Task Focused Robotic Imitation Learning

Pooya Abolghasemi
University of Central Florida

TASK FOCUSED ROBOTIC IMITATION LEARNING

by

POOYA ABOLGHASEMI
M.S. University of Central Florida, 2014
B.S. University of Tehran, 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2019

Major Professor: Ladislau Bölöni

ABSTRACT

For many years, successful applications of robotics were the domain of controlled environments, such as industrial assembly lines. Such environments are custom designed for the convenience of the robot and separated from human operators. In recent years, advances in artificial intelligence, in particular, deep learning and computer vision, allowed researchers to successfully demonstrate robots that operate in unstructured environments and directly interact with humans. One of the major applications of such robots is in assistive robotics. For instance, a wheelchair mounted robotic arm can help disabled users in the performance of activities of daily living (ADLs) such as feeding and personal grooming. Early systems relied entirely on the control of the human operator, something that is difficult to accomplish by a user with motor and/or cognitive disabilities.

In this dissertation, we are describing research results that advance the field of assistive robotics. The overall goal is to improve the ability of the wheelchair / robotic arm assembly to help the user with the performance of the ADLs by requiring only high-level commands from the user. Let us consider an ADL involving the manipulation of an object in the user's home. This task can be naturally decomposed into two components: the movement of the wheelchair in such a way that the manipulator can conveniently grasp the object and the movement of the manipulator itself.

This dissertation we provide an approach for addressing the challenge of finding the position appropriate for the required manipulation. We introduce the ease-of-reach score (ERS), a metric that quantifies the preferences for the positioning of the base while taking into consideration the shape and position of obstacles and clutter in the environment. As the brute force computation of ERS is computationally expensive, we propose a machine learning approach to estimate the ERS based on features and characteristics of the obstacles.

This dissertation addresses the second component as well, the ability of the robotic arm to manipulate objects. Recent work in end-to-end learning of robotic manipulation had demonstrated that a deep learning-based controller of vision-enabled robotic arms can be thought to manipulate objects from a moderate number of demonstrations. However, the current state of the art systems are limited in robustness to physical and visual disturbances and do not generalize well to new objects. We describe new techniques based on task-focused attention that show significant improvement in the robustness of manipulation and performance in clutter.

To my parents and my love Mina

ACKNOWLEDGMENTS

I gratefully thank my advisor who made my stay at UCF a memorable experience. I also would like to thank the members of my committee, Dr. Mubarak Shah, Dr. Gita Sukthankar, and Dr. Bradley Willenberg for their time and valuable comments.

I would also like to thank my collaborators, and friends for their support and friendship and the discussions about interesting ideas. Special thanks to my parents and my love Mina for their love and support during my studies in the most positive way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Recent advances in machine learning (in particular, deep neural networks) led to unprecedented advances in many fields, including computer vision and natural language processing. Over the last several years, many research projects are also applying these technologies to the field of robotics. Although robots are already being deployed in factories, warehouses, and even homes, they are either limited to repetitive applications in well-controlled environments or simple tasks such as vacuuming which do not require manipulation. The focus of this dissertation is assistive robotics however our architectures and techniques can be used in other settings such as mobile robots as well. The goal of assistive robotics is to help disabled or elderly people in the performance of activities of daily living (ADLs). Since open-world manipulation must handle complex perception and a multitude of tasks, most existing technologies such as wheelchair mounted robotic arms are controlled by the user, although some level of automation is desired, especially for people with severe physical and cognitive disabilities. Many ADLs involve object manipulation where motion planning is needed for the arm of the robot to reach an object. In these kinds of tasks, to ensure the arm is able to reach the target object, positioning the base of the robot is important since it strongly affects the ability of the robot to perform the task. We can divide a manipulation task into two control tasks: the positioning of the robot base TP, and the grasping of the object with the robotic arm TG. These tasks can be executed autonomously or manually based on the preferences of the user.

Let us consider a manipulation task by a mobile robot in which the goal is to grasp an object c in the presence of a set of obstacles. The TG task uses a motion planning algorithm A_{grasp} to find a collision-free path to reach the target. It appears that the task of TP is limited to finding a position from where A_{grasp} can successfully perform the grasp, and thus TP is strongly dependent on the grasping algorithm. For instance, if the base of the robot moves to a position from which the arm would not be able to reach the target object, the base should move to another position. Humans, however, have an intuitive understanding that some positions are clearly better than others *without having a specific grasp algorithm in mind*. To do this, we need to (a) quantifying the ease of reach in a way that aligns both with human judgment and motion planning algorithms, and (b) find a way to estimate this metric at a speed suitable for real-time operation.

To transfer this intuition to a robot we define a metric called *ease of reach score* (ERS). In Chapter 3 the ERS is defined by considering the number of distinct ways the object can be reached from a given position for the base of the arm. Defining the ERS this way separates the positioning task TP from the grasping task TG and would be useful for both automated and manual execution of the TG task.

Predicting the ERS and selecting the position based on it will only increase the manipulation chance to be successful but success is not guaranteed. After deciding where to go to perform the manipulation, the robot needs to go there and perform it. Recent researches show the possibility of end-to-end training of deep visuomotor policies that perform object manipulation tasks such as pick-and-place, push-to-location, stacking and pouring. These systems perform all the components of the task (vision, grasp and trajectory planning, and robot control) using a neural network trained by variations of deep reinforcement learning and learning from demonstration (supervised learning). Deep visuomotor policies for manipulator control are neural network architectures that have as input an observation composed of an image or video frame and possibly other sensory data, $\mathbf{o}_t$, a task (or goal) specification, g, and their output is the robot commands, $\mathbf{a}_t = \pi(\mathbf{o}_t, \mathbf{g})$. The robot executes these commands, enacting a change in the external environment, which creates a new observation $\mathbf{o}_{t+1}$, and the cycle repeats. Architecturally, most currently proposed systems follow variations of the generic model of Figure 1.1, which posits the existence of a primary latent encoding, $\mathbf{z}$, the result of the visual processing of the input by a specialized visual network. This encoding, of dimensionality orders of magnitude smaller than the input, is then used by the motor network to generate the next joint angles command for the robot, $\mathbf{a}$. Our approach is pure behavior cloning - we found that the tasks could be learned using the combination of techniques listed above, without the need of refining using dataset aggregation or reinforcement learning.

In the related work chapter we are going to discuss an architecture proposed in [56] which can learn how to perform manipulation tasks from a set of demonstrations. In [56] the demonstrations used to train the policy are clean, undisturbed and lack any physical or visual disturbances. Practically, demonstrations recorded by controlling a real robot cannot cover all possible situations and disturbances the robot might face in a real environment. If a policy is trained using such demonstrations, placing an extra object in the robot's field of view can confuse the motor network by sending the primary latent encoding to an unseen state. To prevent such confusion the latent space should be dependent only on parts of the image that are relevant to the current task. In this dissertation, we are going to investigate techniques and strategies to achieve this goal.

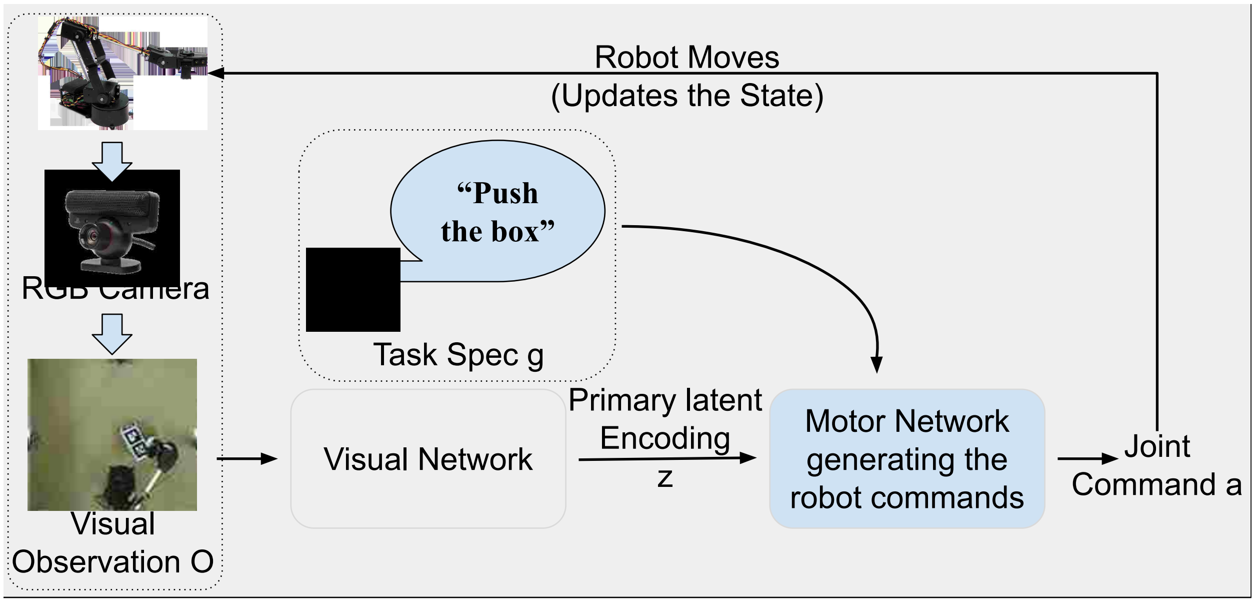


Figure 1.1: The robot executes the commands, enacting a change in the external environment, which creates a new observation $\mathbf{o}_{t+1}$, and the cycle repeats.

The remainder of this dissertation is organized as follows. Chapter 2 describes related works and an overview of a method for learning trajectories using recurrent neural networks and raw images as input. In Chapter 3 we introduce the ease of reach score to better estimate the best position for the robot to perform a manipulation task. In Chapter 4 we augment a visuomotor policy with *Task Focused Attention* (TFA) and illustrate how attention can robustify the policy against visual and physical disturbances. In Chapter 5 we proposed a data augmentation technique (Accept Synthetic Objects as Real) and two novel network architectures that take advantage of it to train end-to-end policies that operate in the presence of clutter.

CHAPTER 2: RELATED WORK

In the introduction, we decomposed the manipulation task into two major segments: (a) The positioning of the robot base TP (b) Performing the manipulation with the robotic arm TG. In the literature, it is also the case that a paper is in most cases written to address only one of these scenarios since each of these scenarios is broad enough to be the subject of discussion for a paper. Hence, we are going to discuss the papers which addressed TP and TG separately.

Positioning of the Robot Base

Placement of a mobile robot is an important problem since it fundamentally affects the ability of the robot to execute the tasks. The importance of the robot's placement in industrial settings to improve the robot's flexibility and throughput is investigated in [67]. One approach to address this problem, introduced by Zacharias, Borst, and Hirzinger [80], is to create the *capability map* of the robotic arm. A capability map contains the information describing which regions of the workspace are reachable from what directions. This map would be useful in finding a proper robot placement for the execution of workspace linear constrained trajectories [81] [19]. Leidner and Brost [41] used object-centric reasoning to find a correct place and time of the movement for the base of the humanoid robot Rollin' Justin for a mobile pick-and-place task.

The inverse reachability approach [71] proposed by Vahrenkamp, Asfour and Dillmann find suitable base poses to reach a target from a particular orientation. However, when it comes to calculating reachability within a target region instead of a single target pose, this method is difficult to use. A technique to generalize the experience of a successful grasp through exhaustive search from different robot poses is presented in [68] by Stulp, Fedrizzi and Beetz. The result of grasps in different positions is classified and used to extract the best base position for a successful grasp.

In [31] Jamone et al. introduced *Reachable Space Map* which the robot learns autonomously and online during the execution of goal-directed reaching movements. This map can be used to estimate the reachability of a fixed object and to plan preparatory movements.

Yang et al. [74] proposed a methodology for automatic reaching analysis of wheelchair users in an indoor environment. Their method is based on a simple model of a person sitting in a wheelchair and an efficient motion planner.

The presence of obstacles in an environment opens up new challenges to the existing methods which work based on inverse kinematics without considering obstacles. The capability map changes when an obstacle is close to the robotic arm. This is because the space needed for the arm such that its end-effector reaches a particular pose might be blocked by an obstacle. Recreating the capability map for a different arrangement of obstacles is a computationally expensive procedure. In [68] the exhaustive search should be repeated since the grasp map of the environment changes by adding an obstacle. To overcome the limitations of the existing works, in chapter 3 we propose a method based on motion planning which does consider obstacles around a target.

Performing the Manipulation

The canonical approach for robot manipulation involves decomposing the task into several well-defined components such as state estimation, planning, and low-level control, which often have further subcomponents such as a model of environment's dynamics in the planner. Recent advances in deep learning led to solutions where some of these components are implemented as neural networks. Examples include object pose estimation [55], robot pose estimation [85]. In some cases, more than one component is implemented with neural networks, such as the inference of grasp and suction affordances and recognizing novel objects [83].

Taking this trend to the extreme are end-to-end learned deep visuomotor policies where all or almost all the canonical pipeline is implemented as a *single* neural network, transforming in one step a visual input into the control signal [43, 57]. The challenge, of course, is that we need to learn a large, opaque neural network, losing the advantages of decomposition and module testing. A deep visuomotor policy for robotic manipulation transforms an input video stream (possibly combined with other sensory input) into robot commands by the means of a single deep neural network. Such a system had been first demonstrated in [43] using guided policy search [42], a method that transforms policy search into supervised learning, with supervision provided by a trajectory-centric reinforcement learning method. In recent years, several alternative approaches have been proposed using variations of both deep reinforcement learning and deep learning from demonstration (as well as combinations of these).

The two major learning models of end-to-end policies are reinforcement learning (RL) and learning from demonstration (LfD). Deep reinforcement learning is a powerful paradigm that, in applications where exploration can be performed in a simulated environment allowing millions of trial runs, can train systems that perform at superhuman level [64], even when no human knowledge is used for bootstrapping [64]. For RL, the task is specified through a reward function and requires the robot to interact with the environment to collect training data. An approach to avoid the requirement of reward engineering by periodic querying of the user is shown in [65]. Unfortunately, for training visuomotor policies controlling real robots, it is very difficult to perform reinforcement runs on these scales. Even the most extensive projects could only collect several orders of magnitude lower number of experiments: for example, in [44] 14 robotic manipulators were used over over a period of two months to gather 800,000 grasp attempts. Even this number of experimental tries is unrealistic in many practical settings. Thus, many efforts focus on reducing the number of experimental runs necessary to train an end-to-end visuomotor controller. One obvious direction is to learn a better encoding of the input data, which can improve the learning rate. In [21], a set of visual features were extracted from the image to be used as state representation for the RL algorithm. The state representations can also be learned using priors from the physical world [33] or being able to predict the future [66, 7, 6] or transferring the learned behavior from previously learned tasks [38, 46, 58].

To avoid real-world trials some studies chose to gather the training data in simulation with testing done in the same simulated environment [18, 57, 63, 62] or the learned behavior will be transferred to a real robot or agent [84, 55]. Despite reducing the number of real-world interactions, training the policy using interactions in a simulated environment has its shortcomings. naturally, the simulated environment cannot represent the real-world conditions accurately in all cases. In [20] the simulator is constantly going through modifications. After each training episode in the simulated environment, the newly trained behavior is deployed on the robotic agent in the real world. A machine learning algorithm will optimize the simulator by comparing the expected result in these real-world trials. Instead of closing the gap between the simulator and the real world, one can train a powerful policy that can generalize well enough to correct its own mistakes when such discrepancies happen between the real world and the simulator using domain randomization [70, 15, 30]. Authors in [55, 24] relied on the imperfect nature of demonstrations performed by humans and their ability to recover from them.

Another alternative is to reduce the number of real-world trials is to use LfD to bootstrap RL and overcome exploration challenges [57, 84]. In [61] Deep Q-learning from Demonstrations (DQfD) is introduced to greatly accelerate the RL learning process by pre-training a Q-function on the demonstration data. Alternatively, demonstrations can be used to bias the exploration phase of RL in a process called reward shaping [10] or Exploration from Demonstration (EfD) [69].

LfD has the advantage that it does not require an engineered reward function, the tasks being specified by demonstrations performed by the user. In a very primitive setting, using only a behavioral cloning/imitation loss, LfD would be limited to reproducing the trajectories of the user, with fast divergence whenever a new situation is encountered. In practice, many techniques can learn well-generalizing policies from demonstrations by combining the imitation loss with other losses, regularization techniques and other ways to introduce inductive bias. Just like in the case of RL, LfD also needs a relatively large number of demonstrations done by an expert. Crowd sourcing [22] or cloud-based [35] approaches can facilitate the data gathering process. However, it is desirable to reduce the number of demonstrations required to learn a new task. To reduce the number of demonstrations for a new task, the deep visuomotor policy should be able to take advantage of the knowledge previously acquired in learning other tasks. In addition to reducing the number of demonstrations for each task, multi-task learning can also act as a regularizer and greatly improve the generalization among different tasks [56, 54]. Another possible approach for this is meta-learning: by utilizing demonstrations from a variety of tasks, under some conditions it is possible to learn a new task from a single demonstration [79].

In the general LfD methods, the experts' influence is only passed to the policy through the demonstrations. Consequently, LfD methods cannot be guided by the expert during test time. Such direct control over the network's decision making is preferable and in some cases necessary. Authors in [16] proposed an approach to condition models trained by LfD on high-level commands at test time. The model is trained to drive a car in a simulation environment and transferred to drive a toy truck in the real environment. The trained policy is able to drive a car and it is still responsive to real-time navigational commands at the test time. Authors in [14] proposed a goal-parameterized visuomotor policy. The policy is trained a subset of all available goals and their corresponding target parameters. The policy receives RGBD images as input to its vision module, additionally, a vector parameterizing the task specification is concatenated to the control module input. The trained policy exhibits zero-shot generalization to unseen target parameterizations. The policy is evaluated on peg-insertion and button-pressing tasks and two robotic arms.

Authors in [45] proposed Learning from Play (LfP) as a way to scale up multi-task learning. As illustrated in both [56, 45] visuomotor policies trained on multiple tasks are superior to policies trained on individual tasks in terms of success rate and robustness. The policy proposed in [45] is task-agnostic, goal conditioned trained on raw unstructured play data record in a simulation environment. The policy is conditioned on its two inputs the current state and the goal state. Despite being task-agnostic authors observed the latent space is conditioned and organized on the nature of the task. In [2] we achieved natural failure recovery with task-focused attention, LfP also exhibits similar behavior.

Robots are expected to perform a wide variety of tasks. To maximize the generalization ability of the policy, longer tasks can be decomposed into smaller well-defined subtasks. Authors in [34] proposed a Deep Neural Network deployed on a humanoid robot capable of decomposing the put in the box task into short sequential subtasks. The proposed network consists of an autoencoder which extracts features from the input images and a Recurrent Neural Network (RNN) based controller.

In robotic studies, the training and testing usually happen on a single robot. Authors in [36] proposed a framework trained on raw RGB images and later transferred the knowledge to a different robot. The authors used 659 demonstrations gathered for a cleaning task in[12]. Camera transformation and random changes in illumination and adding noise to the background tricks are used to expand the recorded dataset and reduced the demonstrations required to achieve the same test error to 20% of the original dataset.

Cooperation between robots is investigated in [60]. The authors used 4 robots, 2 master robots used by the human demonstrator to control the other 2 slave robots to perform the task of serving a salad. The tasks are accomplished by scooping the salad from the initial position and transporting it to the serving plate. An LSTM based network is then trained on the demonstrations.

