6.857 Final Project: Milestone 6

Sebastiani Aguirre Navarro and Rachel Holladay

Abstract— **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 utilize the Dex Net 2.0 data base of grasp, images and stability labels [1]. We begin by training variants of the data set with the Dex Net learning architecture (GQ-CNN) before exploring a variety of different network structures. Specifically, we apply three different network architectures, referred to as Inception Net, Res Net and Andreas Net. Despite our tuning, none of these networks were able to outperform GQ-CNN with respect to accuracy performance. We conclude with several discussion points hypothesizing why all of our networks struggled to learn the objective function.**

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 with respect to our set-up. 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 on-board 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. Note that since attempting a grasp in a real world setting consumes time and can disrupt the environment, we would prefer to be cautious.

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), seeen in Fig. 1, which achieved 85.7% accuracy on their binary classification task.

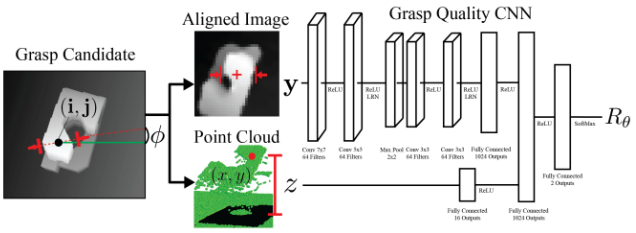


Fig. 1: This is a visualization the GQ-CNN (Grasp Quality Convolutional Neural Network) from [1]. The network takes as input a depth image of the grasp and the distance of gripper to the object and output, after several layers, a prediction of grasp success.

While this is an impressive accuracy rate, we find that it is tied to the fact that nearly 80% of the data is in the same class. When we re-balance the data distribution to have 50% positive examples and negative examples, their same network achieves only 60% accuracy.

One fundamental assumption when learning is that the training sets and test sets are drawn from the same distribution. Maintaining this assumption, we found that this network (and others that we tried), had varying accuracy depending on the distribution. In this case, our results affirm the importance of considering the data distribution when evaluating learning algorithms.

As mentioned, we explored several other architectures, with varying hyper parameters and normalization methods. However, we were unable to beat the accuracy rate achieved with the GQ-CNN.

We make the following contributions:

1) We explore the effect of the data set distribution on the accuracy of the model. Specifically, we modify the distribution of the training, validation and testing set and re-evaluate GQ-CNN.
2) We adapt, train and evaluate three existing networks to our learning problem.
3) Using confusion matrices, we analyze the false positive and false negative ratio of our algorithms. From an algorithmic perspective, this helps us better examine the effect of varying data distributions. From a robotic perspective, this informs how efficient the robot might be when utilizing the results of the network.

We note that our comments with respect to data set distribution can only apply to our specific data set and the models we used. To make a more general claim we would need to more exhaustively test across different architectures and different data sets. Regardless, we

found the effect of the data distribution to be an interesting artifact.

With respect to the structure of the rest of the paper, 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 GQ-CNN (Sec. V). We explore other architectures (Sec. VI) and conclude with a brief discussion (Sec. VII). In particular, we discuss hypotheses on why were we unable to improve our accuracy rate and what makes this problem difficult.

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.

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 to explore the effect of varying distributions.

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).

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

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 their pre-trained GQ-CNN 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 their pre-trained network. This data set is also unbalanced and achieves similarly to above in that it always guesses negative, as seen in Fig. 2d.

Considering the difference between their reported results, we became suspicious of our pre-trained network [2]. We therefore re-implemented their network in Keras, trained it using our balanced data set.

[2]While we cannot eliminate the possibility that we somehow loaded their network incorrectly or made a similar software implementation bug, we believe significant reasoning and debugging that we used their network correctly.

