Planning Multi-Fingered Grasps as Probabilistic Inference in a Learned Deep Network

Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans

Abstract We propose a novel approach to multi-fingered grasp planning leveraging learned deep neural network models. We train a convolutional neural network to predict grasp success as a function of both visual information of an object and grasp configuration. We can then formulate grasp planning as inferring the grasp configuration which maximizes the probability of grasp success. We efficiently perform this inference using a gradient-ascent optimization inside the neural network using the backpropagation algorithm. Our work is the first to directly plan high quality multi-fingered grasps in configuration space using a deep neural network without the need of an external planner. We validate our inference method performing both multi-finger and two-finger grasps on real robots. Our experimental results show that our planning method outperforms existing planning methods for neural networks; while offering several other benefits including being data-efficient in learning and fast enough to be deployed in real robotic applications.

Keywords: Grasping; grasp planning; grasp learning; multi-fingered grasping

1 Introduction and Motivation

Learning-based approaches to grasping [20, 21, 13, 17, 11] have become a popular alternative to geometric [19, 1, 3, 4] and model-based planning [6, 16] over the past decade. In particular grasp learning has shown to generalize well to previously unseen objects where only partial-view visual information is available. More recently, researchers have looked to capitalize on the success of deep neural networks to improve grasp learning. Broadly speaking deep neural network methods for grasp learning can be split into two approaches: predicting grasp success for an image patch associated with a gripper configuration [13, 8, 17, 14, 15, 9, 23] and directly predicting a grasp configuration from an image or image patch using regression [18, 12, 24]. While these deep learning approaches have shown impressive performance for parallel jaw grippers (e.g. [17]) relatively little work has focused

Utah Robotics Center, School of Computing, University of Utah, Salt Lake City, UT, USA
e-mail: qklu,bala,thermans@cs.utah.edu;kautilya.chenna@utah.edu

on the more difficult problem of multi-fingered grasping [23, 24, 10]. We believe two primary difficulties restrict the use of deep learning for multi-fingered grasping (1) the input representation used for grasp configurations in neural networks and (2) the reliance on external planners for generating candidate grasps.

In order to combat these two problems, we propose an alternative approach to grasp planning with deep neural networks, where we directly use the learned network for planning. In our work, we train a network to predict grasp success; however, we use the trained network in a substantially different and novel way from currently employed sampling methods for grasp planning. The grasp representation we use fundamentally enables our approaches' success. In addition of giving the neural network the image patch, z, as input, we also provide the grasp configuration parameters, θ (e.g. joint (preshape) angles, wrist pose, etc.).

Once trained, given a new object patch, z, we perform inference over the grasp parameters, θ, in order to maximize the probability of grasp success $P(Y = 1|z, \theta)$ learned by our convolutional neural network (CNN). We perform this probabilistic inference as a direct optimization over θ using constrained gradient-ascent, which leverages the efficient computation of gradients in neural networks, while ensuring joint angles remain within their limits. Thus, our approach can quickly plan reliable multi-fingered grasps given an image of an object and an initial grasp configuration.

Our planner offers a number of benefits over previous deep-learning approaches to multi-fingered grasping. Kappler and colleagues [10] learn to predict if a given palm pose will be successful for multi-fingered grasps using a fixed preshape and perform planning by evaluating a number of sampled grasp poses. Varley et al. [23] present a deep learning approach to effectively predict a grasp quality metric for multi-fingered grasps, but rely on an external grasp planner to provide candidate grasps. In contrast, our method learns to predict grasp success as a function of both the palm location and preshape configuration and plans grasps directly using the learned network. Saxena et al. [21] also perform grasp planning as inference using learned probabilistic models; however they use separate classifiers for both the image and range data, using hand selected models instead of a unified deep model. Zhou and Hauser [25] concurrently propose a similar optimization-based grasp planning approach to ours using a similar CNN architecture. In contrast to our work, they do not interpret planning as probabilistic inference; they optimize only for hand pose, ignoring hand joint configurations; and they validate only in simulation.

Veres et al. [24] train a conditional variational auto-encoder (CVAE) deep network to predict the contact locations and normals for a multi-fingered grasp given an RGBD image of an object. In order to perform grasping an external inverse kinematics solver must be used for the hand to try and reach the desired contact poses as best as possible. Implicit in such a regression method as proposed in [24] lies the assumption that there exists a unique best grasp for a given object view. In contrast, our method can plan multiple high quality grasps for a given object using different initial configurations. This offers the robot the option of selecting a grasp best suited for its current task. Additionally, we show that our classification-based network can effectively learn with a smaller dataset compared with a regression network, which can not leverage negative grasp examples.

We propose two novel CNN architectures to encode grasp configurations for our planner, but many alternatives would surely also work. Our proposed multi-channel network (c.f. 3.1) has a similar architecture to that of [14], but we train on joint configurations instead of motor commands and use only a single image as input, as we evaluate grasps and do not learn a controller to perform grasping. Our alternative patch-based network described in Section 3.2 was inspired by the fingertip patches used in [23]; however, we construct our patches as a function of the grasp parameters differently in order to improve learning and efficiently perform gradient ascent.