Reward sketching, a new annotation technique used to train reward function for RL, is proposed in [11]. Reward sketching utilizes humans' ability to assess task progression toward a goal state and use it to obtain reward functions. An intuitive and user-friendly interface is used to gather human experts' per timestep annotations by drawing a curve correlated by the task progression. The interface enables a single expert to annotate hundreds of frames per minute. The trained policy reportedly is able to perform tasks faster than a human which is not possible in pure imitation learning settings.

Training end-to-end visuomotor policies using sequences of raw unannotated images is illustrated in [56, 43, 2, 1]. However, additional information regarding object affordances (available actions to an agent to perform on an object) can be utilized to reduce the number of required demonstrations. Such information can also facilitate learning dexterous actions on a wide variety of objects. Masnadi et al. [48, 47] proposed a graphical user interface for users to sketch the object affordances. In [53] authors introduced an agent that is designed to efficiently use the demonstrations and operate in partially observable environments with highly variable initial conditions and sparse rewards.

Instead of the end-to-end training of visuomotor policies, one can train a network to detect the target object's position and then train a separate network to use the estimated position to control the network. Such a network is proposed in [29]. To avoid collecting lots of images in the real world, authors proposed to use 2 VAEs one on the simulation images and one for the real images. The decoder is shared between the two VAEs. Since the object's position in the real and synthetic images is the same, the reconstruction error for both real and synthetic images is calculated with respect to the synthetic image. Now that both real and synthetic objects are mapped to the same space a separate CNN network is used to estimate the object's position. The estimated position is used to train LSTM layers to control the robot to perform a pick and place task.

Here we are going to discuss a paper [56] which proposed a method for teaching a low-cost robotic manipulator to perform several complex picking and placing tasks, as well as non-prehensile manipulations. The proposed architecture provides the foundation for architectures described in Chapter 4 and Chapter 5.

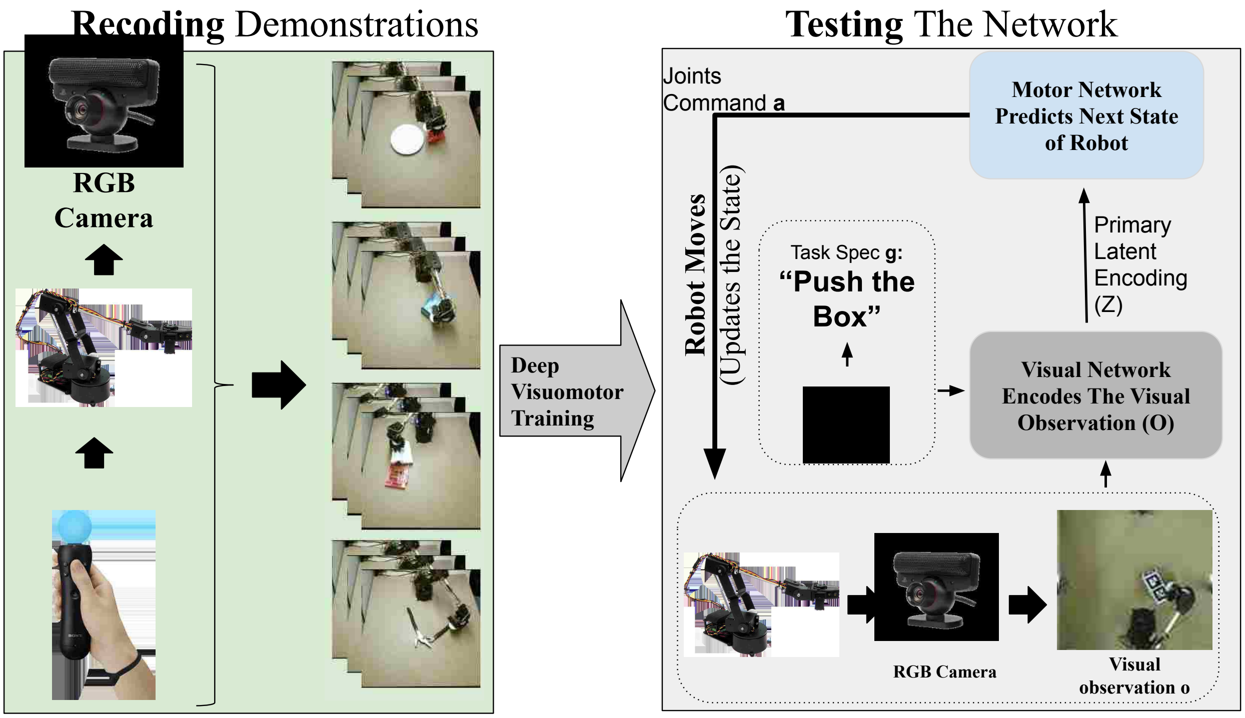

Recoding Demonstrations
RGB Camera
Deep Visuomotor Training
Testing The Network
Joints Command a
Motor Network Predicts Next State of Robot
Robot Moves (Updates the State)
Task Spec g: "Push the Box"
Primary Latent Encoding (Z)
Visual Network Encodes The Visual Observation (O)
RGB Camera
Visual observation o

Figure 2.1: The first step is to record demonstrations for all of our target tasks. Next, the visual and motor networks are trained using the recorded demonstrations. In the testing phase, the trained network is deployed on the robot. In each timestep, the visual network will receive a new image from the camera and generate a latent encoding which will be passed to the motor network. The motor network will predict the next joint commands for the robot and the cycle repeats until the manipulation is successfully performed [56].

As shown in figure 2.1 the proposed method in [56] involves three different stages: 1-Recording the demonstrations 2- Training the network 3- Testing the network on the robot. To use Learning from demonstration, an expert is needed to control the robot and perform the manipulation tasks while image sequences from the scene and commands to the robot are being recorded. These recorded commands and image sequences will be used as demonstrations for the robot to learn from.

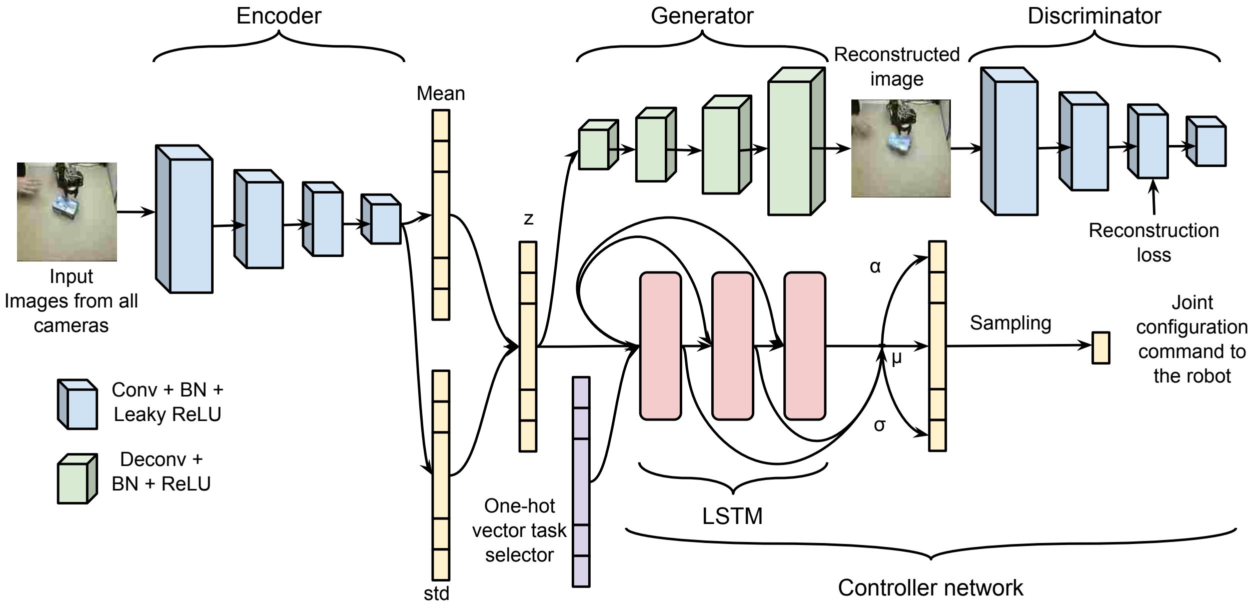


Figure 2.2: Proposed architecture for multi-task robot manipulation learning in [56]. The neural network consists of a controller network that outputs joint commands based on a multi-modal autoregressive estimator and a VAE-GAN autoencoder that reconstructs the input image. The encoder is shared between the VAE-GAN autoencoder and the controller network and extracts some shared features that will be used for two tasks (reconstruction and controlling the robot) [56].

The proposed architecture has two main components: vision and motor network. The vision network's responsibility is to extract a set of features for the controller network to control the robot. The training process happens in two stages. First, both vision and controller networks are trained together. Then, the trained vision network is used to extract the features and save them. The extracted features will be used to further fine-tune the controller.

The testing process involves feeding an image captured from the state of the current scene and extracting the features using the trained vision network. Then these features will be processed by the motor network and we get the next joint command to be performed by the robot. After the robot performs the new command, the state of the scene will change and a new image will be captured and this loop will continue until the desired task is successfully performed.

For more details regarding recording demonstrations and description of tasks and results please refer to the original paper [56]. Here we are going to discuss each component's role, importance, and background in the proposed architecture.

Encoder - Feature Extraction

Inputs to the network are batches of sequences of images and a task selector vector. Convolutional Neural Networks(CNNs) are an obvious choice for processing the images. However, using a separate controller consist of LSTM layers is not as obvious and it is not the only solutuion [43]. Separating the controller and the encoder modules enables us to employ the two-stage training process mentioned before and fine-tune the controller in the second stage. The controller's ability to capture long-term dependencies in its inputs is directly correlated to increasing the sequence length during training. The fine-tuning stage enables us to increase the sequence length because the encoder is not trainable at this stage and the latent encodings can be extracted beforehand, eliminating the need to load images into the GPU.

Autoencoders

The simplest network able to control the network consists of an encoder and a controller. Such a network is expected to produce proper gradients to train its convolutional filters and LSTM layers to directly infer proper commands at each time-step using only the input images, which is not a trivial task. Having an additional loss function to guide the network, especially at the beginning of the training process is preferable. An autoencoder can provide us with such a loss function. The reconstruction error between the generator's reconstruction and the original image eliminates the pressure on the encoder to extract the most relevant features for the controller by forcing it to extract all features to reconstruct the input. The autoencoder's architecture is a Variational Autoencoder and benefits from a reparameterization trick first introduced in [37].

Generative Adversarial Networks

Blurriness is unavoidable when using Mean Squared Error(MSE) in the autoencoder's reconstruction error. More details in the reconstruction indicate the latent encoding ability to transfer more information to the controller. Information regarding the target object's edges is among the vital information for the controller to better perform the tasks. Generative Adversarial Networks(GANs) are known to produce realistic results compare to other classes of Neural Networks like VAE's. The proposed architecture in [56] used VAE-GAN [40] to extract better features. The vision network is a VAE-GAN [40], which will receive the input image from the scene and returns a low dimensional feature vector. The extracted features will be fed to the generator. The generator's job is to reconstruct the original image. Reconstructing the input image will help the network to retain all the relevant information including the robot's position, gripper status, object position, etc. The reconstructed images will then be fed to the discriminator. Based on the idea of generative adversarial networks [25] the Discriminator's job is to distinguish the fake images generated by the generator from the real images captured from the scene (input images). The extracted features are being fed to the controller network as well.

Mixture Density Networks

Mixture Density Networks(MDN) are used in variety of problems such as handwriting prediction [26], generating raw audios [52], speech synthesis [82], sketch drawing [27] and generally any multi-modal sequence generation [26]. They are also applicable to our robotic problem. There are many equally possible solutions to perform a manipulation and reach the same end state. Predicting parameters of a multi-modal distribution enables the network to learn such demonstrations. For example, imagine a scenario in which an object should be pushed to the right but there is an obstacle on its shortest path to the target, in this case, some of the demonstrations might prefer to avoid the obstacle by pushing the object in different ways. Averaging all these demonstrations might result in colliding with the obstacle. Even without an obstacle in the way, since humans are not consistent about which solution to choose, having a multi-modal distribution for the robot's controller output will help the network to avoid excessive averaging between different modes in the demonstrations. That is why the controller network predicts the probability distribution of the joint angles instead of predicting the next joint angle itself. The controller network, is a 3 layer-normalized [3] LSTM [28] layers with skip connection. The controller network predicts the next robot's joint angles given the features extracted by the encoder. The output of the controller network is a mixture of Gaussians[5]. The controller network's loss is calculated according to the Mixture Density Network(MDN) negative log-likelihood loss formula over the supervised data (J) based on the demonstrations (behavioral cloning loss). The behavioral cloning loss is discussed in more detail in Chapter 4.

The features extracted from the input image are shared between the VAE-GAN and the controller network as illustrated in Figure 2.2. The generator and the discriminator are trying to force the encoder to include expressive features for the reconstruction task whereas the controller network is trying to force the encoder to focus on features that will improve the trajectory prediction task.

This competition/collaboration will regularize the feature extraction. The VAE-GAN approach for extracting low-dimensional features from input frames will be the foundation for our architectures in Chapter 4 and Chapter 5.

Attention Mechanisms

The discussed architecture above can be trained using clean, undisturbed demonstrations in which only the target object(s) are present in the scene and nothing else. Consequently, its ability to perform a task autonomously is also limited to such conditions. In our efforts to robustify deep visuomotor policies against physical and visual disturbances such as clutter, we used spatial attention. Different types of attention mechanisms including spatial, temporal and feature level attentions often appear as components of larger networks solving problems like image captioning [72, 76], visual question answering [75, 50, 77], visual text correction [49] or visual expression localization [78]. Although the applications are different, the role of attention networks, i.e., focusing on information-rich parts of the input, remains the same. Our proposed attention mechanism is most similar to [75]. However, in our architectures, we train the attention network with word selection objective. The objective is to select some regions on a video frame regarding a textual input, such that it be able to regenerate the words in the input sentence just based on the visual features of selected image regions.

An approach that is similar to ours in objective, but different in implementation, is described in [17]. Considering manipulation tasks, the authors implement two layers of attention. The first, task-independent visual attention semantically identifies labels and localizes objects in the scene. This labeling relies on training on an external labeled dataset, thus in this respect, the approach is not “end-to-end”. The second, task-specific attention is learned by selecting from the segmented objects, by the task-independent attention, those objects that contribute most to the correct prediction of demonstrated trajectories. The following are main differences between our work presented in Chapters 4 and 5 and the work presented in [17]:

1- Our model supports multi-task learning by jointly encoding the textual command(e.g. “pick up the blue ring/red bowl”) and video frames; however, to the best of our knowledge, each task in [17] requires an additional training process. Moreover, tasks defined in our work are more generalized in the sense that the target object may be different. For example, in [17], the target in all the demonstrations for the first task is a brown mug, however, we can perform tasks, like “pick up”, on multiple objects by just mentioning the object name in the textual command sentence.

2- The attention proposed in [17] relies on Region Proposal Networks (RPN), pre-trained on MSCOCO dataset, and has an extra stage of training to remove extra proposed bounding boxes. Our attention mechanism does not require any region proposals and is trainable end-to-end just using the textual commands and recorded videos of the robot.

3- Our method can ignore the disturbances without additional demonstrations. We record the demonstrations only in benign condition, and we do not have any demonstration in the training set containing any kind of disturbance. Note that, the tasks (e.g. Pick-up vs Pouring) and the environment setup (e.g. robot model and objects) in our work and [17] are fundamentally different.

CHAPTER 3: INTRODUCING THE EASE-OF-REACH SCORE

Let us consider a scenario where a robotic arm positioned at location (x, y) aims to grasp an object c in an environment with obstacles $O = \{o_1, \ldots, o_n\}$ that the arm must avoid. To measure how suitable a certain location is to perform a manipulation from, we will define the *Ease-of-Reach Score (ERS)* such that it captures our intuitions about the preferences over different positions $p(x, y)$. The value of ERS should be 0 for positions from where the grasp is not possible, while 1 for the position from which the grasp can be done under “ideal conditions”. As we want to make the ERS independent of the different human or machine motion planning algorithms, we will base our metric on the *number of distinct grasps possible* from a given position. For instance, if from a given position we have 10 different ways to grasp the object, this position will likely be preferred both by the human and the automatically controlled operator. This position would be preferred to one where there is only one possible grasp that the operator would need to get exactly right to successfully complete the task[1].

Let us now develop a numerical formula for the ERS. We call *Count of Distinct Grasp Trajectories* $CDGT(p, c, O)$ the number of ways the arm can approach an object c to grasp it from base position $p = p(x, y)$ in the presence of the obstacles $O = \{o_1, \ldots, o_n\}$. To discretize the number of grasp poses, we will consider two grasps to be distinct if the approach angle differs by at least $\pi/4$. For the case of our running example, Figure 3.1 shows 17 distinct grasps for the cylindrical cup. As a note, obstacles lower or at best keep the CDGT the same, because they make a previously feasible grasp impossible to achieve.

[1] This chapter’s material has been previously published in 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)2016 under the name ”Real-time placement of a wheelchair-mounted robotic arm” authors: Pooya Abolghasemi, Rouhollah Rahmatizadeh, Aman Behal, Ladislau Bölöni

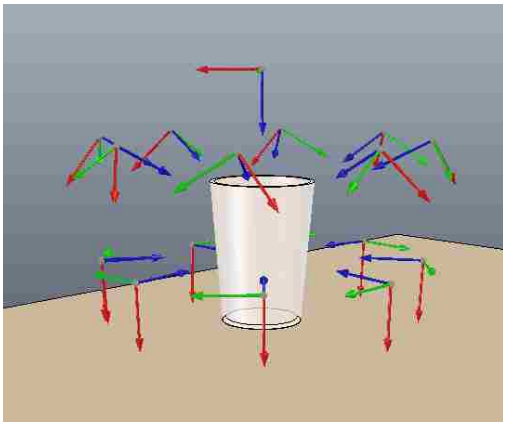


Figure 3.1: The cup placed on the table can be grasped using 17 reachable grasp poses. The blue arrow shows the direction in which the arm approaches the cup. The grasp poses from the bottom are blocked by the table.

$$\forall p\, \forall o_{n+1}\, CDGT(p, c, O \cup o_{n+1}) \leq CDGT(p, c, O) \tag{3.1}$$

The ideal condition for a grasp is an environment with no obstacles and a position from where we can choose the largest number of possible grasps. Starting from these considerations, we will define the ERS as:

$$ERS(p, c, O) = \frac{CDGT(p, c, O)}{\max\limits_{p} \left(CDGT(p, c, \emptyset)\right)} \tag{3.2}$$

Similar to the CDGT, adding new obstacles to the environment will lower or at best keep the ERS the same:

$$\forall p\, \forall o_{n+1}\, ERS(p, c, O \cup o_{n+1}) \leq ERS(p, c, O) \tag{3.3}$$

The best position for the arm is the one where the ERS is maximized:

$$p_{opt} = \arg\max_{p} ERS(p, c, O) \tag{3.4}$$

Figure 3.2 shows the ERS for a scenario with three obstacles and the wheelchair positioned in such a way that the base of the manipulator is at the maximum ERS. Note that the optimal position might not be reachable (due to the fact that the wheelchair on which the robotic arm is mounted has its limitations, for instance, it might collide with the table).

