CS229 Project Report: Imitation Learning for Robot Arm Peg-in-Hole-Insertion
Imitation Learning / Robotics

Hao Li
Department of Mechanical Engineering
li2053@stanford.edu

Kaixin Chen
Department of Mechanical Engineering
kaixinc@stanford.edu

Tianheng Shi
Department of Mechanical Engineering
ts5@stanford.edu

Abstract

This project focuses on building algorithms that can control a robot arm to perform peg insertion using visual representation that has already been encoded by a convolutional neural network. This project propose an new algorithm that uses transformers and out performed a baseline method implemented using fully connected network.

1 Introduction

1.1 Background and Motivation

Inspired by the Amazon Picking Challenge, which can be review in [2], the team found that the combination of robot manipulation grasping task and computer vision contains many challenges. In this competition, the robot was provided with a list of objects' names on a shelf. Given the visual input from the robot, a grasping task and a peg insertion will be performed. This initiates our interest.

In the past decade, machine learning has been in a phase of rapid development. Machine learning in robotics, one of the major applications of machine learning has also been a hot field. The application of machine learning and learning based control in robotics can potentially improve robots' performance and evolve robots from only being able to perform a specific set of tasks with a fixed policy to become capable in complex tasks and even in unseen environments. This is essential that for robots to work autonomously in homes and in public spaces where the tasks, surroundings and potential threats that robots might cause or face are more diverse than the common settings of robots now days, for example, factories.

Among the variety of robot manipulation tasks that a learning based controller can be trained to handle, the focus of this project is peg insertion, as learning to put something at the right place is an important prerequisite for countless complex tasks and peg insertion requires the robot aligning the peg with a hole precisely which made it arguably one of the hardest among similar tasks. Our project does not have a specific setting as peg insertion should be easy to generalize to many settings, for example, plugging something into the power supply in someone's home.

1.2 Problem Statement

For an algorithm to perform peg insertion, good sensory input is a prerequisite to perform precise closed loop control. In our setup this input is from a third person view RGB camera that can see the robot arm's end gripper and the hole. The output of the entire algorithm is the corresponding action that the robot should make. We split this big algorithm into two modules, the vision module handle the input of the camera and process it into more effective representation and the control module takes the more effective representation to generate action. Here in the CS229 project, the focus is to build a novel control module for imitation learning. We assume that we already have a well performing vision module, and the input of our control module is the vision information encoded by the vision module and the output is action.

This is a joint project with CS231N (where we proposed a new algorithm for building a vision module), so this assumption that there is already some working vision module is met by borrowing the baseline vision module built in the CS231N project. However, this CS229 project do not take any credit for the vision module borrowed (and there is not much credit to take either as that vision module is simply calling a PyTorch library). Also, because the two projects focus on different aspects of the same task, the data set and data loader are the same. CS231N project also borrowed the baseline method control module from this project, and not take any credit for it too.

2 Related Work

For peg insertion and many similar robot manipulation tasks like grasping and placing, two categories of approaches have been very popular, those are imitation learning and reinforcement learning.

The first category is reinforcement learning [9] [1] [5], which usually provides the state-of-the-art performance in learning based controlling. Reinforcement learning is achieved while an algorithm interact with the environment to collect data. However, a successful reinforcement learning algorithm depend on two major assumptions. First is a good reward function and second is a interactive environment. In fact, each of the two prerequisites are hard to met. A good reward function captures all possible aspect and the relative weightings between each of the rewards need to be optimal. Also, an interactive environment is hard to get in the setup of peg insertion as in reinforcement learning, it is hard to prevent the robot trying something that damages itself.

The second category is imitation learning. Instead of having the algorithm interact with the environment and collect data, a set of training data is prepared. Our algorithm is an example of this category. Imitation learning algorithms learns more about a task from expert demonstrations. These expert demonstrations can be a driver driving a car [3], a controller performing peg insertion [4] and following human instructions [7]. What is learned by the algorithms varies, some imitation learning algorithms learns the reward that the expert wants to achieve (inverse reinforcement learning), and some learns the execution policy of the expert directly (behavior cloning). While the imitation learning algorithm relaxed the assumptions that there are good reward function and a interactive environment, the problem raised from the fact that many demonstrations in the field of imitation learning are demonstrations by human, and human demonstrations are often sub optimal. While some work bypasses these issues using synthesised optimal demonstrations [4], many works directly addressed this issue [10] [6] and made improvements in algorithms.