Borrowing from the original paper, the network is visualized in Fig. 1. We briefly summarize the network structure here. The depth image is passed through two convolutional layers following by ReLu activation, where the convolution layers both have 64 filters and have sizes 7x7 and 5x5 respectively. This is followed by a max-pooling layer of size 2x2 and stride 1x1. This output is then passed through two more convolutional layers, where the first is following by ReLu activation, where the convolution layers both have 64 filters of size 3x3. We flatten this output is connect it to 1024 hidden units that are activated by a ReLu. The z component is connected to 20 hidden units that are activated by a ReLu and concatenated onto the result above. This is connected to 1024 hidden units activated by ReLU, followed by 2 hidden units. The softmax function is applied to give the output.

The results from reimplemented and training GQ-CNN are shown in Fig. 3. After a few time steps the training and validation accuracy achieve 60%, which is also what our test set reported. The confusion matrix in Fig. **??**, shows our improved accuracy and that we tend to over-predict positive labels. This false positive tendency is actually more problematic for robots then false negatives, since the robot will spend time executing a grasp that is likely to fail.

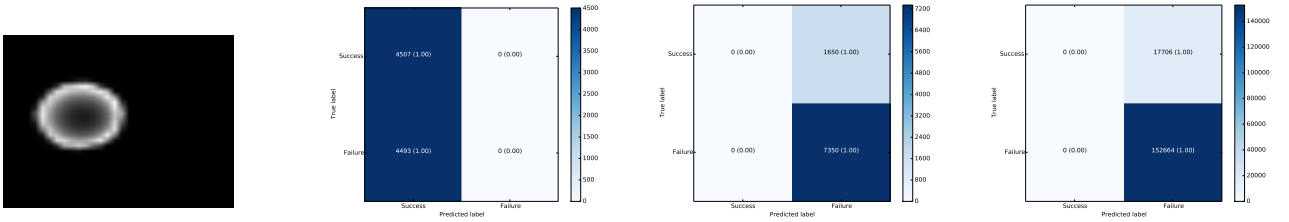
Given that the original paper reported 85% accuracy on a data set that had approximately 80% negative examples, our 60% accuracy on a data set that had 50% examples is comparable. However, taking a step back, a 60% accuracy on a binary classification task leaves much to be desired.

Hoping to improve our accuracy, 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

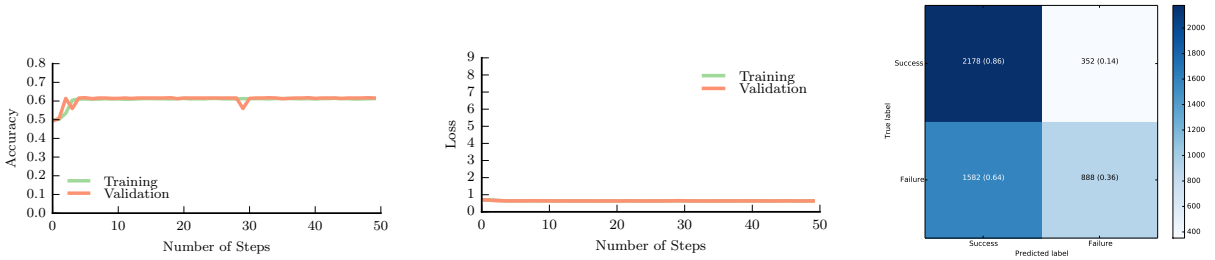
We experimented with three types of network architectures, which we call "Inception Net", "Res Net", "Andreas Net". Each of these networks are inspired by previous work, but were adapted by us to fit our task. For each architecture we will describe their structure, the normalization techniques we applied and the results.

For normalization techniques we applied L_1 regularization on the fully connected layers, batch normalization on the convolutional layer, and dropout. For Inception Net and Res Net, our activation function is the Rectified Linear Unit (ReLu), $f(x) = \max(0, x)$. For Andreas Net, our activation function is the Leaky Rectified Linear Unit, $f(x) = \max(a, ax)$ where a is referred to as the slope. We obtain all of our outputs using the soft-max function over 2 units. We predict the class with higher probability, giving us a binary output.