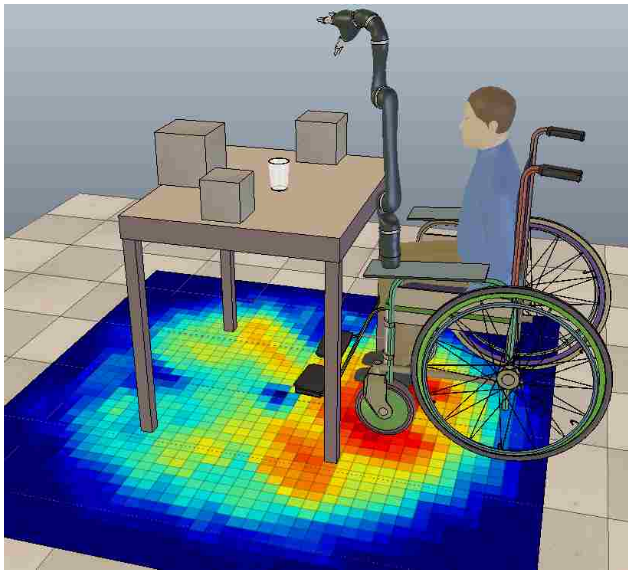


Figure 3.2: ERS for a scenario with three obstacles. In the heatmap, blue represents low ERS, while red represents a high ERS. The wheelchair is positioned such that the robotic arm is located at the maximum ERS.

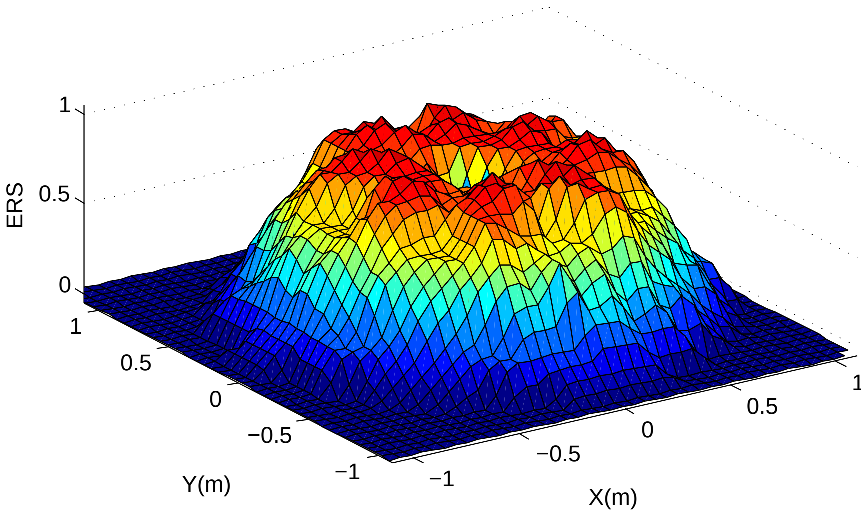


Figure 3.3: ERS for a small object, here a cup, located in position $p = p(0, 0)$, computed using brute-force computations. Blue: low ERS, red: high ERS.

Estimating the ERS

The brute-force calculation of the ERS requires us to solve the motion planning problem for every grasp angle and to repeat this for every point in a grid covering the possible locations of the robot arm. The computational effort depends on the resolution of the grid, but even a very coarse grid (eg. 20 by 20) yields $20 \cdot 20 \cdot 17 = 6800$ motion planning problems. We use Rapidly-exploring Random Trees (RRTs) [39] to find an obstacle-free trajectory to reach a grasp pose close to the target. Even with this fast method, calculating the exact ERS before every decision is not a feasible approach for a realtime solution of the TP task.

Calculating the ERS offline is feasible if there is no obstacle to consider. For instance, Fig 3.3 shows the ERS calculated using this method for a cup positioned at $(0, 0)$ without any obstacles around it. As expected, the ERS has a ring shape - the reach is difficult both if the arm originates too far or too close to the object. The maximum ERS, for this setup, is reached at the distance of 0.5m from the object. Note that this calculation needs to be done only once and is valid for any small object that is graspable by the robotic arm since ERS is agnostic to the shape of the grasp target. The robot can store this map, and recall it whenever it needs to perform the TP task. However, the ERS also depends on the number, location, and size of the obstacles, thus the presence of obstacles leads to a combinatorial explosion of the possible maps. In our setup with n obstacles and 10 different obstacle sizes, the number of maps is $(10 \cdot 20 \cdot 20)^n$, that is $1.6 \cdot 10^7$ for two obstacles and $6.4 \cdot 10^{10}$ for 3 obstacles. It is thus desirable to find a way to quickly estimate the ERS without the need to compute extensive offline libraries. The approach we propose starts from the observation that the maximum ERS is obtained when no obstacles are present, while each obstacle reduces the ERS.

Let us assume that the robot arm is located at an arbitrary position (x, y) and its goal is to reach a particular object while avoiding the obstacles. Obstacles block some of the space it needs to reach the target. The blocking effect of an obstacle depends on the size and shape of the obstacle, its distance to the arm, etc. However, we do not know how much an obstacle will affect the ERS at a particular location of the arm. Furthermore, we do not know how the blocking effect of adjacent obstacles close to the target combines to lower the ERS. We propose an approach that starts from the observation that the ERS for each point in the environment can be estimated considering some general information about the point and its surroundings.

In order to find the value of ERS at a certain location of the arm, we set up a simulation scenario in which a target object is surrounded by 1 to 9 obstacles. The obstacles in the environment are approximated using cuboids with random size and pose. The dimensions of the obstacles can vary between 10-35cm and their distance to the center of the target can vary between 10-100cm. By running the simulation multiple times and measuring the ERS at different locations, we gather the information required for the learning process. In order to learn how the ERS is affected in different situations, we hand-crafted a set of features which intuitively capture the aspects of that situation. In other words, these features are designed in such a way to classify the points on a grid with similar ERS values in the same group while distinguishing between the points which have different ERS values.

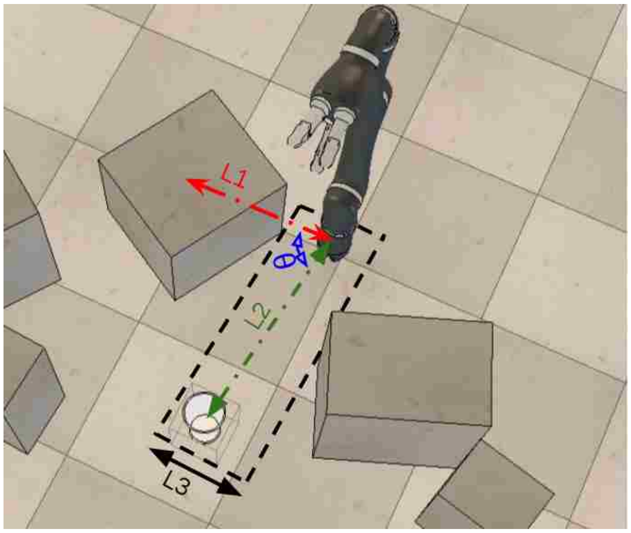


Figure 3.4: Demonstration of features in an example scene. L1: distance to the nearest obstacle, L2: distance to the target, L3: the widest collision-free path from the arm to the target, and θ: the angle between L1 and L2.

The features we found useful are illustrated in Figure 3.4 and explained here:

- *Distance to the closest obstacle (L1).* This feature captures the Euclidean distance between the closest obstacle and the current position of the arm. It is preferred for the arm to be positioned in the furthest possible position from any obstacle. The closest obstacle seems to be the most important one since it probably has the largest effect on ERS.

- *Distance to the target (L2).* The Euclidean distance between the arm and the target which is very important since one of the key factors which affect ERS is how far the arm is located from the target. For instance, when the arm is located far away from the target, reaching the target is not possible, i.e. the ERS is zero.

- *Angle between L1 and L2 (θ).* The angle between L1 and L2 along with the distances L1 and L2 capture the configuration of the arm, target, and the closest obstacle with respect to each other. When the obstacle is located between the target and the arm, the ERS should be lower compared to the situation where the obstacle is located outside the space between the arm and the target. This feature tries to distinguish these situations from each other.

- *Widest path (L3).* It measures the diameter of the widest rectangular obstacle-free path from the arm to the target. This feature tries to estimate how much free space is available for the arm to operate while reaching the target. The wider this path is, the more space the arm has to operate and the result would be a higher ERS.

- *Force field.* In addition to the closest obstacle, other ones also affect the ERS more or less. To estimate the effect of the n closest obstacle, we use a sum of force fields type formula:

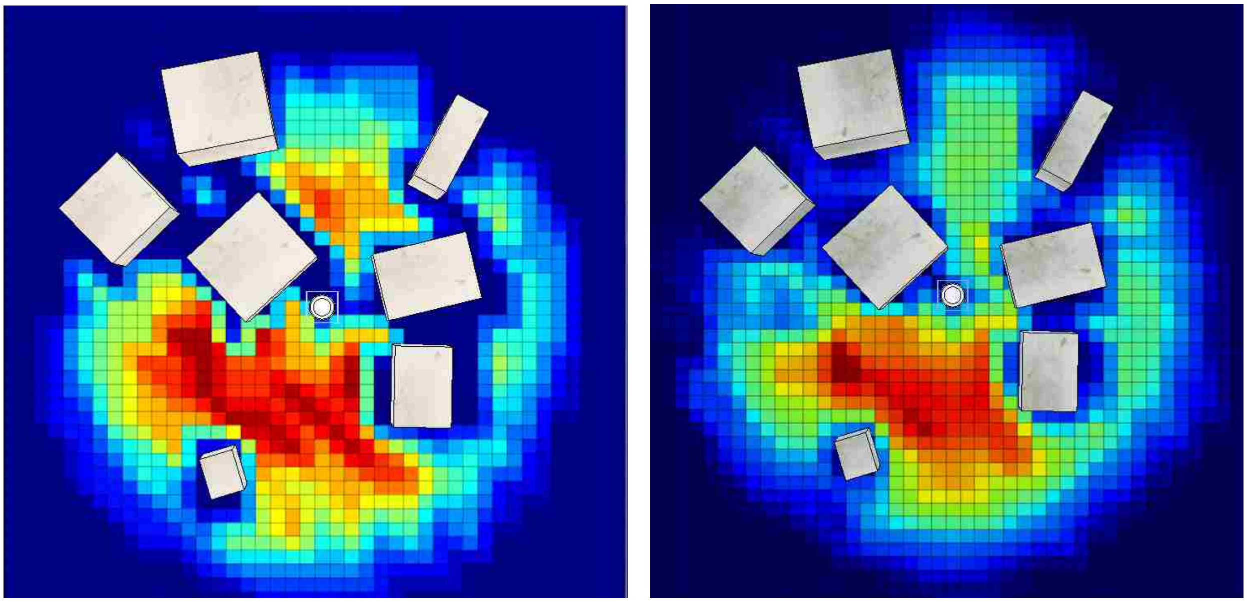
  $$FF(p, O) = \sum_{i=0}^{n} \frac{Volume(o_i)}{Distance(p, o_i)^2} \tag{3.5}$$

  This formula aims to capture the fact that the larger the size, number and proximity of the obstacles to the target object, the more and more difficult the finding of a grasp becomes.

These features had been obtained by experimentally testing an initially larger set and eliminating those that had been found ineffective in reducing the estimation error. Using the data gathered by running the simulation 100 times, we calculate each feature for all the points on the grid to later be used in the learning algorithm.

The cardinality of training data is not merely the number of training scenes, but the number of scenes multiplied by the number of candidate arm position in the scene. For instance, by discretizing the possible positions for the arm into a $(20 \cdot 20)$ grid and repeating the simulation to generate 100 scenes, we gather 40,000 training examples. Calculating the features for each point in the environment provides us with a large number of training samples. This is one of the advantages of the proposed approach which makes the learning reasonably fast without requiring too many simulated environments.

The value of the explained features calculated for each training example and the resulted ERS forms the input to the machine learning algorithm. The output would be a predictor which is able to estimate the ERS based on the features. We use the CART algorithm [9] to create 200 regression decision trees. We use decision trees because decision trees properly handle this kind of non-leaner problems without relying on a large number of training examples. Finally, to enhance the accuracy of the results, we use the bootstrap aggregating ("bagging") method [8] to predict the value of the ERS.



(a) The actual ERS of the cup.

(b) The estimated ERS of the cup.

Figure 3.5: An example scene with three obstacles and heatmaps corresponding to ERS_{act} (upper) and ERS_{est} (lower). In the heatmaps, blue corresponds to low and red to high ERS values.

Experiments and Results

In this section, we validate the proposed approach, by investigating how well the approach solves the positioning problem. We simulate a JACO arm mounted on a wheelchair in V-REP [23] robotic simulator. The techniques proposed in Section 3 allow us to calculate the estimated value ERS_{est} while brute force methods allow us to calculate the actual value ERS_{act}. Figure 3.5 shows an example scenario with a target object (cup) and seven obstacles of various sizes and various orientations. The figure on the top shows the actual ERS as a heatmap under the obstacles, while the bottom figure depicts the estimated ERS. A visual inspection of the figures shows that although not perfect, the approximation and the actual value of ERS are very close.

Let us now try to develop a useful error metric. One approach would be to calculate the average error for every grid point. However, this would be a misleading metric, because for a large number of locations ERS will be trivially zero (for instance, the ones that are located outside the range of the arm). If we calculate the simple average, the error would depend on how far the grid extends from the origin. Instead, we will define the error metric as being the average only for locations where at least one of the ERS_{act} and ERS_{est} are not zero:

$$P = \{p \mid ERS_{est}(p, c, O) > 0 \vee ERS_{act}(p, c, O) > 0\} \tag{3.6}$$

We denote the cardinality of this set of points with $\#P$. Thus for a given target object c and set of obstacles O we define the average relevant error $ARE(c, O)$ as follows:

$$ARE(c, O) = \frac{\sum\limits_{p \in P} \Big| ERS_{est}(p, c, O) - ERS_{act}(p, c, O) \Big|}{\#P} \tag{3.7}$$

Table 3.1: The average relevant error ARE for 700 test scenes.

Scene description	**count**	ARE
1 obstacles	90	0.0276
2 obstacles	79	0.0343
3 obstacles	81	0.0395
4 obstacles	73	0.0435
5 obstacles	67	0.0471
6 obstacles	73	0.0507
7 obstacles	73	0.0544
8 obstacles	73	0.0571
9 obstacles	91	0.0602
	Average	0.0458

The calculation of the ARE is computationally expensive as it requires the calculation of the ERS_{act}. We performed it for 700 test scenes as described in Table 3.1. The first column describes the number of obstacles in the scenes and the second column shows the total number of test scenes with that number of obstacles. The table shows that on average the error stays in a moderate range, but in general increases proportional to the number of obstacles.

From the point of view of a practical robot implementation, however, the error in the ERS is not of high importance. The robotic wheelchair needs to position itself such that the object is easily reachable - the finding of the optimal position is of comparatively small importance. We can divide the errors in the determination of ERS into three major types:

Type 1: $ERS_{act} > 0 \wedge ERS_{est} > 0 \wedge ERS_{act} \neq ERS_{est}$

If this type of error occurs, the system might choose a position which is not exactly the global optima. Practically, this kind of error is not a major issue as long as the difference between the actual value and the estimated one is not considerable. By trying to keep the total error minimized, the chosen position should fall into a small range of the optimal position.

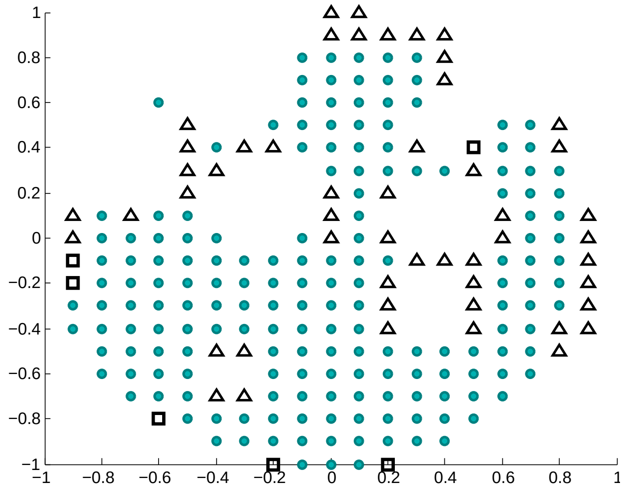


Figure 3.6: The distribution of Type 2 and Type 3 errors by type in a scene with three obstacles. The sign circle denotes locations where both ERS_{act} and ERS_{est} are positive. The grid points with no sign are the ones where $ERS_{act} = ERS_{est} = 0$. Type 2 errors are denoted with $\triangle$ while Type 3 errors with $\square$.

Type 2: $ERS_{act} > 0 \wedge ERS_{est} = 0$

This issue will happen when the system mistakenly estimates ERS to be zero while the actual ERS is not zero, i.e., reaching the target is possible from that location. As long as the ARE is not a large number, we can assume that these points were likely not a good choice for the arm to do the manipulation. However, in highly congested scenes or in scenarios with constraints on the movement of the wheelchair where there are limited options for the arm to select, the presence of Type 2 errors might make the system mistakenly believe that the problem is unsolvable.

Type 3: $ERS_{act} = 0 \wedge ERS_{est} > 0$

This type of error relates to the positions where the grasp is not possible but the estimation suggests otherwise. As a result, if the wheelchair chooses this kind of points to execute the positioning task TP, it would be impossible to perform the grasp task TG. Practically, by performing motion planning for the chosen points, these kinds of errors can be avoided. However, this type of error is the most critical one and should be minimized.

Figure 3.6 depicts the distribution of Type 2 and Type 3 errors in a sample scenario with 7 obstacles. The structure shows that the estimate yielded correct or acceptable values for the majority of positions (but for feasible and unfeasible locations). In a few positions in which Type 2 or Type 3 occurs, all of them located at the boundary between the feasible and infeasible regions. In practice, the system would choose positions at the interior rather than at the boundary of the feasible zone, thus avoiding both types of errors.

CHAPTER 4: ROBUSTIFYING A DEEP VISUOMOTOR POLICY THROUGH TASK-FOCUSED VISUAL ATTENTION

In the related works (Chapter 2), we discussed an architecture which is able to control a robot and successfully perform a set of demonstrated manipulation tasks. In this chapter, we are going to discuss how to robustify our Deep Visuomotor architecture through Task-Focused Visual Attention. Moreover, We are going to discuss the networks' architecture and loss function in greater details[1].

While most demonstrations (supervised data) had been made in unstructured but relatively benign environments, our experiments and personal communication with other researchers had shown that task-independent visual networks for visuomotor policies are highly vulnerable to *physical* and *visual* disturbances. An example of physical disturbance is the robot arm being bumped such that it drops the manipulated object. The desired behavior would be for the robot to immediately notice this, change its trajectory, pick up the dropped object and continue with the manipulation task. Instead, with an otherwise reliably performing policy, we notice situations where the robot arm, having lost the object, continue to go empty-handed through the full trajectory of the manipulation, recovering either much later, or not at all. A visual disturbance may involve distracting mobile objects appearing in the robot's field of view. If the visual disturbance prevents the execution of the task, for instance, by blocking the view of the manipulated object, it is acceptable for the robot to stop or even cancel the manipulation. There are, however, visual disturbances that should not prevent the execution of the task: for instance, hands waving in the visual field of the robot but not covering the manipulated object or the robot arm. We have found that in the case of a task-independent visual network, even such visual disturbances cause the robot to behave erratically –

[1] This chapter's material has been accepted in 2019 Computer Vision and Pattern Recognition(CVPR) ”Pay attention!-Robustifying a Deep Visuomotor Policy through Task-Focused Attention” authored by Pooya Abolghasemi and Amir Mazaheri and Mubarak Shah and Ladislau Bölöni

possibly due to the robot interpreting the situation as a state never encountered before.

In engineered robot architectures such problems can be dealt with by developing explicit models of the possible disturbances, which may allow the robot to reason around the situation. In deep learning systems, one possible brute-force solution is to gather more training data containing physical and visual disturbance events; however, data collection for robotic tasks is time-consuming. Also, there are unlimited visual and physical disturbance scenarios for a single task. It is impossible to record demonstrations to cover all possible scenarios of physical and visual disturbances.

