6.857 Final Project: Milestone 6

Sebastiani Aguirre Navarro and Rachel Holladay

***Abstract*— Insert summary here because piazza suggested it**

I. Introduction

The ability to grasp objects lies at the heart of robotic manipulation and therefore is fundamental to enabling robots to have complex physical interactions with their environment. Grasping a variety of unknown objects is challenging due to sensor and actuator uncertainty and uncertainty with respect to a new object's shape, mass distribution, texture properties, etc. Recently, deep neural networks have been used, with significant success, to address these challenges and enable robotic grasping.

Within the context of this paper, we will make three assumptions. First, we will be grasping objects from a flat, clutter-free surface, such as an uncrowded table top. Second, we assume we have a method of generating *grasp candidates*. Last, the robot has either an onboard camera or the environment the robot is operating in has a camera. Given an image of the scene captured by the camera, our goal is to evaluate which of these candidate grasps are likely to succeed. This creates a binary classification task, where the labels are grasp success and grasp failure.

During execution, we can imagine that our robot with sample several grasps, execute a grasp that has been predicted to be successful via our classification method.

For our data set we will use the Dexerity Network (DexNet) 2.0 data set, presented in [1]. The data set has 6.7 million grasps definitions, images and analytical grasp metrics, that we further detailed in Sec. III. The authors of the data set trained a Grasp Quality Convolutional Neural Network (GQ-CNN), which achieved 85.7% accuracy on their classification task.

Using their network, on both our sampled data and their provided data, we were unable to achieve an accuracy rate higher then the percentage of the largest class. We discuss several possible reasons. We then explore several other architectures, with varying hyper parameters and normalization methods.

We make the following contributions:

1) **RH: fill in**
2) **RH: fill in**

We first review related work (Sec. II) and further detail the data set (Sec. III). Given our data set, we formally define our problem statement (Sec. IV) and then explore various data sets using the pre-trained GQ-CNN (Sec. V). We explore other architectures (Sec. VI) and conclude with a brief discussion (Sec. VII).

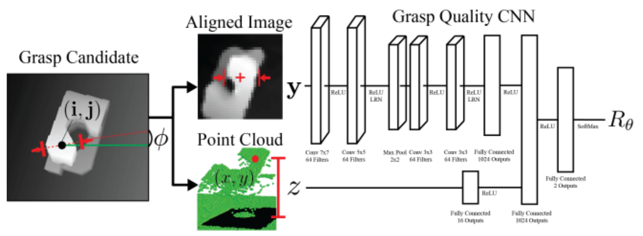


Fig. 1: This is a visualization the GQ-CNN (Grasp Quality Convolutional Neural Network) from [1]. The network takes as input a depth image of the grasp and the distance of gripper to the object and output, after several layers, a prediction of grasp success.

II. Related Work

While we primarily build off of the Dex Net 2.0 paper [1], there is a wide range of literature investigating learning how to grasp objects. The overwhelming majority of the recent work has focused on using CNNs, although there are a few papers that use SVMs, kernel-density estimation and constrained optimization-based techniques [2], [3], [4].

Ten Pas et al [5] developed a grasp detection algorithm similar to the Dex Net paper by generating grasp hypotheses and training a 4-layer CNN to perform binary classification on whether the grasp is viable. They use a different grasp representation and rely on the BigBird data set [6]. Rather then classifying a grasp, Johns et al uses a CNN to learn a grasp function, which provides a score for each grasp [7]. At execution time, then can compare the scores of several grasps and select the best grasp.

The above works focus on using a parallel jaw gripper, a two finger hand where the fingers are parallel to each other and usually move together. While this is a relatively simple hand, it is ubiquitous in industry and research and still allows for complex manipulation tasks [8]. However, people have worked to expand this grasp prediction to more complex, multi-fingered hands using various CNN architectures [9], [10], [11].

