COMP341 - Robot Perception and Manipulation

Project Report

Assignment 1

Grasp Detection with Neural Network

Student Group 3

| Introduction

This assignment includes several attempts to use deep learning techniques to train the neural network to learn a grasp detection, each of the grasp detection value consisting of five elements, they are : (x ; y ; rotation angle ; opening ; jaw size). The structure of this report file is divided into the following seven parts:

1. Grasp Detection using CNN + RGB image
2. Evaluations
3. Grasp Detection using CNN + RGB and Depth Image
4. Grasp Detection in the Wild
5. User Instructions
6. Group Contributions
7. References/Related Works

| Grasp Detection using CNN + RGB image

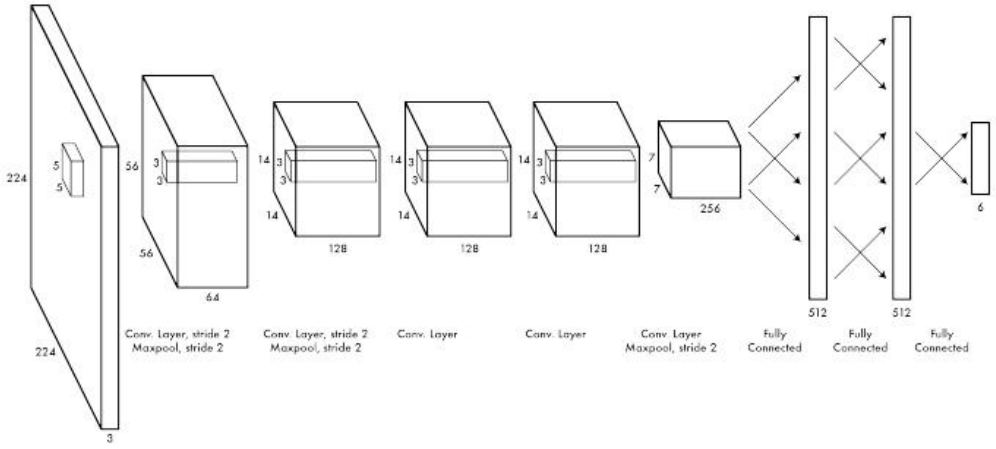
As suggested in the assignment specification, two architectures were used to perform the grasp detection, one of them are being tested and used in the main code for the later sections, the other one, however, due to its worse performance is not used in later sections, but instead its code was provided individually.

Please refer to the user instructions section for more information about the file structure. The following content of this section will explain each of those architectures and illustrate their training loss performances.

1. Direct Grasp Detection with and without batch normalization

This is the version that tested stable and was later used in other sections.

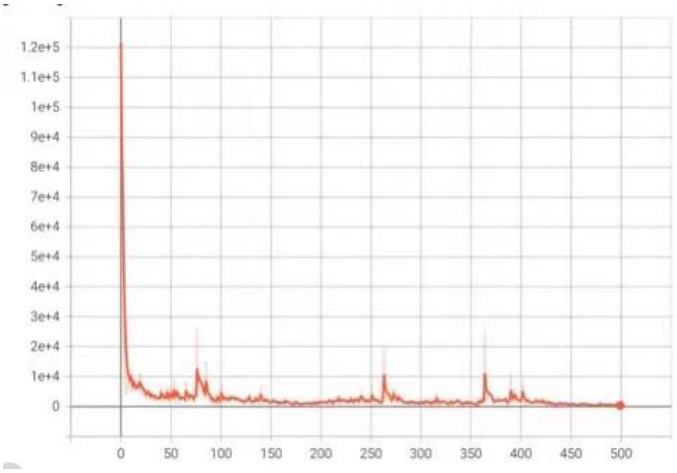
It takes an RGB image tensor of shape (N*3*1024*1024) as input and returns a single grasp detection output array of 5 elements. For each image, it`s ground truth is selected by the one with the highest Jaccard Index value of the current prediction. The architecture of the network is as in below:



