Literature Review

Paper 1: One Shot Imitation Learning

**Abstract**

* Goal is to make robots learn new tasks shown with one demonstration and quickly generalize to new situations of the same task.
* A meta-learning framework is proposed, called 'one shot imitation learning'.
* A neural network is trained such that after being given a demonstration, when given the state corresponding to the second demonstration, it should predict the necessary action.
* Current systems require extensive feature engineering and lots of demonstrations.
* Hope is that by training this model on a variety of tasks, a generalization can be achieved to turn any demonstration into a robust policy for a variety of tasks.

Comparison between traditional and one-shot imitation learning

A screenshot of a social media post

Description automatically generated

**Introduction**

* Showing a task by demonstration has several advantages over using linguistic communication. It is more convenient for certain tasks.
* To make the model work, they use ‘soft attention’ for processing the sequence of states and actions that correspond to that demonstration.

**Related work**

* Behavioral cloning (performs supervised learning from behaviour to actions) and IRL (rewards functions model the demonstration as optimal behaviour) are commonly used techniques.
* Using reinforcement learning requires a large amount of time to learn from trial and error. Furthermore, it requires the design of a reward function which is more difficult than actually showing a demonstration of the optimal task action required.
* This approach relies heavily on ‘soft attention’ model

**Notation**

A screenshot of a social media post

Description automatically generated

**Task**

To stack the blocks in a given order in towers of different height with varying initial configurations. Because the initial condition keeps varying, it requires the learned policy to generalize within a given demonstration.

**Algorithm:**

* Behavioural cloning and DAGGER are used to train the neural network. In this approach, we don’t need to specify a reward function. This makes the algorithm more scalable as it is easier to demonstrate the task than creating a unique reward function for the task at hand.
* Start by collecting a set of demonstrations for each task and add noise to the actions to have a wider coverage of the trajectory space.
* Observation: List of object positions relative to the gripper and if the gripper is open or closed.
* Stage: Used for an event of stacking one block on top of another
* Algorithm Used: Adamax to perform optimization with learning rate of 0.001

**Architecture**

* This is one of the major contributions to this paper, and it consists of the following three modules:
* Demonstration network
* Context network
* Manipulation network
* **Demonstration Network:**

**Temporal Dropout:**

* Receives demonstration trajectory as input and produces an embedding of the demonstration to be used by the policy.
* Stacking of blocks can produce sequences spanning extremely long-time steps, and training with such large sequences can be demanding both in time and memory.
* They randomly discard a subset of the time steps during a demonstration, and this process is called temporal dropout. Let p = proportion of time steps discarded. They use p = 0.95

A screenshot of a social media post

Description automatically generated

**Neighbourhood Attention:**

* Since the policy will have varying number of blocks during demonstration as its input, it needs to have the ability to process variable-dimensional inputs.
* Soft attention is a process which maps variable dimensional inputs to fixed dimensional outputs.
* This can result in loss of information and becomes a problem as the number of blocks increases, as the amount of information lost will also increase.
* Manipulation network is the simplest network, uses MLP to plan an action to

Need to Explore:

* What is meta-learning?
* What is soft attention? – Focusing on a given subset of the total input rather on the entire input, where the chosen subset is in focus and the rest is blurred. Soft attention is deterministic, while hard attention is stochastic.
* What is DAGGER?

Shallow-Depth Insertion: Peg in Shallow Hole Through Robotic In-Hand Manipulation

Abstract:

* A method to perform shallow hole insertion using a parallel jaw gripper
* Different from the well-studied peg-in-hole assembly in terms of the need for dexterity along with the way of grabbing the part
* Traditionally, the peg is held on the sides which will make contact with the wall of the deep hole
* Suitable for simple low DOF parallel jaw grippers with no tactile sensing

Deep Reinforcement Learning for Industrial Insertion

Tasks with Visual Inputs and Natural Rewards

Abstract:

* Connector insertion tasks requires first order modelling to capture physical effects.
* Traditional controllers in such tasks can be inaccurate and need to be manually tuned
* Paper shows that complex insertion tasks can be solved with RL where the goal is natural sparse reward signals (+1 for successful electrical connectivity and 0 if not) and goal images

Introduction:

* Reinforcement learning relies on trial and error
* Deep RL handles high dimensional inputs
* Deep RL has thus far not seen wide adoption in the automation community due to several practical obstacles such as sample efficiency: tasks must be completed without excessive interaction time or wear and tear on the robot
* Another obstacle is specifying the goal when sparse rewards are present and designing a reward function in such situations manually.

Approaches:

1. End-to-end approach that learns a policy from images (which serve as state representation and goal specification). Visual data also provides robustness to sensor and actuator noise
2. Using simple sparse rewards. Sample requirements are reduced by using the prior knowledge on the task. To handle this challenge, we extend the residual RL approach, which learns a parametric policy on top of a fixed, hand-specified controller, to the setting of vision-based manipulation.

* The use of neural networks in RL is called “deep” RL.

Methods:

Various RL algorithms are evaluated to choose the most appropriate for the insertion of various cables task:

1. Markov Decision Process (MDP)
2. Efficient Off-Policy Reinforcement Learning
3. Residual Reinforcement Learning
4. Learning from Demonstrations

Experiments:

* Residual RL with easy-to-obtain reward signals
* Two types of natural rewards that are intuitive for users to provide: an image directly specifying a goal and a binary sparse reward indicating success.
* Questions to be answered:

1. aim to answer the following questions: (1) Can such trained policies provide comparable performance to policies that are trained with densely-shaped rewards?
2. Are these trained policies robust to small variations and noise?

Results:

1. Experiments show that a **successful and consistent vision-based insertion policy** can be learned from relatively few samples using residual RL. The distance between images corresponds to a relatively dense reward signal which is sufficient to distinguish the different stages of the insertion process
2. All methods are able to achieve very high success rates in the sparse setting.
3. Results of the experiment with sparse rewards outperforms standard RL having perfect state information, with standard RL exhibiting only initial performance superiority. Residual RL and RL with learning from demonstrations both solve the task relatively quickly, while RL alone takes about twice as long to solve the task at the same performance.

Future Work:

* One practical direction for future work is focusing on multi-stage assembly tasks through vision. This would pose a challenge to the goal-based policies as the background would be visually more complex. Moreover, multi-step tasks involve adapting to previous mistakes or inaccuracies, which could be difficult but should be able to be handled by RL. Extending the presented approach to multi-stage assembly tasks will pave the road to a higher robot autonomy in flexible manufacturing.

To explore:

1. How do you manually shape this reward function?