Also it is worth noting that there are some overlaps between imitation learning and reinforcement learning. For example, inverse reinforcement learning learning a reward function first and the use reinforcement learning with the learned reward function. Also there are a few other works that do not belong to any of the two categories.

3 Dataset

The dataset used is from Stanford Vision and Learning Lab, we did not collect the data our self. There are a approximately 80 demonstrations of peg insertion over time and each demonstration provides about 100 time steps, giving us about 8,000 data points. The train and test split is approximately 70:30. In each of the time step, the data includes the image taken at that time step and the corresponding ground truth action (demonstration action). The image input are RGB image at each time step with a resolution of 480 by 640. The ground truth action is three dimensional and is discrete, this is because the robot arm's end gripper can only move either negative one, zero or one unit step along each of the x, y and z directions at each of the time step. This make the problem into classification problem with 27 classes (three times three times three), the size of the unit step is 1.5mm.

The data was not shuffled. The data is not only loaded in a sequence, but as a sliding window. This is different to what was implemented in the CS231N project. The sliding window works as the following: an extra hyper parameter "history length" was defined, and the data loader will not only load the vision input and ground truth action at the current times step, but also the history of the current time step up to history length. Namely, if history length is zero, it would be an ordinary data loader, and if history length is one and current time step is three, the data loader will load the vision input and ground truth actions at time step two and time step three.

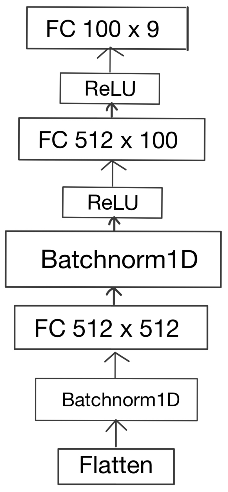
The image at each of the time step was prepossessed. It is first down sampled to 180 by 120 pixels and then randomly cropped to 162 by 108. Color jitter was added afterward. these steps serves as data argumentation. The ground truth action was also prepossessed. To make the ground truth actions more suitable for a classification problem, the ground truth action are mapped to a class number, we define making a negative unit step along certain axis as class zero, making zero unit step as class one and making one unit step as class two.

4 Methods

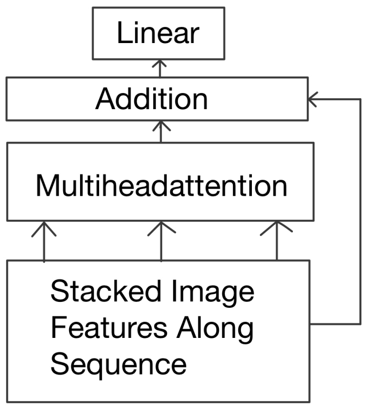
For the image feature extraction, Hao and Kaixin designed a model with ResNet18 pretrained on ImageNet for Class CS231n. That model is used as the upper stream feature extractor for this project. Under the field of imitation learning, behavior cloning is the method chosen. Behavior cloning directly learns the policy of the expert. It is a supervised learning algorithm to have the robot directly learn the manipulation policies. Among all models we used, the problem of peg insertion was framed as a series of classification tasks.

4.1 Models

For the baseline model, a 3-layer MLP model was chosen. The input of the model is the images from a fixed camera from dataset. No pose information or secondary camera images are being used. It is consists of three linear layers that mapping the flattened images features from size of 512 to 512, 100 and 9 sequentially. Batchnorm1d layers are inserted in between. ReLU function is used for the activation function. It's structure is shown in Fig. 1a. The output dimension is designed as 9. It can be treated as an one-hot vector that contains a 3-class probability distribution for Cartesian coordinates x-y-z. The 3 classes for each axis would be correspond to moving to negative direction by unit length, staying still, and moving to positive direction by unit length.