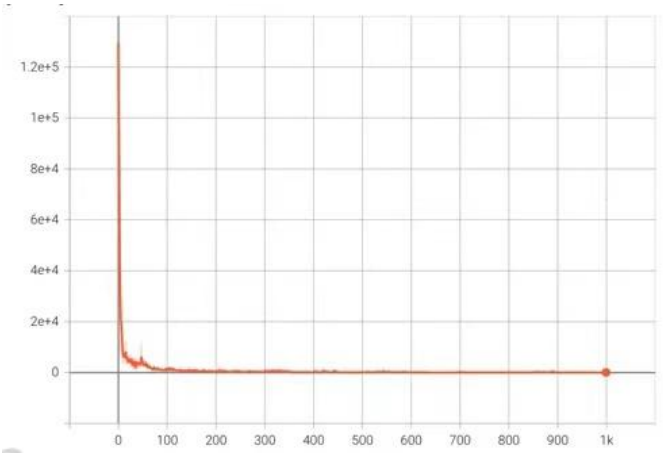
This architecture is referenced from Joseph Redmon and Anelia Angelova`s paer[0].

This architecture is later improved by us by adding, for each convolutional layers, a batch normalization.

Please see the following figure of its performance of loss function values over episode, note that each episode takes a batch of 5 images as input.



Without Batch Normallization



With Batch Normallization

2. Multi-Grasp Detection

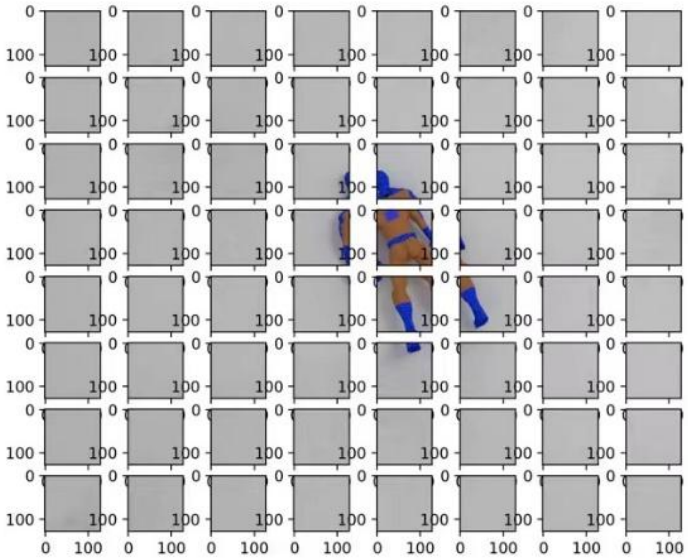
This part of the code is NOT included in the main code file, instead it was provided individually in our submission, please refer to the user instruction section for more details about this.

Please also note that this Multi-Grasp Detection is an implementation of the Multi-Grasp Detection method proposed by Joseph Redmon and Anelia Angelova`s paer[0], because of that, the networks architecture remains the same as Direct Grasp Detection.

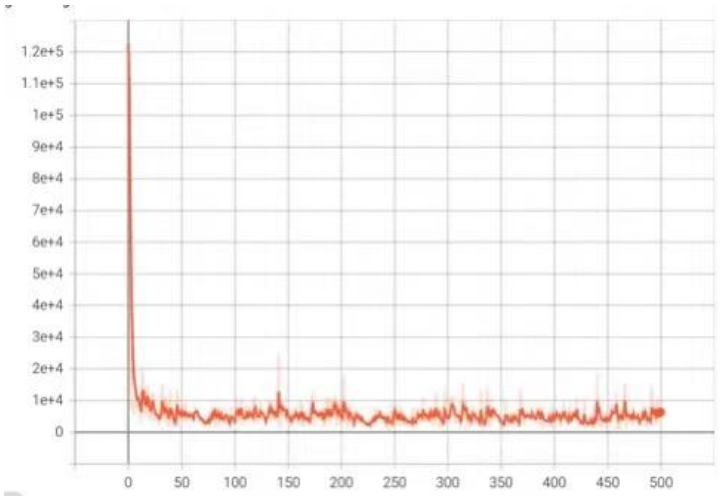
The network in this part takes the input tensor of shape (64*3*128*128), where it is dividing the RGB image into a grid of 8*8 cells, each cell has shape (128*128*3), since

the original image has shape (1024*1024*3).

For each of the cell, the neural network produces 6 elements output: (heatmap; x ; y ; rotation angle ; opening ; jaw size), where the heatmap is a probability of a single region contains a grasp.



