DSCI6005 : Final Project

A Report on Deep Learning for Detecting Robotic Grasps

Jaichitra (Jc) Balakrishnan

**Abstract**

Deep learning has significantly advanced computer vision and natural language processing. While there have been some successes in robotics using deep learning, it has not been widely adopted. In this project, the proposed robotic grasp detection system predicts the best grasping pose of a parallel-plate robotic gripper for novel objects using the RGB-D image of the scene. The proposed model uses a deep convolutional neural network to extract features from the scene. The model also uses as input a ResNet trained model that classifies the object type. This is in addition to the image and grasp boundary inputs from the Cornell baseline dataset. The multi-modal model achieved an accuracy of 89.21% on the standard Cornell Grasp Dataset

**Introduction**  
Grasp detection is a visual recognition problem in which the robot uses its sensors to detect graspable objects in its environment. The sensors used for perceiving the robot’s environment are typically 3-D vision systems or RGB-D cameras. The key task is to predict potential grasps from sensor information and map the pixel values to real world coordinates and orientation. This is a critical step in performing a grasp as the subsequent steps are dependent on the coordinates calculated in this step. The calculated real world coordinates are then transformed to position and orientation for the robot’s end-of-arm tooling.

The project targets the problem of detecting a ‘good grasp’ from RGB-D imagery of a scene. The project uses standard Cornell Grasp Dataset as a baseline to compensate for the lack of access to a Baxter robot. The project uses two neural networks to predict a grasp configuration for an object. It uses a 19-layer deep convolutional residual neural networks to classify the object type. The output of this neural network is then fed to another 4-layer convolutional network as a parallel input to the original image and grasp baseline dataset to extract features from the RGB-D images and predict the grasp configuration.

**Data and Pre-Processing**

**Data**

Data is available at http://pr.cs.cornell.edu/grasping/rect\_ data/data.php. This dataset consists of 240 images. Each image has multiple grasp rectangles labeled as successful (positive) or failed (negative), specifically selected for parallel plate grippers.



**Pre-processing**

Baseline dataset of images as captured by the Baxter Robot had 4 fields – X, Y, Z, RGB, Index. It was not possible to recreate the image matrix as they were made with an older version of the Baxter Camera and OpenCV, both of which are not available anymore. And the baseline grasps rectangles cannot be used if the image matrix is not recreated. So, the image dataset was then appended with other orientations of the images caused by rotating the images and the grasps at various angles. The images were cropped to the bounding box. This was done to restrict the feature deep learning to the grasp aspect of the object and not the background or other aspects of the image. The grasp rectangles were shifted to the new X and Y co-ordinates to align with the cropping of the images around the bounding box. The last preprocessing involved Depth scale factoring. Grasp points are captured by the robot’s arm cameras and the distance from the camera and the object brings in a depth feature that needs to be reduced as the images are rotated. At this stage, the dataset consists of 885 images of 240 different objects. In total, there are 8019 labeled grasps with 5110 positive and 2909 negative grasps.

**Data splits – Object-wise Image splitting**

Object-wise split ensures that all objects are represented in the test and train split. This is necessary as the grasp of an object may not relate to another object type. Image-wise splitting of all the images was done randomly for these objects. So, the training data is exposed to all objects. This is helpful to test how well did the network generalize to the objects it has seen before at a different position and orientation. The test train split is 80-20. 0.1 percent of the training set is also randomly selected in each epoch run of the model in training as validation set

**Class Imbalance Check**

**Positive and Negative Grasp Classes**

Baseline dataset of positive and negative grasps had very similar number of positive and negative grasps. So, there was no class imbalance.



After the pre-processing, some of the rotated grasps fall outside the bounding box and must be disregarded. This caused a shift in the class balance but the number of positive and negative grasps were still comparable.