Within the grasp learning community, and in fact, within the robotics learning community, there is a pull between real data collected through a robotic platform and data generated from a physics simulator. While data collected on a robot better captures reality (since physics simulators are far from perfect), data collection is difficult and time-consuming. Pinto et al collected, at the time, a record amount of data at 50k data points of grasps collected across 700 robot hours [12]. Levine et al later collected 800,000 grasp attempts over a two month period, using between six and fourteen robot arms at once [13]. While these approaches allowed them to train a CNN without over fitting or using simulation data, such data collection is not always practical and require a huge amount of engineering effort. Bousmalis showed how to augment a smaller amount of real data with simulation to improve accuracy, thus attempting to combine the merits of both [14].

While most of this work focuses on using color (RGB) or depth (RGBD) images [15], there is growing interest in using tactile feedback, inspired by how humans feel as they grasp. Calandra et al combines vision and touch sensing to build a visuo-tactile CNN that predicts grasp outcomes from a combination of the modalities [16]. This can go one step further in using tactile feedback to learn how to readjust while grasping [17].

Dex Net 2.0 is the second of three pieces of research. Dex Net 1.0 solves the same grasping problem, but uses a multi-armed bandit model to correlate the rewards of a proposed grasp with previously seen grasps [18]. The similarity metric between grasps is learned from a Multi-View CNN. Dex Net 3.0 uses a CNN to learn suction points, leveraging recent interest in using suction for pick and place motions [19].

III. Data Set

We opted to use the Dex Net 2.0 data set due to its size, ease of use and parameterization [1]. Large published grasping data sets are rare within robotics, both because the field (data based learning for manipulation) is new and because such data sets are generally difficult to collect.

Mahler et al define a generative graphical model defined over the camera pose, object shape and pose, friction coefficient, grasp, depth image and success metric. To generate the data set they make i.i.d (independent and identically distributed) samples from their generative graphical model, resulting in 6.7 million data points.

The data set is defined over 1,500 object meshes that were used in Dex-Net 1.0 [18], collected from a variety of other data bases and standardized with respect to position. For each object, they generated 100 parallel jaw grasps via rejection sampling of antipodal pairs and evaluated a robust epsilon quality grasp metric on each grasp [20]. Additionally, each object is paired with a rendered depth image (2.5D point cloud[1]) from the sampled camera pose.

The data set of 6.7 million data points has 21.1% positive examples. This is unsurprising, since it is much more difficult to find successful grasps, as compared to failed grasps.

[1]The images are 2D matrices that are referred to as 2.5D in robotics literature because they display depth information.

From the data set we randomly sample, with replacement, k data points. In some cases we sample such that we guarantee some ratio of positive versus negative examples.

IV. Problem Statement

We now formally define our learning problem. We take as input a 32x32x1 depth image and the distance between the gripper and the camera (referred to as the z value).

The depth image, called the "aligned image", is transformed to be centered and axis aligned according the grasp point. Hence this image captures the scene and grasp in one representation. An example depth image in shown in Fig. 2a. We are solving a binary classification problem and hence the output of our network will be 0 or 1 labels. A positive label refers to a grasp predicted to be successful and a negative label is a predicted grasp failure. Our label in the data set is given by the robust epsilon quality grasp metric (defined in [20]), which is thresholded by the value 0.002 to create binary labels.

We split our data into into 80%, 10% and 10% for the training, validation and testing sets respectively. We measure success by the percentage of correct labels for each set.

For training we use Keras, an open source neural network library [21], that is powered by TensorFlow [22].

V. Balancing Data Sets

As mentioned previously, Dex-Net 2.0 contains approximately 20% positive examples. Naïvely, you could predict a negative label for all instances and be correct 80% of the time. Interesting, the paper reported an 85.7% accuracy, which does slightly better than an all negative classifier.