(a) Baseline Model



(b) MultiheadAttention Based Model

Figure 1: Model Structures

To be the baseline model, the team implemented a transformer based model with pytorch framework.

The intuition behind choosing a transformer based model is because of the expert demonstration used are human demonstration and the fact that human demonstrations are sub-optimal. This includes the fact that human decisions and demonstrations are non-Marcovian: the current action not only depend on the current state but also the history of states. As such, making the history of visual inputs available to the controller can potentially help the controller to learn the human demonstration better. [6] has demonstrated that using an RNN network potentially improves the performance of behavior cloning with human demonstration. However, transformers have various benefits over RNN: for example, it is easier to parallel. As such, transformer has been given high expectation.

The multiheadattention layer from pytorch implemented the self-attention layer which is a essential layer of a transformer[8]. Since history length could effect the performance of the model, the input sequence length was set as a hyperparameter. The sequence history lists the team tried are 1, 2 and 3. The detailed structure of the transformer based model is shown in Fig. 1b.

4.2 Evaluation Metric

Since the behavior clone task here can be seen as an action prediction/classification task, accuracy is used as the evaluation metric.

$$Accuracy = \frac{\sum_{t=0}^{n}\sum_{j=0}^{2} 1\{a_{predicted\_tj} = a_{experts\_tj}\}}{3n} \tag{1}$$

Here, t corresponds to time step and j corresponding to Cartesian coordinate axis.

5 Experiment Details

Multiple loss functions was tried for the optimization. At the end, KL divergence was selected and used for all the training runs. To avoid underflow issues during computations, both the logits from model output and the target one-hot vector of the ground truth action were scaled into log space.

During the training for the baseline model, Adam optimizer was used. Learning rate was set to 0.001 with linear decay. The batch size was 16. The training process was terminated after the losses decay became negligible.

For Multihead-Attention based models with history length are 1, 2 and 3, the models are trained with the following hyper-parameters: Learning rate was set to 0.02 with no decay. The batch size was 8, the model was trained for 80epochs. In addition, l2 regularization of strength of 1×10^{-5} was used.

6 Result

The learning curves for all models are presents in Fig. 2. The baseline behavior cloning model yields the training accuracy of 0.824 and validation accuracy of 0.7981. For the transformer models, the input history sequence of length 2 and 3 didn't beat the baseline model, having the validation accuracy of 0.7771 and 0.6789. However, the transformer model with input history sequence of 1 achieved better results: having a validation accuracy of 0.8328.

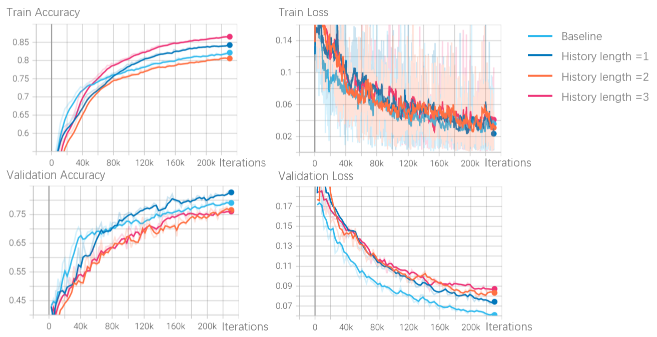


Figure 2: The Result obtained after training out novel and the baseline method

After the validation, the team performed real robot test with the multihead-attention based model. A screenshot of a success case is shown in Fig, which serves as our qualitative result. **??**. The final test success rate is 92%.