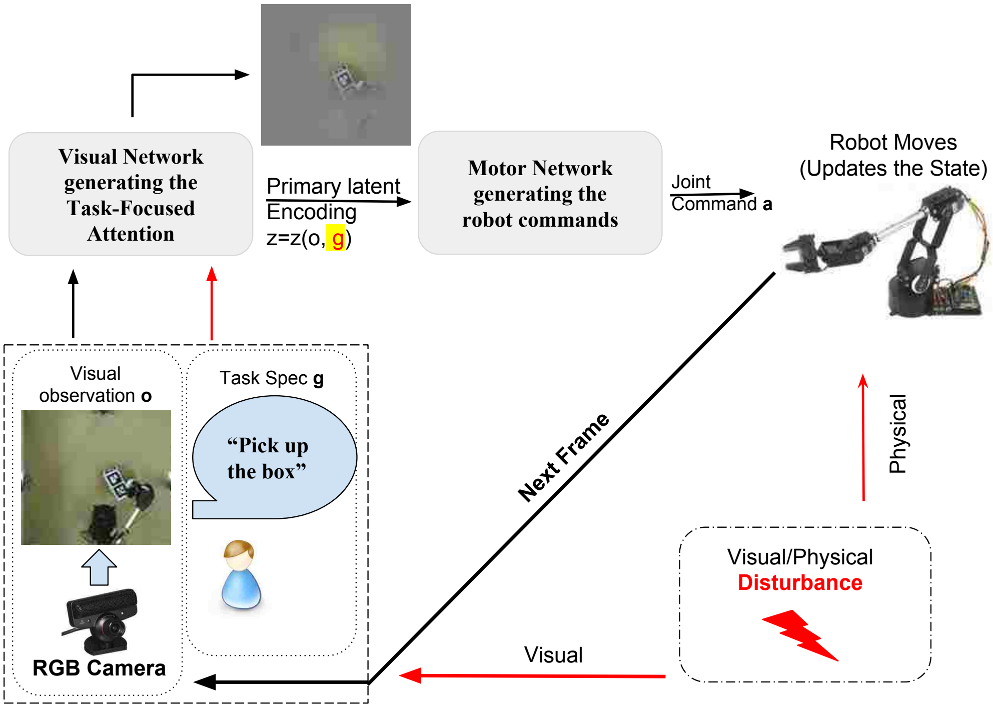

Visual Network generating the Task-Focused Attention
Primary latent Encoding z=z(o, g)
Motor Network generating the robot commands
Joint Command a
Robot Moves (Updates the State)
Visual observation o
Task Spec g
"Pick up the box"
RGB Camera
Next Frame
Physical
Visual/Physical Disturbance
Visual

Figure 4.1: The robot performs a given command. Our proposed network attends the image regions that matter the most and is robust to physical and visual *disturbance*.

Task Dependent Visual Network:

The principal idea of this chapter is that performance benefits can be obtained if we make the vision system pay attention to relevant regions of each frame regarding the current task or user command. Humans are known to exhibit selective attention - when observing a scene with a particular task in mind, features of the scene relevant to the task are given particular attention, while other features are de-emphasized or even ignored. This had been illustrated in the famous experiments of Chabris and Simmons [13]. In this chapter, we are going to propose *Task Focused (Visual) Attention* (TFA) as an auxiliary network to increase the robustness of the robot manipulator network to physical and visual disturbances, without the need of any additional training data. Thus, our objective is to create a system that implements selective visual attention similar to what human perception is doing: we want the robot to focus on the objects of the scene that are relevant to the current manipulation task. We conjecture that using TFA, $\mathbf{z}$ will better represent the objects and colors that are the subject of the attention, allowing for more precision in grasping and manipulation (See Figure 4.2).

The goals of this chapter are as follows:**1-** We describe a novel architecture for a visuomotor policy trained end-to-end from demonstrations, which features a task-focused visual attention system. The visual attention system is guided by a natural language description of the task and focuses on the currently manipulated object. **2-** We show that, under benign conditions, the new policy outperforms a closely related baseline policy without the attention model over pick-up and push tasks using a variety of objects. **3-** We show that in the case of a severe physical disturbance when an external intervention causes the robot to miss the grasp or drop the already grasped object, the new policy recovers in the majority of situations, while the baseline policy rarely recovers. **4-** We show that the task-focused visual attention allows the policy to ignore a large class of visual disturbances, that interfere with the task for the baseline policy. We show experimentally that the system exhibits the “invisible gorilla” phenomenon [13] from the classic selective attention test.

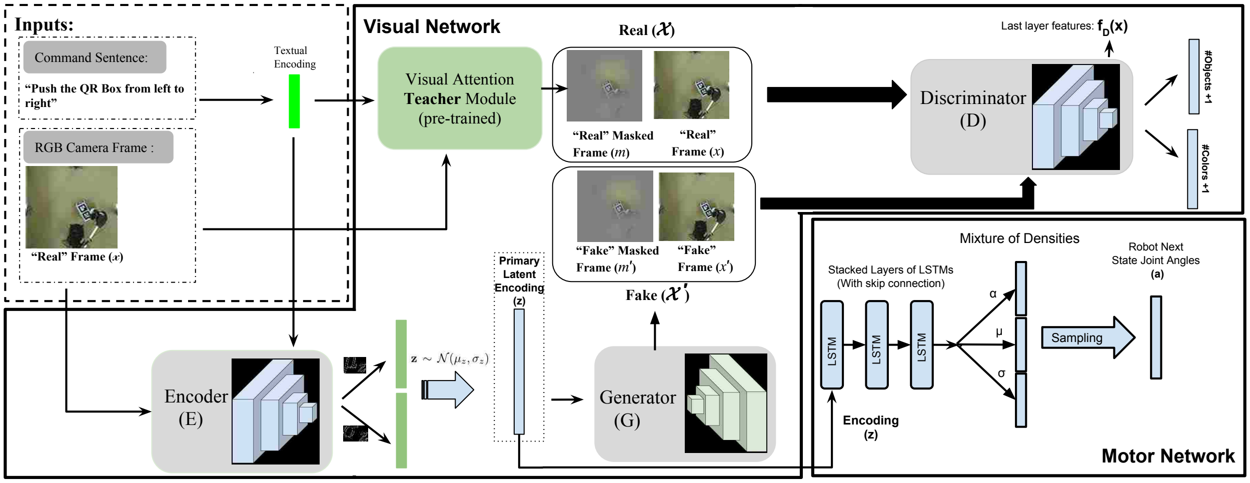

Inputs:
Command Sentence:
“Push the QR Box from left to right”
RGB Camera Frame :
“Real” Frame (x)
Textual Encoding
Visual Network
Visual Attention Teacher Module (pre-trained)
Real ($\mathcal{X}$)
“Real” Masked Frame (m)
“Real” Frame (x)
Last layer features: $f_D(x)$
Discriminator (D)
#Objects +1
#Colors +1
“Fake” Masked Frame (m')
“Fake” Frame (x')
Fake ($\mathcal{X}'$)
Primary Latent Encoding (z)
$z \sim \mathcal{N}(\mu_z, \sigma_z)$
Encoder (E)
Generator (G)
Mixture of Densities
Stacked Layers of LSTMs (With skip connection)
LSTM
LSTM
LSTM
α
μ
σ
Sampling
Robot Next State Joint Angles (a)
Encoding (z)
Motor Network

Figure 4.2: **The proposed visuomotor architecture**. Given an image captured from the scene and a command sentence provided by the user, the **Encoder (E)** produces the *Primary Latent Encoding (z)*. **z** is the input to the **Motor Network**, which decides the next state of the robot joint angles. Also, **z** is the input to a **Generator (G)**, which produces “Fake” frame and masked frame. A pre-trained Visual Attention **Teacher** Module masks the original frame by spatial attention computed employing the textual input. The **Discriminator (D)** must discriminate between real/fake frames and masked frames, and also classify the object and color of the object being manipulated.

As shown in Figure 4.1, our architecture contains a **Motor Network** and **Visual Network**. The Motor Network, often but not always, contains a recurrent neural network and is trained on a loss that favors the execution of the specified task, **g**. This training may take several forms. In the case of RL, we need a source of rewards. If the task is specified by demonstrations (our case), the training may be executed in a supervised fashion using a behavioral cloning loss.

Figure 4.2 illustrates the architecture in greater details. The Visual Network contains an Encoder module that encodes the input frame into the *Primary Latent Variable, z*. To get a richer representation **z**, we incorporate two other modules. First, a teacher network which computes an attention map and masks the input frame. We train the teacher network separately. Second, a GAN network that takes **z** as input and generates two reconstructed frames, the input frame, and the masked input frame.

Teacher Network for TFA

The teacher network for the TFA can be trained offline, does not require additional training data or pixel-level annotation of objects. Since the Teacher network is trained offline and not with other components, its architecture is not of any importance to the rest of the network architecture. One can even use annotated images instead of extracting the TFA using a network.

I am not going to the details of the Visual Attention Module here since the architecture is one of my co-author's contribution. In short, the teacher network is using a VGG-19 to extract features from the image. It also extracts a set of features from the text input describing the task. After combining these visual and textual features it will use them to generate spatial attention for the input image. The spatial attention will be used to pool the features from the last convolution layer of VGG-19 and the pooled features will be passed on to a classifier. The classifier should be able to classify the pooled featured to their corresponding category based on the object present in the input image. This will force the spatial attention to focus on the part of the image which contains the object. Please refer to the paper for more detail regarding the teacher network architecture [2].

The Visual and Motor Networks

Our architecture follows the generic architecture for the visuomotor policy in Figure 4.1. It consists of a **Visual Network** sub-module that extracts a primary latent encoding, **z**, and a **Motor Network** that transforms **z** into actions, which in our case are joint angle commands (next state of the robot arms). However, our architecture makes several specific decisions to take advantage of the available text description of the current task and the TFA.

Visual Network

The objective of the Visual Network is to create a compact primary latent encoding that captures the important aspects of the current task. An ongoing problem is that the encoding needs to work within a certain limited dimensionality budget. Intuitively, general-purpose visual features extracted from the image would waste space by encoding aspects of the image that are not relevant to the task. On the other hand, focusing only on the attention field may ignore parts of the image that are important for the task. For instance, in Figure 4.3- bottom right masked frame, the robot arm itself is not visible.

Our proposed architecture for the visual network, shown in Figure 4.2, incorporates several techniques that allow it to learn a representation that efficiently encodes the parts of the input that are relevant to the *current task*. The overall architecture follows the idea of a VAE-GAN [40]: it is composed of an encoder, a generator, and a discriminator. The *Primary Latent Encoding (*$\mathbf{z}$*)* is extracted from the output of the visual encoder (E).

The visual network receives a raw frame $\mathbf{x}$ and a one-hot representation of the user command (input sentence), denoted by $I_c \in \{0,1\}^{|V|}$. In fact, I_c is indicates which words of the dictionary are appearing in the textual input command. We assume that $\mathbf{z} \sim \mathcal{N}(\mu_z, \sigma_z)$, and:

$$[\mu_{\mathbf{z}}|\sigma_{\mathbf{z}}] = E(x, I_c), \tag{4.1}$$

where μ_z, $\sigma_z \in \mathcal{R}^{d_z}$, and d_z is the length of the Primary Latent Encoding (z). In fact, E is a multi-layer convolutional neural network with a $2d_z$ dimensional vector which splits into μ_z and σ_z.

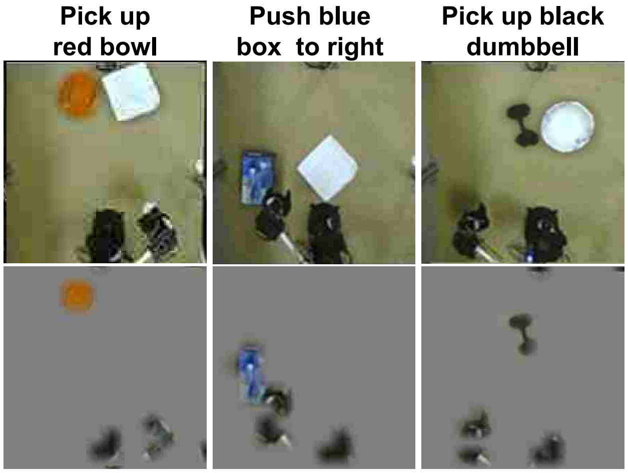


Figure 4.3: Examples of task-focused visual attention. We provide the command sentence on top of each column. The first row shows frames from RGB camera and the second row is the same image masked by the attention, produced by **teacher network**. We denote the first/second row images by x/m in our equations.

The generator, (G), takes the Primary Latent Encoding $\mathbf{z}$ as input and produces two images, a reconstruction frame, and a reconstructed frame masked with attention ("Fake Frame" and "Fake masked frame" in Figure 4.2). Notice that a novel aspect of our proposed architecture is that the generator does not only create a reconstruction of the input, $\mathbf{x}'$, but also an approximation of the faked masked frame, m'. Moreover, Unlike traditional GAN discriminators, the discriminator D employed in our architecture performs a more complicated classification [59]. Masked and unmasked frames(m/m', x/x') are both inputs to the discriminator, and it classifies the objects (s) and color (c) of the object of interest, as well as whether the input was fake or real. The discriminator has two outputs of lengths of $|s| + 1$ and $|c| + 1$. $|s|$ and $|c|$ are respectively the number of colors and objects in the vocabulary $|V|$ and the "+1" is for the "fake" class. We make the set of s and c tags by parsing all the input sentences (user's textual commands) in the training.

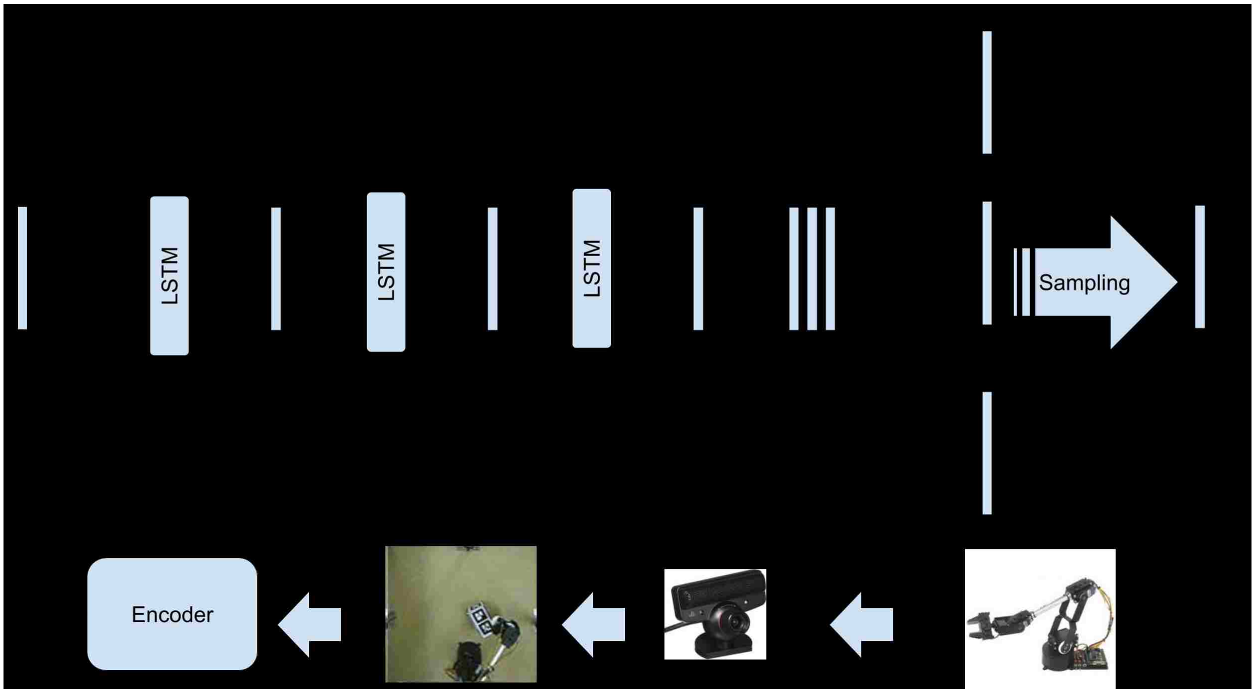


Figure 4.4: Motor Network Architecture used in our framework. Given the Primary Latent Variable z, the motor network predicts the next state of the robot and produces 7 numbers (corresponding to 7 joint angles of the robot) to move the joints of the robot. We use 3 stacked layers of LSTMs with skip connections. Also, we use layer normalization in between LSTMs. We concatenate the outputs of all the LSTMs and generate the μ, σ, and the mix coefficients α for the Mixture Density Network (MDN) described in the main manuscript. We use 20 Gaussians for the MDN of our implementation. The states of all LSTMs get updated frame by frame. In fact, after each move of the robot, a new frame is captured by the RGB camera, fed to the Encoder, and then the next z vector is fed back to the Motor Network. Each frame corresponds to one time-step for LSTMs.

Motor Network

The motor network in our architecture (see Figure 4.4) contains both recurrent and stochastic components. It takes as input the primary latent encoding, $\mathbf{z}$, which is processed through a 3-layer LSTM network with skip connections [26]. Note that the memory cells of LSTMs get updated through the time by doing the task (frame by frame). The output of the final LSTM layer is fed into a mixture density network (MDN) [5]. MDN provides a set of Gaussian kernels parameters namely μ_i, σ_i and the mixing probabilities $\alpha_i(x)$, all $\in \mathcal{R}^{|J|}$, and $1 \leq i \leq N_G$. Here, $|J|$ is the number of robot joints (specific to the robot) and N_G is the number of Gaussian components. The $|J|$-dimensional vector describing the next joint angles is sampled from this mixture of Gaussians.

Loss Function and Training

In this section, we describe the discriminator loss function $\mathcal{L}_D$, and the generator loss function $\mathcal{L}_G$. All the parameters in the Discriminator have been optimized to minimize $\mathcal{L}_D$, and parameters of the visual Encoder, Generator, and Motor network are optimized by the loss value $\mathcal{L}_G$ in a GAN training manner. In following, to prevent repetition of equations, we use the unifying tuples $\mathcal{X}' = (x', m')$ and $\mathcal{X} = (x, m)$ as fake and real data respectively. To clarify, $(x', m') = G(\mathbf{z} \sim E(x, I_c))$, while x is the real frame from RGB camera, and m is the masked real frame by the teacher network (Section 4).

Discriminator Loss

If the discriminator D is receiving real data $\mathcal{X}$, it needs to classify the object and color contained in the user's textual command input:

$$\begin{aligned}\mathcal{L}_{real} = &- \mathbb{E}_{\mathcal{X}, s \sim p_{data}}[\log(P_D(s|\mathcal{X}))] \\ &- \mathbb{E}_{\mathcal{X}, c \sim p_{data}}[\log(P_D(c|\mathcal{X}))],\end{aligned} \tag{4.2}$$

where P_D is the class probabilities produced by the discriminator for both colors and objects.

Similarly, if D receives $\mathcal{X}'$, it should classify them as fake:

$$\begin{aligned}\mathcal{L}_{fake} = &- \mathbb{E}_{\mathcal{X}'\sim G}[\log\left(P_D(|s|+1\big|\mathcal{X}')\right)] \\ &- \mathbb{E}_{\mathcal{X}'\sim G}[\log\left(P_D(|c|+1\big|\mathcal{X}')\right)].\end{aligned} \tag{4.3}$$

Finally, if D receives raw and masked faked frames, generated by G with the latent representation $\mathbf{z} \sim \mathcal{N}(0,1)$:

$$\begin{aligned}\mathcal{L}_{noise} = &- \mathbb{E}_{z\sim noise}[\log\left(P_D(|s|+1\big|G(z))\right)] \\ &- \mathbb{E}_{z\sim noise}[\log\left(P_D(|c|+1\big|G(z))\right)].\end{aligned} \tag{4.4}$$

The overall loss of the discriminator is thus $\mathcal{L}_D = \mathcal{L}_{real} + \mathcal{L}_{fake} + \mathcal{L}_{noise}$.

