advent of deep learning and the inevitable data hunger, some researchers tackle the grasping problem via scaling up the real-world data collection pipelines. For example, [103] and [77] collected numerous grasp attempts and trained a CNN to estimate a grasp success likelihood. Similarly, [63] utilized half a million real-world grasp attempts, and trained a large Q-Network via the developed reinforcement learning framework called 'QT-Opt'. They achieved closed-loop grasping that generalizes to unseen objects. This self-supervision (or self-play) idea has also been used in pick-and-place systems. Recently, [12] has proposed a noteworthy pick-and-place system with an average placement error of 2.5, that operates without object models given a top-down view of the goal configuration. Their method uses the orthographic projection of the scene to generate image patches, where each image patch (and its rotated version) simply corresponds to a top-down action of the gripper.The presented approach, at its core, finds the most relevant image patches, one from the grasp-bin, one from the place-bin before placement, and one that is coming from the goal configuration, using a score that is calculated via a neural network model. Although effective, their method requires a search via sliding window in the image space.

Another notable robotic pick-and-place system (for the stowing task of Amazon Robotics Challenge 2018) that manipulates both known and novel objects in cluttered environments was presented by [148]. Different from the standard grasping pipelines, their approach followed a 'grasp-first-then-recognize' plan initially predicting pixel-wise, object agnostic (model-free) affordances for distinct grasping primitives, e.g., suction-down, grasp-down, etc. The approach then executes the action with the highest affordance and detaches the object from the clutter using the selected grasping mode, recognizes the category of the object by pairing it with product images using a distance metric, and place it in the correct bin. In this work, all the required labeling was manually done on real images using a special interface that enabled the labelers to select positive and negative grasp points, grasp angles, and suction points. This work provided a good showcase towards a scalable real-world robotic manipulation. As most of the other existing grasping pipelines, policy executed open-loop actions and could have performed better via closing the loop.

In [146], the authors formalized the kit assembly task as a shape matching problem where the goal is to learn a descriptor (an embedding) that forms a geometric correspondence between the grasped object's surface and the target placement location. To achieve this, they proposed an approach called *'Form2Fit'* that is composed

of the following modules: Suction module, place module, matching module, and a planner. Suction and place modules take an image heightmap, and respectively predict favorable picking and placement locations of dense pixel-wise confidence values. Their main contribution, the matching module (a Siamese network), learns pixel-wise orientation sensitive correspondences between the objects on the table and their placement locations in the kit by mapping every pixel into a d-dimensional vector space where the closer L2 distance indicates better object-to-placement correspondence. Finally the planner ingrates all the information coming from all the modules and produces the final pick and place decisions, i.e., picking point, placing point, relative angle of rotation between pick and place. Although their system presented an important step towards generalizable kit assembly, it only handles 2D planar (face-down) kits, and does not work with fully transparent materials.

Following up on the successful applications of transfer learning in core computer vision tasks, [142] investigate the effect of visual pre-training for pick and place tasks that utilize affordance predictions that are followed by suction and grasping primitives. They show that pre-trained vision models for various tasks such as foreground segmentation, edge detection, normal estimation, etc. can highly benefit the affordance learning process. They additionally show that off-the-shelf vision models, improves both the training speed and the final performance of the presented real-world affordance based manipulation pipeline. In [118], the authors propose a method to be able to see transparent objects which is a notably important problem considering the commonality of these objects in our everyday life, e.g. dishes. Initially, given a single RGB image they estimate (1) surface normals, (2) masks of transparent surfaces, (3) occlusion boundaries - via training networks using synthetic data. Then, using the initial depth image, they find the final/refined depth image using a global optimization. It is important to note that this work enables previously developed or readily available robotic manipulation algorithms to work with/on transparent objects.

Visuomotor policies have: (1) a visual model potion that converts the raw image into a low dimensional representation; (2) a policy model portion that combines these representations and low dimensional measurements, such as from robot encoders, to either predict an action or execute a pre-defined motion. Representing objects with a parametrized pose defined using a fixed object template, i.e. a CAD model, fails to capture large intra-class shape variations. Apart from the planar grasp points and 6D object poses, it is also possible to manipulate object using visual descriptors, i.e., object key-points, which ignores task-irrelevant geometric details. In [87], the authors presented a method, called *kPAM*, to acquire category-based semantic 3D key-points which were then used to plan a trajectory for the manipulation target pose. Building upon the previous work, [39] introduced a dense point cloud representation, to enable planning with greater shape awareness. Given a point cloud, the full 3D shape was estimated using a fully-convolutional shape completion network and then combined with the key points to plan the manipulation trajectory. Planning costs included both the distance between the object and target keypoints,

together with a collision cost to prevent the object mesh colliding with other environment objects. This line of work has been continued by [40], in which the authors introduced low-tolerance contact-rich manipulation pipeline, called *kPAM 2.0*, that uses the extracted category level key-points in a feedback loop. Similarly, in [34], the authors used a self-supervised correspondence learning approach to improve the ability of robots to learn visuomotor policies. In their approach, a dense correspondence model was used to predict the correspondence locations of a (either learned of pre-defined) descriptor set. They then used these locations as the input of the policy is then trained using the imitation learning.

3.5 Imitation Learning

The purpose of imitation learning is to train robust agents that efficiently learn the required policy by imitating the expert's behaviour, e.g., a human operator. As collecting the expert behaviour is usually too costly in robotics, the demonstration data set does not cover all possible circumstances, making the target distribution different from the source distribution. This situation is often referred to as 'covariate shift'. In imitation learning - different from the reinforcement learning - the agent does not have access to the reward signal, which is quite hard to define in real robotic object manipulation settings. Inverse reinforcement learning methods tries to recover possible reward functions using expert's trajectories that ideally represent the expert's intentions.

In a fully observable setting, e.g., a tele-operated system, it is possible to obtain trajectories as a sequence of the state and the action/control. If the control inputs are not available, the trajectory can be represented as a sequence of system states or sequence of partial observations. Assuming the underlying system is Markovian, then the system dynamics could be represented as a graphical model shown in Figure 3.5. It should be noted that observations may not satisfy the Markov property. Especially if we consider human demonstrators, most of the time, it is unlikely for a person to execute (or sample) the exact same behaviour given a fixed observation. However, more sophisticated methods, e.g., the methods that consider the observation history, could perform poorly in behavioral cloning settings.

Over time, an extensive literature has developed on the subject. Modern imitation learning literature could be traced back to the seminal work from [104] in which the authors managed to successfully and autonomously drive a car by learning a (3-layer neural net) policy that maps images from a camera and a laser range finder to discretized steering wheel directions. [10] presented a comprehensive survey of 'learning from demonstrations' and structures learning problem into the demonstration gathering phase, i.e., data collection, and the policy derivation phase, i.e., learning. [23] introduced an interactive behavioral cloning algorithm that queries additional demonstrations if the agent is not near the training data points by a specified threshold. Similarly, in [116] the authors tackled the covariate shift (data mismatch) prob-

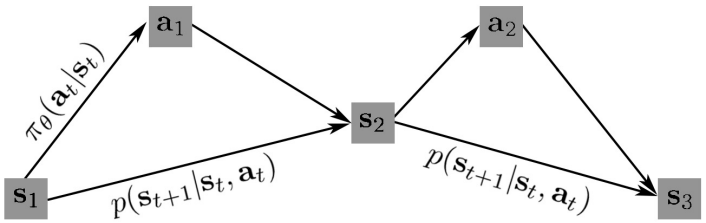


Figure 3.4: Graphical model of an MDP

lem, and proposed the well-known DAgger algorithm that learns a stationary deterministic policy through online learning where the expert specifies the correct actions given the input state that is rolled out via the learners current policy iteration. Several variations of this work have been proposed over the years studying the safety of the learned policy with statistical confidence measures [89, 90, 65]. [17] trained a convolutional neural network to obtain the function that map pixels to steering commands. They showed that the proposed system automatically learns useful latent representations for the driving task. e.g., useful road features. Recently, [28] introduced an epistemic-uncertainty aware imitation learning framework, called *SAFARI*, that actively gathers informative expert demonstrations to prevent failures.

The quality of the demonstration is decidedly critical in imitation learning, this creates a natural need to develop effective and efficient robot-human interfaces to to allow human experts control the robot optimally. For object manipulation [6] presented that kinesthetic teaching is faster/easier/preferable compared to teleoperation for human experts, yet the agent learns the desired skill better with the data set created via teleoperation. [149] introduced a virtual reality based robot control interface (teleoperation system) for object manipulation, and learned policies for variety of tasks using 30 minutes of demonstration for each. This work provided further evidence on the the importance of the quality of collected data set and the robot-human interface. Ideally, we would want a robot to learn a skill from just a few number of demonstrations, as we humans do. Along this line of research [29] presented a meta-imitation learning framework that is able learn an effective new policy from a single demonstrations from a new task, e.g., having an agent being able to generalize over different configurations of block towers given that it knows how to stack 2 blocks on top of each other. [81] developed a third-person imitation learning framework via learning a context-aware (e.g. view-point, background, object-positions, etc.) translation model using the expert videos of the same task. Merging previous two ideas with pose detection and temporal convolutions, [144] showed it is possible to achieve one-shot imitation learning from a third person view.

Other noteworthy studies on imitation learning include learning how to fly a helicopter via IRL [4], learning how to ball-paddle [69], and learning human arm movements [13].The general taxonomy and the detailed algorithmic analysis of the

field can be found respectively in [14] and [97].

Our work on one-shot imitation was mostly inspired by [62], in which the authors proposed a course-to-fine one-shot imitation approach. Different than ours, they did not utilize camera extrinsics, and estimated the viewpoint alignment actions (it is referred to as bottleneck pose) using a model that was trained after the expert interaction. The required dataset is collected in an automated manner with real robot maneuvers. Their approach requires a time-consuming data-set generation part and its success highly depends on the performance of learning-based alignment model. Note that, we only use machine learning to provide high-level guidance. One other similar approach was proposed in [11] where the authors iteratively align the live observation frame with the demonstration frame – trying to track the entire demonstration trajectory. Their method performs re-alignment using the point correspondences coming from the optical flow model, together with a given foreground mask of the demonstration. Although practical, their method fails with erroneous optical flow calculations as the image space displacements get larger. In comparison, our approach does not rely on any external pre-trained models and can realign live and demonstration end-effector frames even if the image space discrepancies are comparatively large.

3.6 Sim2Real

The real world is complex and hard to model (an environment where MDP assumptions fail). What robot sees is high dimensional, noisy, multi-modal observations from sensors which are destined to degrade, perform poorly, and eventually break. It is well-known that the actuation and sensing differ vastly between simulation and real-world. Therefore the policies that successfully work in simulation environments often do no transfer well to real world applications as the small modelling errors could lead to large control errors for the downstream tasks. Additionally, it is hard to accurately model sensors and simulate physical systems without making good number of assumptions, e.g., convex objects, discrete time steps, simple friction models, rigid bodies, etc. Even with really detailed models, in order to get realistic simulations, it is essential to get all relevant model parameters with sufficient accuracy via system identification, such as, mass, rotational inertia and damping. In robotics, this situation is often called the 'sim2real gap'.

There has been outstanding improvements in sim2real after the success of deep learning [73, 45] and deep reinforcement learning [92] which are well known for their sample inefficiency for learning. This presents a challenge in robotics as the real world data collection and labeling is costly and may be consequential. There has been some work to get around this data availability problem of robotics. One trivial, yet costly, solution is to scale up the real-world data collection framework using a fleet of robots. [5] investigated learning a dynamics model – to be able plan for novel future tasks – via random interactions, gathering 400 hours of real robot experience. [103] collected 700 robot hours of grasp attempts to train a CNN-

based classifier that outputs the grasp likelihood given an input image patch for 18-dimensional pre-defined top-down rotation angles. Similarly, [77] trained a CNN that outputs the grasp success probability given the monocular input image pairs and the motion input, using 800000 grasp attempts over the course of two months using 6 to 14 robotic manipulators. Recently, a large-scale open database of robot experience, called *RoboNet*, which is composed of 15 million video frames of seven different robot platforms, was proposed for vision-based robot learning [25]. In this work, the authors demonstrated the sample efficiency gains via pre-training with RoboNet across various tasks. Another solution to the robotic data availability problem is to develop more sample-efficient reinforcement learning algorithms, for example utilizing model-based RL [115] and meta-learning methods [33].

A potential remedy to the aforementioned data availability problem is the use of simulated data which is a cheap, fast, and safe substitute, that comes with free data labels. Via simulation, it is possible to create/simulate normally unlikely scenarios and rare events. Further, simulation enables fast prototyping and testing of robotic systems.