The contributions of this paper are as follows:

- We present a method for performing grasp planning as probabilistic inference in a deep neural network, that
  - is the first work to directly optimize over grasp configurations inside a learned deep network;
  - is data-efficient compared with direct regression CNNs;
  - can naturally predict a variable number of high quality grasps;
  - can improve initial grasp configurations which would fail to result in successful grasps;
  - requires far fewer grasps as initializations than sampling-based approaches.
- We propose two novel CNN architectures for use with our planning algorithm.
- We provide a multi-finger grasp dataset from simulation with more realistic data than currently existing simulation datasets.
- We experimentally validate the effectiveness and efficiency of our inference method for both multi-finger and two-finger grasp planning on real robots.

In the next section we provide a formal description of our grasp planning approach. We follow this in Section 3 with an overview of our approach to multi-finger grasp learning and the novel CNN architectures for predicting grasp success. We then give a thorough account of our experiments and results in Section 4. We conclude with a brief discussion in Section 5.

2 Grasp Planning as Probabilistic Inference

Following [3] we define the grasp planning problem as finding a grasp pre-shape configuration (hand joint angles and palm pose). The robot then moves to this pose and executes a controller to close the hand forming the grasp on the object. We focus on scenarios where a single object of interest in isolation exists in the scene. Importantly, we assume no explicit knowledge of the object beyond this sensor reading. The problem we address states, given such a grasp scenario, plan a grasp preshape configuration that allows the robot to successfully grasp and lift the object without dropping it.

We propose planning grasps by finding the grasp configuration which maximizes the grasp success probability. We use a deep neural network to predict the probability of grasp success, Y, as a function of the sensor readings, z, and hand configuration, θ. We formalize probabilistic grasp inference as an optimization problem:

$$\begin{aligned} \underset{\theta}{\text{argmax}} \quad & p(Y=1|\theta,z,w) = f(\theta,z,w) \\ \text{subject to} \quad & \theta_{min} \leq \theta \leq \theta_{max}. \end{aligned} \tag{1}$$

In Eq. 1 $f(\theta, z, w)$ defines a neural network classifier with logistic output trained to predict the grasp success probability as a Bernoulli distribution over Y. The parameters w define the neural network parameters. We encode joint limits and the reachable workspace of the robot hand as constraints on the decision variables. We can now directly optimize over θ in order to infer the maximum likelihood grasp estimate, $\theta^\star$.

In order to efficiently solve this optimization we use a gradient-ascent approach, which leverages the structure of the neural network to efficiently compute gradients. The gradients are computed using the well-known backpropogation algorithm with respect to the grasp input, θ, instead of the more typical optimization of network parameters, w, during learning. The complete algorithm takes as input the current object image, z, and an initial grasp configuration, θ^0. We compute the success probability by evaluating a forward pass of the neural network, then update the grasp parameters, θ, using backpropogation in backwards pass through the network. We iterate these forward and backward passes until convergence. We handle the linear constraints using gradient projection and apply a backtracking line search to determine the gradient step length at each iteration. Initial hand configurations could be generated in a number of different ways, we describe the heuristic approach we use in Section 4.1.

Our formulation allows for a number of straightforward extensions, which we do not consider in this work. First, adding a prior over θ would allow one to infer the Maximum a posteriori (MAP) grasp estimate, which could also be learned from data. This could encode, for example, generally effective preshapes independent of the object. Additionally other constraints could be added to the optimization such as collision avoidance constraints or the full forward kinematics of the robot arm. Finally, other sensing modalities could be given as input to the neural network if available such as tactile or haptic information. In the next section we define two neural network architectures we use in evaluating our approach. However, many other networks would likely work well within our framework, as long as they use the same inputs and outputs required by our planner.

3 Deep Networks for Multi-fingered Grasp Learning

We present two novel neural network architectures designed to predict the probability of grasp success for multi-fingered hands. Importantly, the networks operate as a function of both a grasp configuration and RGBD image of the object of interest. For a given RGBD image we compute the surface normals and curvature from the associated point cloud giving an 8-channel representation (i.e. RGB (3), depth (1), normal (3), and curvature (1)). In the remainder of this section we first detail the specifics of the two networks. We then explain the training algorithm used, followed by a brief discussion of the gradient computation necessary to perform the inference described in the previous section.

3.1 A Multi-channel Grasp Configuration and Image CNN

We define a multi-channel deep neural network which takes as input a grasp configuration and RGBD image grasp patch and predicts as output the probability of

grasp success. We extract a 400×400 pixel object patch from the 8-channel RGBD representation. In our experiments we examine both keeping this object patch fixed and moving it as a function of the palm pose of the grasp. The configuration RGBD CNN takes the grasp patch and the grasp preshape configuration as the input to learn to predict the grasp success probability. We pass the image patch through two convolution layers and one max pooling layer. We first process the grasp configuration, θ, through one fully connected layer. We tile the resulting grasp configuration features point-wise across the spatial dimensions of the response map of the *pool1* features output from the image feature channel (c.f. [14]). This generates a concatenation of the grasp configuration and image features which pass through one convolution layer, then one max pooling layer, followed by two-fully connected layers, and a final logistic regression output layer. Figure 1 shows this "config-CNN" structure in detail. The config-CNN contains 61k parameters in total.