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

Fig. 2: The far left panel shows a depth image that serves as input to our network. The three panels on the right show the confusion matrices for various data set learned using the pre-trained GQ-CNN network. In each case the network simply outputs, for every new instance, the label of of the largest classes.



(a) Accuracy (b) Loss (c) Confusion Matrix

Fig. 3: After re-creating and re-training the GQ-CNN network with our balanced data set, we achieve the accuracy and loss shown above on our training and validation sets. We also show the confusion matrix, as compared to the earlier results.

For each network, we experimented with various normalization constants, network hyper-parameters (i.e. pool size), slopes, etc. Within this paper we report only on the best combination of parameters.

For each network we plot the accuracy and loss across epochs for the training and validation set and report the final accuracy on the test set. We also, for most networks, show the confusion matrix. For all of the following results we used 50,000 data points (split into 80%/10%/10% for training, validation and testing) from our balanced data set.

Overall, none of these networks were able to achieve an accuracy rate higher then our re-implemented GQ-CNN. We can rank the networks from worst to best by accuracy: Res Net, Inception Net, Andreas Net.

A. Inception Net

The first network architecture we will explore is the Inception Network [23], visualized in Fig. 4. 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 at ReLU as the activation function. We then 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 size 3x3 and stride 1x1. We apply batch normalization, ReLu activation and a dropout of 70%. The outputs are flattened with global average

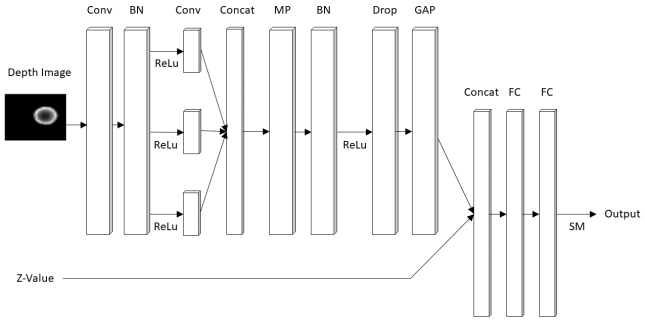
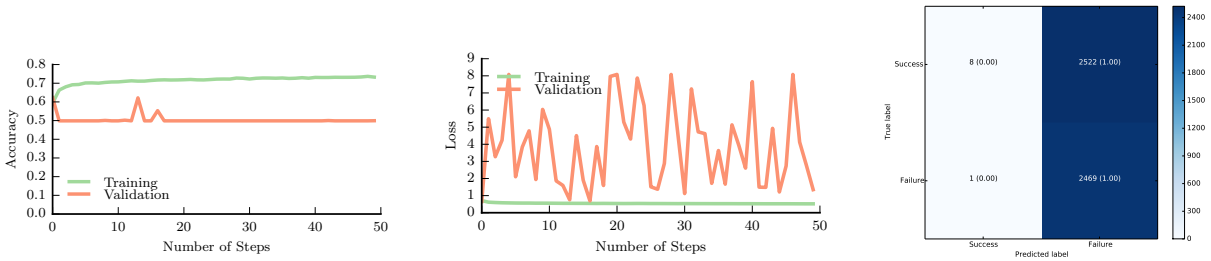


Fig. 4: Inception Net Architecture. We provide the following labels: Convolution (Conv), Batch Normalization (BN), ReLU (Rectifer Linear Unit), Concatenate (Concat), Dropout (Drop), Global Average Pooling (GAP), Fully Connected (FC), SoftMax (SM).

pooling and then the z value is concatenated on. We then pass through two fully connected units of size 25 and 2 respectively. We apply the softmax function to these final two units to obtain our output for binary classification.