In their seminal paper, [129] explored a simple but influential technique called 'domain randomization', aiming to bridge the reality gap between the simulated and the real-world robotics for an object localization task. They showed that it is possible to train a high-accuracy object detector using only the randomized simulated data. They further illustrated that the amount of texture randomization plays a significant role on the performance and large amounts of simulated training data could remove the need of pre-training. The idea of randomizing the input images using different colors and textures is shown in Figure 3.6.

Other researchers extended this approach to get full 6D pose [9], to more challenging textures [31], and to a wider range of objects [132]. The method also was used in object detection for autonomous vehicles [131], visuomotor control [102, 59], and cloth manipulation [88]. It has been shown that this technique works with other types of sensors, i.e., depth camera [85]. In [128], the authors showed that seeing a wide range of unrealistic objects during training could help with the generalization to realistic objects – which are not used in training – during test time.

Zero-shot sim2real for object manipulation was also explored using key-point representations. For example, [105] proposed a key-point-based visual servoing framework, i.e., KOVIS, that achieves target reaching with mm-level clearances. Similarly, [133] developed a precise coarse-to-fine ring-on-peg pipeline fusing classical model-based motion planing with an end-to-end policy that was trained in simulation with learned task-relevant key-points. They showed that sub-mm sim2real accuracy is achievable, and key-point based image representations leads to superior policies compared to standard RGB, depth, or stereo IR based convolutional architectures.



Figure 3.5: Illustration of a target(left) and source(right) domain.

There is also a vast amount of work in domain randomization for dynamics in which the physical parameters, visual appearances, applied noise to policy inputs, sensor readings, friction coefficients, object locations, camera rotations, gear backlash, etc., are randomized. For example, in [101], authors train a recurrent neural net policy capable of adapting the random variations in the environment dynamics. A similar idea was proved to be useful in dexterous manipulation [7, 9]. Interestingly, [134] presented a recent detailed study comparing prior approaches, and showed that a random force injection into the simulation enables zero-shot sim2real as well as previously developed more complex methods.

3.7 Uncertainty Quantification

As detailed in [94], there are two important elements in Bayesian learning : Prior beliefs and the background knowledge about the system of interest are used to design prior probability distributions of the model parameters; and the prediction of the future observations/states are estimated using the posterior distributions of the model parameters that are acquired by updating the priors with the available dataset. For neural-nets however, the properties of the priors are generally constructed for learning stability, and the naive calculation of the posterior distributions are computationally very demanding, if not intractable.

[37] showed that the Monte-Carlo dropout method (applied after every weight layer) during testing provides a way to estimate the model's (MLP) predictive distribution via generating samples from it. Proposed approach introduced a practical Bayesian tool for deep neural nets. It is mostly referred to as *MC-drop*, and has been widely popular due to its simplicity. Later, [38] developed another dropout

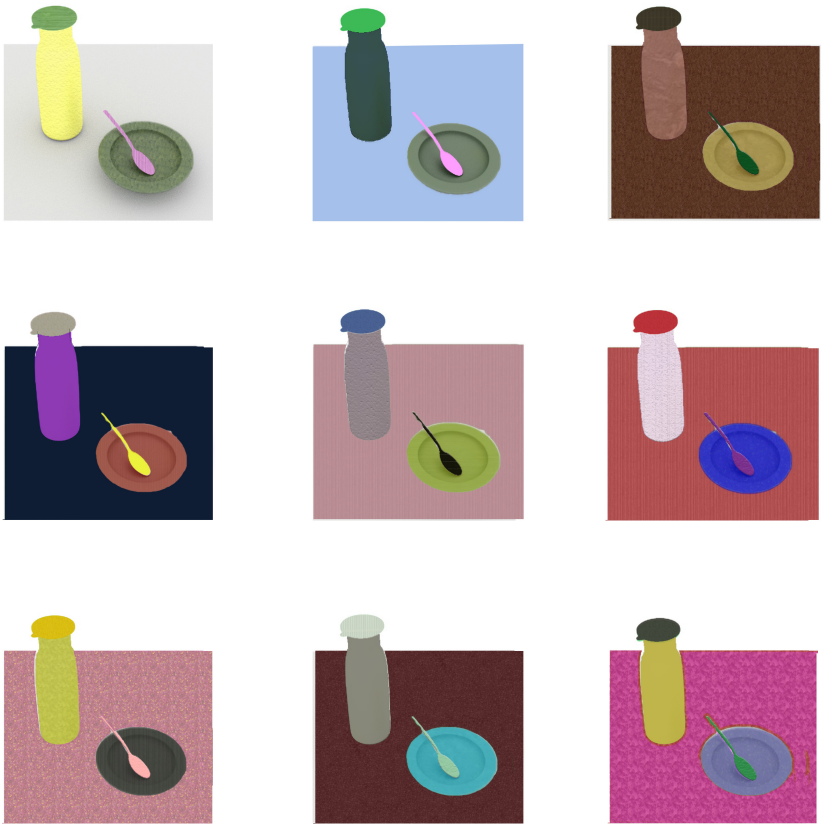


Figure 3.6: Illustration of a visual (color and texture) randomization.

variant that achieves better calibration by automatically tuning each layer's dropout probabilities. [67] divided uncertainties into aleatoric and epistemic, and presented a Bayesian deep learning framework for computer vision tasks. They empirically showed that the aleatoric uncertainty does not depend on the amount of training data and whereas the epistemic uncertainty does – and therefore should be tracked in safety-critical applications.

[72] proposed an alternative and relatively simpler method for the estimations of the predictive uncertainty of deep neural networks – which are recognized to form overconfident predictions even for out-of-distribution input data. Utilizing a proper scoring rule – which can be defined as a numerical score that assesses the quality of the trained model's predictive distribution – they proposed a non-Bayesian framework in which they combine deep ensemble learning and adversarial training. Recently, [46] showed that using a multi-input multi-output neural network architecture, it is possible to train multiple independent subnetworks that cohabit in one network, without any explicit separation during training. Therefore, with a single forward pass, their approach evaluates the predictions of all the subnetworks – gaining the benefits on ensembles while bypassing a significant computation cost that is normally required by prior methods. In [49], the authors present two methods

as a baseline to undertake the problem of silent classification failures in machine learning. They show that (1) simple statistics extracted from output softmax values could be used to determine whether the input is misclassified or coming from a different data distribution, and (2) an additional abnormality module that utilizes an frozen internal representation coming from an auxiliary decoder can be trained with in-data and out-of-data distributions could be used to distinguish between normal and abnormal test examples.

As a concluding remark to this section, it is interesting that the sensitivity estimation of the output neurons with respect to the uncertainty in the weight space was studied very early on by [27].

Chapter 4

A Study on Policy Uncertainties

Higher-level decision making systems that uses DL-based feature extractors, such as detection or segmentation models, could output erroneous predictions due to silent mistakes done by the lower-level DL-based components. It has been shown that a small additive perturbation in the input space of a deep neural network could create large perturbations at the output of the last layer [124]. As the usage of deep learning models increases, it becomes more urgent to devise means to interpret the results with their uncertainties and to develop models with out-of-distribution robustness. Especially in safety critical applications, such as, medical diagnosis, self-driving cars, autonomous drones, and domestic robotics, this situation poses a great challenge.

Herein, we present our research on predictive uncertainty estimations of neural network models. We focus on three specific approximate methods, namely, MC-dropout [37], deep ensembles (DE) [72], and MIMO [46]. These have been selected because of their implementation simplicity, and the previously reported superior performances compared to other sophisticated methods [98]. Detailed description of each method is provided in the following subsections.

We first compare these approaches by studying a 1D regression problem. We evaluate the the quality of the predictive uncertainty estimates in terms of calibration and out-of-distribution robustness. We then turn our attention to visual imitation learning problem. We simulate a ring-on-peg task and inspect how these estimation methods compare with each other when mapping visual inputs, i.e., wrist-cam images, to end-effector movements.

4.1 Method

4.1.1 MC-dropout

Dropout technique was introduced as a stochastic regularization method for deep neural networks to overcome overfitting [120]. The idea is to form subnetworks by randomly dropping neurons or feature maps from the base network during training. It has an adjustable hyperparameter, mostly referred to as the *dropout rate*, which

determines the probability of dropping/zeroing a unit (and its connections) during the forward and backward passes. Even though dropout's relations with model averaging was previously investigated in [120], the rigorous mathematical connection with the Bayesian machinery was presented by [37]. In their seminal work, the authors offered a novel interpretation of dropout showing its use as an approximate Bayesian variational inference. Activating dropout during both training and testing, and drawing multiple output samples given an input by stochastic forward passes, enables us to marginalize over the posterior distributions of the model parameters, i.e., network weights. This routine is referred to as *Monte Carlo dropout*, or *MC-dropout* for short, and illustrated in Figure-4.1. Acquired outputs are then used to estimate the mean and variance of the model predictions. Although the routine has been shown to be effective in improving the model performance [66, 67, 44], the quality of the uncertainty calibration has been identified to be coupled with the dropout-rate and architectural design choices [135, 38].

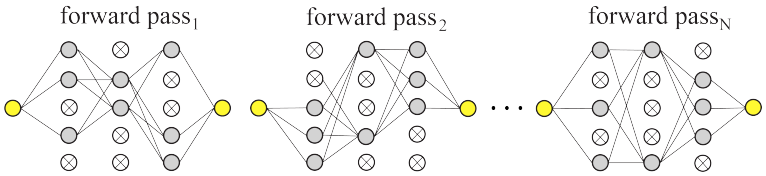


Figure 4.1: Illustration of MC-dropout. Each forward pass drop randomly neurons according to the dropout-rate.

4.1.2 Deep Ensembles

Ensembling multiple models, has been a frequently used method to increase the performance, i.e., predictive accuracy, of ML-based approaches, e.g., in Kaggle competitions. The idea is to compose a predictor using a population of base models. When the errors coming from each base model are uncorrelated, the ensemble's testing performance becomes superior than its members. For neural network ensembles, partial independence between base models is often acquired by random parameter initializations and using a different set of training hyper-parameters. [72] further shows that ensembles of neural networks – illustrated in Figure-4.2 – can also be used to estimate the predictive output distributions given that each single neural network model in the ensemble is trained with a proper scoring rule. Although *deep ensembles* (DE) approach requires a minor modification in the output layer to account for the heteroscedastic uncertainty, it is one the most scalable and reliable approaches.

For example, in a 1D regression setting – assuming that each neural network model outputs a Gaussian distribution with mean $f_{\theta_\mathbf{i}}(\mathbf{x}^n)$ and variance $\sigma^2_{\theta_i}(\mathbf{x}^n)$, where $\theta_\mathbf{i}$ corresponds to the model parameters of the i^{th} model in the ensemble, and $(\mathbf{x^n}, \mathbf{y^n})$ tuple comes from the dataset $\mathcal{D}$ of size N where $\mathcal{D} = \{\mathbf{x}^n, y^n\}_{n=1}^N$ – each neural network model in the ensemble is trained minimizing the following negative log-likelihood:

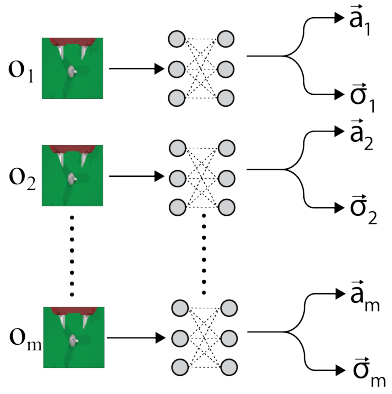


Figure 4.2: Illustration of Deep Ensembles (from a robotics perspective). $\mathbf{o_i}$ corresponds to the observation that is fed into the i^{th} model. During testing ensemble prediction is acquired by feeding the same input to all the base models, $\mathbf{o_1} = \mathbf{o_2} = ... = \mathbf{o_m}$. $\boldsymbol{a_i}$ depicts the action output estimated by the i^{th} model given $\mathbf{o_i}$. $\boldsymbol{\sigma_i}$ describes the i^{th} stardard deviation of the respective action output.

$$
\begin{aligned}
-\log p_\theta(y^n|\mathbf{x}^n) &= -\log\left(\frac{1}{\sqrt{2\pi\sigma_\theta^2(\mathbf{x}^n)}}\exp\{-\frac{(y^n - f_\theta(\mathbf{x}^n))^2}{2\sigma_\theta^2(\mathbf{x}^n)}\}\right) \\
&= -\frac{1}{2}\log\frac{1}{2\pi\sigma_\theta^2(\mathbf{x}^n)} + \frac{(y^n - f_\theta(\mathbf{x}^n))^2}{2\sigma_\theta^2(\mathbf{x}^n)} \\
&= \frac{\log 2\pi}{2} + \frac{\log\sigma_\theta^2(\mathbf{x}^n)}{2} + \frac{(y^n - f_\theta(\mathbf{x}^n))^2}{2\sigma_\theta^2(\mathbf{x}^n)} \\
&= \text{constant.} + \frac{\log\sigma_\theta^2(\mathbf{x}^n)}{2} + \frac{(y^n - f_\theta(\mathbf{x}^n))^2}{2\sigma_\theta^2(\mathbf{x}^n)}
\end{aligned}
\tag{4.1}
$$