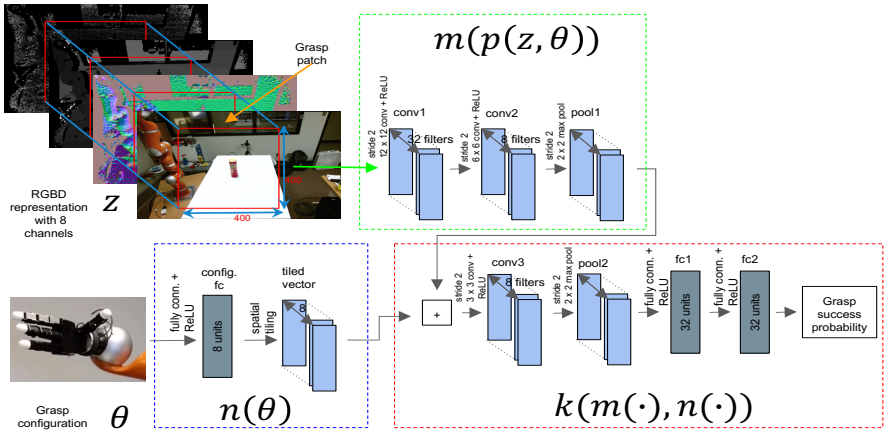


Fig. 1: The config-CNN architecture. Top left: the RGBD representation, z, extracted grasp patch, p, and image patch network channel $m(\cdot)$. Bottom: the Allegro hand and grasp configuration network channel $n(\cdot)$. Right: the combined feature network layers and output $k(\cdot)$.

3.2 An Image Patch Based CNN for Grasping

As an alternative to the config-CNN, we define an image based network inspired by [23]. We extract image patches related to the palm pose and fingertip locations for a given grasp configuration, θ. Each image patch is passed through two convolution layers, one max pooling layer, and one fully connected layer *fc1*. We concatenate the features from each finger and palm *fc1* and feed these into one fully connected layer followed by a logistic regression output. We show the structure of the "patches-CNN" in Figure 2 implemented for the Baxter parallel jaw gripper as an example. However, extending it for use on a multi-finger hand is straightforward.

We use the same 8 channel image representation as with the config-CNN. We extract a 200×200 pixel patch centered at the projected palm location and oriented to align with the project palm orientation. We extract 100×100 pixel patches for each finger centered at the projected fingertip location in the image rotated to align with the projected palm orientation. Our patches-CNN has 316k parameters in total.

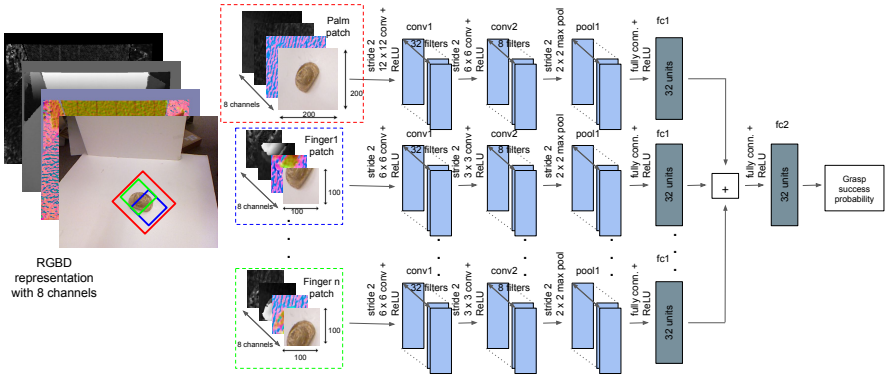


Fig. 2: The patches-CNN architecture and patch input. Finger and palm patches are extracted from the 8-channel RGBD representation on the left. The red bounding box represents the palm patch. The blue and green boxes define the patches for the two Baxter gripper fingers.

3.3 Training

We train our classifiers using the common cross entropy loss function:

$$\underset{w}{\text{argmin}} \quad \sum_{i=1}^{M} -y_i \log(f(\theta,z,w)) - (1-y_i)\log(1-f(\theta,z,w)) \tag{2}$$

where y_i defines the ground truth grasp label as 1 if successful and 0 if failed. We optimize for training by stochastic gradient descent (SGD) using Adam optimizer with mini-batches of size 8 for $6,000$ iterations. The learning rate starts at 0.001 and decreases by $10\times$ every $2,000$ iterations. We apply dropout to prevent overfitting keeping weights with a probability of 0.75 and apply Xavier initialization [5] for the weights. For the multi-finger grasp CNN training, we oversample positive grasps making sure at least one positive grasp exist in each mini-batch. The same training set up is used for the patches-CNN, except we train for $60,000$ iterations with the learning rate decreasing by 10x every $20,000$ iterations. When training on the two-fingered grasps dataset from [13] we do not oversample, but mirror each grasp patch to double the number of training examples. We implemented and trained all of our neural network models using TensorFlow (`http://tensorflow.org/`.).

3.4 Computing Gradients for Image Patches in Inference

Since, we define the patches used as input to our CNNs, as a function of the grasp configuration, θ, we must compute the gradient of the patch with respect to the configuration parameters for use in inference. In the patches-CNN the gradient takes the form:

$$\frac{\partial f}{\partial \theta} = \sum_{i=0}^{N} \frac{\partial f}{\partial p_i} \frac{\partial p_i}{\partial \theta} \tag{3}$$

where we define the sum over the derivatives with respect to the image patches p_i associated with the $N-1$ fingers and palm.