To investigate this, we begin by using the **RH: pretrained dex net network** on the data sets we created by sampling the entire Dex Net data base. We can constrain our sampling to produce a data set of 10,000 samples that is 50% positive examples and 50% negative examples (referred to as a "balanced" data set). Using their network we achieve approximately 50% accuracy because, as shown in our confusion matrix in Fig. 2b, the network learns to always output a positive label.

If we do not make such a constraint and sample randomly, we expect to have the original distribution: 20% positive examples and 80% negative examples (referred to as an "unbalanced" data set). As shown by the confusion matrix in Fig. 2c, the network achieved approximately 80% accuracy by always guessing negative.

To confirm our thoughts a step further, we use the data they provide on the network they provide (**RH: confirm**). This data set is also unbalanced and achieves similarly to above in that it always guesses negative, as seen in Fig. 2d.

This result was initially surprising. Due to the few percentage point disparity between our accuracy and theirs, we hesitate to conclusively claim their network did not learn features. While we varied the size of the data set we use, we did not come close to their magnitude (in the millions) and therefore it is possible that the network simply needs a tremendous amount of data to learn effectively.

To explore further, we experiment with varying the architecture of the CNN. This both allows us to explore the effect of the architecture (as well as its hyper-parameters) and to see if we can improve upon our accuracy rate.

VI. Testing Architectures

We experimented with three types of network architectures, which we call "Inception Net", "Res Net", "Andreas Net". For each architecture we will describe their structure, the normalization techniques we applied and the results.

RH: how many data points did we use? RH: What activation functions did we use

RH: For normalization we mainly used L_1 or L_2 regularization for the fully connected layers and dropout and batch normalization on the convolutional layers.

RH: Give summary of how these did

A. Inception Net

The first network architecture we will explore is the Inception Network [23], visualized in Fig. 3. **RH: Any intro about this network (ie from the original paper** This network style was originally designed to increase the depth and width of a network while keeping computational load the same while operating on ImageNet [24].

The inception network begins with 1 convolution layer in the beginning with 10 filters of size 3x3. We apply batch normalization to the output of this layer and pass that to three parallel convolutional layers of sizes 1x1, 3x3, and 5x5, each with 16 filters. These outputs are concatenated on the depth dimension and passed through a max pooling layer of 3x3. We apply batch normalization and a dropout of **RH: k%**. The outputs are flattened with global average pooling and then the z value is concatenated before passed to a classifier of one hidden layer of 20 units **RH: two FC?**.

RH: INPUT RESULTS

B. Res Net

Our residual network (called "Res Net" in this discussion) is visualized in Fig. 4. It consists of one convolutional layer, with 8 filters of size 7x7. At this point, the output of this layer branches, such that this same output is passed through two more convolution layers of 32 filters 3x3 and 16 filters 1x1 used as dimension reduction. The output of these two layers is added to their input and then passed to another convolution layer of 8 filters of 1x1 for further dimension reduction and then flattened with global average pooling. After each convolutional layer, we apply batch normalization.

Like for the other network, the z value is concatenated to this output before passing it to a classifier with a fully connected layer of 10 hidden units.

RH: insert results

C. Andreas Net

The last architecture we consider is from [25] and will be referred to as "Andreas Net". It is visualized in Fig. 5. The network, like GQ-CNN, was designed for robotics grasping and uses depth images as input. However, [25] creates a closed-loop controller that guides the gripper to the object to be grasped. Thus their CNN learns the distance to the nearest grasp function used by the controller. Despite its original use as a regression network, we adapt it to our classification task.

The depth image is passed through one convolutional layer, with 8 filters of size 7x7, and then a max-pooling layer. The z-value is passed through a full connected layer and then is tiled. **RH: How does this work for one integer? RH: explain tiling** The outputs of each of these are summed before being passed through a convolutional layer with 8 filter of size 7x7 and another max pooling layer. Finally, we process through two fully connected layers of size **RH: something**.