Having an ensemble that is composed of M base neural network models, the final predictor is a mixture of Gaussians, and therefore, the ensemble mean and the variance given a new input $\mathbf{x}$ is calculated by:

$$
\begin{aligned}
\overline{f}(\mathbf{x}) &= \frac{1}{M}\sum_{m=1}^{M} f_{\theta_m}(\mathbf{x}) \\
\overline{\sigma^2}(\mathbf{x}) &= \frac{1}{M}\sum_{m=1}^{M}\left(\sigma_{\theta_m}^2(\mathbf{x}) + f_{\theta_m}^2(\mathbf{x})\right) - \overline{f}^2(\mathbf{x})
\end{aligned}
\tag{4.2}
$$

4.1.3 MIMO

Neural networks are in general massively over-parametrized models. Previous results on network pruning suggest that a neural network has enough capacity to fit several subnetworks that co-habit the same network at the same time. Building upon this observation, [46] shows that using a multi-input-multi-output (MIMO) model architecture it is possible to jointly train multiple *independent* subnetworks within

one high-capacity network. MIMO approach requires two modifications to the base neural network architecture: First, the input layer should be adjusted to take M datapoints instead of a single data point, e.g., $H \times W \times C \rightarrow H \times W \times (M \times C)$. M depicts the chosen number of ensemble units. Second, the output layer should be modified to output M heads resulting in a model that makes M predictions from the last hidden layer of the original network. Note that, the required modification in the input and output layers causes a slight yet negligible increase in the total number of parameters of the original neural network model. Training is done via independently sampling M random inputs and their respective labels, where each output head is trained to predict the label of its corresponding input head, e.g., the first output head should predict the label of the first input head. During testing, the same input is fed through all the input channels to get M independent predictions in a single forward pass. These multiple independent predictions are then used to acquire an ensemble output.

More specifically, if we denote the dataset of size N by $\mathcal{D} = \{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$, where $(\mathbf{x}^n, \mathbf{y}^n)$ corresponds to the n^{th} input-label pair, then the neural network model that captures the prediction uncertainties output a probability distribution $p_\theta(\mathbf{y}|\mathbf{x})$ is trained minimizing the negative log-likelihood of $\mathcal{D}$. In the MIMO setting – that is shown in Figure- 4.3 – the input datapoint is constructed concatenating M inputs from $\mathcal{D}$, namely, $\{x_1, x_2, ..., x_M\}$, and the network outputs M matching predictive distributions:

$$p_\theta(\mathbf{y_1}|\mathbf{x_1}, ..., \mathbf{x_M}), p_\theta(\mathbf{y_2}|\mathbf{x_1}, ..., \mathbf{x_M}), ..., p_\theta(\mathbf{y_M}|\mathbf{x_1}, ..., \mathbf{x_M})$$

During testing, a single input $\mathbf{x}$ is concatanetad to itself M times, where $\{\mathbf{x_1} = \mathbf{x_2} = ... = \mathbf{x_M} = \mathbf{x}\}$, and the network outputs M predictive distributions:

$$p_\theta(\mathbf{y_1}|\mathbf{x}, ..., \mathbf{x}), p_\theta(\mathbf{y_2}|\mathbf{x}, ..., \mathbf{x}), ..., p_\theta(\mathbf{y_M}|\mathbf{x}, ..., \mathbf{x})$$

.

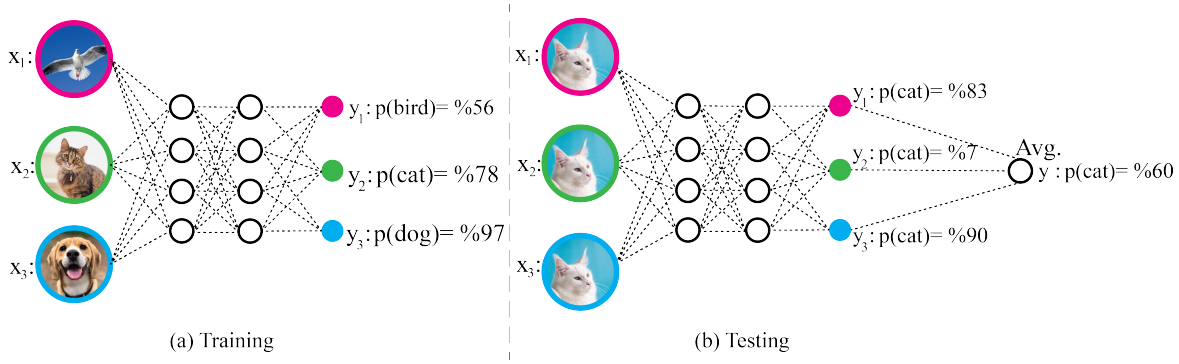


Figure 4.3: Illustration of the MIMO approach for $M = 3$ (adapted from [46]). Training is done by randomly sampling M input-label pairs, minimizing the negative log-likelihood of the MIMO model. During evaluation, the input is concatenated with itself M times and ensemble statistics are calculated using M corresponding outputs.

4.2 Experiments

4.2.1 1D Regression Problem

We implemented and empirically compared deep ensembles (DE) [72], the MC-dropout [37], and MIMO [46]. We first studied the synthetic 1D regression problem that was initially proposed by [50], and further used by [72]. The function we have tried to learn, $f(x) = x^3 + \epsilon$, is illustrated in Figure-4.4, where ϵ is the Gaussian corruption $\sim$ np.random.randn(0,9). Training datasets were always sampled uniformly between -4 and 4 for all the experiments, independent of the dataset size. Note that, [50] and [72] studied this problem using a training dataset of size 20. Herein we also analyzed the effect of the dataset size. The network architecture we used is composed of 2 hidden layers of 50 neurons with ReLU activations. Note that, depending on the utilized UQ method, we introduced the required modifications to this base architecture. For the MC-dropout method, the dropout-rate was set to 0.2. We have ensembled 5 models in our DE implementation. Similarly, the number of input and output heads, i.e., the subnetworks, was set to 5 in our MIMO implementation. And finally, we have used 20 forward passes to estimate the output mean and variance using MC-dropout. All the implementation details can be found in **https://github.com/m13uz/1d_UQ**.

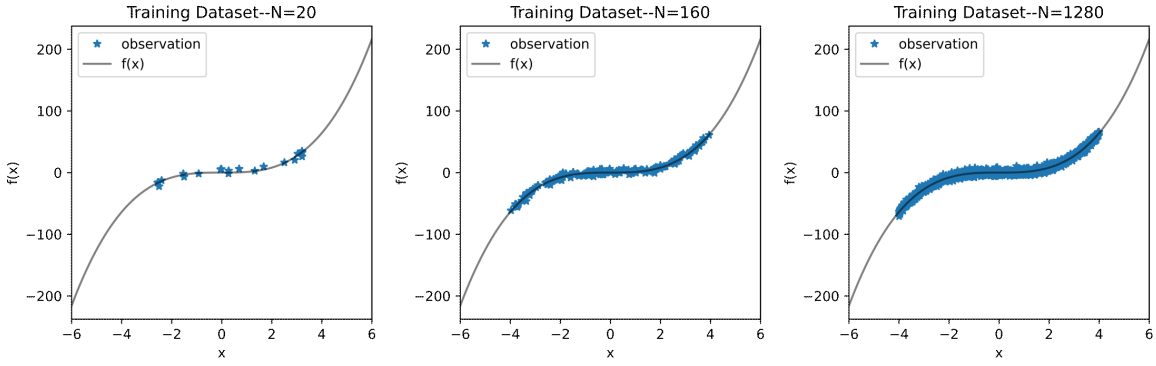


Figure 4.4: Illustration of the 1D regression problem. From left to right: training data sets of size 20, 160, 1280.

We initially trained each model with 20 noisy observations of $f(x)$, sampled between -4 and 4. Figure-4.5-top-row shows the model predictions together with the estimated uncertainties; and Figure-4.5-bottom-row illustrates how the prediction errors change with respect to the estimated prediction uncertainties. Although each method estimated higher output uncertainties as the testing inputs are sampled further away from the training distribution, the out-of-distribution robustness was best achieved with the DE approach – see the drastic increase in the estimated predictive ensemble uncertainty, compared to the other methods, as the input distribution gets further away from the training distribution. Inspecting the fitted lines to inditribution (id) and out-of-distribution (ood) testing errors and uncertainties, DE approach stood out to be the best in terms of the id and ood calibration—the output

errors and output std's increase in a coupled trend. We then studied the effect of the dataset size by increasing the number of observations to 160 and 1280. The results are presented in Figure-4.6. Similar to the scenario with the 20 observations, the best ood robustnes and calibration was achieved by DE method. It should be noted that the MIMO's calibration quality and ood robustness could have been increased by expanding the neural network model's capacity.

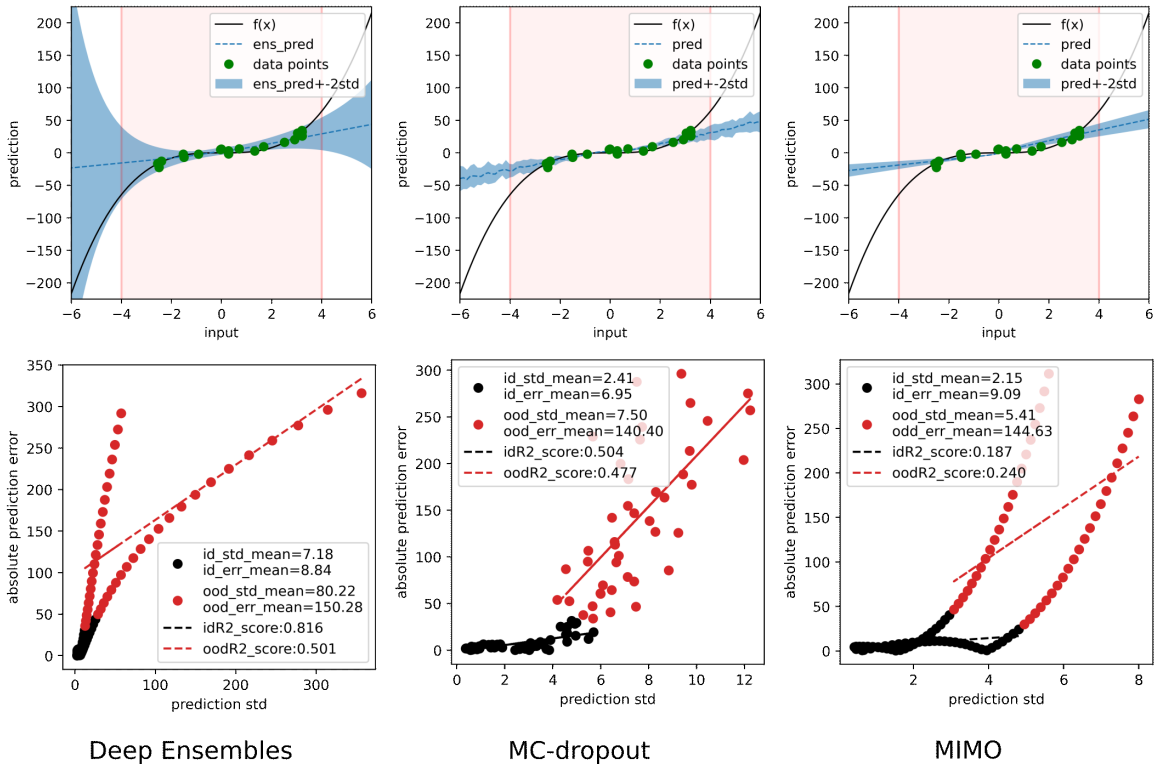


Figure 4.5: 1D Regression and uncertainty quantification results estimated (from left-to-right) by [72], [37], and [46] using a training dataset of size 20. Top Row : the model predictions (estimate mean values) of each model is shown with a dashed blue line; and output uncertainties, i.e., $+-2\sigma$, are shown as the blue regions around the estimated mean values. Red regions encapsulate the training regions, i.e., x ranges from -4 to 4; sampled observations are denoted by green dots; and the true function is illustrated by a solid black line. Bottom row: illustrates how the prediction errors changes with respect to the prediction uncertainties. In-distribution (id) and out-of-distribution (ood) testing inputs are respectively colored black and red. Black and red dashed lines describe the fitted linear lines to in-distribution and out-of-distribution points. Corresponding R^2 scores are detailed in the legends.