We can decompose the config-CNN $f(\cdot)$into sub-modules m, k, and n so that $f = k(m(p(z,\theta)),n(\theta))$ as shown in Fig. 1. This gives the following equation in

computing the gradient:

$$\frac{\partial f}{\partial \theta} = \frac{\partial k}{\partial n}\frac{\partial n}{\partial \theta} + \frac{\partial k}{\partial m}\frac{\partial m}{\partial p}\frac{\partial p}{\partial \theta} \tag{4}$$

We use backpropogation to compute most of the terms in Equations 3 and 4; however, for both networks we use finite differences to estimate the patch gradient with respect to the hand configuration:

$$\frac{\partial p}{\partial \theta} = \frac{p(z, \theta + \varepsilon) - p(z, \theta - \varepsilon)}{2\varepsilon} \tag{5}$$

4 Experimental Validation

In this section, we describe our simulation-based training data collection process and the experimental evaluation of our grasp inference approach. We conduct multi-finger grasp experiments using the same four-fingered Allegro hand mounted on a Kuka LBR4 arm in simulation and on the real robot. We compare to a heuristic grasp procedure, as well as the two dominant approaches to deep-learning based grasp planning: sampling and regression. Finally, we show the applicability of our inference method to two-finger grippers by preforming real-world experiments on the Baxter robot. All data and software used in this paper are available for use at: `https://robot-learning.cs.utah.edu/project/grasp_inference`.

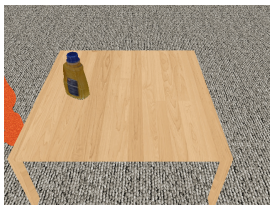
4.1 Multi-finger Grasping Data Collection

We developed a grasping system using ROS that runs both in simulation and on the real robot. We collected simulated grasps data using the Allegro hand mounted on the Kuka LBR4 arm inside the Gazebo simulator with the DART physics engine (https://dartsim.github.io/). We use Blensor [7] to generate RGB and depth images simulating a Kinect camera we use in real-world experiments. Example images generated by blensor can be seen in Figure 3.

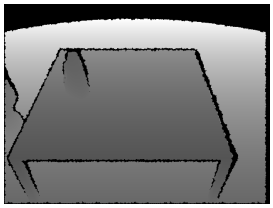
For a given trial we place the selected object, with a predetermined support surface facing down, at a location chosen uniformly at random from a 0.2×0.2m rectangle area with a uniformly random orientation. We then perform object segmentation by fitting a plane to the table using RANSAC and extract the points above the table. We compute a 3D bounding box of the segmented object using PCA used to generate heuristic grasps associated with the bounding box's top face and its two faces closer to the camera.

We generate a grasp for a given face by setting the palm to be a fixed distance from the face center (2cm for top grasps and 3cm for side grasps) in the direction of its surface normal. For data collection we add zero mean Gaussian noise with a standard deviation of 1mm along the normal to get a more diverse set of grasps. We rotate the palm to align with the bounding box face. For side grasps we have the thumb pointing towards the top, for top grasps we randomly select to align with

either the major or minor axis of the face and add an additional small amount of random noise. We generate a preshape by randomly sampling joint angles for the first two joints of all fingers within a reasonable range, fixing the last two joints of each finger to be zero. The preshape of the Allegro hand has 14 parameters: 6 for the palm pose and 8 for joint angles. In data collection, each preshape reachable for the arm using the RRT-connect motion planner in Moveit is executed and recorded. In experiments, all 3 preshapes are used as initializations for inference. For grasping the object we first use a position controller to move the finger preshape joints to the desired preshape, followed by moving the arm to reach the desired palm preshape pose. We then use a simple grasp controller to close the hand. The controller closes the fingers at a constant velocity stopping each finger independently when contact is detected by the measured joints velocities being close to zero. We found slightly different controllers to work well for top grasps versus side grasps. For most grasps the controller closes the last three joints of non-thumb fingers and the last two joints of the thumb. Importantly, the last three joints includes both preshape and non-preshape joints. However, for overhead grasps the controller closes only the second joint of non-thumb fingers and only the third joint of the thumb.



(a) The RGB image.



(b) The depth image.

Fig. 3: The RGB and depth image generated by Blensor for one grasp trial of the "detergent" object. The bottom part of the orange robot arm can be seen in the left side of the image.

If after closing the hand the robot can lift the object to a height 0.15m without dropping it, the system labels the grasp as successful. We used this grasp system to collect a dataset containing 1507 grasp trials. The dataset covers all 125 objects of the Bigbird [22] dataset with an average of 12 grasps per object, depending on how many grasps were reachable by the planner. Of the collected grasps attempted 159 were successful.

4.2 Grasp Classification CNN Evaluation

We first compare the performance of our two proposed network architectures config-CNN and patches-CNN. We train the networks using a random 80% of the collected data and test with the remaining 20%. We repeat this procedure to perform a five-fold cross validation. We compare two different cross validation scenarios. In the first the training set contains examples of all objects in the test set. In the second scenario objects in the testing set are held out from the training set to simulate observing novel objects. We refer to these scenarios as "seen" and "unseen" respectively. In both scenarios, each fold contains approximately the same percentage of successful grasps as the complete data-set.

The patches-CNN performs poorly for the classification of multi-finger grasps for both seen and unseen scenarios. The network predicts all test grasps as failure

cases. Different training parameters (e.g. learning rate, dropout probability, the number of batches, the number of training iterations, and the oversampling of successful grasps) were attempted to improve learning, including using the successful settings from the config-CNN. As such we do not report detailed results from the patches-CNN. The comparative success of the config-CNN shows that having direct access to the grasp configuration parameters allows the network to learn a more useful feature representation than providing this information indirectly through the image patch.