RH: results

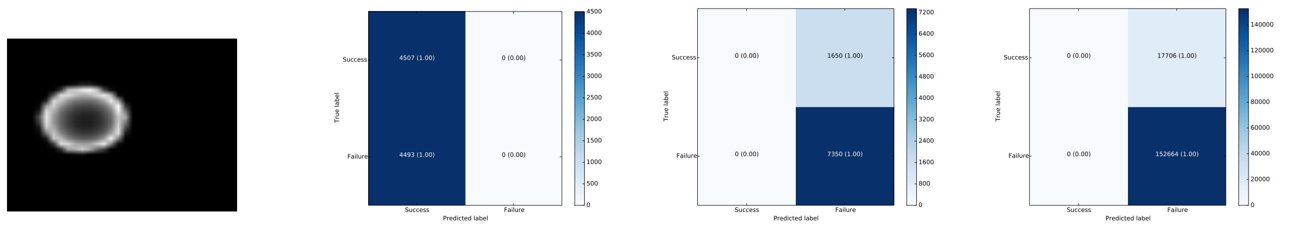
VII. Discussion

The goal of this project was to use modified depth images to perform binary classification whether a grasp would succeed, as measured by a grasp stability metric. We leveraged the recently published Dex Net 2.0 data base by sampling data points [1].

We began by learning using the Dex Net framework, their GQ-CNN, with various input sampling techniques. Given the unsatisfactory performance we next experimented with three architectures: Inception Net, Res Net and Andreas Net.

Overall, we did not achieve the accuracy we were hoping for. Taking a step back, we hypothesize three possible reasons. The first is that, although we tried various networks, we did not find the best possible network or combination of hyperparameters. To investigate this, we would continue experimenting with architectures.

Our second guess is that we did not train our network with enough data. For computational reasons, we sampled sets from the large Dex Net database and thus trained on a much smaller set. The key to this kind of learning is large quantities of data, so is possible our issues would be alleviated by training on larger sets.



(a) Example Depth Image (b) Balanced Data (c) Unbalanced Data (d) Replicated Paper Results