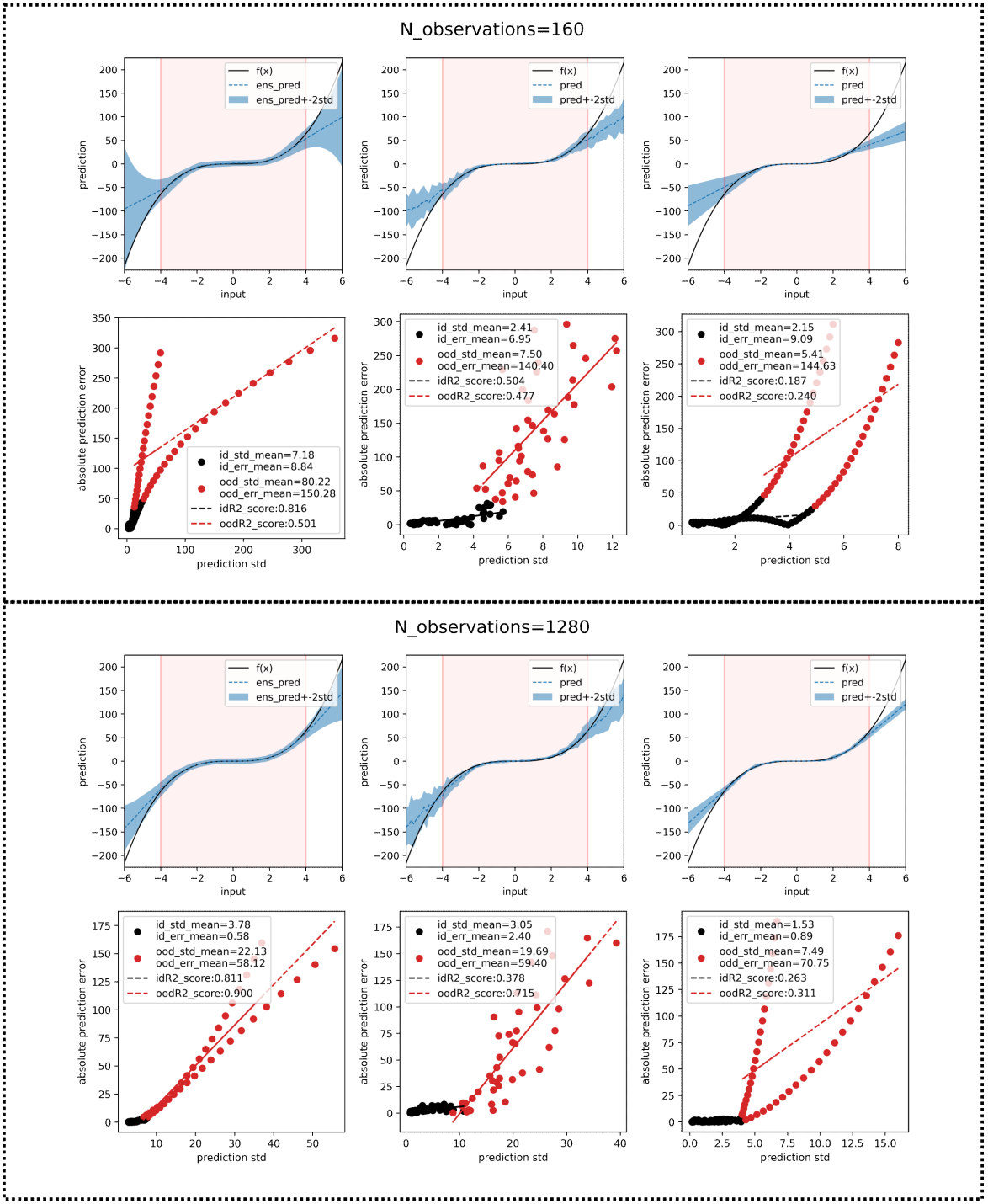


Figure 4.6: 1D Regression and uncertainty quantification results via (from left-to-right) by DE, MC-dropout, and MIMO models using a training dataset of size 160 and 1280. The first and the third row shows the model predictions with corresponding output uncertainties, i.e., $+-2\sigma$. The second and the forth row illustrates how the prediction errors changes with respect to the prediction uncertainties. See Figure-4.5's caption for more information on the figure legends.

4.2.2 Visual Imitation Problem

To better understand and compare the predictive uncertainties estimated by DE, MC-dropout, and MIMO methods/models, we evaluated them in a visual imitation setting, on a simulated *ring-on-peg* task. We used Coppeliasim [113] with PyRep [60] to construct our experiments. A typical scene is illustrated in Figure-4.7. The task requires the ring that is attached to the gripper to be perfectly aligned with the square peg, so that a direct downward movement would successfully complete the insertion. We used a floating parallel jaw gripper with an attached wrist cam. The wrist-cam was designed to be a simulated replica of the Realsense D435, and its pose with respect to the gripper was fixed throughout the experiments. The RGB images were captured at the 320x180 resolution. All visuals are resized to 128x128x3 tensors.

Solving the task requires to learn a policy (a position-based controller) that maps the captured visual inputs to position control commands: x, y, γ, where γ is the rotation around local z-axis of the gripper. Training data was collected by randomly moving the gripper on top of the peg—the workspace was a 16cm-by-16cm region, and the relative orientation of the ring with respect to the peg, i.e., γ, was sampled between -8 and 8 degrees. The gripper always faced downwards, and its height was set to $z = 15$cm above the green table-top surface. We did not assume any prior knowledge of the 3D object models, camera intrinsics, and camera extrinsics. Therefore, visuomotor policy simply needs to learn to perform the relative SO2 pose estimation of the ring w.r.t. the peg from a captured RGB image.

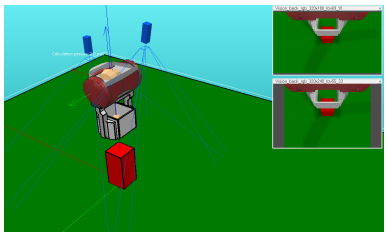


Figure 4.7: Illustration of the ring-on-peg task.

To isolate and to compare the uncertainty quantification (UQ) capabilities of DE, MC-dropout, and MIMO techniques, we trained neural models up until their validation losses get into the same ballpark ($1e^{-6}$) using a training dataset of size 8k. For DE, the output mean is calculated via averaging all base model predictions (the population is composed of 20 models), for MC-dropout we forward pass the same input 20 times and take the average of all the acquired outputs, and finally for MIMO we construct the input tensor by stacking 3 input images ($M = 3$) and calculate the output mean using the predictions of all 3 output heads. MC-dropout and MIMO models share almost the same architecture with 3M parameters (only the first and the last layer of MIMO require a minor modification), and each model in the DE population has 0.9M parameters.

In-distribution Testing We first evaluated these models' absolute output error - output standard deviation profiles with an in-distribution testing dataset. These plots are shown in Figure-4.8 – each column, from left to right, represent a control degree-of-freedom, namely, x, y, γ; and each row, from top-to-bottom, corresponds to DE, MIMO, and MC-dropout models. In terms of the R-scores, MC-dropout consistently performs the worst—failing to attach high uncertainties to wrong outputs. As can be seen in Figure-4.8, there are scenarios when all the methods output low output uncertainties attached to high prediction errors. This situation, however, is the most frequently observed with MC-dropout approach, and is the least frequently encountered with DE approach.

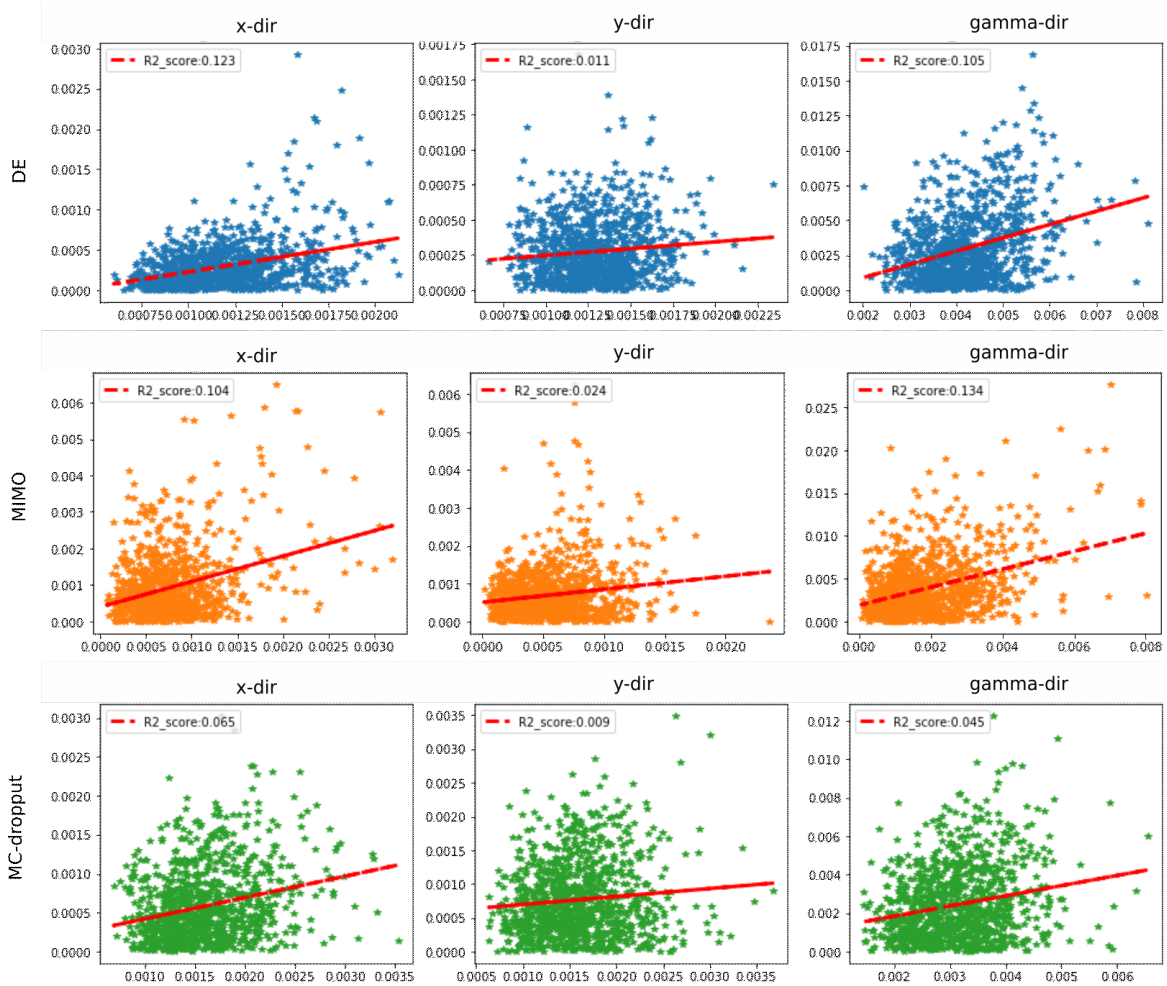


Figure 4.8: Output std (x-axis) vs Absolute error(y-axis) figures. Top-to-Bottom: DE, MIMO, MC-dropout. Left-to-Right: x, y, γ directions.

Out-of-distribution Testing We then compared each methods' out-of-distribution (ood) detection capabilities via creating ood datasets. We simulated 3 different ood scenarios—modifying the original experiment scene and the dataset generation process:

1. Height randomization (hr): *We randomize the height of the gripper during testing.* The corresponding ood dataset is generated by collecting 10 trajectories of top-down movements. At the beginning of each trajectory, we first induce a pose disturbance, $\delta x_i, \delta y_i, \delta \gamma_i$ at $z_{start} = 0.20$m, within the original workspace limits. We then move the gripper downwards until its height reaches $z_{end} = 0.11$m, following a top-down linear trajectory, while collecting observations. All the observations of the same (top-down) trajectory are labeled the same, i.e., $\delta x_i, \delta y_i, \delta \gamma_i$. The trajectory length is 50—composed of 50 steps with the same vertical step-size dz. Note that the models (DE, MC-dropout, MIMO) were trained with a dataset that was generated at a fixed gripper height, $z_{training} = 0.15$m. See Figure-4.9 top row.