Our accuracy and loss are shown in Fig. 5. While our training accuracy quickly reaches 70%, our validation accuracy only rarely spikes above nominal 50%. This poor learning is reflected in the plot of our loss function (Fig. 5b) for our validation set, which never stabilizes. Our final accuracy on our training set was 50%, which, is unimpressive given a binary classification task on a balanced data set. Looking at our confusion matrix in Fig. 5c, our network almost always predicted failure. While from a robotic execution perspective this means we are unlikely to execute a failure-likely grasp, the



(a) Accuracy of Inception Net (b) Loss of Inception Net (c) Confusion Matrix

Fig. 5: We plot the accuracy rate and loss of the Inception Net over training epochs for the training and validation set. This network seems to overfit based off of the difference between the accuracy of the training and validation sets. Our confusion matrix in the rightmost column shows we over predicted failure.

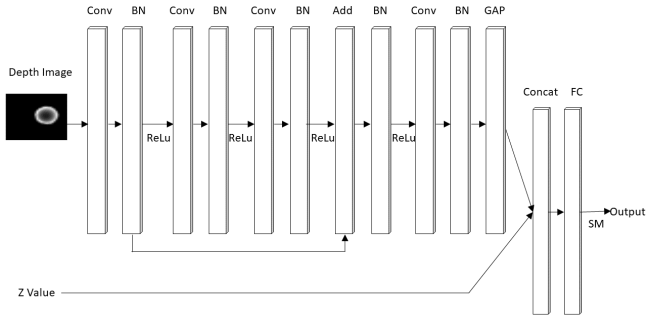


Fig. 6: Res-Net Architecture. We provide the following labels: Convolution (Conv), Batch Normalization (BN), ReLU (Rectifer Linear Unit), Add (Add), Global Average Pooling (GAP), Fully Connected (FC), SoftMax (SM).

over-prediction of failure also means we are unlikely to execute any grasp.

For tuning this network, we tried using a larger number of filters at each step, testing with 64, 32 and 20 filters. However, the initial loss of the network was very high, and it would not decrease substantially or at all during training. Likewise, max pooling of 2x2 was tested, however this did not bring any benefits to the training either. An extra inception "layer" was added after the dropout, however this also resulted in unusually high losses. It is worth noting that because dimensions of our data is small, it was not feasible to add many layers. We selected a high dropout rate in order to let the network "learn" which of the mappings learned better features and thus was not modified during the experiments. As for the z input, a fully connected layer of 16 units was tried, but it did not have much effect on our validation accuracy.

B. Res Net

Our residual network (called "Res Net" in this discussion) is visualized in Fig. 6. It consists of one convolutional layer, with 16 filters of size 3x3. We then apply batch normalization and ReLu as an activation function. At this point, the output of this layer branches, such that this same output is passed through two more convolution layers of 8 filters 7x7 and 32 filters 3x3used as dimension reduction. After each of those two convolution layers is a layer of batch normalization and activation by ReLu. The output of these two layers is added to their input and then passed through batch normalization and activation by ReLu. This is then passed to another convolution layer of 8 filters of 1x1 for further dimension reduction. We then apply batch normalization and ReLu activation once more before flattened with global average pooling.

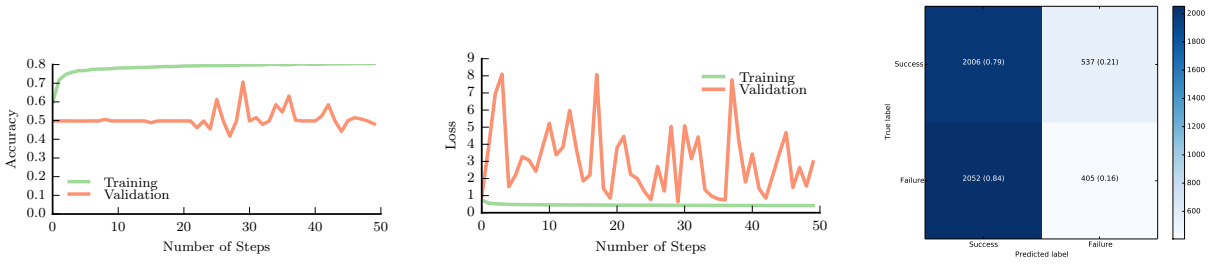
Like for the other network, the z value is concatenated to this output before passing it to a classifier with a fully connected layer of 2 hidden units. We obtain our output via the softmax function.