The average receive operator characteristic (ROC) curve and the area under the ROC curve (AUC) for the config-CNN can be seen in Figure 4 along with the performance of selecting a label at 50% probability for comparison. Table 1 shows the mean and standard deviation of the accuracy and F1 score across all validation folds for both seen and unseen objects. We treat predictions with grasp probability above 0.4 as positive, based on the ROC curve and our desire for high recall. As we can see, the config-CNN predicts multi-finger grasp success reasonably well for both seen and unseen objects, significantly outperforming chance.

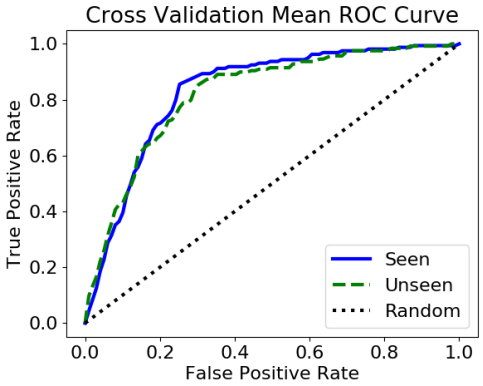


Fig. 4: The average ROC curve of five-fold cross validation for seen and unseen objects. For seen objects, the AUC mean is: 0.83 the AUC std: 0.02. For unseen objects, the AUC mean is: 0.82, the AUC std: 0.06. For random guessing, the AUC mean is: 0.5.

Table 1: Accuracy and F1 score of the RGBD config-CNN of the 5-fold cross validation for seen and unseen objects. We list the mean with standard deviation in parentheses.

Experiment	Accuracy	F1
Seen	0.766 (0.038)	0.405 (0.016)
Unseen	0.756 (0.05)	0.385 (0.096)
Random	0.5 (-)	0.172 (0.02)

As an additional comparison to our proposed networks we implemented a regression network to directly predict the grasp configuration parameters, θ, as a function of the 8-channel RGBD object patch. We implemented a network architecture similar to the RGBD channel of our config-CNN, except that we removed the third convolution layer used for concatenating the configuration and grasp patch features. The output layer of the regression CNN consists of separate linear regression outputs for each grasp parameter. We apply ridge regression to regularize the network with a strength of 0.5 in addition to the standard least-squares loss function.

In order to predict good grasps, the regression CNN can only learn from the 159 successful grasp examples in our dataset and makes no use of the remaining 1348 samples. Unsurprisingly this is far too few samples to reasonably learn in

our regression CNN with 90k parameters. The regression CNNs in [18, 12] have at least 60M parameters and the CVAE deep learning model in [24] has more than 2M parameters, thus we find it highly unlikely that they will work with our data-set.

We further validate the regressors performance by examining the distance between the predicted grasp and the ground truth example for a given object pose. The mean Euclidean distance between the predicted preshape palm location and the successful ground truth preshape palm location was above 0.4m and the minimum distance was above 0.2m. Thus the grasp regression CNN cannot predict palm locations within an effective distance of the object for grasping. The predicted palm orientations and joint angles were also far from the ground truth preshapes. We examined alternative deeper and shallower network structures for regression, but found none which could effectively learn from such a limited amount of data.

4.3 Multi-finger Grasp Inference in Simulation

We first evaluate our inference procedure for multi-finger grasp planning by performing grasping experiments on 10 objects of daily life from the Bigbird dataset. We test grasps for each object at 5 different random poses on the table. For each pose we generate three initial grasp configurations following the heuristic described in Section 4.1. We apply our CNN gradient ascent planner to each initialization and select the grasp preshape configuration with highest predicted grasp success probability for execution on the robot. The robot automatically selects to close with the top or side grasp controller depending on which bounding box face was used for initialization. If the planned grasp is not reachable by the arm motion planner, then we generate a new random object pose for evaluation.

We used the following parameters in performing inference. We constrained the palm location to be within a $0.1 \times 0.1 \times 0.1$m cube around the palm location given as initialization. We restrict the palm orientation to stay within 0.3 rad of the initial pose expressed in XYZ Euler angles. All preshape joint angles were restricted to the same limits used with the heuristic grasp. The maximum number of iterations for the inference is 100. We set the initial step size for backtracking line search to 0.001 and allow it a maximum of 10 iterations for each gradient ascent step. We set the control parameter of backtracking line search to 0.5.

We found that changing the object image patch as a function of the palm pose did not significantly change the grasps resulting from inference. As such, we only compute the change in grasp configuration using the gradient $\frac{\partial k}{\partial n}\frac{\partial n}{\partial \theta}$ (c.f. Sec. 3.1). This approximation sped up the inference procedure significantly without noticeably changing the grasps generated by the planner.