2. Scene object randomization (sor): *We introduce a previously unseen spherical object that passes by the peg.* The corresponding ood dataset is created by collecting observations while the distractor object enters and leaves the frame. Gripper/ring pose disturbances with respect to the peg are sampled within the training limits at $z = z_{training}$. The object follows a linear pre-determined path. Figure-4.9 middle row illustrates a few input examples of this ood scenario. Note that all the models were trained without any distractor objects.

3. Scene multi object randomization (smor): *We randomly place a number previously unseen distractor objects around the peg.* Gripper/ring pose perturbations with respect to the peg are sampled within the training limits at $z = z_{training}$. See Figure-4.9 bottom row.

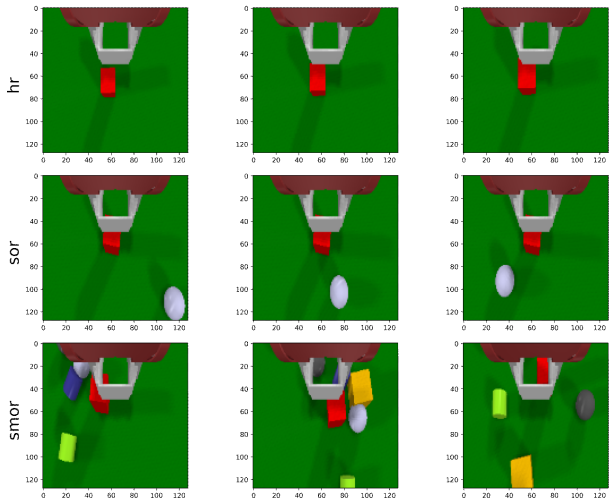


Figure 4.9: Generated out-of-distribution testing datasets for the ring-on-peg task. Top-to-bottom: hr, sor, smor.

for all different ood scenarios, namely, (top-to-bottom) height randomization (hr), scene object randomization (sor), and scene multi object randomization (smor),
We show how the estimated output uncertainties percent change with respect to their averaged in-distribution values. From left to right, bar plots illustrate the percent increase in the estimated uncertainties, which are calculated respectively via

DE, MIMO, and MC-dropout methods. MC-dropout model consistently yields the lowest amount percentile change in output predictive uncertainties for all the ood scenarios—the worst ood detection capability. Overall, MIMO approach seems to be the most sensitive to ood inputs—could be learning a better embedding (training manifold) of the training dataset due to its training scheme. We should still note, however, that DE approach still stood out as the best in terms of assigning high uncertainties to high output errors. For example, see Figure-4.11, where we illustrate each model's output error-output std profiles of the x-offset for the ring position. Shaded gray zones represent the mean+-2std of the absolute errors and output stds when testing with the in-distribution data. As can be seen, DE performs the better in both determining ood inputs, and attaching high predictive uncertainties to wrong policy outputs.

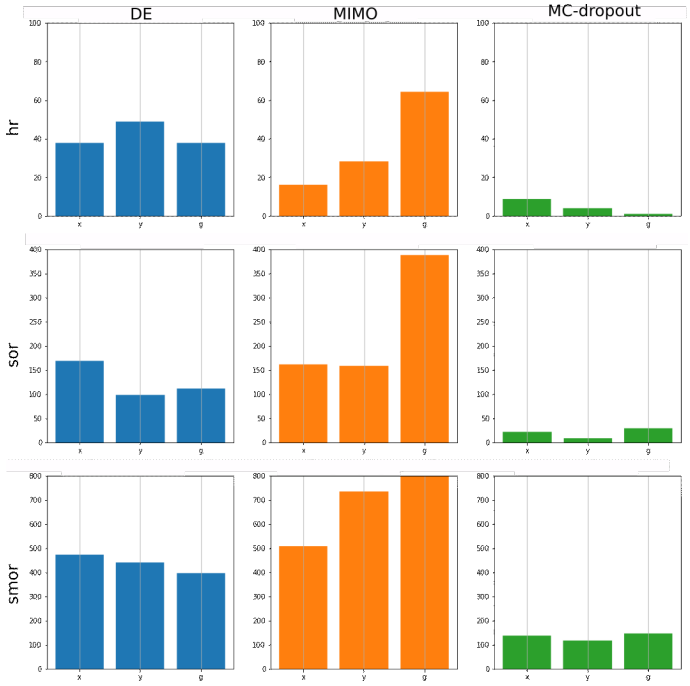


Figure 4.10: Out-of-distribution (ood) robustness of each method for the ring-on-peg task. Each control DOF, x, y, γ, is inspected separately. Each bar plot illustrates the percentile change of the averaged testing output std compared to its in-distribution testing mean. Top-to-bottom ood scenarios: hr, sor, smor.

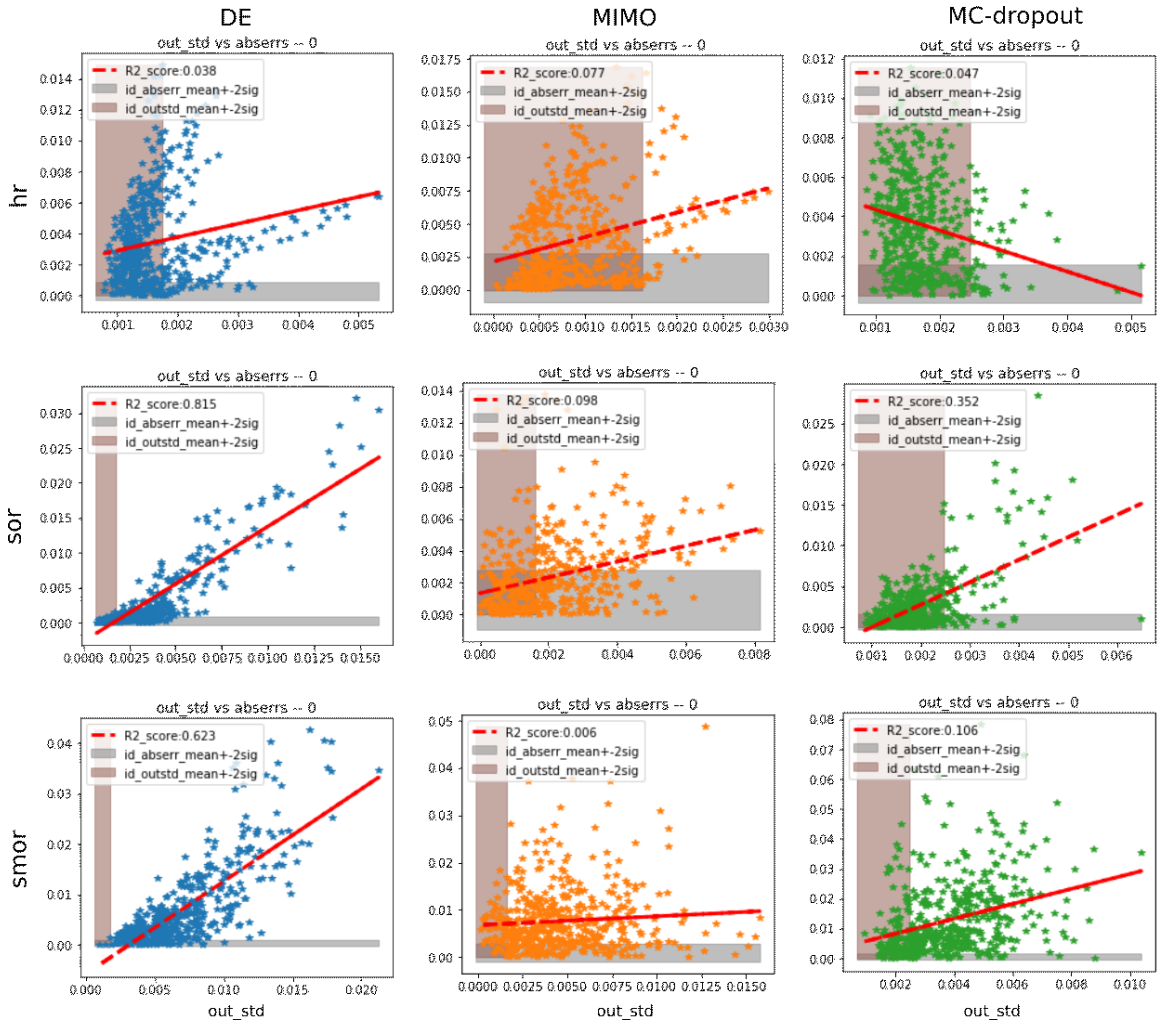


Figure 4.11: Output error-output std profiles of the x-offset when testing with ood datasets. Each row illustrates the testing results with the following ood datasets: hr, sor, and smor. From left to right each column respectively represents DE, MIMO, MC-dropout

Chapter 5

Align-and-Repeat

Visual-imitation learning is a powerful framework that enables robots to learn control policies using expert demonstrations. Collecting the expert behavior is usually too costly and does not cover all possible circumstances. We tackle these issues by proposing a practical and effective method that enables robots to achieve one-shot imitation for table-top manipulation tasks without using any 3D object models and task-relevant prior knowledge. Our method only requires a calibrated wrist-cam for depth measurements, and a single expert demonstration, i.e., end-effector velocities. For each task, we accomplish one-shot imitation via dividing the robot motion into two phases, namely, the view-point alignment and the interaction. Our approach first learns to imagine the orthographic render of the workspace via training an offline neural-scene-rendering model to guide the end-effector for rapid retrieval of all the task-relevant visual information of the live scene. After enough live-scene information is collected, the first frame of the live scene is aligned with the first frame of the demonstration scene *via estimating a rigid transformation matrix*, so that, we can simply repeat the expert actions. This enables us to perform complex manipulation behavior without explicitly learning a policy. We run a number of manipulation experiments in simulation comparing our approach with similar baselines.

5.1 Method

A common table-top manipulation task involves various different interaction trajectories that lead to task completion. In this work, the expert is expected to provide one of these acceptable interactions. We further assume: (1) the wrist-cam is rigidly attached to the robot arm, i.e., the relative pose of the end-effector tip in the camera frame is invariable; (2) camera intrinsics and extrinsics are known, (3) the object of interaction could be randomly placed on the workspace and could only rotate around its local z-axis (the axis perpendicular to the table). This section specifies a detailed overview of the proposed align-and-repeat method, providing a description of each module.

5.1.1 Demonstration Scene

Formation. Given a demonstration table-top scene S_{demo}, our approach starts with collecting N depth maps, $D_{demo} = (d_1, d_2, ..., d_N)$, where $d_i \in R^{H \times W}$, together with corresponding extrinsic matrices that map camera frames to robot base frame $X_{demo} = (x_1, x_2, ..., x_N)$, where $x_i \in R^{4x4}$, using a pre-scripted policy that follows a planar path parallel to the table-top. This procedure allows us to patch the missing visual information of initially unseen regions due camera properties and self-occlusions. Assuming a fixed-wrist cam, using the camera intrinsic matrix, K, we then convert each depth reading into a point-cloud representation $P_{demo} = (p_1, p_2, ...p_N)$ where $p_i \in R^{H \times W \times 3}$. These point clouds are then converted into the robot base frame using the logged extrinsics, $H_{demo} = (h_1, h_2, ..., h_N)$, where $h_i \in R^{H' \times W'}$, and fused to render an orthographic top-down view, h_{demo}, of our demonstration work-space. This representation enables us to filter out any points that are outside of our work-space, those that belong to the gripper, or any missing depth values due to sensor noise. Additionally, independent of the object's position and orientation, the 2D area that is covered with the object mask is always the same given reliable scene re-constructions with dense sampling. To take advantage of this, we then define a function, $V(.)$, that estimates the 2D area of the object of interaction given the orthographic render of the scene. This is achieved by summing up the number of non-zero pixels of the orthographic view, $V(h_i) = \sum_{m,n}^{H',W'} \alpha(h_{i_{mn}})$ where $\alpha(h_{i_{mn}}) = 1$ if rendered $h_{i_{mn}} > 0$, and else $\alpha(h_{i_{mn}}) = 0$. Note that the robot-base is assumed to be located on the table-top, making heightmap values of the table-top zero. Estimated $V(h_{demo})$ value is later used as a comparison criterion during live testing – having a similar value simply means that all the object-relevant visual information is collected.

Interaction. After rendering an accurate heightmap of the demonstration scene, h_{demo}, we let the expert control the robot to solve the given task, while recording the provided interaction trajectory $\tau = (\boldsymbol{a_0}, \boldsymbol{a_1}, ...\boldsymbol{a_T})$, where $\boldsymbol{a_i} = [v_x^i, v_y^i, v_z^i, w_\alpha^i, w_\beta^i, w_\gamma^i, G]$ depicts the end-effector velocities (with respect to the end-effector's frame) together with the open/close command G for the gripper. This recorded discrete time policy is referred to as $\pi_{inter}[n] : \mathbb{N} \rightarrow R^7$, and directly cloned during live testing.