Fig. 2: FILL IN CAPTION

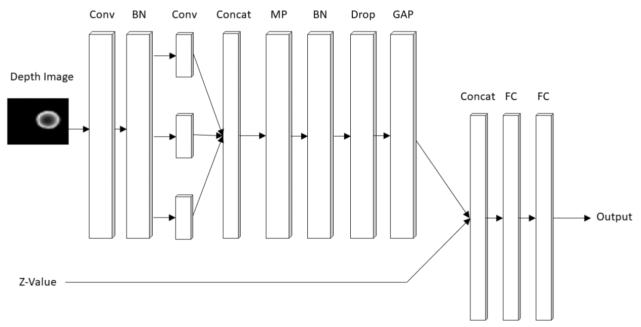


Fig. 3: Inception Net Architecture

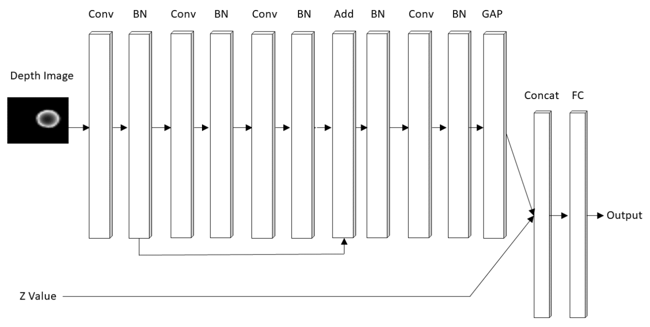


Fig. 4: Res-Net Architecture

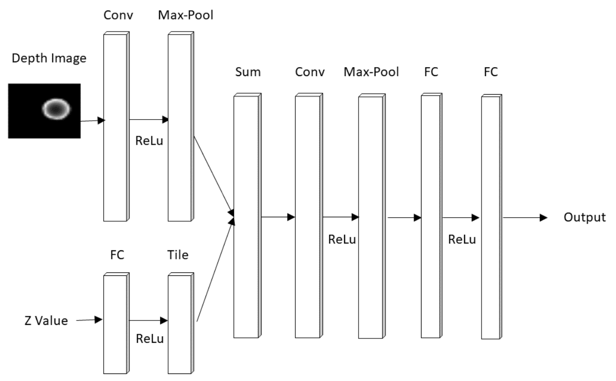


Fig. 5: Andreas Net Architecture

Our third possible reason relates to the nature of the data set. Considering we did not create the data set, editing it in a significant way was outside of our control. It is possible that the depth image and distance is not sufficiently powerful enough representation to learn grasp stability. Several other grasp learning algorithms leverage a color input, since color can often characterize objects [26]. While we attempted to access a data set with color images, the repository was not usable in its published form (it referenced data that was not publicly accessible) and the authors of the repository did not reply to our several inquires.

Our learning objective is a hotly published research area and much of the work we reference is extremely recent (i.e. Dex Net 2.0 was presented in July of this year and Andreas Net was published to Arvix less than a month ago). **RH: some conclusion sentence!**

ACKNOWLEDGMENT

We would like to thank the entire course staff for making this course possible and our project TA for his helpful guidance. We would also like to thank Jeff Mahler and Sherdil Niyaz (a few of the authors of the DexNet 2.0 paper) for answer our questions throughout this project.

DIVISION OF LABOR AND CODE

RH: Insert description of work distribution. Our code base is available at: `https://github.com/finalProject6867`

REFERENCES

[1] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.
[2] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *ICRA*, pp. 3304–3311, IEEE, 2011.
[3] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *IJRR*, vol. 35, no. 8, pp. 959–976, 2016.
[4] IEEE, *Bridging the gap: One shot grasp synthesis approach*, 2012.
[5] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, p. 0278364917735594, 2017.
[6] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *ICRA*, pp. 509–516, IEEE, 2014.
[7] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *IROS*, pp. 4461–4468, IEEE, 2016.
[8] M. Mason, S. Srinivasa, and A. Vazquez, "Generality and simple hands," *Robotics Research*, pp. 345–361, 2011.

[9] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *ISRR*, 2017.

[10] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in *IROS*, pp. 4415–4420, IEEE, 2015.

[11] Y. Zhou and K. Hauser, "6dof grasp planning by optimizing a deep learning scoring function," in *R:SS Workshop*, 2017.

[12] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *ICRA*, pp. 3406–3413, IEEE, 2016.

[13] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *IJRR*, p. 0278364917710318, 2016.

[14] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," *arXiv preprint arXiv:1709.07857*, 2017.

[15] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *IJRR*, vol. 34, no. 4-5, pp. 705–724, 2015.

[16] R. Calandra, A. Owens, M. Upadhyaya, W. Yuan, J. Lin, E. H. Adelson, and S. Levine, "The feeling of success: Does touch sensing help predict grasp outcomes?," *arXiv preprint arXiv:1710.05512*, 2017.

[17] Y. Chebotar, K. Hausman, Z. Su, G. S. Sukhatme, and S. Schaal, "Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning," in *IROS*, pp. 1960–1966, IEEE, 2016.

[18] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. 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 *ICRA*, pp. 1957–1964, IEEE, 2016.

[19] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning," *arXiv preprint arXiv:1709.06670*, 2017.

[20] D. Seita, F. T. Pokorny, J. Mahler, D. Kragic, M. Franklin, J. Canny, and K. Goldberg, "Large-scale supervised learning of the grasp robustness of surface patch pairs," in *SIMPAR*, pp. 216–223, IEEE, 2016.

[21] F. Chollet, "Keras (2015)," *URL http://keras. io*, 2017.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, pp. 1–9, IEEE, 2015.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.

[25] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images," *arXiv preprint arXiv:1706.04652*, 2017.

[26] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. Chavan-Dafle, R. Holladay, I. Morona, P. Qu Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *arXiv preprint arXiv:1710.01330*, 2017.