Our accuracy and loss are shown in Fig. 7. Our training accuracy nears 80% and our validation accuracy varies between 50% and 70%, depending on when we stop our network. This stopping criteria is reflected in our loss graph, which also fluctuates for our validation set. Doing slightly worse then Inception Net, our final test accuracy is 48%. Looking at our confusion matrix, our learning algorithm over predicted success, in stark contrast to our Inception Net. In practice, given the large quantity of false positives, our robot would execute many grasps that are likely to fail, wasting time and resources.

For tuning this network, we also tried increasing the number of filters to 64 and 30, and the kernel sizes where permuted (switched around layers). However, the model did not do any better or worse than the set of hyper-parameters described above. While we experimented with adding a convolution layer to the identity path in the Res Net of 32 filters and kernel size of 5x5, this was not successful since the network never escaped from the 50% accuracy. We also tried using L_1 regularization on the fully connected layers in the reported architecture. However this increased significantly the initial loss and it did not help with the final validation accuracy.

C. Andreas Net

The last architecture we consider is from [25] and will be referred to as "Andreas Net". It is visualized in Fig. 8. The network, like GQ-CNN, was designed for robotics grasping and uses depth images as input. However,



(a) Accuracy (b) Loss (c) Confusion Matrix

Fig. 7: We plot the accuracy rate and loss of the Res Net over training epochs for the training and validation set. There is still overfitting in this network, there is odd oscillation of the validation accuracy. Our confusion matrix in the rightmost column shows we over predicted success.

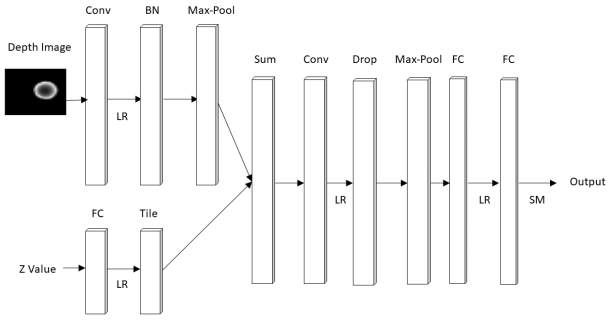


Fig. 8: Andreas Net Architecture. We provide the following labels: Convolution (Conv), Leaky Rectified Linear Unit (LR), Max-Pooling (Max-Pool), Fully Connected (FC), Tiling (Tile), Sum (Sum), Fully Connected (FC), SoftMax (SM).

[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 20 filters of size 5x5. This output is activated by a Leaky ReLu with slope 0.3 before a batch normalization layer. We pass this to a max-pooling layer of size 2x2 and stride 1x1. The z-value is passed through a full connected layer of 20 units with L_1 regularization of $\lambda = 0.1$. We pass this through Leaky ReLu, again with slope 0.3, and then tile the output [26]. The tile operation consists of taking the 1-D vector of 20 elements and, for each scalar, a 14x14 "feature map" is formed by repeating this scalar. The output is 20 14x14 "feature maps" are added channel-wise to each of the outputs of one of the convolutional layers. The outputs of each of these, the processed depth image and processed z-value, are summed.

This result is passed through convolutional layer with 50 filter of size 5x5, followed by Leaky ReLu, with slope 0.3. We then apply dropout at rate 0.6 and then a max-pooling layer with size 2x2 and stride 1x1. We flatten this output and pass it through a fully connected layer of 20 units and L_1 regularization with $\lambda = 0.1$. This is activated by Leaky ReLU (again slope 0.3) and finally connected to a fully connected layer with 2 hidden units (again with L_1 regularization at $\lambda = 0.1$). We apply softmax to these two units to generate our output.

