Alvin Lee

12/14/22

Capstone Project Write Up

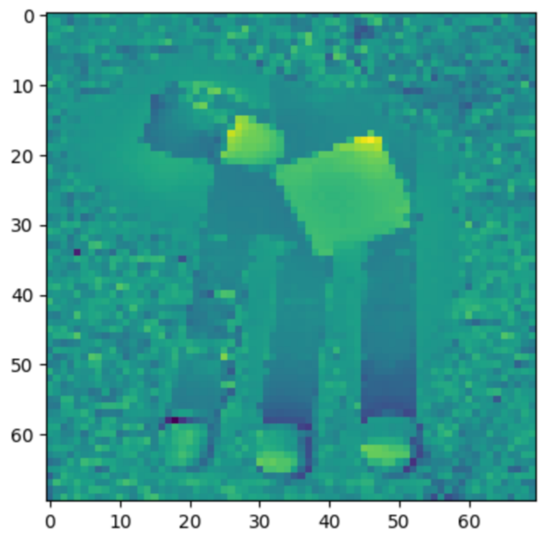
Introduction

In this project, I was tasked with predicting the state of a 4-fingered robot hand using RGBD (RGB + Depth) images. The goal of the project was to implement a simple supervised learning algorithm that could take in RGBD images of the hand from a top view and output the positions of the tips of each finger in meters.

Method

To accomplish this, I first had to develop a dataset class that could load the data, apply necessary augmentations, and return a tensor at each iteration of our training procedure. I started by implementing PyTorch's abstract dataset class to facilitate the loading and preprocessing of the dataset. This allowed me to train our model efficiently and effectively on the dataset.

I first started by defining the '__init__' method, which was used to initialize the dataset. In this method, I loaded the data as well as a transformation method if there was one. I then implemented the '__getitem__' method which was used to retrieve a single sample from the dataset. I also set two variables, angle, and color, so that the user could set which camera angle or color was used to train the data. I instantiated an instance of the dataset class and passed it to the PyTorch ‘DataLoader” class which was used to iterate over the dataset and feed the data to the model.



To the left, we have an example of what the image looked like after resizing it to a 70x70 image, and a normalization to the mean and standard deviation of the data.

When creating the dataset, I chose to choose a small batch size, diving the length of the dataset by 15 for each batch. I also shuffled the data so that each time I trained the model, it would be able to generalize better.

The proposed algorithm is a supervised learning algorithm, which means that it is trained on a dataset of labeled examples. In this case, the dataset consists of RGBD images of the robot hand, along with the corresponding positions of the tips of the fingers. The algorithm uses a convolutional neural network (CNN) to process the RGBD images and extract relevant features. I implemented the CNN architecture using the PyTorch framework. The CNN consists of three convolutional and pooling layers followed by two fully connected linear layers at the end. I used the ReLU activation function for all layers, except the output layer, where I did not use an activation function.

Once the CNN was implemented, I trained the model using the preprocessed RGBD images and their corresponding finger positions. I chose to use the Adam optimizer rather than the Stochastic gradient descent (SGD) algorithm mainly because Adam can automatically tune the learning rate for each parameter. I knew that I had the processing power capable of using Adam. The loss function defined uses PyTorch's MSELoss, as we are trying to minimize root mean squared loss.

Results and Discussion

The results of our experiment show that using a convolutional neural network (CNN) for state prediction of a 4-fingered robot hand using RGBD images is a successful approach. Our model was able to accurately predict the 3D positions of the fingers in the hand, achieving a mean squared error of less than the benchmark given by the example model.

Throughout the project, there were many adjustments I made to tune and optimize the model. I noticed that many adjustments had little effect on the effectiveness of the model. For example, I noticed that normalizing the data did not have much effect on the prediction of the finger state. This is likely since in the image, the fingertips are white, and the rest of the hand is black, meaning that they were already quite distinct. This means that training the data on the unnormalized data was still effective.

One change that brought me beneficial results was changing the activation function for the output layer. Initially, I tried using a non-linear activation function on the output layer, however I was not getting the results I wanted. I decided to stick with a linear activation layer for the output instead, and just have the model train on it. This resulted in the most accurate predictions, giving me close to the benchmark score.

There were also many changes to the CNN that did not improve the performance of the model. I found that increasing the number of layers had little effect on the performance on the model. Also, increasing the number of convolutional features also had little effect on the performance. To investigate this, I trained several versions of the CNN with different settings. I found that at the end, the performance that was the most effective was having 6 convolutional features, a kernel of size 5, and three convolutional layers.

Overall, the model performed well on the task of predicting the state of the 4-fingered robot hand using RGBD images. The use of a CNN allowed me to effectively learn the relationships between the input images and the positions of the fingertip, resulting in accurate predictions. I was able to achieve a low MSE loss on the test set, indicating that the model was able to generalize well to unseen data. The project demonstrated the effectiveness of using a CNN for predicting the state of a robot hand using RGBD images. By carefully designing the dataset class and carefully tuning our model, I was able to achieve good performance on this challenging task.

Future Work

This project showed me the potential of machine learning in the real world. One potential direction for future work I would be interested in is exploring the use of other types of data. For example, how this could be used in prosthetics. It would be interesting to investigate the use of reinforcement learning and using the state predictions generated by the CNN model as part of a reinforcement learning algorithm. This could allow the robot to learn how to perform tasks such as moving or grasping objects.