Generator Loss

The Generator (G) must reconstruct a real looking frame and masked frame that contains the object of interest. In fact, G tries not only to look real but also presents the correct object in both of its outputs. Hence, it has to fool the discriminator which tries to distinguish between fake frames and different objects and colors:

$$\begin{aligned}\mathcal{L}_{GD} = &- \mathbb{E}_{\mathcal{X}',s\sim p_G}[\log p_D(s\big|\mathcal{X}')] \\ &- \mathbb{E}_{\mathcal{X}',c\sim p_G}[\log p_D(c\big|\mathcal{X}')].\end{aligned} \tag{4.5}$$

The training of GANs is notoriously unstable. A possible technique to improve stability is *feature matching* [4]– forcing G to generate images that match the statistics of the real data. Here, we use features extracted by the last convolution layer of D for this purpose and we call it $f_D(x)$. The generator must produce outputs that have a similar f_D representation to real data. We define the loss term $\mathcal{L}_{fea}$ as a distance between the real inputs x/m and generated ones x'/m' features [59]:

$$\mathcal{L}_{fea} = ||f_D(x) - f_D(x')||^2 + ||f_D(m) - f_D(m')||^2. \tag{4.6}$$

To regularize the Primary Latent Encoding ($\mathbf{z}$), we minimize the KL-divergence between $\mathbf{z}$ and $\mathcal{N}(0,1)$:

$$\mathcal{L}_{prior} = D_{\mathrm{KL}}(E(x, I_c) \,||\, \mathcal{N}(0,1)). \tag{4.7}$$

Additionally, a reconstruction error of "fake" Frame/Masked generated by G is defined by:

$$\mathcal{L}_{rec} = ||x' - x||^2 + ||m' - m||^2. \tag{4.8}$$

Motor Network Loss:

The motor loss is calculated according to the MDN negative log-likelihood loss formula over the supervised data based on the demonstrations (behavioral cloning loss):

$$\mathcal{L}_{motor} = -log\left(\sum_{i=1}^{N_G} \alpha_i(x) \cdot P_{\sim\mathcal{N}(\mu_i,\sigma_i)}(J)\right). \tag{4.9}$$

Finally, we write the Generator loss as $\mathcal{L}_G = \mathcal{L}_{DG} + \mathcal{L}_{rec} + \mathcal{L}_{prior} + \mathcal{L}_{motor}$.

We collected demonstrations for the tasks of picking up and pushing objects using an inexpensive Lynxmotion-AL5D robot. We controlled the robot using a PlayStation controller. For each task and object combination, we collected 150 demonstrations. The training data consists of joint-angle commands plus the visual input recorded in 10 fps rate by a PlayStation Eye camera mounted over the work area. The training data thus collected was used to train both the Visual and the Motor Networks. Note that this robot does not have proprioception – any collision or manipulation error needs to be detected solely from the visual input.

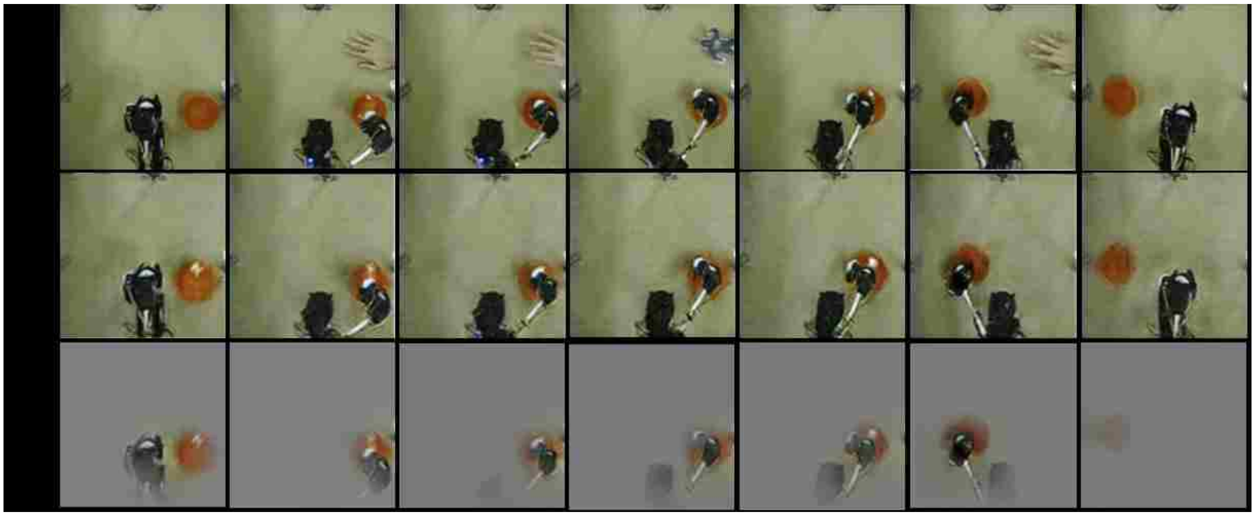


Figure 4.5: Execution of the pushing task with the sentence “Push the red bowl from right to left”. Top row: original input image, middle row: fake frame generated by the **Generator(G)**, bottom row: fake masked image with TFA generated by **G**. You can compare the fake masked frames presented in this figure with attention maps generated by the **teacher network** in Figure 4.3. Notice that visual disturbances such as the hand and the gorilla do not appear in the reconstructed image.

Table 4.1: The upper half of the table shows the rate of successfully performing the desired manipulation with different sentence commands. The model with **TFA** has superior results to a model without it [56]. We also train a version of our model without the Discriminator, named *Traditional VAE*. The model trained without D cannot effectively perform the manipulations since the adversarial loss helps to learn rich Primary Latent Variable ($\mathbf{z}$). Also, in *Just Encoder experiment*, we just use the Encoder as the visual network. The lower half of the table shows the rate of successfully performing the desired command while being disturbed by an external agent. The model with **TFA** is by far better than a model without it [56] in all cases.

Textual Command Sentences	**Pick up ...**							**Push ... from left to right**					
	Red Bowl	White Towel	Blue Ring	Black Dumbbell	White Plate	Red Bubbles	Mean pick up	Red Bowl	White Plate	Blue Box	B/W QR-box	Mean Push	Mean
Method	**Benign Condition**												
Just Encoder (%)	20	20	0	40	0	10	15.0	40	10	0	0	12.5	14.0
Traditional VAE (%)	60	60	20	20	50	30	40.0	50	**60**	**30**	30	42.5	41.0
(w/o TFA) (%)	70	50	30	40	60	10	43.3	80	**60**	10	20	42.5	43.0
with TFA (%)	**80**	**80**	**60**	**50**	**80**	**40**	**65.0**	**100**	**60**	**30**	**60**	**62.5**	**64.0**
	With Disturbance												
(w/o TFA) (%)	10	10	0	0	0	0	3.3	0	30	0	0	7.5	5.0
with TFA (%)	**70**	**80**	**60**	**60**	**40**	**40**	**58.3**	**90**	**50**	**30**	**50**	**55.0**	**57.0**

Performance Under Benign Conditions

The first set of experiments studies the performance of the visuomotor controller under benign conditions, that is, under situations when the robot is given a textual command, I_c in Sec. 4, and it is left alone to perform the task in an undisturbed environment. To compare our approach against a baseline, we have reimplemented and trained the network described in [56], which can be used in the same experimental setup, but it does not feature a task-focused visual attention. Note that the success rates are not directly comparable with [56], due to the more complex objects used here and the different camera position and environment of our robot. We trained the [56] model on our own dataset, tuned its hyper-parameters and also tried to get the best possible results by adding all the loss terms explained in Sec. 4.

Table 4.1 compares the performance of the four approaches for all the tasks, averaged over 10 tries each. We note that the proposed architecture using “TFA” outperforms the “w/o TFA” on all tasks. As an ablation study, we remove the discriminator and train the system as a traditional VAE (compared to VAE-GAN). Also, in another experiment, we trained the E just by using the motor network loss without any GAN. We confirm the contribution of the adversarial loss and the GAN network to produce a rich primary latent variable $\mathbf{z}$. We observe that not having the adversarial loss will reduce the sharpness of the reconstructed images and fade out the details. Note that the model without adversarial loss fails to manipulate objects that require precise positioning like the black dumbbell or the blue ring, however, it can push the white plate much better as the plate is a big symmetric object. Please refer to the supplementary materials to compare the reconstructed images with and without the adversarial loss.

Recovery After Disturbance

In the second series of experiments, we investigate the controller’s ability to recover from a physical and visual disturbance. We are comparing the baseline model and our model which uses TFA. Physically disturbing means to disturbed the robot either by (a) pushing the object just when the robot was about to pick it up or (b) forcefully taking away the object from the robot after a successful grasp. For the push tasks, we bring in one or two hands into the scene (Figure 4.5). We make different visual disturbances by bringing in the hand in random positions, waving it, sometimes covering the whole top part of the scene. In some cases, we even put other random objects like a paper gorilla. Under the described situations we count the manipulation as a success if the robot notices the disturbance and recovers by successfully redoing the task. We remind the audience of this dissertation that due to the limitations of the Lynxmotion-AL5D robot, the *only* way the robot can detect the disturbance is through its visual system. Table 4.1 shows the experimental results for scenarios with physical/visual disturbance. We notice that the results here are drastically better than the baseline. In the absence of TFA, the recovery rate is close to zero. In most cases, after losing the object, the robot tried to execute the manipulation without noticing that it does not grasp the object. With the help of TFA, however, the robot almost always notices the disturbance, turns back and tries to redo the grasp. This phenomenon is illustrated in our supplementary material video. Averaged over all the objects, the recovery rate is only 5% for the baseline policy in pickup and push tasks, while it is **57%** for the policy with the TFA (see Tables 4.1). Note that physical disturbance doesn't necessarily drop the robot's success rate since disturbing the robot occurs only when it is about to successfully perform the task, therefore the robot's success rate with and without the physical disturbance are not comparable. In other words, the robot starts doing the task, a human judge decides if the robot is doing well and if it is, the human judge starts to disturbing the robot. We discard any tries that the robot is likely to fail even without disturbance.

The Disappearing Gorilla:

The proposed architecture allows us to ignore many of the possible visual disturbances. Experiments comparing the architecture to one without TFA confirm that this is indeed the case. Another way to study whether the policy ignores the visual disturbance is to reconnect the generator during test time and study the reconstituted video frames (which are a good representation of the information content of primary latent encoding). Figure 4.5 shows the input video frames (first row), the reconstructed video frames (second row) and the generated masked frames (third row). While the robot was executing the task of pushing the red bowl to the left, we added some disturbances such as waving a hand or inserting a cutout gorilla figure in the visual field of the robot. Notice that in the reconstructed frames, the hand and the gorilla disappear, while the subject matter is reconstructed accurately. As these disturbing visual objects are ignored by the encoding, the task execution proceeds without disturbance. While we must be careful about making claims on the biological plausibility of the details of our architecture, we note that the overall effect implements a behavior similar to the selective attention experiments[2] of Chabris and Simmons [13], purely as a side effect of an architecture implemented for a completely different goal.

Comparison of the Results

In section 3 we discussed that our proposed approach with Task-Focused visual Attention (TFA) makes the robot policy robust to various types of visual and physical disturbances. Table 4.1 shows that the average performance of the model without TFA is about 50% less than the proposed network. Here, by visualizing the generated images from the two named experiments, we qualitatively show the reasons behind robustness and effectiveness of visual attention during the disturbance.

Figure 4.6 shows the original and reconstructed frames from the "w/o TFA" experiment. The frames show the sequence of a task when two other objects, namely a human hand and an eyeglass box, enter the scene and disturb the internal representation (starting frame 8). We notice that the reconstructed frames become very blurry and inaccurate. For example, from frame 23 to 30, the object of interest is completely missing in the reconstruction. We conjecture that the disturbance forced the primary latent encoding z to move to an unseen state from which the generator cannot reconstruct meaningful images.

Figure 4.7 reproduces same disturbance scenario as in Figure 4.6 but using the model with TFA. We notice that the attention disregards the obstacles and disturbances and the quality of reconstructed frames do not drop drastically. Also, we see that the disturbing objects such as the hand are removed from reconstructions.

Figure 4.8 illustrates several frames for a given textual command in each row. The first column of each row shows the original frames (denoted by x in the main manuscript), and the masked

[2] https://youtu.be/vJG698U2Mvo

frames produced by the teacher network (denoted by m in the main manuscript). The second and third columns show the fake (reconstructed) frame and the fake masked frame generated by the generator (x' and m' in the main manuscript); however, the third column shows the results out of generator when it is trained merely based on a reconstruction loss, without any discriminator. The quality difference between the second and third columns reconstructions explains the performance difference between "traditional VAE" and other experiments with the "VAE-GAN" setting (see Table 4.1).

We notice that in some cases, the attention produced by the generator is even better than the teacher network attention. For example, compare the masked frames of the first and second columns of the second and fourth examples in Figure 4.8. We believe that this phenomenon is due to the rich Primary Latent Variable, z that our network learns. In fact, the fake masked frame must be rich enough that the discriminator predicts the correct object and color of the task and it provides some complementary information to the Encoder.

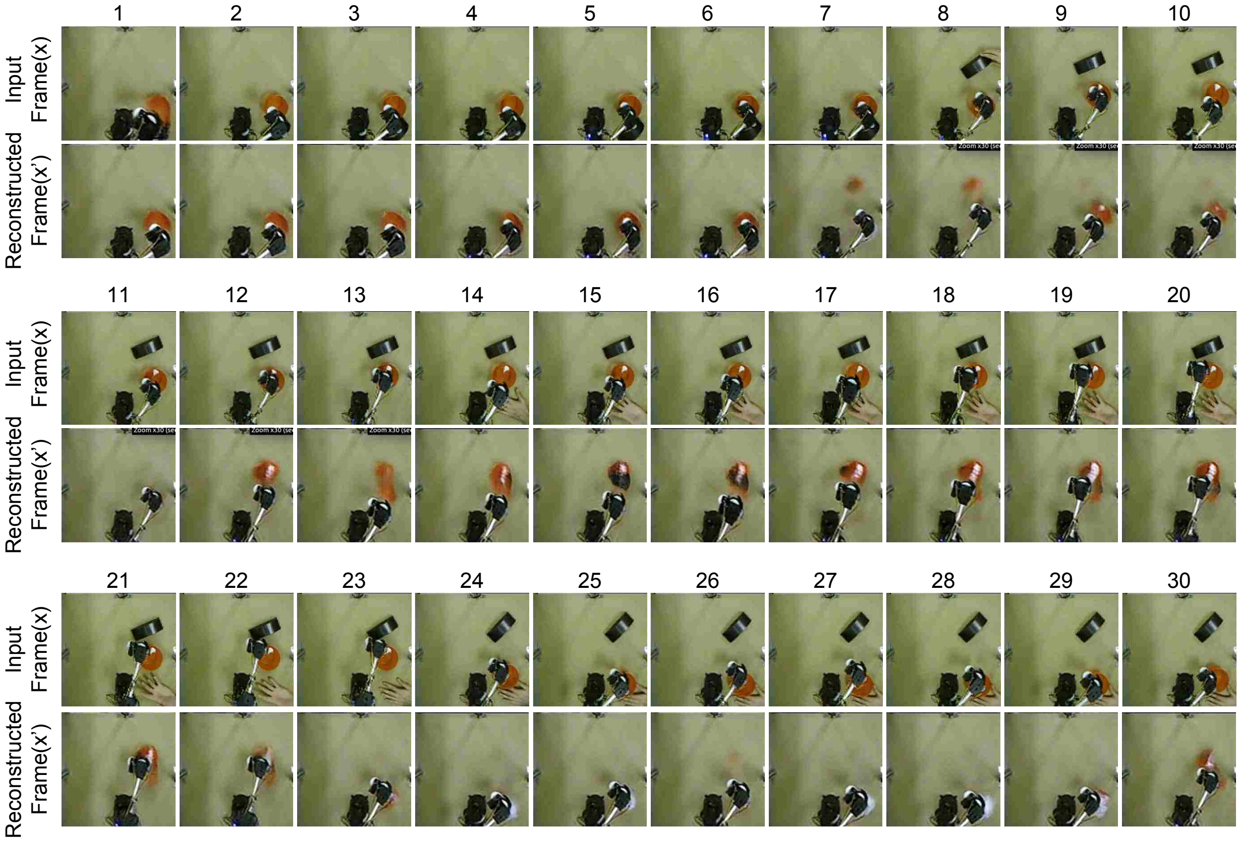


Figure 4.6: A sequence of frames from an experiment with visual disturbance. The textual command for this experiment is: ”push the red bowl from left to right”. Here, we show the frame reconstructions of the “w/o TFA” model. In many frames like 23-29, the model has failed to reconstruct the input frame properly, showing that the Primary Latent Variable z in this model is not robust to visual disturbance.

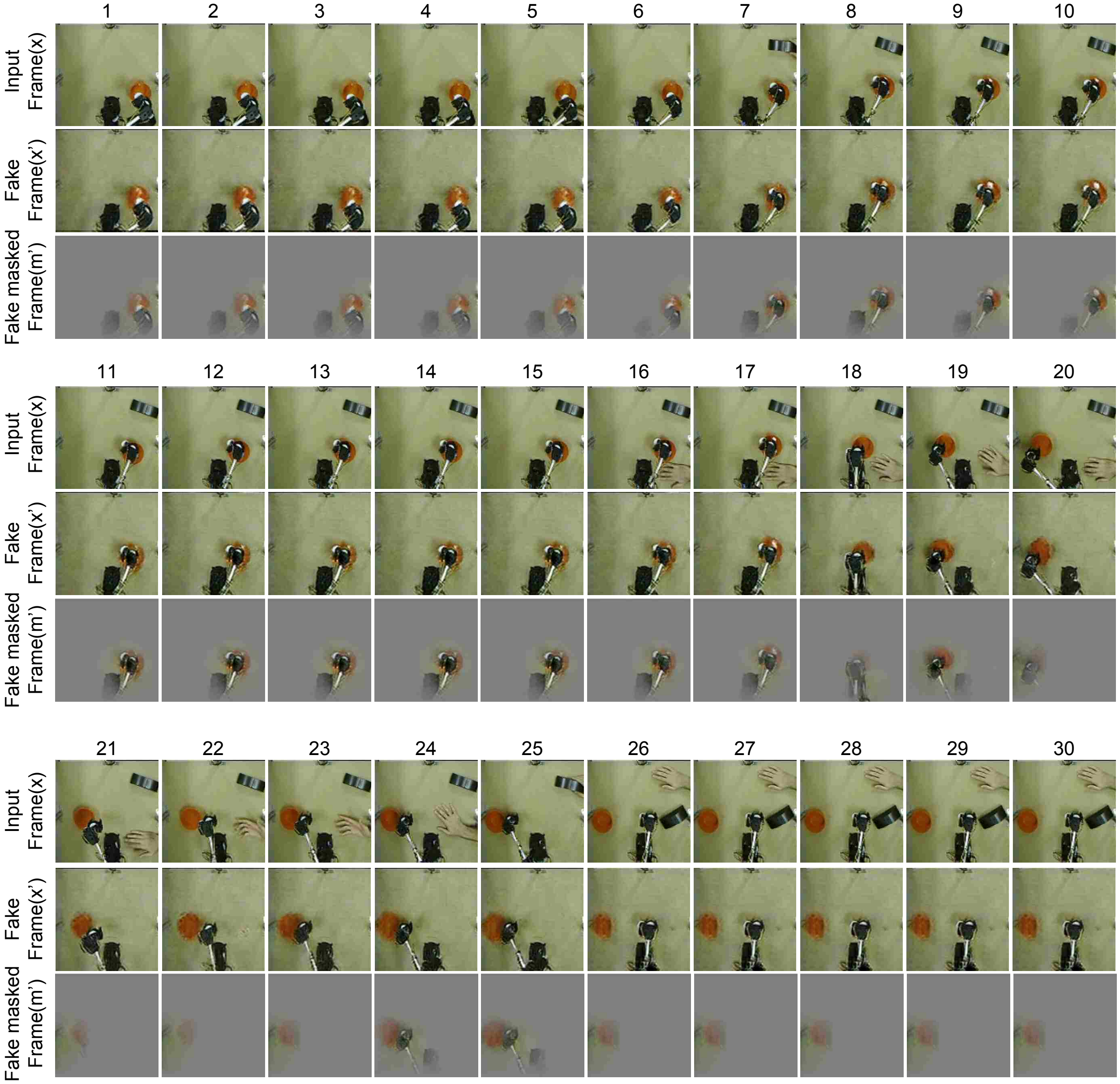


Figure 4.7: A sequence of frames of an example with a scenario similar to the one in Figure 4.6, using the TFA-augmented model. We notice that the attention stays on the correct object and the model has a better reconstruction of the object in both masked and unmasked frames compared to Figure 4.6.