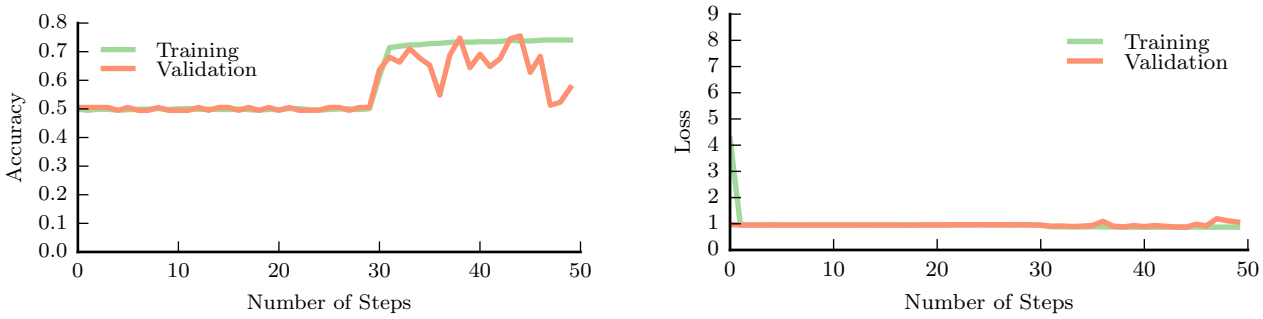
Our accuracy and loss are shown in Fig. 9. After 30 training steps, our training and validation accuracy improve with our training accuracy near 70% and our validation wavering between 50% and 70%, depending on when we stop training. For both networks our loss over time shows little variance. Our final test accuracy is 56%, an improvement over Inception Net and Res Net, but still slightly worse then our re-trained version of the GQ-CNN.

In developing the network, we first tried the original "Andreas Net" (i.e described in [25]), without dropout. This did not perform better than random classification. However, the introduction of the dropout increased the the final test accuracy to 56%. We also replaced the Leaky ReLU functions with normal ReLU functions, but this decreased the accuracy back to 50%. An additional layer of 50 filters and kernel size of 5x5 was added before the flattening operation, therefore essentially reducing the feature maps to 1x1. This did not lead to good generalization.

As an alternative to the flatten operation, a global average pooling operation was tested instead. While this operation has been successfully used in other object detection networks, it did not seem to add much to our network and thus we continued to use the flatten operation. For the L_1 regularization factor, in addition to 0.1 we also tried 0.01 and 0.3, to see whether decreasing or increasing the regularization would affect the final accuracy. However, even when changing slightly this factor, we saw high initial loss even after 50 epochs and the network performance was quite poor.

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].



(a) Accuracy (b) Loss

Fig. 9: We plot the accuracy rate and loss of the Andreas Net over training epochs for the training and validation set. Of our new networks this performs the best, although its validation accuracy also oscillates.

We began by learning using the Dex Net framework, their GQ-CNN, with various input sampling techniques. Using their pre-trained 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 then re-train their network to achieve similar performance to that reported in their original paper, but found this rate to be unsatisfactory on a binary classification task

Aiming for better performance we next experimented with three architectures: Inception Net, Res Net and Andreas Net. We were unable achieve better performance with any of these networks. Furthermore, despite our regularization attempts, each of these network struggled to generalize as well as the GQ-CNN.

Overall, even the best performing network (GQ-CNN) 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. While we were able to achieve comparable performance (on a re-scaled data set), the GQ-CNN is reported to have 4.5 million trainable parameters, which means that we had many more parameters then data points.

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 [27]. 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). Therefore, we realize that there is significant amount of progress to be made in learning robotic grasps.

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

Both partners contributed evenly throughout the duration of the project. We collaborated in design our research questions, areas of exploration and design. Rachel took the lead on processing the data set, figure generation and writing the final results. Sebastiani took the lead on designing the network hyper-parameters and training the networks. 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] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng, "Tiled convolutional neural networks," in *Advances in neural information processing systems*, pp. 1279–1287, 2010.

[27] 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.