5.1.2 Live Scene

Overview. Given a new live scene S_{live}, our aim is to align our robot's gripper pose w.r.t. the object of interaction with what it was at the initial time-step of the provided expert demonstration. This alignment is achieved by two computations: (1) using a neural scene rendering model $f_\theta(.)$ – that predicts the orthographic map of the workspace given the current render – we first estimate and execute optimal velocity command that would move the gripper towards currently unexplored yet expected to be information rich regions to have a more complete top-down render; (2) we estimates and executes the alignment transformation, $T \in R^{4x4}$, after the value of the live-render, $V(h_{live})$ reaches to its previously recorded value, $V(h_{demo})$ within a certain threshold. Note that as we keep moving the gripper we keep collecting scene information and synchronously update h_{live}. After the alignment phase

is completed, π_{inter} takes over and executes the exact expert actions in an open-loop manner. Although this procedure allows us to accurately mimic complicated manipulation behaviors without explicitly learning a policy, any error in alignment could introduce an aggravating offset during the interaction.

Neural Scene Rendering. Even though it is convenient to have a method that only uses a wrist-cam, this design choice usually leads to a considerably reduced field of view largely occluded by the gripper. Moreover, for table-top manipulation scenarios, if the object is large, it is simply not possible to see the entirety of its 3D shape. One simple way around this issue is to follow a pre-scripted (or random) policy to guide the gripper movement aiming to collect workspace visuals from different viewpoints until a reliable orthographic map could be constructed. This type of solution, however, would be fairly time-consuming considering the allowable joint velocities in domestic environments due to safety reasons. As a response to this matter, we train an encoder-decoder type neural scene rendering model f_θ, that predicts the heightmap of the live scene - that is not yet fully explored- given the current heightmap. Simply subtracting the binary mask of the current heightmap from the model prediction reveals the expected orthographic view of the unexplored regions, and enables us to estimate the optimal gripper velocity. This guidance increases the rate of the information gain compared to a random exploration. Estimated gripper velocity at each step is logged into a buffer and the final output is calculated via averaging all the previous estimation. The rendering model $f_\theta(.)$ is trained offline with a dataset that is generated leveraging the recorded heightmap of the demonstration scene, h_{demo}. Taking inspiration from denoising-autoencoders [45], we construct the training dataset of this model as follows. (1) We first crop h_{demo} around the object mask that is acquired via simple background subtraction. (2) We then randomly cover different parts of it to form a noisy, incomplete top-down render of the object. (3) Finally, we randomly rotate and distribute both the noisy-crop and the original-crop creating an input-label pair of an imagined workspace. Model f_θ learns to complete the shape of the interaction object and used to predict the complete orthographic map of the live scene.

Alignment. Having reliable orthographic renders, h_{demo} and h_{live}, we construct the transformation, $T \in R^{4\times4}$, via simply estimating the relative object translation and rotation between the S_{demo} and S_{live}. Applying this same transformation to the end-effector in S_{live} (using a position-controller), aligns the object-to-end effector poses in S_{live} and S_{demo} – in other words after the alignment the captured depth map should look exactly the same with the depth map that is captured at the initial time step of the provided interaction policy. Note that, even in the case of perfect alignment, although not significant, in reality there would be inevitable discrepancies due to sensor noise, illumination variations, etc. Relative demo-to-live translation of the object of interaction is directly found comparing extracted heightmaps h_{demo} and h_{live} through masking the objects. Relative rotation, on the other hand, is found by cross-

correlating the cropped object heightmaps and selecting the angle that yields the maximum cross-correlation value. The precision of the rotation estimation depends on the rotational-step size. Even though decreasing the angular search step requires more computation, the time overhead could be avoided through parallelization.

5.2 Experiments

First, we evaluate the viewpoint alignment component of our approach via object reaching experiments. Second, we study the value of utilizing a neural-scene renderer for guided exploration. And finally, we rollout the full one-shot imitation learning pipeline on several everyday table-top tasks.

5.2.1 Setup

We designed our simulation experiments in CoppeliaSim [113] with PyRep [60] using a floating parallel jaw gripper. The wrist-cam's intrinsic parameters were modified to match the Realsense D435, and its pose was fixed with respect to the gripper throughout the experiments. The workspace was a 50cm-by-50cm region on a table-top unless otherwise specified. Live scenes were created via randomly placing and orienting the object of interaction. The relative orientation of the object ranged -45 to 45 degrees compared to its local pose in the demonstration scene. We assumed that the expert trajectory, τ, starts when the gripper was facing downwards at a known height, i.e., $z = 0.5$m. Therefore the view-point alignment in the live scene requires us to find 3D position control value $[\Delta x, \Delta y, \Delta\gamma]$. The images and depth measurements are captured at the 320x180 pixel resolution. For machine learning-based baselines, all captured visuals are resized to 64x64 pixels.

5.2.2 View-point Alignment

We initially evaluate our method's view-point alignment capabilities via constructing target reaching tasks using a selected subset of ShapeNet objects[20], namely, *camera*, *shoe*, *mug*, and *kettle*. Captured depthmaps and grayscale images of these objects are shown in Figure-5.1. The task is to move the end-effector to a **reference pose** that is manually defined with respect to the object in the demonstration scene. Herein, each demonstration scene was constructed via placing the object and the gripper at the workspace center. This relative pose of the gripper tip with respect to the object was recorded. During testing the aim is to control the gripper's position and orientation to achieve the same object-gripper alignment of the demonstration, i.e., the same relative reference pose. We compared our method with vanilla behavioral cloning (BC) model that maps input images to end-effector velocites, $v = [v_x, v_y, \omega]$. These models were trained with 1, 50, and 100 expert trajectories (BC-1, BC-50, BC-100). We also compared our approach against the course-to-fine (C2F) method, that was introduced in [62], with required modifications. C2F is composed of two neural network models (one for approach and one for last-inch

correction) that output 3D continues actions $a = [\Delta x, \Delta y, \Delta\gamma]$ given an RGB image to align the gripper with respect to the object in the live scene. First, our approach uses depth, therefore, we provided that information to all other baselines. Second, we have used grayscale images instead of RGB, as the used objects were texture-less and monochromatic. BC models and C2F models share the same CNN architecture that is provided in [62] with an adjustment in the number of input channels, and trained using a mean squared-error loss. Third, as we perform the alignment at a known gripper height, we have used the correction (reference pose estimation) models of [62] in a 2-step visual servoing sequence using only the current images at each step. Note that, [62] compares naive visual-servoing with filtering methods and shows that sequential state estimation during approach increases the success rate. Fourth, we assumed to have a detailed and reliable orthographic map of the workspace bypassing the guided neural scene rendering phase.

Reference pose alignment results, that are estimated running each method in 25 random live scenes, are illustrated in Table-5.1. For each object, we demonstrate the mean values of the absolute translation and rotation errors, together with their standard deviations. These errors were calculated comparing the final pose of the gripper with the reference pose from the demonstration scene. We see that our method significantly outperforms all other baselines, solely relying on properly constructed h_{demo} and h_{live} without any required machine learning. Figure-5.2 illustrates the procedure we utilized to determine the rotation between the objects (i.e., mug) in S_{demo} and S_{live} via the cross-correlation operation: at each iteration the live-heightmap-crop (illustrated in Figure-5.2-Mid-Left) is rotated by 1 degree and cross-correlated (i.e., searched) with the demo-heihgtmap-crop (illustrated in Figure-5.2-Right). Figure-5.2-Left shows the results of this cross correlation operation. Figure-5.2-Mid-Right, shows the version of the live-heightmap-crop that yields the highest correlation value.

BC agents gets better as we increase the number of provided trajectories, however, even 100 demonstrations were not enough to fully outrun the effects of covariate shift. Furthermore, we observed that BC agents are prone to making severely large reaching errors depending on the input, e.g., for the *kettle* task, testing involved a number of drastically wrong policy rollouts for BC-100 agent. C2F stands out to be the second best among the all baselines probably because its models were specifically trained to estimate the correction to reach the reference pose during testing. The performance of C2F could have been further improved (1) by utilizing multiple observations that are gathered on-the-fly as the gripper gets closer to the reference pose [62], (2) with an extensive hyper-parameter tuning, (3) by increasing the dataset size.

5.2.3 Guided Exploration

Our method requires an orthographic workspace render to estimate the required correction during the alignment phase. As using only a mounted wrist-cam significantly

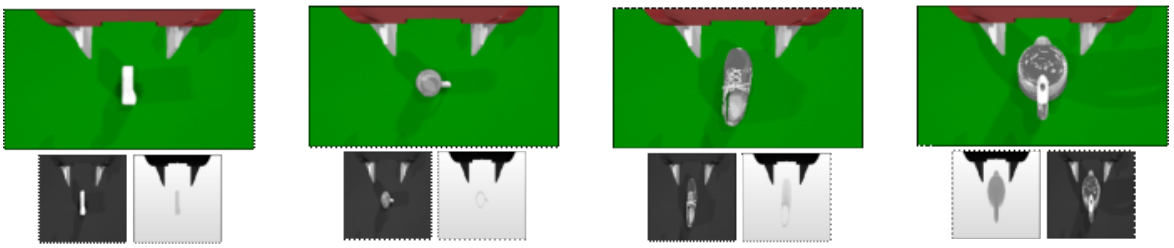


Figure 5.1: The set of ShapeNet[20] objects that are used in viewpoint alignment experiments. Shoe, mug, camera, kettle. The top row illustrates the RGB images of demonstrations scenes as they were captured by the wrist-cam. The bottom row illustrates the corresponding resized grayscale images and depthmaps.

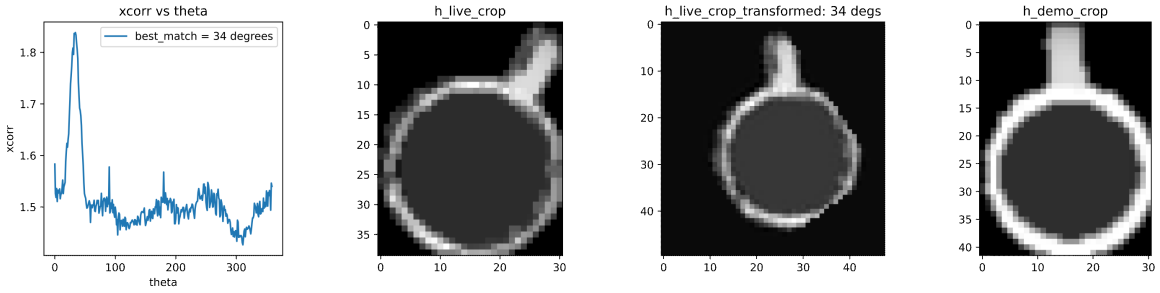


Figure 5.2: Illustration of the cross-correlation procedure that is used to estimate the relative rotation of the mug in S_{demo} and S_{live}. Left: Demonstrates the cross-correlation values as a function of θ, that are estimated via rotating and sliding the live-heightmap-crop on top of the demo-heightmap-crop. Right: Illustrates the demo-heightmap-crop which is acquired from h_{demo}. Mid-Left: Illustrates the live-heightmap-crop which is acquired from h_{live}. Mid-Right: Best matching version of the live-heightmap-crop that yields the highest correlation value.