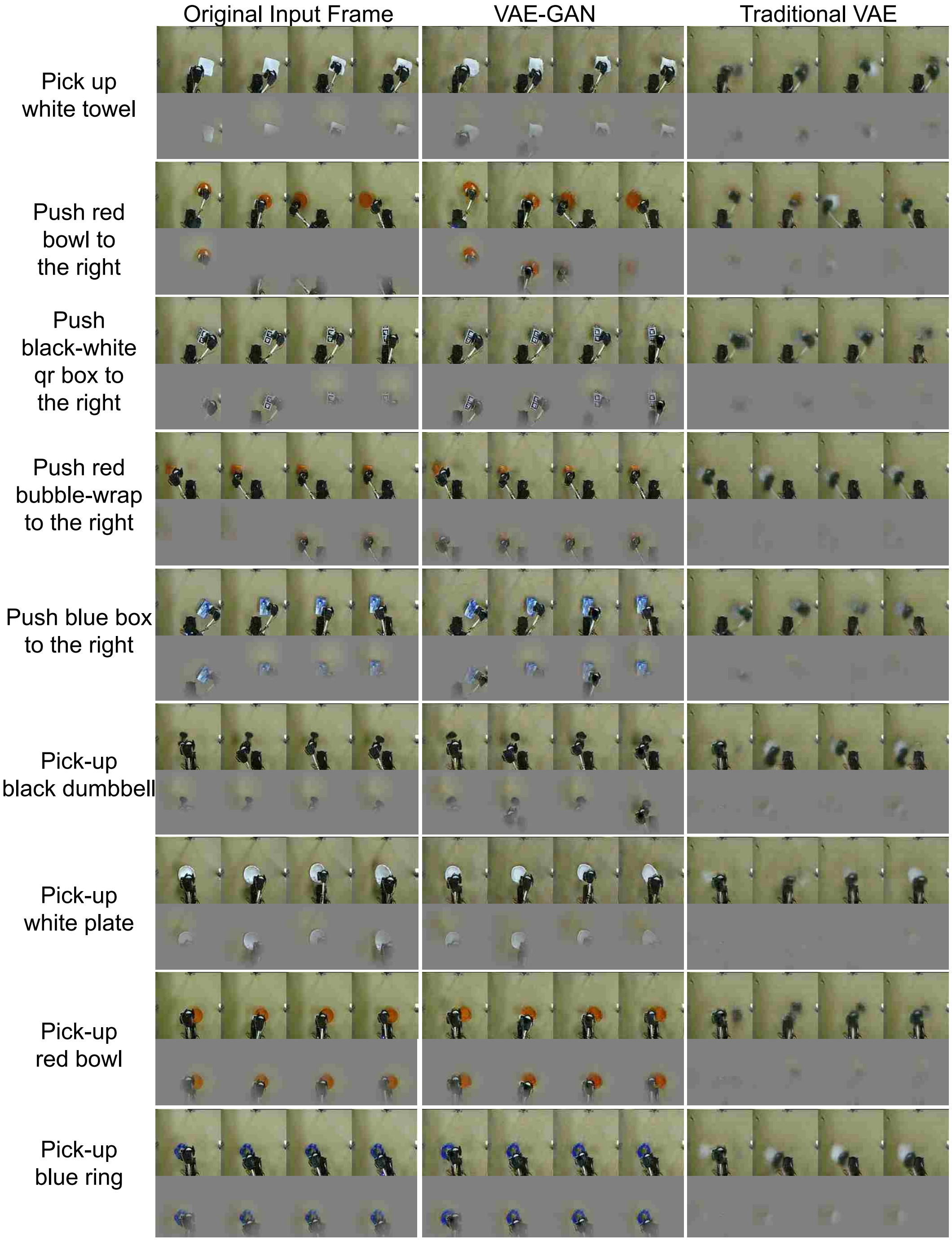


Figure 4.8: A comparison between the original frame reconstructed frames by VAE-GAN and Traditional VAE. We also show the real masked frames by attention (using the teacher network), generated fake masked frame by VAE-GAN and Traditional GAN. By comparing the second and third columns of this figure, we can justify the performance drop of the "Traditional VAE" experiment in Table 4.1

CHAPTER 5: MANIPULATION IN CLUTTER

Till now we have discussed how a robot should be positioned to have a better chance of a successful manipulation (Chapter 3), we briefly discussed an architecture which enabled the robot to learn from human demonstrations in Chapter 2, we made the architecture robust against visual and physical disturbances by using attention (Chapter 4). Here in this chapter, we are going to enable our robot to perform in cluttered environments. In the previous chapter, we introduced the term visual disturbance. Visual disturbance can be considered clutter but our definition in the previous chapter didn't cover all visual disturbances.[1].

One of the challenges which had not yet been consistently solved by deep visuomotor policies is the purposeful manipulation of objects in the presence of random clutter. For the purpose of this chapter, we define clutter as objects in the scene that do not need to be manipulated. If a scene contains a bowl and a towel, with the task to pick up the bowl, then the towel is clutter. If the robot needs to push aside the towel to pick up the bowl, then the towel becomes the target object for the push subtask.

In the following, we first discuss why operation in clutter is a particular problem for end-to-end learned policies and then outline our proposed solution. In a typical deep visuomotor policy there is an internal representation bottleneck we will call the *primary latent encoding* $\mathbf{z}$ separating the network into a vision component $f_v(\cdot) \rightarrow \mathbf{z}$ and a motor component $f_m(O, T) \rightarrow a$. It is tempting to use an off-the-shelf pre-trained network as a vision component such as VGG-19 or ResNet and to keep the size of the encoding small. A low dimensional $\mathbf{z}$ allows us to keep the number of demonstrations and/or reinforcement trials low. For instance, it was found that if $||\mathbf{z}|| \approx 64$, the

[1] This chapter's material has been published in ”Accept Synthetic Objects as Real: End-to-End Training of Attentive Deep Visuomotor Policies for Manipulation in Clutter” authored by Pooya Abolghasemi and Ladislau Bölöni

motor network can be trained from scratch with less than a hundred demonstrations per task, a number which can be further reduced with techniques such as meta-learning.

Clutter, however, presents a problem to a small, pre-trained latent encoding. As the off-the-shelf encoding does not depend on the task specification T, the encoding $\mathbf{z} = f_v(O)$ will need to represent the entire scene, leaving it to the motor network to sort out what is important. If $||\mathbf{z}||$ is small, there is simply no space for an accurate representation for many objects in the scene. The quality of representation for the target object will suffer, impacting the success rate of the policy.

There are several ways this problem can be solved. We can increase the size of $||\mathbf{z}||$ - but this requires a corresponding, usually higher-than-linear increase in the training data. For instance, we would need to provide a large number of demonstrations done in various clutter scenarios. Another possibility is to use a higher-dimensional encoding $\mathbf{z}$ but enforce on it a pre-defined object-oriented representation [32]. A benefit of this approach is that it is more explainable. However, to some degree, it backs off from the end-to-end learned paradigm towards more engineered approaches.

The approach we take retains the fully end-to-end trained model. As a first step, we require the visual component of the network to create a primary latent encoding that depends on the current manipulation task $\mathbf{z} = f_v(O, T)$ - this can be seen as a way to reserve a larger part of the representation to the target object. Nevertheless, this does not entirely eliminate the need for a variety of demonstrations in conditions of clutter. Our approach is based on the observation that such extra demonstrations convey new visual data, but very little new motion information. Human demonstrators already have a mental mechanism for ignoring clutter – thus they will usually deliver the same robot trajectory whether clutter is present or not. This leads us to the idea that it should be possible to train a visuomotor policy that performs under clutter conditions without requiring *any* demonstration in clutter.

Accept Synthetic Objects as Real

Our objective is to teach a robot arm to manipulate objects of different types under conditions of clutter. We will perform this by training a visuomotor policy that takes as input a video stream of the scene and generates commands to the robot. To be able to make a one-to-one comparison to previous approaches, we will reuse one of the existing datasets for which both training data and code for previous approaches is publicly available [2]. This scenario includes training data for picking up 8 different object types and pushing to place 5 different object types. The objects are distinguished by shape and color and have different degrees of rigidity including a rigid plastic bowl, a foam dumbbell, a piece of bubble wrap and a cotton towel.

The training data contains manipulation demonstrations done without the presence of clutter, with the only objects visible in the scene being the object to be manipulated and the robot arm itself (Figure 5.1 Demonstration Observation column or O_D).

As we discussed before, collecting training data in the presence of clutter is not an ideal way to use human effort, as humans will simply ignore the clutter. A better approach would augment the training data by adding clutter objects after the fact, using a visual manipulation of the images. This is shown in Figure 5.1 Augmented Demonstration Observation or O_{DA}.

Unfortunately, this approach, by itself, does not work. While copy-pasted synthetic objects may look natural to the human eye, they have low-level artifacts that allow the vision network to recognize them. As in these images, the target object is always real, and clutter objects are synthetic, the network learns to ignore clutter based on its synthetic nature. Such policies are not able to ignore clutter when tested with real objects. Attempts to improve our image processing capabilities to the degree that the artificially created clutter would be indistinguishable from the real one would be fighting against the training of the vision system which is motivated to distinguish them.

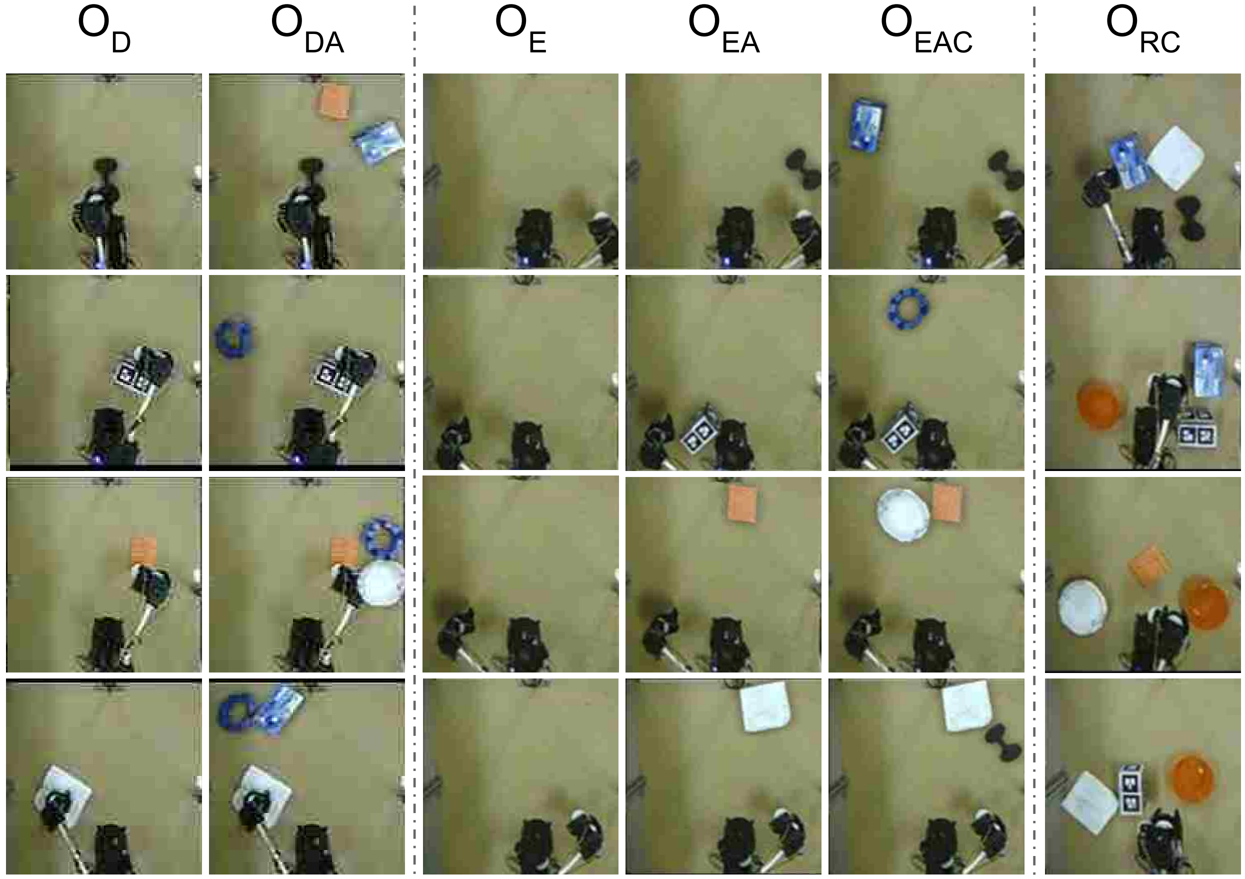


Figure 5.1: Faking clutter and accepting it as real. O_D**:** Demonstrations captured in an uncluttered environment [2]. O_{DA}**:** O_D images augmented by synthetic clutter. O_E**:** Empty images contain only the robot. O_{EA}**:** O_E images augmented by a synthetic target object. O_{EAC}**:** O_{EA} images augmented by synthetic clutter. O_{RC}**:** Real clutter(random objects in random positions).

As our object is not to generate more convincing images but to train the robot, we propose an approach that intervenes in the end-to-end training loss of the robot. Instead of improving the synthetic objects to be indistinguishable from real objects, we train the vision system to Accept Synthetic Objects as Real. This technique requires both specific type of generated training images as well as training regime and losses relying on them. Columns 3-5 in Figure 5.1 show the new type of training images generated. We started by recording a small set of images with only the robot arm in random positions and no other object (Empty Observations or O_E). Then we created images with one synthetic target object (Augmented Empty Observations or O_{EA}) and with a synthetic target object and several synthetic clutter objects (Clutter Augmented Observations or O_{EAC}). Note that there is no visual difference between the target object and the clutter objects: all of them are synthetic, the only feature of the target object is that it is referred to in the task specification. We can generate an arbitrary amount of training data through this technique. However, this training data is useful only for training the vision component, as it does not contain purposeful robot arm motion. We have another type of input images which we named Real Clutter Observations or O_{RC} which contains random real objects in random positions to give the model more generalization power.

With this set of images we use a training regime where the vision system does not distinguish between synthetic and real objects: it can accept the synthetic target object as real one in columns O_{EA} and O_{EAC}, and it can remove the clutter objects from O_{EAC} to create a generated scene as in O_{EA}.

In the following two sections we describe two different network architectures for visuomotor policies that take advantage of the ASOR model. We start by recognizing that any model where the primary latent encoding $\mathbf{z}$ prioritizes the target object specified by the task specification can be seen as attention-based. This can be an *explicit attention* where the network actually creates a heatmap style 2D overlay for the visual field, a model used in ASOR-EA discussed in Section 5. Alternatively, we can use ASOR in a network where such an explicit model does not exist – an *implicit attention* model such as the ASOR-IE discussed in Section 5.

ASOR With Implicit Attention

The ASOR-IA network architecture, shown in Figure 5.2 uses a VAE-GAN based visual network evolved from the one introduced in [56] featuring an encoder E, generator G, and discriminator D.

Next, we discuss the discriminator's loss function $\mathcal{L}_D$, and the generator's loss function $\mathcal{L}_G$. All the parameters in the Discriminator have been optimized to minimize $\mathcal{L}_D$, and parameters of the Encoder, Generator and the motor network are optimized by the loss value $\mathcal{L}_G$.

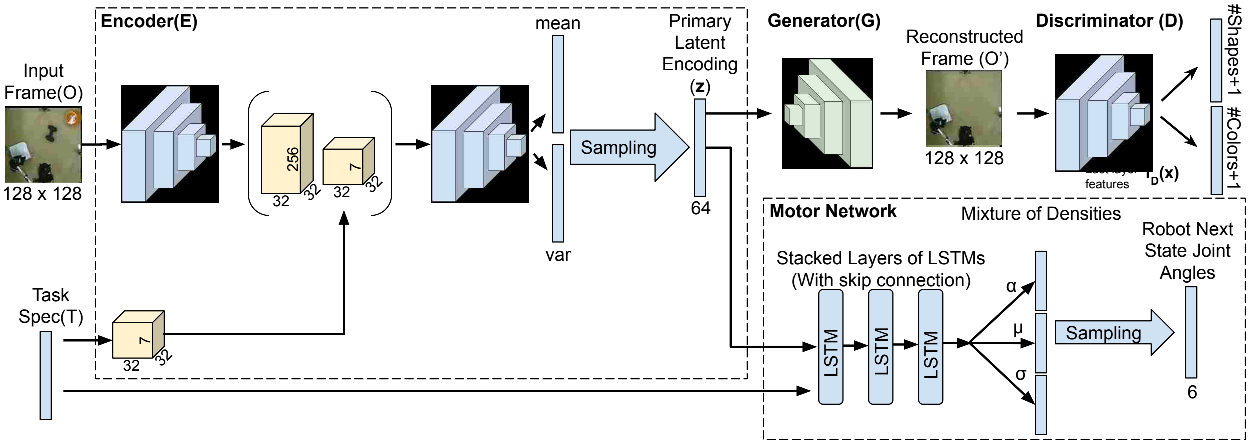


Figure 5.2: The ASOR-IA network architecture. In each time-step the **Encoder(E)** receives the task specification and an image(O) as input. Features extracted from the task specification will be concatenated to feature maps extracted from the image somewhere in the middle of the CNN layers. The **Generator(G)** is trained to reconstruct the image given the primary latent encoding **z**. The reconstructed image should only contain the target object and ignore all the unnecessary information. **z** and the task specification is then passed to the motor controller to generate the next robot joint angles.