Please see the following figure of its performance of loss function values over episode, note that each episode takes a batch of 1 image, but 64 cells as input.



This implementation is not used in later analyzation since we have got some problem with its backpropagation, that is also the potential reason why the above graph is not welly converging.

| Evaluations

The evaluation method we used is Rectangle Metric, based on Jaccard Index, as mentioned in the lectures. Since we have multiple ground truths, therefore, for each prediction result, we iterate through all its ground truth values, if one of the ground truths values and the predicted value satisfies both the following conditions, then the predicted value is referred as correct:

1. The difference of the rotational angle prediction is no larger than 30 degrees.
2. The Jaccard index (or IOU ratio) of two parallel rectangles is no larger than 0.25.

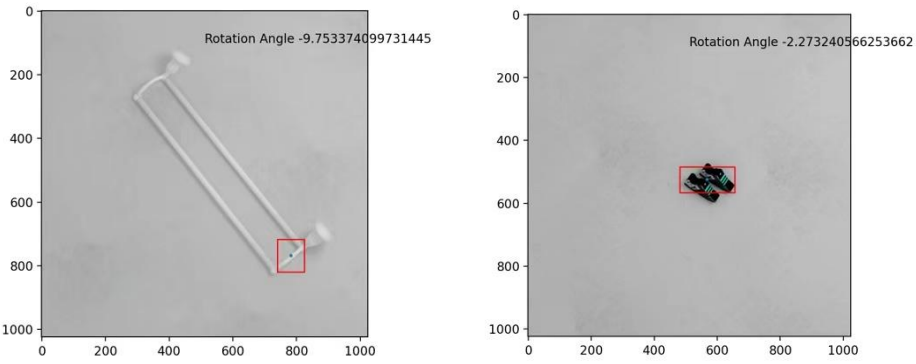
Jaccard index (or IOU ratio) in our implementation is calculated as the Intersection area divided by Union area between the predicted rectangle and the ground truth rectangle, assuming no rotation is performed (parallel), as shown in below.

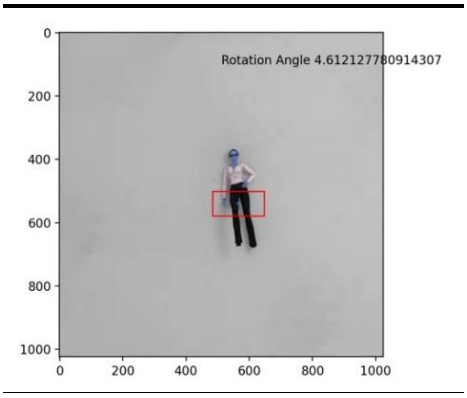
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

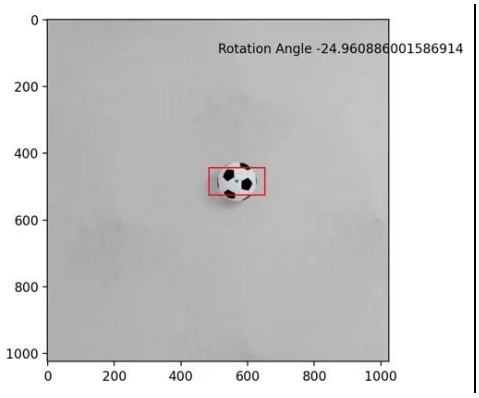
By using the above method, we evaluated the results produced by the direct grasp detection with batch normalization and without over the testing data, both trained 500 episode. The direct grasp detection with batch normalization predicted 13 correct predictions over 19 images, achieves **73%** accuracy, whereas the one without batch normalization produces only 8 correct predictions over 19 images and achieves **43%** accuracy. This is because the later one converges far slower than the one with batch normalization, under the same number of training episode, furthermore, the batch normalization also helps the earlier one avoids gradient explosion / dilation in DNN.

Please see the following visualized result produced by the direct grasp detection with batch normalization. Note that the rotation of the rectangle in degrees is not performed but instead marked in the result as negative stands for rotation anticlockwise about the rectangle center.