Method	Mug		Camera		Kettle		Shoe	
	pos[mm] mean-std	ori[deg] mean-std	pos[mm] mean-std	ori[deg] mean-std	pos[mm] mean-std	ori[deg] mean-std	pos[mm] mean-std	ori[deg] mean-std
BC-1	137.9-87.2	89.1-46.1	78.3-68.4	23.9-20.6	146.1-96.5	232.3-164.7	232.3-164.7	99.2-46.1
BC-50	129.5-103.9	107.9-53.1	22.8-55.1	3.5-4.7	11.4-21.4	2.8-2.0	20.1-39.5	9.1-17.6
BC-100	12.2.5-17.5	6.6-8.6	9.3-13.4	2.5-4.7	48.9-58.1	58.1-65.9	8.2-11.8	46.7-74.6
C2F	10.1-8.7	6.3-0.7	13.1-15.0	2.2-3.0	9.9-12.9	5.0-8.1	11.8-10.7	4.0-6.0
Ours	0.9-0.7	1.1-0.8	0.9-0.7	0.9-0.6	1.2-0.9	0.7-0.5	1.0-1.4	0.5-0.4

Table 5.1: View-point alignment results. Mean values and standard deviations of the absolute reaching errors were calculated across 25 live scenes.

reduces the field of view, we proposed a neural-scene-renderer model, $f_\theta(h_i)$ that imagines the currently not explored or occluded regions of the workspace given the available orthographic map, h_i. This is especially important when the task requires the robot to interact with a large object. To demonstrate the importance of this component we designed an experiment where the task is to collect enough depth measurements of the object of interaction, i.e., *drawer* [61], as fast as possible to acquire the complete orhographic render of the testing scene. For example, consider the the depth measurement that is illustrated in Figure-5.3a. If we create an orthographic render using only this information, then we can only obtain what is illustrated in the top-row of Figure-5.3b. If, however we keep compiling depthmaps from different viewpoints, the final orthographic render portrays the entire geometry of the object, as shown in the bottom-row of Figure-5.3. $f_\theta(.)$ is an encoder-decoder type of model with a Unet[114] architecture. The training dataset of size 10K was created by cluttering h_{demo} as described in the methods section. Several examples of input-label pairs are shown in Figure-5.3c. Offline dataset generation and model training together roughly takes 5 minutes on a workstation with 6-core 3.5 GHz intel i7-7800X CPU and GeForce RTX 2080 GPU. Having a trained model that *imagines* what is currently unknown enables us to guide the end-effector movement maximizing the information content per control step. As can be seen in Figure-5.3d, trained model learns to patch the orhographic renders completing object's top-down shape. We compared out approach with: [1] a random-walk policy - that moves the end-effector in a random direction with a fixed step size; [2] a scripted-walk policy that follows a diagonal trajectory that visits the middle points of all four edges of the workspace. Both baselines, together with our approach outputs 2D position control commands with a step size of 1 cm. The workspace for this experiment was set to 1m-by-1m region. Our results are summarized in Figure-5.4, where each x-axis and y-axis respectively denotes the number of steps taken and the live top-down 2D area estimation of the object of interaction, i.e., $V(h_{current})$. The plot on the left demonstrates the averaged object-relevant novelty that is discovered per-step that were estimated over 20 live-scenes - highlighting the importance of guidance. Guided-walk gathers the necessary object-relevant information faster than the other baselines, on average in 25 steps. The plot on the right illustrates all the outcomes of 20 experiments. As can be seen, scripted-walk could gather more object-relevant information as the exploration continues. This happens simply because the orthographic render of the top-surface is represented more accurately fusing more depth information – the effect of minor Gaussian filtering to patch missing pixel values is reduced. Note that, during live testing, it is possible for the model, f_θ, to find different ways to patch the current orthographic render, cancelling out the optimal direction information. This local-minimum-like situation was eliminated using a buffer averaging earlier calculations. Another possible solution to this circumstance could have been an epsilon-greedy type of guidance.

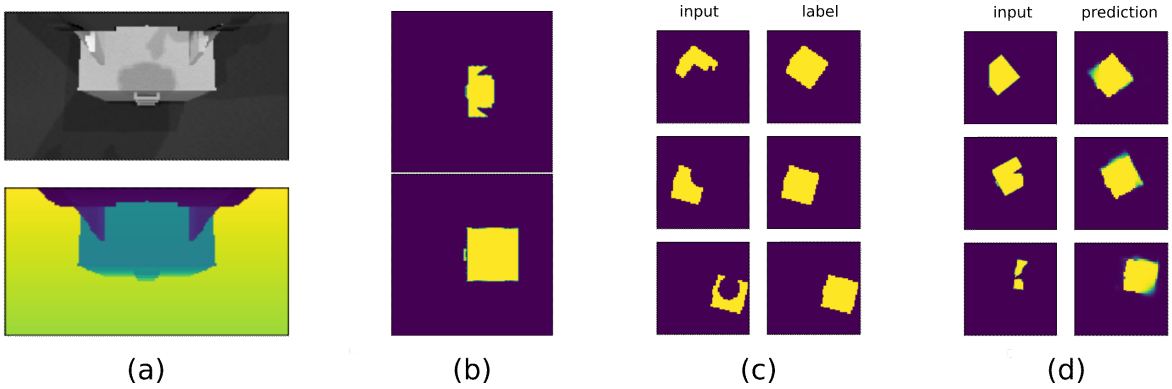


Figure 5.3: (a-top,a-bottom): grayscale image and depthmap captured from by the wrist-cam in an example testing scene. (b-top, b-bottom): rendered orthographic maps respectively using a single depthmap, and multiple depthmaps. (c) A few input-label pairs from the constructed dataset that is used the train scene-rendering model f_θ. (d) A few model predictions given the current orhographic renders.

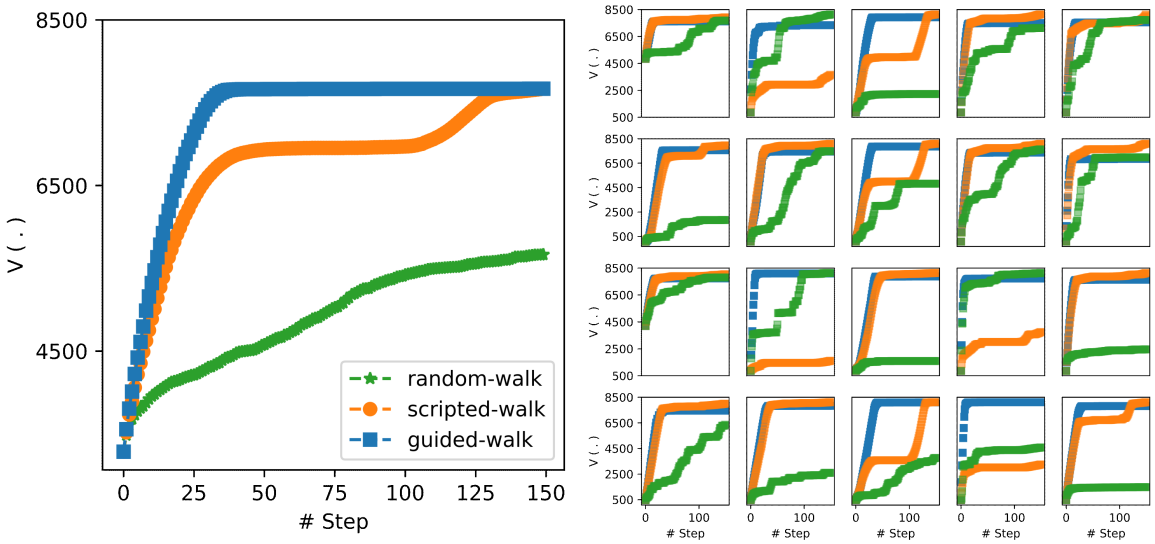


Figure 5.4: (Left): The illustation of the averaged object-relevant novelty, $V(h_{live}(n_{step}))$, that is discovered per-step that are estimated over 20 live-scenes via guided-walk, scripted-walk, and random-walk. (Right): Outcomes of all the experiments, following the same color code.

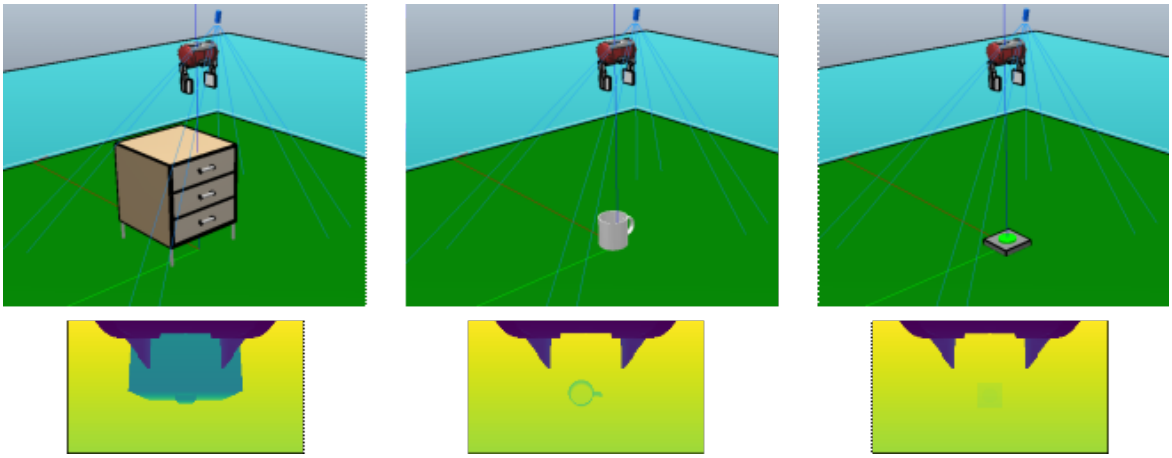


Figure 5.5: Tasks: open the drawer, pick up the mug, push the button. The bottom row illustrates the respective depthmaps of the first demonstration frame.

5.2.4 One-Shot Imitation

View-point alignment and guided exploration experiments have independently evaluated the significance of each component of our framework. We herein evaluated the full one-shot imitation pipeline including the expert interaction policy, π_{inter}. We chose 3 everyday tasks of different difficulties, namely, *Drawer*, *Mug*, and *Button*. Experiment scenes – that are shown in Figure-5.5 – were constructed using ShapeNet [20], and RLBench [61]. *Drawer* task requires the bottom-most drawer to be opened via grasping and pulling its handle. *Mug* task requires the mug to be properly grasped by its holder and lifted. *Button* task requires the green button to be pushed inward with a fully-closed gripper. The workspace size for the *Drawer* task was increased to 1m-by-1m region. During the demonstration phase, the task-relevant object was placed at the centre of the workspace, then h_{demo} was constructed following a scripted trajectory by fusing the depthmaps of different viewpoints, and finally the expert policy was executed. For each task, the scene-rendering model was trained with a dataset of size 10K, following the randomization procedure explained in Section 5.1.2. Success criteria for each task were defined either via comparing the coordinates of the object-gripper interaction points, i.e., grasping way-point, or the post-interaction joint-angles. Every experiment were also evaluated by eye to process any errors that could occur due simulation failures of contacting surfaces.

We compared our approach with our modified implementation of C2F method following the same design choices previously described in Section 5.2.2. We trained two C2F models changing the dataset size. First C2F baseline was trained with 2.5K datapoints (this approximately corresponds to a 15 minute of real robot interaction time per task). For C2F* model, we doubled the dataset size to 5K datapoints. The corresponding datasets were constructed automatically as described in [62]. During this automated data collection, we randomized lights to make sure that trained C2F models are robust to illumination changes. We omit naive behavioral cloning baseline due to its many fold more data requirement.

Task	Button	Mug	Drawer
	S-F	S-F	S-F
C2F	13-7	12-8	11-9
C2F*	16-4	13-7	13-7
Ours	20-0	19-1	20-0

Table 5.2: One-shot imitation learning results. Results were obtained across 20 live scenes. S: Success, F: Failure

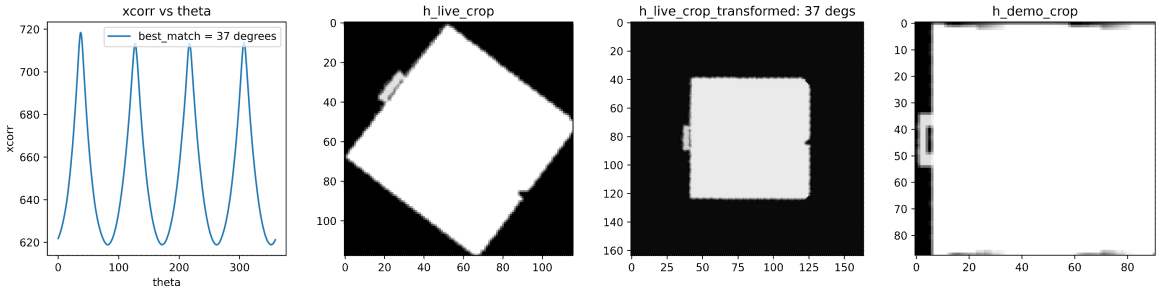


Figure 5.6: Illustration of the cross-correlation procedure that is used to estimate the relative drawer rotations in S_{demo} and S_{live}.

Table-5.2 illustrates the results for our one-shot imitation learning experiments across 20 live scenes per task. We see that our approach outperforms C2F and C2F*, i.e., our regression based baselines, with a significant margin, demonstrating that our method can learn visual robot manipulations tasks from a single expert trajectory. Although it is shown that the C2F method could automatically get better simply by increasing the dataset size, i.e., C2F* – which does not require any additional human efforts –, and its performance could improve with further filtering methods as shown in [62], we observed that they are prone to catastrophic failures, especially when the object of interaction is not completely visible by the wrist-cam. For example, C2F agents collided with the drawer in several live scenes. We believe that real-life domestic deployments of neural-network-based regression methods for robot manipulation would require calibrated policy output uncertainty estimations, so that, harmful scenarios could be avoided.