The discriminator classifies the input frames based on target object's shape(s) and color(c), it also will classify fake frames in a separate class [59]. The discriminator's outputs are of lengths $|s| + 1$ and $|c| + 1$. $|s|$ and $|c|$ are respectively the number of unique shapes and colors and the "+1" represents the "fake" class. If the discriminator D receives a real frame from $\mathcal{O}$, it needs to classify the frame correctly in the corresponding object's shape and object's color class provided in the task specification (‘Push the red bowl/pick-up the white towel‘):

$$\begin{aligned}\mathcal{L}_{real} = &- \mathbb{E}_{\mathcal{O},s\sim p_{data}}[\log\left(P_D(s\middle|\mathcal{O})\right)] \\ &- \mathbb{E}_{\mathcal{O},c\sim p_{data}}[\log\left(P_D(c\middle|\mathcal{O})\right)] \end{aligned} \tag{5.1}$$

where $\mathcal{O} \in \{\mathcal{O}_{\mathcal{D}}, \mathcal{O}_{EA}\}$ and P_D is the class probabilities produced by the discriminator for object's shape and color. Similarly, if D receives fake frames generated by the generator, it should classify them as fake:

$$\begin{aligned}\mathcal{L}_{fake} = &- \mathbb{E}_{\mathcal{O}'\sim G}[\log\left(P_D(|s|+1\middle|\mathcal{O}')\right)] \\ &- \mathbb{E}_{\mathcal{O}'\sim G}[\log\left(P_D(|c|+1\middle|\mathcal{O}')\right)] \end{aligned} \tag{5.2}$$

where $\mathcal{O}'$ is the generator's reconstruction of any of the 5 input types. Finally, if D receives fake frames, generated by G with the latent representation $\mathbf{z} \sim \mathcal{N}(0, 1)$:

$$\begin{aligned}\mathcal{L}_{noise} = &- \mathbb{E}_{z\sim noise}[\log\left(P_D(|s|+1\middle|G(z))\right)] \\ &- \mathbb{E}_{z\sim noise}[\log\left(P_D(|c|+1\middle|G(z))\right)]. \end{aligned} \tag{5.3}$$

The overall loss of the discriminator is thus $\mathcal{L}_D = \mathcal{L}_{real} + \mathcal{L}_{fake} + \mathcal{L}_{noise}$.

The Generator(G) must reconstruct a real looking frame(O') that only contains the object described in the input sentence. G tries not only to look real but also presents the correct object in its output.

$$\begin{aligned}\mathcal{L}_{GD} = &- \mathbb{E}_{\mathcal{O}',s\sim p_G}[\log P_D(s \big| \mathcal{O}')] \\ &- \mathbb{E}_{\mathcal{O}',c\sim p_G}[\log P_D(c \big| \mathcal{O}')]. \end{aligned} \tag{5.4}$$

The generator needs to learn to remove the irrelevant objects present in the scene. The discriminator as described in equation 5.1, only receives single object images. As a result, it should be able to easily classify multi-object images as fake. Moreover, to enforce such behavior and to stabilize the training process we use feature matching [59] technique and the reconstruction error. Using these two terms in our loss function, we will force the network to reconstruct images of type O_{DA} and O_{EAC} as close as possible to the corresponding images of type O_D and O_{EA}. These reconstruction losses will also force the attention module to include information regarding the robot's position.

$$\begin{aligned}\mathcal{L}_{rec} = &||O'_D - O_D||^2 + ||O'_{DA} - O_D||^2 \\ &+||O'_{EA} - O_{EA}||^2 + ||O'_{EAC} - O_{EA}||^2. \end{aligned} \tag{5.5}$$

$$\begin{aligned}\mathcal{L}_{fea} = &||f_D(O'_D) - f_D(O_D)||^2 + ||f_D(O'_{DA}) - f_D(O_D)||^2 \\ &+||f_D(O'_{EA}) - f_D(O_{EA})||^2 + ||f_D(O'_{EAC}) - f_D(O_{EA})||^2 \end{aligned} \tag{5.6}$$

where $f_D(x)$ is features extracted by the discriminator in the last layer for input x. To regularize the primary latent encoding ($\mathbf{z}$), we minimize the KL-divergence between $\mathbf{z}$ and $\mathcal{N}(0, 1)$:

$$\mathcal{L}_{prior} = D_{\text{KL}}(E(O, T) \mid\mid \mathcal{N}(0, 1)). \tag{5.7}$$

Finally, we write the generator's loss as $\mathcal{L}_G = \mathcal{L}_{DG} + \mathcal{L}_{rec} + \mathcal{L}_{prior} + \mathcal{L}_{motor}$, where the $\mathcal{L}_{motor}$ is calculated based on the behaviour cloning loss similar to what we discussed before.

ASOR with Explicit Attention

The second architecture we discuss, ASOR-EA generates explicit attention maps for the input frames. The network, shown in Figure 5.3 has 5 primary components: encoder, generator, discriminator, attention and the motor network. First, the attention module extracts an attention mask to cover the parts of the input frame that are not relevant to the current task (the clutter). The masked frame M is the result of pixel-wise multiplication of the attention map A and the input frame O. Given the masked frame M, the encoder-generator pair will try to extract a set of features descriptive enough to create reconstructed versions of the original frame O' and of the masked frame M'. The reconstruction of the masked frame M' is necessary to ensure that the information regarding the attention map is preserved in the latent encoding to be used later by the motor controller. O' is fed to the discriminator network.

The attention network combines features from the task specification and the input frame to extract the attention map. Several convolutional layers augmented by batch normalization are used to obtain K spatial visual feature vectors. These convolution layers will provide spatial visual features with size $\phi_f \in \mathcal{R}^{K \times d_\phi}$, where d_ϕ is the number of features for region k_i. The task specification encodes the target object as two concatenated one-hot vectors describing the shape and color of the object respectively. Several fully-connected layers are used to transform the task encoding to shape $u \in \mathcal{R}^{d_\psi}$. To combine the visual and task features repeated the u vector k times and combined it with the visual features using a technique similar to [75, 50]:

$$\psi = \tanh(\phi_f \times W_f \oplus u), \tag{5.8}$$

where $W_f \in \mathcal{R}^{d_\phi \times d_\psi}$ is a mapping matrix, $\oplus$ is element-wise summation. The combined feature vector is of size $\mathcal{R}^{k \times d_\psi}$. To compute the final attention map we calculate

$$p = \text{sigmoid}(\psi \times W_p), \tag{5.9}$$

$$p_{\mathit{TFA}} = \text{ReLU}(p - t), \tag{5.10}$$

where $W_p \in \mathcal{R}^{d_\psi \times 1}$ represents a fully-connected layer, and t is a hyper-parameter. t can be a constant or it can be a statistical metric such as a running average over p. Here we set t to the constant value of 0.5. ReLU and the hyper-parameter t are needed to force the network to completely hide the information from the irrelevant regions by assigning a score of exactly 0 to most regions. Without these, the attention map will keep small values in most regions and the encoder can easily reconstruct the original image. The final $p_{\mathit{TFA}} \in [0, 1]^k$ contains attention scores of all k regions.

Similar to the teacher network in [2] the network should be able to reconstruct the task specification from the pooled spatial features weighted by the attention map, p_{TFA}. Weighted pooled features are defined as $p_f = \sum_{i \in k} p_{\mathit{TFA}_i} \phi_{f\,i}$ where $p_f \in \mathcal{R}^{d_\phi}$ and are passed to two multi-layer perceptrons (MLP) $\hat{s} = \tau_1(p_f)$, $\hat{c} = \tau_2(p_f)$. These MLP layers try to classify the pooled features based on target object's shape and color. In other words, the network is trying to focus the attention on regions which provide better information to reconstruct the task specification and an image which only contains the target object.

We use the L1 norm of the p_{TFA} and softmax cross entropy between the f_s and f_c from the task specification and $(\hat{s}, \hat{c})$ to calculate $\mathcal{L}_A$.

The masked frame M is generated using the attention map, $M = O \otimes p_{\mathit{TFA}}$ where $\otimes$ is element-wise multiplication. Examples of input frames and the masked frames using the computed attention p_{TFA} are shown in Figure 5.4.

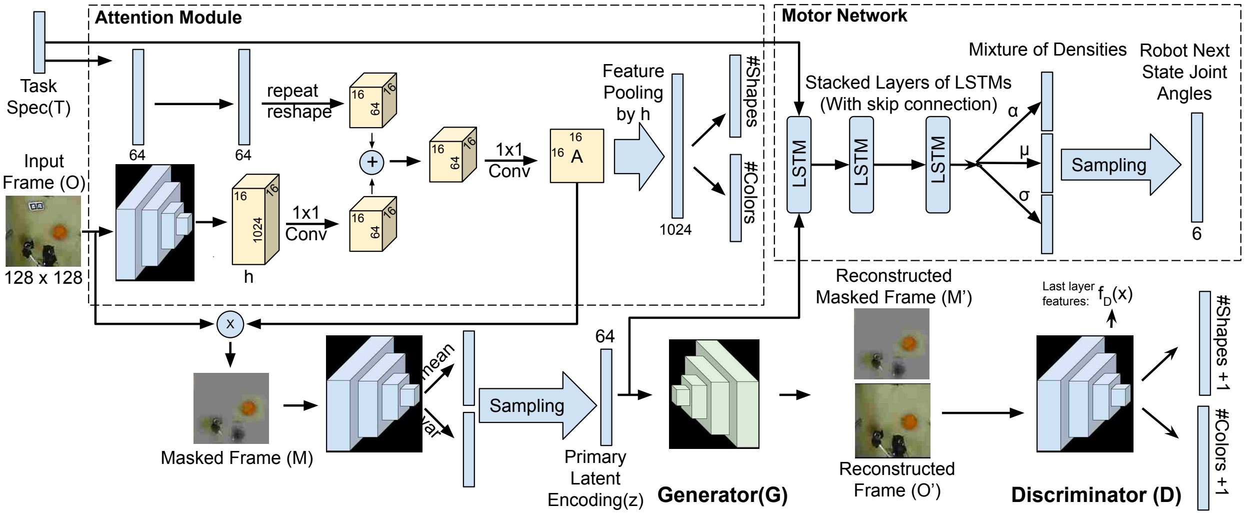


Figure 5.3: The ASOR-EA network architecture. The task specification and the image capture are the inputs to the attention network. The attention network generates the attention map and should be able to classify the input image based on the features pooled from feature maps pooled from the last convolution layer of the attention map. The VAE-GAN will receive a masked image of the original input by the attention map and should be able to reconstruct the input image without the clutter objects.

The encoder E receives M as input, $[\mu_{\mathbf{z}}|\sigma_{\mathbf{z}}] = E(M)$ where μ_z, $\sigma_z \in \mathcal{R}^{d_z}$, and d_z is the length of the primary latent encoding $\mathbf{z}$. E is a multi-layer convolutional neural network with a $2d_z$ dimensional vector output split into μ_z and σ_z. We assume that $\mathbf{z} \sim \mathcal{N}(\mu_z, \sigma_z)$. The generator receives $\mathbf{z}$ and reconstructs the input frame O and the masked frame M. Note that in this case information regarding the target object and the text encoding is not passed to the encoder. The encoder should rely solely on the masked frame to extract the required information for G to reconstruct the original frame and the masked frame. In addition to the $\mathcal{L}_A$, the reconstruction loss between different types of input images will also force the attention to focus on parts of the image related to the task.

The loss functions of the discriminator, generator, and motor network are similar to the one for ASOR-IA. The only difference is that $\mathcal{L}_A$ and the reconstruction loss for the masked frame M' is added to the generator's loss $\mathcal{L}_G$.

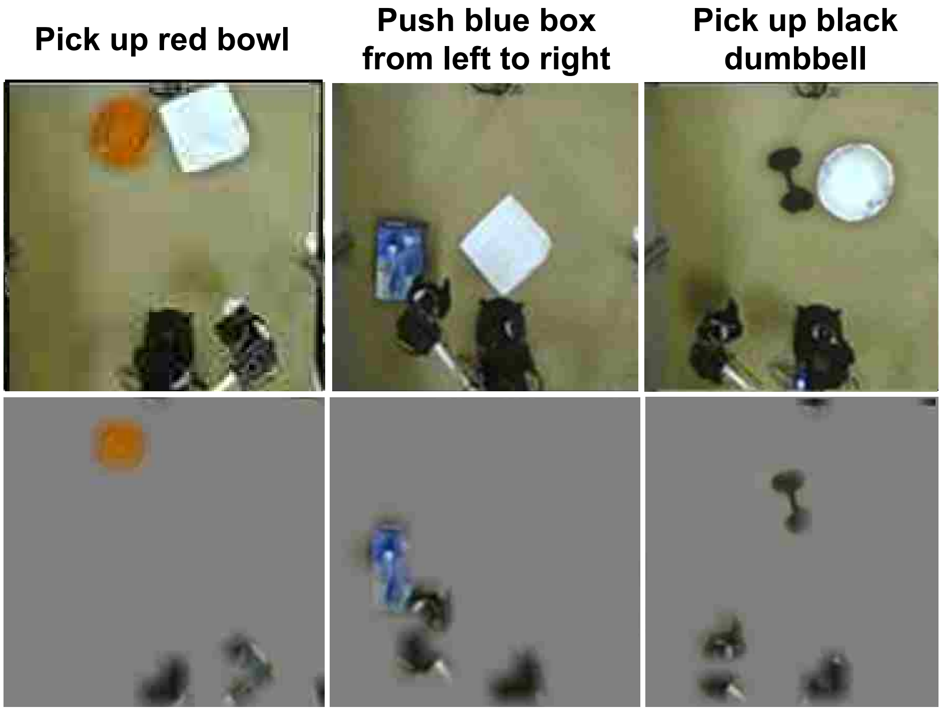


Figure 5.4: Examples of the operation of the attention module in ASOR-EA. The first row is the input image O and the second row is the masked image M for the task specification on the top of the column.

Experiments

Comparing robot control approaches is a general challenge due to difficult to reproduce environmental settings. It is to be hoped that in future years with the spread of inexpensive, replicable platforms such as REPLAB [73] and PyRobot [51], such one to one comparisons will be easier to make. For the purpose of this dissertation, we will test the proposed approach over 10 tasks for which training data is publicly available [2]. The four algorithms we compare are the one from [56] which does not use an attention model, the algorithm from [2] which uses a language induced attention model called TFA, ASOR-IA as described in Section 5 and ASOR-EA as described in Section 5. All algorithms were trained with the same demonstrations.

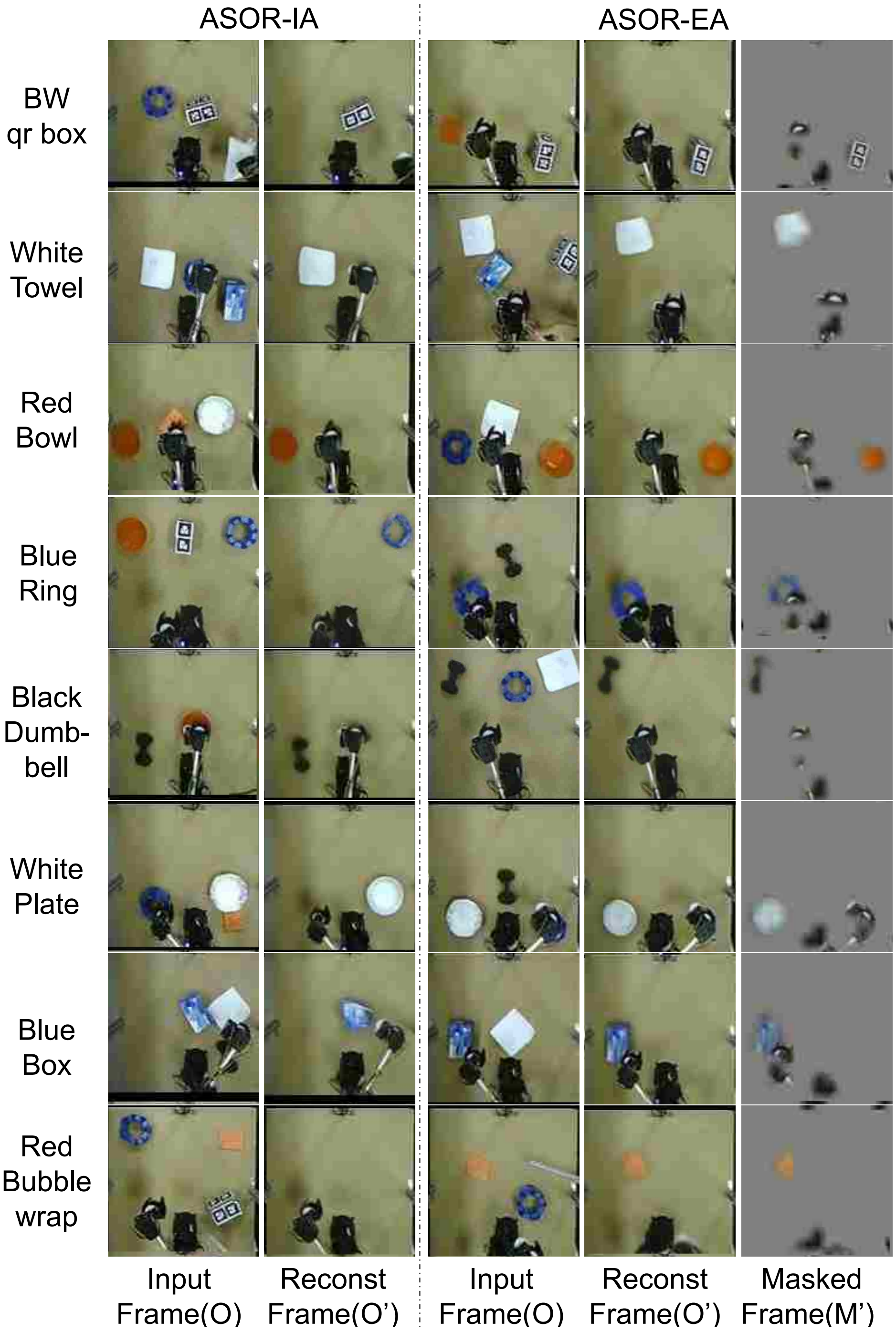


Figure 5.5: Each row illustrates ASOR-IA and ASOR-EA reconstructions for a specific object. The first column is the input frame(O) to ASOR-IA and the second column is its output(O'). The third column is the input frame(O) to ASOR-EA and the next two columns are the generator's reconstructions O' and M'.

Table 5.1: Success percentages: no clutter / with clutter

	Task spec.	w/o att. [56]	w/ att. [2]	ASOR-IA	ASOR-EA
Pick up	Red Bowl	70 / 0	80 / 0	80 / 60	**90 / 80**
	White Towel	50 / 0	80 / 0	**100** / 70	**100 / 100**
	Blue Ring	30 / 0	60 / 0	60 / 40	**80 / 80**
	Black Dumbbell	40 / 0	50 / 0	40 / 50	**60 /70**
	White Plate	60 / 0	80 / 0	80 / 60	**100 / 90**
	Red Bubblewrap	10 / 0	**40 / 0**	30 / **20**	**40 / 20**
	All pick up	43.3 / 0.0	65.0 / 0.0	65.0 / 50.0	**78.3 / 70.0**
Push to right	Red Bowl	80 / 0	**100** / 0	80 / 80	**100 / 100**
	White Plate	60 / 0	60 / 0	**70** / 50	60 / **70**
	Blue Box	10 / 0	30 / 0	60 / 50	**90 / 90**
	BW QR-box	20 / 0	60 / 0	60 / 40	**80 / 70**
	All push tasks	42.5 / 0	62.5 / 0	67.5 / 57.0	**82.5 / 82.5**
Overall		43.0 / 0.0	64.0 / 0.0	66.0 / 53.0	**80.0 / 75.0**

The task set was composed of 6 tasks of picking up objects (a red bowl, a white towel, a blue ring, a black foam dumbbell, a white plate and a red bubblewrap) and 4 tasks for pushing objects (a red bowl, a white plate, a blue box and a black-and-white QR patterned box). Figure 5.5 illustrates ASOR-IA and ASOR-EA reconstructions for each of these objects. For each algorithm, we repeated the experiments 10 times under two types of conditions: in an uncluttered scene with only the target object and the robot visible and in a cluttered scene where 2-3 other objects were also present. A trial was counted as a success if the manipulation was completed in 2 minutes.