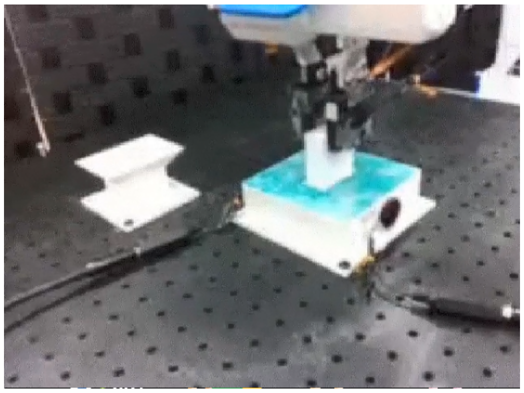


Figure 3: Screenshot during Real Robot Test

Experiment	Discription	History length	Best Training Accuracy	Best Validation Accuracy
1	Fully Connected (Baseline)	0	0.824	0.7981
2	Transformer	1	0.8428	0.8328
3	Transformer	2	0.8086	0.7771
4	Transformer	3	0.8657	0.7689

Table 1: Results for each of the experiments

6.1 Discussion

For the baseline model, the validation accuracy has reached to a decent value, and the performance of the novel method with history length of 1 beats the baseline. Note that history length of 1 includes the information of two time steps, which is not the baseline.

For the baseline and history length of 1 and 2, the gap between training and validation accuracy is small, showing that there isn't much over fitting. However, over fitting is observed in history length of 3.

As expected, the application of transformer furthermore increased the performance of the behavior cloning task. From the results, it can be observed that the length of the input history sequence can affect the performance. This intuitively makes sense because the decision of the current time frame is based not only current state but a history of states. However, as observed, the performance of the model is not increasing with the length of history. An explanation would be that our model do not have temporal encoding. Namely, it might be ambiguous to the network that which input if the earlier history and which one is the more recent history. The longer the history the more ambiguity there is, and eventually causing the models with longer history to perform worse.

7 Future Works

Due to computing power limitations and time limits, for the Multihead-Attention based model, only 3 history lengths were tried. Hyperparameters tuning could be done more for this model, including longer history length, different learning rate, batch size, etc. In addition to tuning, since it was hypothesised in the discussion that the lack of temporal encoding is causing the models with longer history to perform worse, adding temporal encoding should also be tried.

Besides, RNN model or LSTM models could also be tried as an alternative. While transformers generally trains faster, there might be possibilities that LSTM can out perform transformers.

References

[1] Cristian C. Beltran-Hernandez, Damien Petit, Ixchel G. Ramirez-Alpizar, and Kensuke Harada. Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach. *Applied Sciences*, 10(19), 2020.

[2] Eppner Clemens, Höfer Sebastian, Jonschkowski Rico, Martín-Martín Roberto, Sieverling Arne, Wall Vincent, and Brock Oliver. Four aspects of building robotic systems: lessons from the amazon picking challenge 2015. *Auton Robot*, 42, 2018.

[3] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning, 2017.

[4] Yifang Liu, Diego Romeres, Devesh K. Jha, and Daniel Nikovski. Understanding multi-modal perception using behavioral cloning for peg-in-a-hole insertion tasks, 2020.

[5] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016.

[6] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[7] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[9] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester, and Jon Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 754–760, 2019.

[10] Songyuan Zhang, Zhangjie Cao, Dorsa Sadigh, and Yanan Sui. Confidence-aware imitation learning from demonstrations with varying optimality, 2021.

Appendices

A Contributions

Hao Li: Participated in project proposal. Coordinated with Stanford Vision and Learning lab for data. Performed data prepossessing and reformatting. Designed the data loader. Designed all the code architecture and APIs using Pytorch Lightening. Participated in debugging the ML model. Designed the novel algorithm.

Tianheng Shi: Participated in project proposal and midterm milestone. Read several papers for literature review and potential implementation ideas. Designed the imitation learning baseline model. Took leading role in debugging the ML model. Redesigned the implementation of the loss function. Participated in debugging the AWS server connection issues. Participated in attempts of designing a novel algorithm . Participated in experimenting.

Kaixin Chen: Participated in project proposal and midterm milestone. Set up an AWS server for training and corresponding environments. Read several papers for literature review and potential implementation ideas. Designed the original loss and accuracy metrics. Took leading role in debugging the ML model. Participated in attempts of designing a novel algorithm. Performed almost all of the experiments.