We examine our grasp planner on both seen and unseen objects. For the seen scenario we train the learner on the entire training set. In the unseen case we only hold out the test object from the training set to simulate the robot viewing a novel object. We compare our planner to three baselines. First we perform the three heuristic grasps for each object pose used by the planner. We ignore cases where the arm planner could not find a plan to reach the grasp pose. This gives a maximum of 15 heuristic grasps per object. The second approach evaluates the three heuristic grasps using the learned network and selects the one with highest predicted success prob-

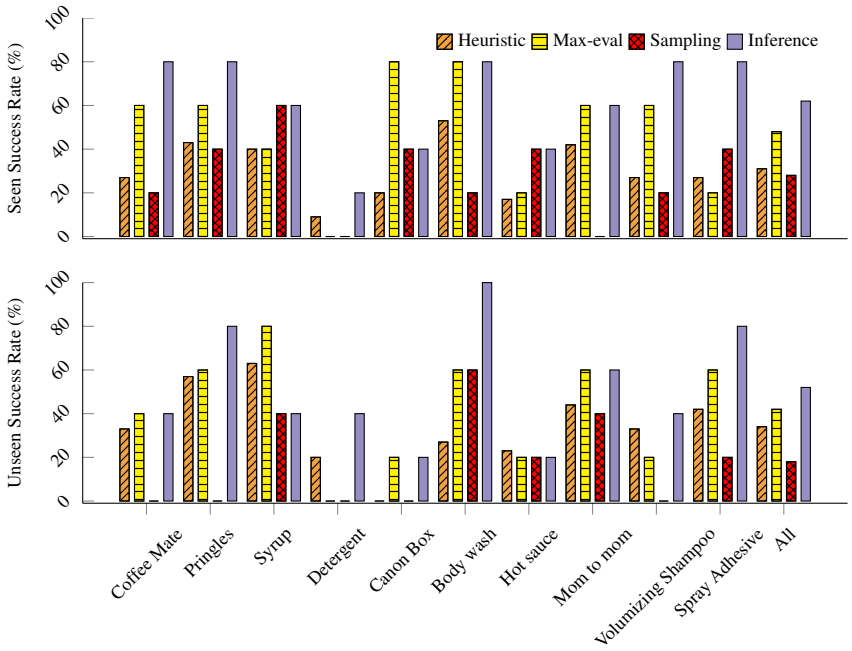


Fig. 5: Simulation Success Rate Results: Success rate for executed grasps on previously seen objects (top) and previously unseen objects (bottom) in simulation.

ability to execute. We term this approach "max-eval" and executed it on an additional 5 random object poses per object. Our final comparison performs a sampling approach where we generate 150 random grasp configurations. We evaluate all 150 generated grasps with our learned network and execute the grasp with highest predicted probability. Unlike the other three methods, the heuristic approach produced a variable number of reachable grasps for each object pose. In total we collected 121 heuristic grasp attempts for seen and 110 for unseen.

Figure 5 shows the grasping success rates for our method and the baselines comparisons. We report the rate of successful grasps for each object in both the previously seen and unseen cases. We note that our inference planner performs significantly better than the alternative approaches for both scenarios. Most importantly, for grasp initializations which would fail, the gradient-based optimization can refine these failure grasps to be successful. For example for the unseen scenario, a side grasp initialized for the "spray adhesive" object failed, but succeed after inference at the same object pose. We report the average success probability predicted by the network in Figure 6 for the initial heuristic grasps and the grasps found after inference for both the seen and unseen cases. We see that the inference is clearly improving the predicted probability, but has fairly low confidence in its prediction.

Our planner performs best with objects that can be easily enveloped from the side. It never selects precision grasps. For larger objects such as boxes the inference planner generally selects top grasps. For shorter objects, our arm planner had a difficult time finding plans which would not collide with the table, especially for side grasps. In terms of timing the inference-based planner takes approximately 2 to 3

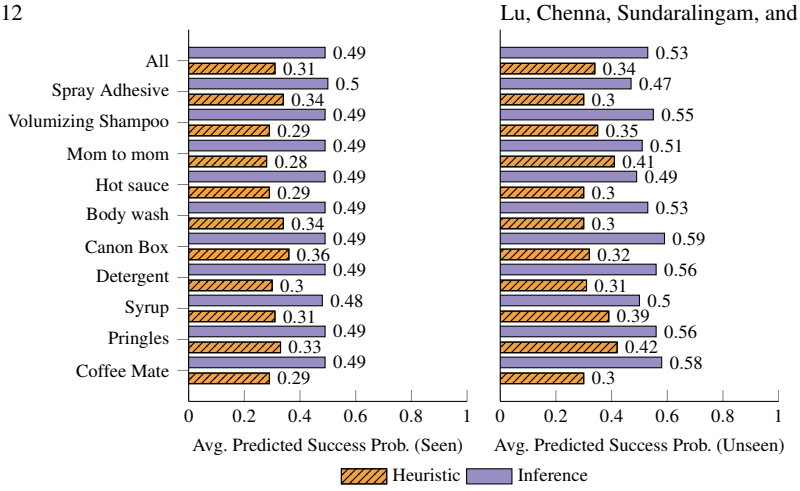


Fig. 6: Simulation Predicted Grasp Success Probability Results: Probability of grasp success predicted by the network for multi-fingered grasps for previously seen (left) and unseen (right) objects in simulation.

seconds for each initialization, while sampling takes around 6 seconds to evaluate the 150 sampled grasp preshapes, comparable to the time to perform inference with 3 initializations. Max-eval takes 0.1 to 0.2 second for 3 initializations. The computer used to run the training and simulation experiment has an Intel-i74790k with 64GB RAM and an Nvidia GeForce GTX 970 graphics card (1.1 GHz base clock rate and 4G memory) running Ubuntu 14.04 with ROS Indigo.