Table 5.1 shows the experimental results as percentages of the successes for the no-clutter / clutter case, with the best results for each task setting highlighted in **bold**[2].

As expected, we find that for almost all trials, the results were worse in clutter. Neither of the previous approaches managed to complete any trial successfully in the presence of clutter. Note that the architecture proposed in [2] is trained with task-focused attention and is able to ignore

[2] Checkout our YouTube video `https://youtu.be/GchuLQhG3ug`

physical and visual disturbances like a human hand and generally disturbances that are not present in the training set but clutter in our experiments contains objects involved in other demonstrations. ASOR-IA and ASOR-EA outperformed the previous approaches in the no-clutter setting, with ASOR-EA being the better of the two. More importantly, they managed to complete a significant portion of the tasks in clutter as well - in fact, ASOR-EA outperforms in clutter the previous best algorithm operating in an uncluttered setting[3].

[3] Code is available at `https://github.com/pouyaAB/Accept_Synthetic_Objects_as_Real`

CHAPTER 6: CONCLUSIONS

In this dissertation, we discussed research results that advance the field of assistive robotics. We divided the problem into two part: 1. Finding the best position to have a better chance of performing the manipulation. 2. Performing the manipulation itself. In Chapter 3, we investigated the problem of positioning a wheelchair mounted robotic arm such that reaching a target object would be easier while avoiding obstacles. First, we explained a metric called ease of reach score (ERS) which simply says how easily reachable a target is when the arm is located in a certain position. Then we argued that exhaustive search for finding the ERS at each point is a very computationally expensive process. Therefore, we proposed an alternative method to estimate the value of ERS using machine learning techniques. By simulating the proposed method in V-REP simulator in which the wheelchair mounted robotic arm tries to reach a target object, we showed that the estimation technique provides sufficient accuracy for practical use. We also showed that this estimation takes less than a second which makes the proposed method suitable for real-time applications.

In Chapter 2, we briefly discussed an architecture suitable for imitation learning. In Chapter 4, we proposed a method for augmenting the architecture from [56] with a task-focused visual attention model. The attention is guided by a natural language description of the task – it effectively tells the policy to “Pay Attention!” to the task and object at hand. Our experiments show that under benign conditions, the resulting policy consistently outperforms a related baseline policy from [56]. More importantly, paying attention has significant robustness benefits. In severe adversarial conditions, where a bump or human intervention forces the robot to miss the grasp or drop the object, we demonstrated through experiments that the proposed policy recovers quickly in the majority of cases, while the baseline policy rarely recovers. In the case of visual disturbances such as moving foreign objects in the visual field of the robot, the new policy is able to ignore these disturbances which in the baseline policy often trigger erratic behavior.

In Chapter 5, we used data augmentation techniques along with modifications in the proposed architecture to enable the network to perform in cluttered environments. We performed experiments in benign and cluttered environments. Our experiments show that under benign conditions, the resulting policy consistently outperforms a related policy from Chapter 4. Moreover, the augmented architecture can perform the manipulation tasks in cluttered environments as well.

Future Direction

Architectures proposed in Chapter 4 and Chapter 5 are designed to block the flow of information from unnecessary regions of the environment. By doing so we tried to imitate nature's solution to process a huge amount of information also known as selective attention [13]. However, in humans, the flow of information is not completely blocked. As soon as the mind decides there is something important in other regions it will focus it's attention to not miss anything of value. Architectures proposed in this dissertation are not the final architecture ready to perform in the real world (street, house, school, ...) but we are building the final one step by step. One can imagine an architecture based on our work with multiple attention modules. Each attention module will be responsible to pay attention to a set of events or objects. For example, one can focus on detecting fast moving objects since they need immediate attention and processing. The other one can pay attention to obstacles that might prevent the manipulation. We might even design one to detect humans only to avoid hurting them during the manipulation. The results of all these selected regions can be processed by separate vision and controller modules and the resulting commands from each of them will affect the final decision. For example, if the module responsible for avoiding humans demands the robot to turn right, it's decision should be prioritized above all other modules. In other situations commands from different modules can be combined and control the robot such as avoiding an obstacle. While the robot tries to go around an obstacle the main controller module will still generate commands moving the robot toward the obstacle since the main controller does not see the obstacle. However, the module responsible to detect obstacles will generate commands to avoid the collision. The combination of these commands will the final command for the robot to perform. A hierarchy of networks can be designed to account for many such situations.

LIST OF REFERENCES

[1] Pooya Abolghasemi and Ladislau Bölöni. “Accept Synthetic Objects as Real: End-to-End Training of Attentive Deep Visuomotor Policies for Manipulation in Clutter”. In: *arXiv preprint arXiv:1909.11128* (2019).

[2] Pooya Abolghasemi et al. “Pay Attention! - Robustifying a Deep Visuomotor Policy Through Task-Focused Visual Attention”. In: *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-2019)*. Sept. 2019.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer Normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[4] Jianmin Bao et al. “CVAE-GAN: Fine-Grained Image Generation Through Asymmetric Training”. In: *arXiv preprint arXiv:1703.10155* (2017).

[5] Christopher M Bishop. *Mixture Density Networks*. Tech. rep. Aston University, 1994.

[6] Wendelin Böhmer et al. “Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations”. In: *KI-Künstliche Intelligenz* 29.4 (2015), pp. 353–362.

[7] Byron Boots, Sajid M Siddiqi, and Geoffrey J Gordon. “Closing the learning-planning loop with predictive state representations”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 954–966.

[8] Leo Breiman. “Bagging Predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[9] Leo Breiman et al. *Classification and Regression Trees*. CRC press, 1984.

[10] Tim Brys et al. “Reinforcement Learning From Demonstration Through Shaping”. In: *Twenty-Fourth Int’l Joint Conf. on Artificial Intelligence*. 2015.

[11] Serkan Cabi et al. “A Framework for Data-Driven Robotics”. In: *arXiv preprint arXiv:1909.12200* (2019).

[12] Nino Cauli et al. “Autonomous Table-cleaning from Kinesthetic Demonstrations Using Deep Learning”. In: *Joint IEEE Int’l Conf. on Development and Learning (ICDL) and Epigenetic Robotics (EpiRob)*. 2018.

[13] Christopher Chabris and Daniel Simons. *The Invisible Gorilla: And Other Ways Our Intuitions Deceive Us*. Harmony, 2010.

[14] Jonathan Chang et al. “Learning Deep Parameterized Skills from Demonstration for Retargetable Visuomotor Control”. In: *arXiv preprint arXiv:1910.10628* (2019).

[15] Paul Christiano et al. “Transfer From Simulation to Real World Through Learning Deep Inverse Dynamics Model”. In: *arXiv preprint arXiv:1610.03518* (2016).

[16] Felipe Codevilla et al. “End-to-End Driving via Conditional Imitation Learning”. In: *IEEE Int’l Conf. on Robotics and Automation (ICRA-2018)*. IEEE. 2018, pp. 1–9.

[17] Coline Devin et al. “Deep Object-Centric Representations for Generalizable Robot Learning”. In: *arXiv preprint arXiv:1708.04225* (2017).

[18] Coline Devin et al. “Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer”. In: *Proc. of IEEE Int’l Conf. on Robotics and Automation (ICRA-2017)*. 2017, pp. 2169–2176.

[19] J. Dong and J.C. Trinkle. “Orientation-Based Reachability Map For Robot Base Placement”. In: *Proc. of IEEE/RSJ Int’l Conf. on Intelligent Robots and System (IROS)*. Sept. 2015.

[20] Alon Farchy et al. “Humanoid Robots Learning to Walk Faster: From the Real World to Simulation and Back”. In: *Proc. of the 2013 Int’l Conf. on Autonomous agents and multi-agent systems*. Int’l Foundation for Autonomous Agents and Multiagent Systems. 2013, pp. 39–46.

[21] Chelsea Finn et al. “Deep Spatial Autoencoders for Visuomotor Learning”. In: *arXiv preprint arXiv:1509.06113* (2015).

[22] Maxwell Forbes et al. “Robot Programming by Demonstration With Crowdsourced Action Fixes”. In: *Second AAAI Conf. on human computation and crowdsourcing*. 2014.

[23] Marc Freese et al. “Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator”. In: *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 51–62.

[24] Yang Gao et al. “Reinforcement Learning From Imperfect Demonstrations”. In: *arXiv preprint arXiv:1802.05313* (2018).

[25] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[26] Alex Graves. “Generating Sequences with Recurrent Neural Networks”. In: *arXiv preprint arXiv:1308.0850* (2013).

[27] David Ha and Douglas Eck. “A Neural Representation of Sketch Drawings”. In: *arXiv preprint arXiv:1704.03477* (2017).

[28] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[29] Tadanobu Inoue et al. “Transfer learning from synthetic to real images using variational autoencoders for precise position detection”. In: *2018 25th IEEE Int’l Conf. on Image Processing (ICIP)*. IEEE. 2018, pp. 2725–2729.

[30] Stephen James, Andrew J Davison, and Edward Johns. “Transferring End-to-End Visuomotor Control From Simulation to Real World for a Multi-Stage Task”. In: *arXiv preprint arXiv:1707.02267* (2017).

[31] Lorenzo Jamone et al. "Interactive Online Learning of the Kinematic Workspace of a Humanoid Robot". In: *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and System (IROS)*. 2012, pp. 2606–2612.

[32] Yiding Jiang et al. "Language as an Abstraction for Hierarchical Deep Reinforcement Learning". In: *arXiv preprint arXiv:1906.07343* (2019).

[33] Rico Jonschkowski and Oliver Brock. "State Representation Learning in Robotics: Using Prior Knowledge about Physical Interaction." In: *Robotics: Science and Systems*. 2014.

[34] Kei Kase et al. "Put-in-Box Task Generated from Multiple Discrete Tasks by a Humanoid Robot Using Deep Learning". In: *IEEE Int'l Conf. on Robotics and Automation (ICRA-2018)*. IEEE. 2018, pp. 6447–6452.

[35] Ben Kehoe et al. "Cloud-Based Robot Grasping With the Google Object Recognition Engine". In: *2013 IEEE Int'l Conf. on Robotics and Automation*. IEEE. 2013, pp. 4263–4270.

[36] Jaeseok Kim et al. "Cleaning Tasks Knowledge Transfer Between Heterogeneous Robots: A Deep Learning Approach". In: *arXiv preprint arXiv:1903.05635* (2019).

[37] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[38] George Konidaris and Andrew Barto. "Autonomous Shaping: Knowledge Transfer in Reinforcement Learning". In: *Proc. of the 23rd Int'l Conf. on Machine learning*. ACM. 2006, pp. 489–496.

[39] James J Kuffner and Steven M LaValle. "RRT-Connect: An Efficient Approach to Single-Query Path Planning". In: *IEEE Int'l Conf. on Robotics and Automation (ICRA)*. Vol. 2. 2000, pp. 995–1001.

[40] Anders Boesen Lindbo Larsen et al. "Autoencoding Beyond Pixels Using a Learned Similarity Metric". In: *arXiv preprint arXiv:1512.09300* (2015).

[41] Daniel Leidner and Christoph Borst. “Hybrid Reasoning for Mobile Manipulation Based on Object Knowledge”. In: *Workshop on AI-based Robotics at IEEE/RSJ Int’l Conf. on Intelligent Robots and Systems (IROS)*. 2013.

[42] Sergey Levine and Vladlen Koltun. “Guided policy search”. In: *Int’l Conf. on Machine Learning*. 2013, pp. 1–9.

[43] Sergey Levine et al. “End-to-End Training of Deep Visuomotor Policies”. In: *Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[44] Sergey Levine et al. “Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection”. In: *Int’l Journal of Robotics Research* 37.4-5 (2018), pp. 421–436.

[45] Corey Lynch et al. “Learning Latent Plans from Play”. In: *arXiv preprint arXiv:1903.01973* (2019).

[46] Michael G Madden and Tom Howley. “Transfer of Experience Between Reinforcement Learning Environments With Progressive Difficulty”. In: *Artificial Intelligence Review* 21.3-4 (2004), pp. 375–398.

[47] Sina Masnadi et al. *A Sketch-Based System for Human-Guided Constrained Object Manipulation*. 2019. arXiv: `1911.07340 [cs.HC]`.

[48] Sina Masnadi et al. “Sketching Affordances for Human-in-the-loop Robotic Manipulation Tasks”. In: *2nd Robot Teammates Operating in Dynamic, Unstructured Environments (RT-DUNE)* (2019).

[49] Amir Mazaheri and Mubarak Shah. “Visual Text Correction”. In: *The European Conf. on Computer Vision (ECCV)*. Sept. 2018.

[50] Amir Mazaheri, Dong Zhang, and Mubarak Shah. "Video Fill in the Blank Using LR/RL LSTMs With Spatial-Temporal Attentions". In: *Proc of IEEE Int'l Conf. on Computer Vision (ICCV-2017)*. Oct. 2017.

[51] Adithyavairavan Murali et al. "PyRobot: An Open-source Robotics Framework for Research and Benchmarking". In: *arXiv preprint arXiv:1906.08236* (2019).

[52] Aaron van den Oord et al. "Wavenet: A Generative Model for Raw Audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[53] Tom Le Paine et al. "Making Efficient Use of Demonstrations to Solve Hard Exploration Problems". In: *arXiv preprint arXiv:1909.01387* (2019).

[54] Lerrel Pinto and Abhinav Gupta. "Learning to Push by Grasping: Using Multiple Tasks for Effective Learning". In: *2017 IEEE Int'l Conf. on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2161–2168.

[55] Rouhollah Rahmatizadeh et al. "From Virtual Demonstration to Real-World Manipulation Using LSTM and MDN". In: *Thirty-Second AAAI Conf. on Artificial Intelligence (AAAI-2018* (2018).

[56] Rouhollah Rahmatizadeh et al. "Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-to-End Learning from Demonstration". In: *Proc. of IEEE Int'l Conf. on Robotics and Automation(ICRA-2018)*. 2018, pp. 3758–3765.

[57] Aravind Rajeswaran et al. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations". In: *arXiv preprint arXiv:1709.10087* (2017).

[58] Jan Ramon, Kurt Driessens, and Tom Croonenborghs. "Transfer Learning in Reinforcement Learning Problems Through Partial Policy Recycling". In: *European Conf. on Machine Learning*. Springer. 2007, pp. 699–707.

[59] Tim Salimans et al. “Improved Techniques for Training GANs”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242.

[60] Ayumu Sasagawa, Sho Sakaino, Toshiaki Tsuji, et al. “Imitation Learning for Human-robot Cooperation Using Bilateral Control”. In: *arXiv preprint arXiv:1909.13018* (2019).

[61] Andrew Sendonaris and COM Gabriel Dulac-Arnold. “Learning From Demonstrations for Real World Reinforcement Learning”. In: *arXiv preprint arXiv:1704.03732* (2017).

[62] Hassam Ullah Sheikh and Ladislau Bölöni. “Emergence of Scenario-Appropriate Collaborative Behaviors for Teams of Robotic Bodyguards”. In: *Proc. of the 18th Int’l Conf. on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2019, pp. 2189–2191.

[63] Hassam Ullah Sheikh and Ladislau Bölöni. “Universal Policies to Learn Them All”. In: *arXiv preprint arXiv:1908.09184* (2019).

[64] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), p. 354.

[65] Avi Singh et al. “End-to-End Robotic Reinforcement Learning without Reward Engineering”. In: *arXiv preprint arXiv:1904.07854* (2019).

[66] Satinder P Singh et al. “Learning predictive state representations”. In: *Proc. of the 20th Int’l Conf. on Machine Learning (ICML-03)*. 2003, pp. 712–719.

[67] Domenico Spensieri et al. “Optimal robot placement for tasks execution”. In: *Procedia CIRP* 44 (2016), pp. 395–400.

[68] Freek Stulp, Andreas Fedrizzi, and Michael Beetz. “Learning and Performing Place-Based Mobile Manipulation”. In: *Int’l Conf. on Development and Learning (ICDL)*. 2009, pp. 1–7.

[69] Kaushik Subramanian, Charles L Isbell Jr, and Andrea L Thomaz. “Exploration From Demonstration for Interactive Reinforcement Learning”. In: *Proc. of the 2016 Int’l Conf. on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2016, pp. 447–456.

[70] Josh Tobin et al. “Domain Randomization for Transferring Deep Neural Networks From Simulation to the Real World”. In: *2017 IEEE/RSJ Int’l Conf. on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[71] Nikolaus Vahrenkamp, Tamim Asfour, and Rudiger Dillmann. “Robot Placement Based on Reachability Inversion”. In: *IEEE Int’l Conf. on Robotics and Automation (ICRA)*. 2013, pp. 1970–1975.

[72] Kelvin Xu et al. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”. In: *Proc. of Int’l Conf. on Machine Learning (ICML-2015)*. 2015, pp. 2048–2057.

[73] Brian Yang et al. “REPLAB: A Reproducible Low-Cost Arm Benchmark for Robotic Learning”. In: *Int’l Conf. on Robotics and Automation (ICRA-2019)*. 2019, pp. 8691–8697.

[74] Jing Yang, Patrick Dymond, and Michael Jenkin. “Reaching Analysis of Wheelchair Users Using Motion Planning Methods”. In: *Impact Analysis of Solutions for Chronic Disease Prevention and Management*. Springer, 2012, pp. 234–237.

[75] Zichao Yang et al. “Stacked Attention Networks for Image Question Answering”. In: *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-2019)*. 2016, pp. 21–29.

[76] Quanzeng You et al. “Image Captioning With Semantic Attention”. In: *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-2016)*. 2016, pp. 4651–4659.

[77] Dongfei Yu et al. “Multi-level Attention Networks for Visual Question Answering”. In: *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-2017)*. 2017, pp. 4187–4195.

[78] Licheng Yu et al. “MAttNet: Modular Attention Network for Referring Expression Comprehension”. In: *Proc. of IEEE Conf. on on Computer Vision and Pattern Recognition (CVPR-2018)*. 2018.

[79] Tianhe Yu et al. “One-Shot Imitation from Observing Humans Via Domain-Adaptive Meta-Learning”. In: *arXiv preprint arXiv:1802.01557* (2018).

[80] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. “Capturing Robot Workspace Structure: Representing Robot Capabilities”. In: *Proc. of IEEE/RSJ Int’l Conf. on Intelligent Robots and System (IROS)*. 2007, pp. 3229–3236.

[81] Franziska Zacharias et al. “Positioning Mobile Manipulators to Perform Constrained Linear Trajectories”. In: *Proc. of IEEE/RSJ Int’l Conf. on Intelligent Robots and System (IROS)*. 2008, pp. 2578–2584.

[82] Heiga Zen and Andrew Senior. “Deep Mixture Density Networks for Acoustic Modeling in Statistical Parametric Speech Synthesis”. In: *IEEE Int’l Conf. on acoustics, speech and signal processing (ICASSP-2014)*. IEEE. 2014, pp. 3844–3848.

[83] Andy Zeng et al. “Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching”. In: *Proc. of IEEE Int’l conf. on Robotics and Automation (ICRA-2018)*. 2018, pp. 1–8.

[84] Yuke Zhu et al. “Reinforcement and Imitation Learning for Diverse Visuomotor Skills”. In: *arXiv preprint arXiv:1802.09564* (2018).

[85] Yiming Zuo et al. “CRAVES: Controlling Robotic Arm with a Vision-based Economic System”. In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-2019)*. 2019, pp. 4214–4223.