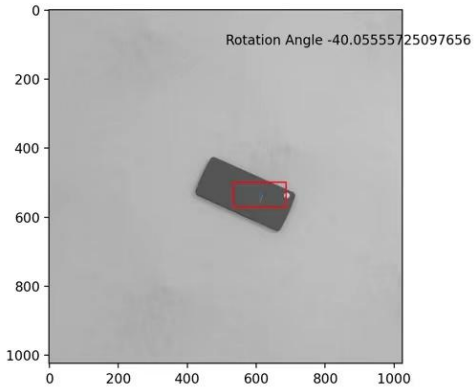
Correct Predictions

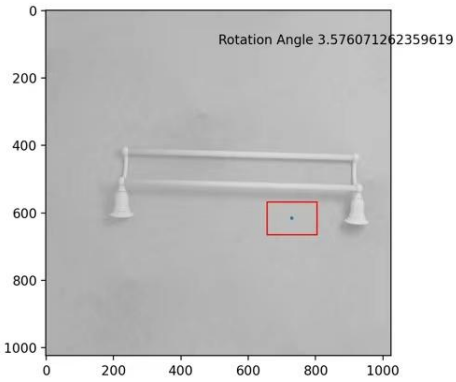






Incorrect Predictions

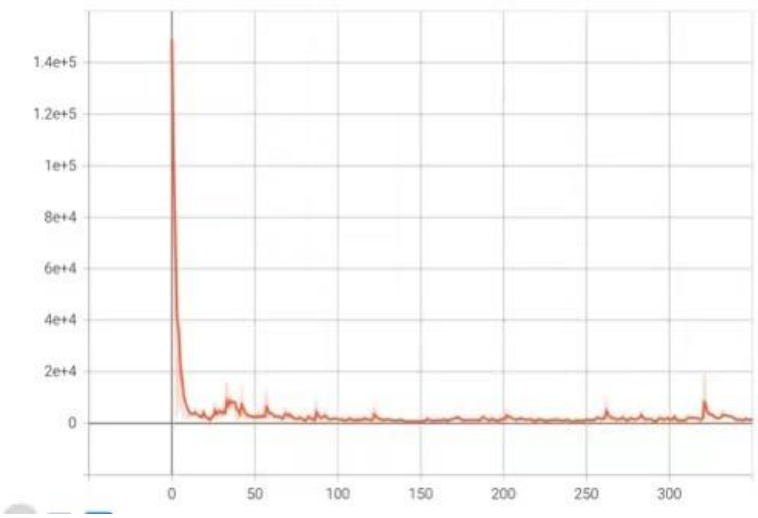




| Grasp Detection using CNN + RGB and Depth Image

In this section, instead of passing only the RGB image (1024*1024*3) into the network, we pass the image with shape (1024*1024*4) into the network, with additionally a depth channel. In this case, the architecture of the neural network are mostly remains the same, but with the first convolutional layer receiving an input of 4 channels instead of 3.

Note that in our implementation, the "*perfect_depth.tiff*" was used for each of the image. Please see the following figure of its performance of loss function values over episode, note that each episode takes a batch of 5 images and their corresponding depth data.

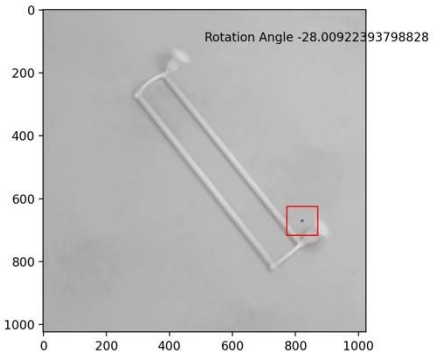


After the careful evaluation process, we obtained the accuracy rate of **47.3%** by using RGBD data. It is clear to tell that this performance is not as expected, higher than only use RGB data, it is probably due to the lake of training time, so called under-fitting, since for the RGBD input we have got 4 channels, that brings 1*1024*1024 more weights to the network, therefore we need to train the network many times more than the current 500 episode.

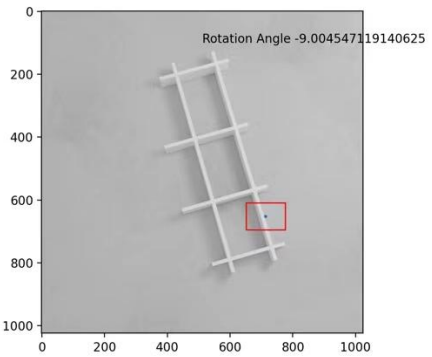
Furthermore. In such a case of deep neural network, it is also worth to mention that a ResNet can be implemented to improve the performance, however, due to the time limitation we have not got enough time to test that out.