While being significantly superior in terms of the success rates, one natural current limitation of our approach compared to the learning-based regression baselines is the computation time that is required for the alignment phase. As described in the section 5.1.2, our current implementation of the proposed method finds the required rotation matrix that aligns the demo and live viewpoints via applying a a sequential cross-correlation operation. At each iteration the live-object-crop from h_{live} is rotated by 1 degree and slided over the demo-object-crop: a single 2D cross correlation evaluation requires m^2 floating point operations if the live-object-crop's size is $\approx m \times m$. This procedure is illustrated in Figures-5.6-5.2. For example, using the Scipy [137] implementation of cross-correlation on a 4-core 2.7 GHZ intel

i7-6829HQ CPU, alignment phase for the *button* and the *mug* tasks takes 3-to-5 seconds, whereas the *drawer* task takes around a minute of wall clock time. This limitation, however, could be resolved by parallelization. Furthermore, the search space could be significantly reduced by getting an initial coarse rotation estimate which is then fine-tuned using smaller steps. For the *Mug* task, the only failure of our approach was due to a minor error in estimating the transformation, T, because the mug was randomly placed at the very edge of the workspace.

Chapter 6

Conclusions

We initially have studied how existing predictive uncertainty quantification methods for neural network models, namely, deep-ensembles[72], MC-dropout[37], and MIMO [46], behave in regression settings. We empirically compared each method in terms of their calibration and out-of-distribution robustness. Overall, although deep-ensemble method stood out to be the best in terms of the uncertainty calibration, we have observed a number of testing scenarios where all the implemented methods fail to attach high uncertainties to wrong model predictions.

We then have proposed a novel, practical, and data-efficient VIL method that enables robots to carry out complex everyday manipulation tasks using a single expert demonstration. Our method does not require any task-specific engineering, 3D object models, and substantial human effort. We have divided the robot motion into alignment and interaction phases combining ideas from classical vision and neural scene rendering. During alignment, we first have collected sufficient live-scene information in order to estimate the transformation that aligns the gripper pose in the live scene so that we can directly clone provided expert actions. Using machine learning only during alignment to achieve coarse-grained guidance has provided us a safer and interpretable controller. Our simulation experiments have shown the effectiveness of our approach compared to data-hungry visual imitation methods. We have further demonstrated the benefit of using an orthographic neural-scene-renderer and provided a comparative study with similar one-shot baselines. Overall, our approach proves to be a strong method for single-step table-top manipulation tasks. Future work includes real-world experiments that account for real sensor noise and erroneous camera calibration. We also plan to extend this approach to multi-step manipulation tasks. Other opportunities include incorporating learning-based 6D object pose estimation methods for novel objects with classical approaches to reduce the compute time that is required to estimate the relative transformation matrices for the view point alignment. Additionally, the performance of the neural-scene-renderer could be improved via using a backbone that is pre-trained with number of different table-top manipulation tasks in simulation.

Bibliography

[1] *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. IEEE, 2019. pages 19

[2] We need to go deeper, Apr 2020. pages 17

[3] P. Abbeel, Jun 2020. pages 14

[4] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010. pages 26

[5] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in neural information processing systems*, pages 5074–5082, 2016. pages 27

[6] B. Akgun and K. Subramanian. Robot learning from demonstration: kinesthetic teaching vs. teleoperation. *Unpublished manuscript*, 2011. pages 26

[7] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. pages 29

[8] R. Alghonaim and E. Johns. Benchmarking domain randomisation for visual sim-to-real transfer. *arXiv preprint arXiv:2011.07112*, 2020. pages 2

[9] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. pages 28, 29

[10] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. pages 25

[11] M. Argus, L. Hermann, J. Long, and T. Brox. Flowcontrol: Optical flow based visual servoing. *arXiv preprint arXiv:2007.00291*, 2020. pages 27

[12] L. Berscheid, P. Meißner, and T. Kröger. Self-supervised learning for precise pick-and-place without object model. *IEEE Robotics and Automation Letters*, 5(3):4828–4835, 2020. pages 2, 23

[13] A. Billard and M. J. Matarić. Learning human arm movements by imitation:: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2-3):145–160, 2001. pages 26

[14] A. G. Billard, S. Calinon, and R. Dillmann. Learning from humans. In *Springer handbook of robotics*, pages 1995–2014. Springer, 2016. pages 27

[15] W. Boerdijk, M. Sundermeyer, M. Durner, and R. Triebel. Self-supervised object-in-gripper segmentation from robotic motions. *arXiv preprint arXiv:2002.04487*, 2020. pages 22

[16] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017. pages 22

[17] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. pages 26

[18] T. Bouwmans, S. Javed, M. Sultana, and S. K. Jung. Deep neural network concepts for background subtraction: A systematic review and comparative evaluation. *Neural Networks*, 2019. pages 22

[19] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. pages 20

[20] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. pages 47, 49, 52

[21] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. pages 19

[22] X. Chen, R. Girshick, K. He, and P. Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2061–2069, 2019. pages 19

[23] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009. pages 25

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005. pages 17

[25] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019. pages 2, 28

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. pages 16

[27] J. S. Denker and Y. LeCun. Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859, 1991. pages 31

[28] N. Di Palo and E. Johns. Safari: Safe and active robot imitation learning with imagination. *arXiv preprint arXiv:2011.09586*, 2020. pages 26

[29] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017. pages 26

[30] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011. pages 5

[31] J. S. Dyrstad and J. R. Mathiassen. Grasping virtual fish: A step towards robotic deep learning from demonstration in virtual reality. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1181–1187. IEEE, 2017. pages 28

[32] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. pages 17

[33] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017. pages 1, 28

[34] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019. pages 25

[35] V. Florence, J. J. Corso, and B. Griffin. Robot-supervised learning for object segmentation, 2019. pages 22

[36] Y. Gal. Uncertainty in deep learning. *University of Cambridge*, 1(3):4, 2016. pages 2

[37] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. pages 29, 32, 33, 36, 37, 55

[38] Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In *Advances in neural information processing systems*, pages 3581–3590, 2017. pages 29, 33

[39] W. Gao and R. Tedrake. kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion. *arXiv preprint arXiv:1909.06980*, 2019. pages 24

[40] W. Gao and R. Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letters*, 6(2):2962–2969, 2021. pages 25

[41] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. pages 18

[42] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. pages 18

[43] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. pages 17

[44] P. Goel and L. Chen. On the robustness of monte carlo dropout trained with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2219–2228, 2021. pages 33

[45] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016. pages 10, 27, 46

[46] M. Havasi, R. Jenatton, S. Fort, J. Z. Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai, and D. Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020. pages 30, 32, 34, 35, 36, 37, 55

[47] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. pages 18, 19

[48] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. pages 17

[49] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. pages 30

[50] J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015. pages 36

[51] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. pages 20

[52] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. pages 7

[53] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. pages 17

[54] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017. pages 20

[55] R. Hu, P. Dollár, K. He, T. Darrell, and R. Girshick. Learning to segment every thing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4233–4241, 2018. pages 19

[56] T.-W. Hui, X. Tang, and C. Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8981–8989, 2018. pages 22

[57] S. D. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2117–2126. IEEE, 2017. pages 22

[58] S. James, M. Bloesch, and A. J. Davison. Task-embedded control networks for few-shot imitation learning. *arXiv preprint arXiv:1810.03237*, 2018. pages 1

[59] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv preprint arXiv:1707.02267*, 2017. pages 2, 28

[60] S. James, M. Freese, and A. J. Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019. pages 39, 47

[61] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. pages 50, 52

[62] E. Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. *arXiv preprint arXiv:2105.06411*, 2021. pages 2, 27, 47, 48, 52, 53

[63] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018. pages 23

[64] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017. pages 21

[65] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019. pages 26

[66] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016. pages 33

[67] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017. pages 11, 30, 33

[68] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. pages 5

[69] J. Kober and J. Peters. Learning motor primitives for robotics. In *2009 IEEE International Conference on Robotics and Automation*, pages 2112–2118. IEEE, 2009. pages 26

[70] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. pages 7, 17

[71] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020. pages 22

[72] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017. pages 30, 32, 33, 36, 37, 55

[73] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. pages 5, 27

[74] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. pages 6

[75] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. pages 7, 17

[76] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009. pages 21

[77] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018. pages 23, 28

[78] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. pages 19

[79] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. pages 17

[80] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. pages 19

[81] Y. Liu, A. Gupta, P. Abbeel, and S. Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018. pages 26

[82] 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*, pages 3431–3440, 2015. pages 19

[83] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. pages 17

[84] D. J. MacKay and D. J. Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003. pages 8, 9

[85] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017. pages 28

[86] A. Makhzani and B. Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013. pages 8

[87] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019. pages 2, 24

[88] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:1806.07851*, 2018. pages 28

[89] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer. Dropoutdagger: A bayesian approach to safe imitation learning. *arXiv preprint arXiv:1709.06166*, 2017. pages 26

[90] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. *arXiv preprint arXiv:1807.08364*, 2018. pages 26

[91] F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016. pages 19

[92] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. pages 27

[93] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. pages 17

[94] R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012. pages 29

[95] A. Y. Ng and M. I. Jordan. *Shaping and policy search in reinforcement learning*. PhD thesis, University of California, Berkeley Berkeley, 2003. pages 12

[96] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1187–1200, 2013. pages 22

[97] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018. pages 27

[98] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019. pages 32

[99] K. Park, A. Mousavian, Y. Xiang, and D. Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10710–10719, 2020. pages 22

[100] K. Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7668–7677, 2019. pages 21

[101] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018. pages 29

[102] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017. pages 28

[103] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016. pages 23, 27

[104] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. pages 25

[105] E. Y. Puang, K. P. Tee, and W. Jing. Kovis: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7527–7533. IEEE, 2020. pages 28

[106] M. Ranzato, C. Poultney, S. Chopra, and Y. L. Cun. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007. pages 8

[107] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. pages 19

[108] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. pages 19

[109] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. pages 19

[110] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. pages 18

[111] A. Richtsfeld. The object segmentation database (osd), 2012. pages 19

[112] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. 2011. pages 8

[113] E. Rohmer, S. P. N. Singh, and M. Freese. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com. pages 39, 47

[114] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. pages 19, 50

[115] U. Rosolia and F. Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017. pages 28

[116] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. pages 2, 14, 25

[117] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. pages 5, 6

[118] S. S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. Cleargrasp: 3d shape estimation of transparent objects for manipulation. *arXiv preprint arXiv:1910.02550*, 2019. pages 24

[119] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. pages 17

[120] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. pages 32, 33

[121] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018. pages 2, 16

[122] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998. pages 12

[123] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. pages 17

[124] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. pages 2, 11, 32

[125] C. Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010. pages 12

[126] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018. pages 21

[127] M. Tian, L. Pan, M. H. Ang Jr, and G. H. Lee. Robust 6d object pose estimation by learning rgb-d features. *arXiv preprint arXiv:2003.00188*, 2020. pages 22

[128] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018. pages 28

[129] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017. pages 2, 28

[130] P. Tokmakov, C. Schmid, and K. Alahari. Learning to segment moving objects. *International Journal of Computer Vision*, 127(3):282–301, 2019. pages 22

[131] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977, 2018. pages 28

[132] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018. pages 21, 28

[133] E. Valassakis, N. Di Palo, and E. Johns. Coarse-to-fine for sim-to-real: Sub-millimetre precision across the workspace. *arXiv preprint arXiv:2105.11283*, 2021. pages 2, 28

[134] E. Valassakis, Z. Ding, and E. Johns. Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics. *arXiv preprint arXiv:2008.06686*, 2020. pages 2, 29

[135] F. Verdoja and V. Kyrki. Notes on the behavior of mc dropout. *arXiv preprint arXiv:2008.02627*, 2020. pages 33

[136] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008. pages 8

[137] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020. pages 53

[138] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019. pages 22

[139] F. Wenzel, K. Roth, B. S. Veeling, J. Świątkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020. pages 9

[140] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. pages 20, 21

[141] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation. *arXiv preprint arXiv:1907.13236*, 2019. pages 19

[142] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin. Learning to see before learning to act: Visual pre-training for manipulation. 2020. pages 24

[143] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy. *arXiv e-prints*, pages arXiv–2008, 2020. pages 2

[144] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018. pages 26

[145] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1941–1950, 2019. pages 21

[146] K. Zakka, A. Zeng, J. Lee, and S. Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. *arXiv preprint arXiv:1910.13675*, 2019. pages 23

[147] K. Zakka, A. Zeng, J. Lee, and S. Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410. IEEE, 2020. pages 2

[148] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018. pages 23

[149] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. pages 26