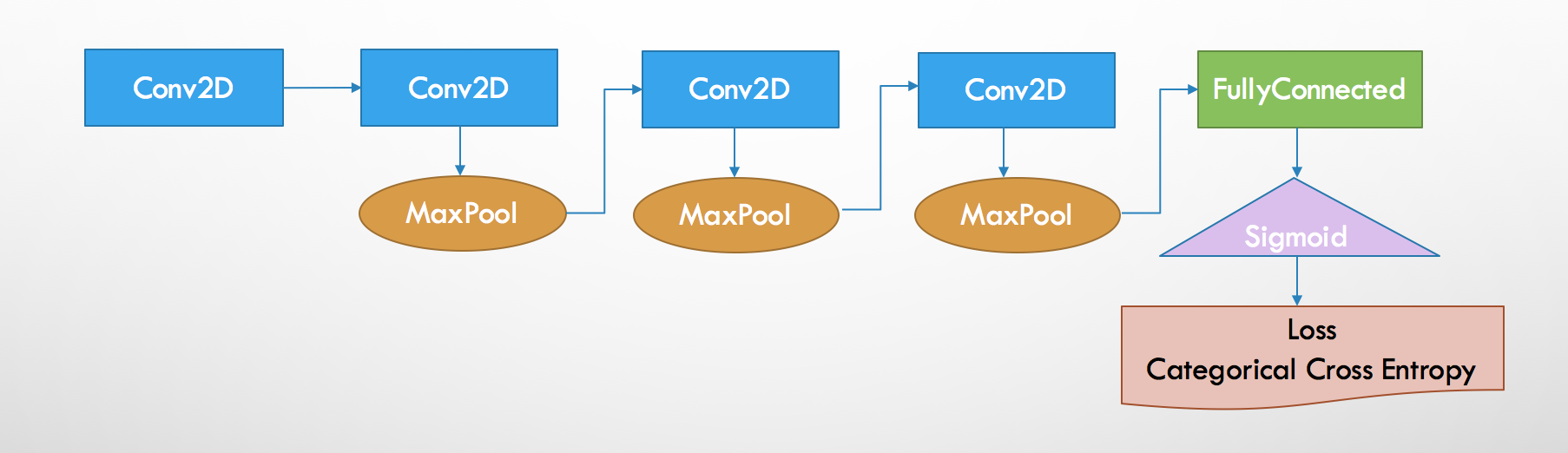
**Object types**

Images of all objects were rotated equal number of times. So, the object classes are equally represented in the form of rotations. Some object categories were repeated in the original dataset causing a desired class balance check so it is not the same number of classes but at the same time, it is not imbalanced.

**Training & Model Architecture**

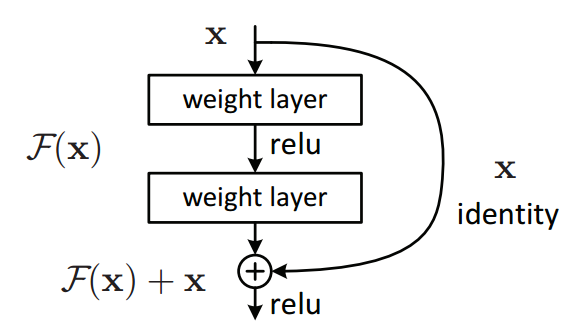
**Object Grasps**

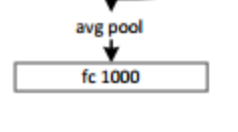
CNN Architecture with 4 layers were created. The kernel sizes and filters were adjusted at each step along with dropouts to ensure optimal training of the model. The model learns the features to predict a set of grasps, which are then scored and classified as positive and negative grasps.



**Object Types**

A residual net of 14 layers was used. A residual net deep convolution neural network is made by equal "residual" blocks. The idea of the residual network is use blocks that re-route the input, and add to the concept learned from the previous layer. The idea is that during learning the next layer will learn the concepts of the previous layer plus the input of that previous layer. This would work better than just learn a concept without a reference that was used to learn that concept. Another way to visualize their solution is remember that the back-propagation of a sum node will replicate the input gradient with no degradation.



**Hyperparameters**

**Fixed**

|  |  |
| --- | --- |
| **Hyper parameter** | **Value** |
| **global\_step** | **0** |
| **initial\_learning\_rate** | **0.01** |
| **decay\_steps** | **64** |
| **decay\_rate** | **0.9** |
| **learning\_rate** | **Exponentional decay function of above parameters** |

**Varied**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Run1** | **Run2** | **Run3** |
| **Batch Size Hyper parameter** | **500** | **250** | **50** |
| **Epochs Hyper parameter** | **40** | **100** | **1000** |
| **Validation Accuracy Metric** | **0.7** | **0.8** | **0 OR 1** |
| **Test Accuracy Metric** | **0.64** | **0.7** | **0.37** |

**Accuracy as a Metric**

While accuracy is used as a metric for this model, it is necessary to note that in case of robotics, a good metric is if the object grasp is done successfully. In the absence of a robot, we go with accuracy as a metric.



But it is interesting to see if maybe True Positives or True Negatives are a better metric. Based on the application, it iss also necessary to think if a good grasp identified as a bad grasp is a more sensitive error than the alternative

****

**Tools and Utilities**

* Keras
* TENSORFLOW
* Tensorboard
* Google cloud
* OPENCV
* Lua

**Conclusion**

In this paper, the model shows a multi-modal approach to robotic grasp detection system. It uses a model to predicts the graspability of objects for a parallel plate robotic gripper using RGB-D images, along with a model that classifies the object type. It shows that DCNNs can be used in parallel to extract features from inputs and can be used to predict the grasp configuration for an object. It has been demonstrated that the use of deep residual layers helped extract better features from the input image, which were further used by the fully connected layers to output the grasp configuration.

Future project is to implement the following architecture to the grasp detection system