The following shows to you a sample of image for the RGBD network predicted correctly and incorrectly, one for each.

Correct



Incorrect

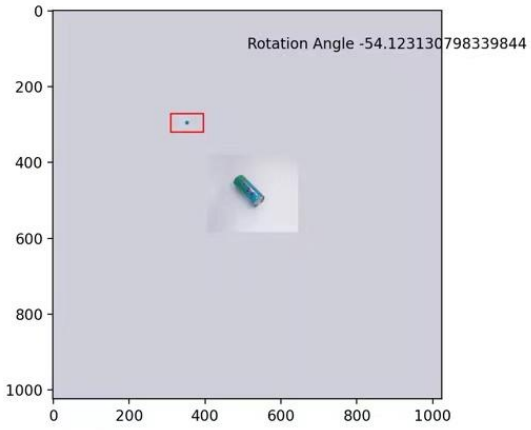


| Grasp Detection in Wild

We have collected 10 wild data from the internet[1] to test out our direct grasp detection architecture with batch normalization. The collected images are shown in blow:

Before feed the image into our network, each of them has been filled into 1024*1024 pixels to fit the network`s requirements. Unfortunately, we have obtained accuracy of **0%** on wild images, the issue was firstly assumed to be the padding method, however, no matter how we fill the image, the network is still not output well. We then turned to try image normalization before feeding, the result is still not good.

Please see the following of one of the wild data result:



The issue was caused maybe by the problem proposed by MobileNet V2, the general problem of the deep neural network, that is, the deeper NN will cause more and more information loss, as shown below:

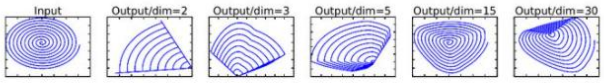


Figure 1: Examples of ReLU transformations of low-dimensional manifolds embedded in higher-dimensional spaces. In these examples the initial spiral is embedded into an n-dimensional space using random matrix T followed by ReLU, and then projected back to the 2D space using T^{-1}. In examples above $n = 2, 3$ result in information loss where certain points of the manifold collapse into each other, while for $n = 15$ to 30 the transformation is highly non-convex.

With that be in mind, a possible improvement is to use Residual Deep Neural network for training, however, that part is not implemented by our team due to the time limitation.

| User Instructions

Our final submission file contains the following instances:

- am01.py
- am01_attmp2.py
- data_net.pkl
- data_net2.pkl
- d_data_net.pkl
- Data
- wild
- log

The main implementations are in **am01.py**, run the file and follow the command line interface to walk through all four steps required by assignment specification. Note that this file uses the direct grasp detection with batch normalization. The training process will automatically save the network model once all 500 episode is done.

The Multi-Grasp Detection implementations are in **am01_attmp2.py**, this file only contains its network and training implementation.

data_net.pkl contains the saved model for the direct grasp detection with batch normalization mentioned in this paper, trained 500 episodes.

data_net.pkl2 contains the saved model for the direct grasp detection without batch normalization mentioned in this paper, trained 500 episodes.

d_data_net.pkl contains the saved model for the direct grasp detection with batch normalization and depth information as mentioned in this paper, trained 500 episodes.

Please note that the RGB evaluation option in the command line interface only loads model that namely as **data_net.pkl,** if you wish to see the evaluation of **data_net.pkl2,** please alter the file name.

Data folder contains the original data provided by the assignment specification.

wild folder contains the wild data collected by our team.

log folder is generated automatically by Tensorboard to record the real-time performance of DNNs.

| Group Contributions

Yuhang Song 201403970
Yiming Li 201435477
Yang Zhang 201433135
Shenpu Zhou 201418077
Jiahao Li 201384517

| References / Related Works

[0] Redmon, Joseph, and Anelia Angelova. "Real-time grasp detection using convolutional neural networks." *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015.

[1] https://www.kaggle.com/oneoneliu/cornell-grasp