4.4 Real Robot Multi-finger Grasp Inference

We evaluate our config-CNN trained on the simulated grasp data on the real robot. We compare our proposed CNN gradient ascent inference to the heuristic grasp and the "max-eval" grasp planner using the same heuristic initializations. We do not compare to the regression network on the real robot, because of its poor performance in simulation. We perform inference and max-eval following the same procedures used with the simulated data, except for one minor modification. In order to overcome the issue of planned grasps being in collision with the table, we limit the minimum z-axis value of the hand position to remain above the table. We perform experiments on 5 objects from the YCB dataset [2], only the "pringles" can objects was present in the training data; however, it was a different flavor. We evaluate each object at 5 random poses.

Figure 7 summarizes the results of these experiments. We note that the inference results perform better than our comparison methods in all cases. Similar with simulation, the gradient-based optimization can refine these failure grasps to be successful. For example, a side grasp initialized for the "pitcher" object failed, but succeed after inference at the same object pose. In Figure 8 we show several example grasps generated by our grasp inference planner. We see that while most grasps are enveloping, different finger spreads are used to accommodate the different object geometries. The inference-based planner and max-eval have similar running time with simulation experiments run on similar hardware.

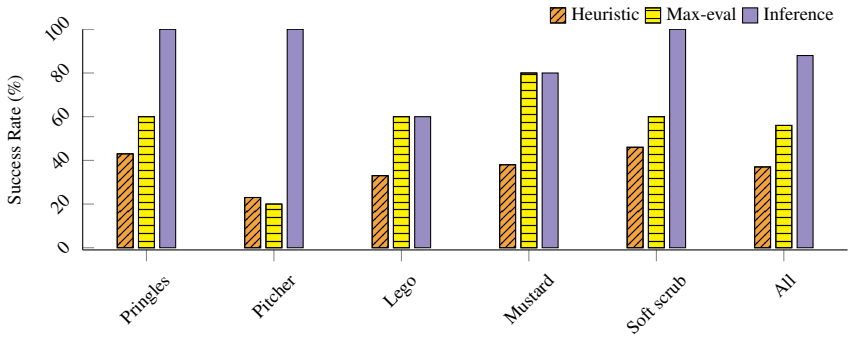


Fig. 7: Comparison of success rates for multi-fingered grasping on the real robot. Pringles was seen in training, all other objects are previously unseen.

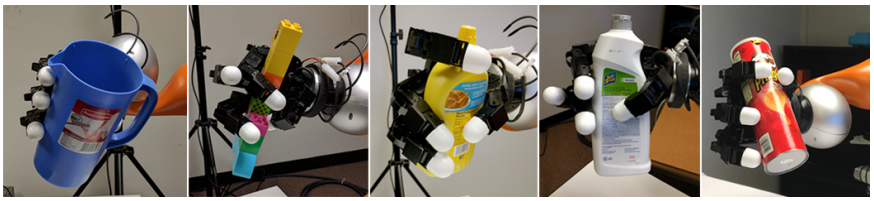


Fig. 8: Examples of successful grasps generated by our inference approach to grasp planning.

4.5 Experiments for Two-finger Gripper Grasp Inference

As a final evaluation we apply our grasp inference approach to planning two-fingered griper grasps on the Baxter robot. We train our patches-CNN (c.f. Figure 2) to predict two-finger gripper grasp success using the Cornell grasp dataset from [13]. Our patches-CNN network has similar classification performance on an offline validation dataset as the deep network results reported in [13]. To highlight the role of the inference procedure, and not the network, we chose to evaluate the patches-CNN, and not the config-CNN, as it is more similar to previous approaches tested on this dataset than our config-CNN.

We initialize our gradient ascent inference using four different configurations (grasp rectangle center, orientation, and gripper width) for a given object location. For all initializations we set the bounding box center to the estimated center of the segmented object and provide a random orientation. We initialize the gripper width to be 30 pixels. We perform inference initialized with all four configurations and select the configuration with highest predicted success probability to execute. We set the maximum number of iterations for gradient ascent to 10 and set the initial step size of the backtracking line search to 0.05. We set the maximum number of iterations for the backtracking line search to 5 per ascent step and the control parameter of backtracking line search to 0.5. We transform the rectangle found through inference to a 3D grasp pose using the same method as [13]. If the robot can then lift the object without dropping it to a height 0.2m, we label it successful. We selected 10 objects, some coming from the YCB dataset, with varying shape, texture, and visual properties comparable to the variability of the items used in [13]. Among these 10 objects different examples of the stapler, screw driver, and umbrella all appear in the

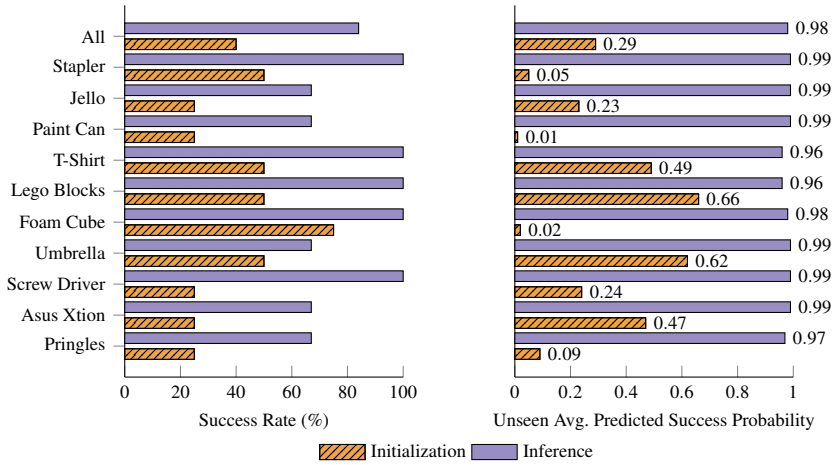


Fig. 9: Success rate for executed grasps and the associated probability of grasp success predicted by the network for Baxter 2-finger grasps.

Cornell dataset; however, the other 7 objects are entirely novel. For each object we evaluate grasps at three random poses. As a comparison to our inference approach we test all four random initial configurations for each object.

Figure 9 shows the success rate and success probability predicted by the patches-CNN for both inference and the initial heuristic grasps for all experiments. The inference significantly improves upon the initializations with an average success rate of 84% across all objects. This performance is comparable to the success rate shown in [13]. However, our inference takes only 4 initializations to achieve the same success rate, far fewer than the number of grasp samples required in [13]. Inline with our multi-finger experiments, we see that the CNN gradient inference can improve initial configurations which would fail to generate successful grasps. However, the trained network is over confident in its prediction of success probability predicting at lest 96% confidence, while only being successful 84% of the time.

5 Conclusions

We presented a novel approach for multi-fingered grasp planning formulated as probabilistic inference in a learned deep neural network. Our planning algorithm generally achieves higher grasp success rates compared with sampling-based approaches currently used for grasping with neural networks, while still running fast enough for use in a deployed robotic system. We showed the successful application of our grasp optimization approach on two novel neural network architectures. Additionally, our CNN classification approach to learning grasp success allows for more data-efficient learning of grasps compared to directly predicting grasps as output of a neural network regression model.

Our multi-fingered grasping results leave room for significant improvement. Importantly any improvement to the underlying classifier will immediately be leveraged by our planner. As immediate extensions we are examining the use of learned

priors over grasp parameters, θ, as well as using priors over the neural network weights w, to have better-calibrated predictions of the probability of grasp success, for reliable use within task-level planners. Finally, we wish to explore active data collection to further improve the data efficiency of our learning algorithm while also learning a wider variety of grasps.

Acknowledgements Q. Lu and B. Sundaralingam were supported in part by NSF Award #1657596.

References

[1] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-Driven Grasp Synthesis-a Survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[2] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research. In *International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.

[3] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dimensionality Reduction for Hand-Independent Dexterous Robotic Grasping. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3270–3275, 2007.

[4] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Gaussian Process Implicit Surfaces for Shape Estimation and Grasping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2845–2850, 2011.

[5] Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[6] Roderic A Grupen. Planning Grasp strategies for Multifingered Robot Hands. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 646–651, 1991.

[7] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. BlenSor: Blender Sensor Simulation Toolbox. *Advances in visual computing*, pages 199–208, 2011.

[8] Marcus Gualtieri, Andreas ten Pas, Kate Saenko, and Robert Platt. High Precision Grasp Pose Detection in Dense Clutter. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 598–605, 2016.

[9] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Deep Learning a Grasp Function for Grasping under Gripper Pose Uncertainty. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4461–4468, 2016.

[10] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging Big Data for Grasp Planning. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4304–4311, 2015.

[11] Marek Kopicki, Renaud Detry, Maxime Adjigble, Rustam Stolkin, Ales Leonardis, and Jeremy L. Wyatt. One-Shot Learning and Generation of Dexterous Grasps for Novel Objects. *The International Journal of Robotics Research (IJRR)*, 35(8):959–976, 2016.

[12] Sulabh Kumra and Christopher Kanan. Robotic Grasp Detection using Deep Convolutional Neural Networks. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[13] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep Learning for Detecting Robotic Grasps. *The International Journal of Robotics Research (IJRR)*, 34 (4-5):705–724, 2015.

[14] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *The International Journal of Robotics Research*, page 0278364917710318, 2016.

[15] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In *Robotics Science and Systems (RSS)*, 2017.

[16] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[17] Lerrel Pinto and Abhinav Gupta. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3406–3413, 2016.

[18] Joseph Redmon and Anelia Angelova. Real-Time Grasp Detection using Convolutional Neural Networks. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1316–1322, 2015.

[19] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. An Overview of 3D Object Grasp Synthesis Algorithms. *Robotics and Autonomous Systems*, 60 (3):326–336, 2012.

[20] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic Grasping of Novel Objects using Vision. *The International Journal of Robotics Research (IJRR)*, 27(2):157–173, 2008.

[21] Ashutosh Saxena, Lawson L. S. Wong, and Andrew Y. Ng. Learning Grasp Strategies with Partial Shape Information. In *AAAI National Conf. on Artificial Intelligence*, pages 1491–1494, 2008.

[22] Ashutosh Singh, Jin Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A Large-Scale 3D Database of Object Instances. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 509–516, 2014.

[23] Jacob Varley, Jonathan Weisz, Jared Weiss, and Peter Allen. Generating Multi-Fingered Robotic Grasps via Deep Learning. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4415–4420, 2015.

[24] Matthew Veres, Medhat Moussa, and Graham W Taylor. Modeling Grasp Motor Imagery through Deep Conditional Generative Models. *IEEE Robotics and Automation Letters*, 2(2):757–764, 2017.

[25] Yilun Zhou and Kris Hauser. 6DOF Grasp Planning by Optimizing a Deep Learning Scoring Function. In *Robotics: Science and Systems (RSS) Workshop on Revisiting Contact - Turning a Problem into a Solution*, 2017.