Extending GG-CNN through Automated Model Space Exploration using Knowledge Transfer

Mario Ríos Muñoz, Lambert Schomaker, S. Hamidreza Kasaei
Faculty of Science and Engineering
University of Groningen
Nijenborgh 9, 9747 AG Groningen, The Netherlands

***Abstract*— The ability to manipulate objects is a key aspect for future service robots. While autonomous manipulation has been accomplished in industrial settings, the uncertainty and inherent open-ended nature of unstructured environments like the home make for a challenging problem. One of the key areas in manipulation research is grasp affordance detection. CNNs have proven successful in predicting grasp candidates and the state of the art is capable of generalizing to unforeseen objects. However, the process to finetuning and optimizing the architecture for these networks is still manual and tedious. In this work we show how by formulating architecture optimization as a search problem it is possible to improve the performance by up to 5% on the state of the art GG-CNN architecture increasing its number of parameters by just 8%.**

I. INTRODUCTION

The classic approach to Grasp Pose Detection has been matching the object to be grasped to a database of known geometries, and adopting one of the previously defined poses according to some physics-related metric. The problem with this approach is its lack of scalability and generalization. Motivated by this limitation, an interest for open-ended, few-shots classification learning has risen. The goal is to develop algorithms that can extend the number of object classes throughout time, when representation of an object lies too far from the known categories. The new classes can be bootstrapped by the existing ones, and only a few additional grasp examples are needed to fully learn the new poses.

An alternative to dynamically extending the number of classes in a model is detaching object classification from grasp prediction. Pose generation can be formulated as a regression on geometric features, independently of the object class. For this type of problems, the use of CNNs has been particularly successful. However, the problem with optimally constructing this kind of models is that the number of possible combinations of layer hyperparameters is too big to be tested manually or exhaustively.

II. PROPOSED SOLUTION

Net2Net is a knowledge transfer technique proposed in [1] to accelerate and optimize the training of models. It introduces two transfer operators: *Net2WiderNet* and *Net2DeeperNet*. In the context of CNNs, the first operator adds filters to a convolutional layer, and the second adds convolutional layers to the model. The transfer of knowledge is achieved by initializing the added weights s.t. the output of the extended network equals the original one.

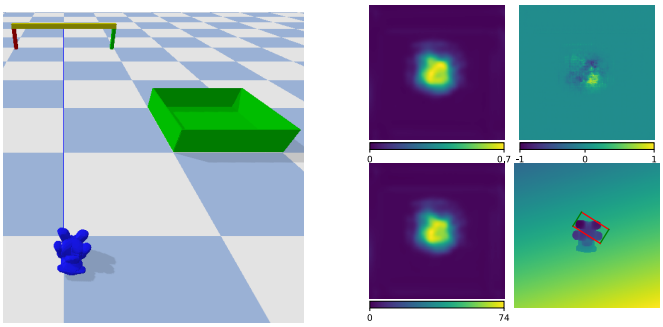

Fig. 1. *(Left)* Simulation environment with a free-floating 6DoF parallel-jaw gripper. A grasp is considered successful if the CoM of the object is dropped within the target bin. *(Right)* Pixelwise prediction of extended GG-CNN. Upper left is the normalized grasp quality distribution. Upper right is the $-\pi/2, \pi/2$ normalized orientation. Bottom left is the gripper width in pixels. Bottom right is the resulting oriented box for the absolute maxima in the quality distribution.

In [2] the *Net2Net* framework is used to automate model fine tuning for a 3D shape recognition task. They formulate the architecture optimization problem as a beam search in model space. The root node of the search is an initial simplified model and the offspring is generated by applying the transfer operators to the teacher network. The branching factor at a given depth is a function of the number of layers compatible with a transfer operator and the number of operators (2), and is limited by the size of the beam.

GG-CNN [3] is a small (64K parameters) fully-convolutional network that can yield a pixelwise prediction (fig. 1) of grasp quality, rotation and width of a parallel jaw gripper in real-time (19ms). In this work we employ a beam search exploration algorithm to either:

1) Increase the performance by adding a small number of parameters to the model.
2) Optimize the network by reducing the number of parameters while keeping the performance rate.

III. EXPERIMENTS AND RESULTS

To evaluate the performance of the new models we implement a custom simulation using pybullet. Our benchmark consists on picking 40 objects from the ShapeNetSem dataset [4] and dropping them within a target bin (see figure 1). Each task is repeated 5 times with a random initial pose of the object. Training is conducted over the Cornell Grasping Dataset [5]. As a heuristic to guide the search we use the

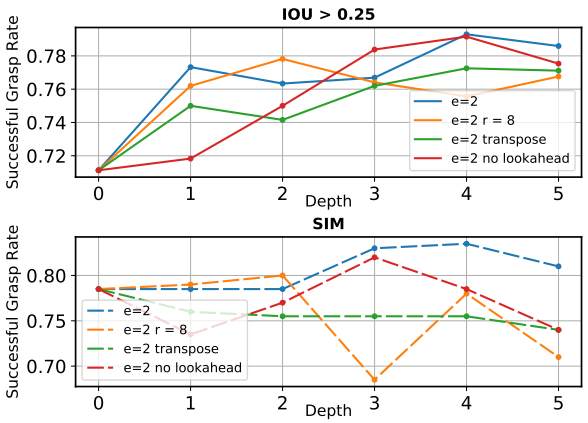

Fig. 2. Experiments extending the original GG-CNN architecture: e denotes the number of training epochs after applying a transfer operator; r is the number of epochs we train the selected candidate after a full lookahead step; *transpose* implies that the transposed-convolutional layers are expanded as well

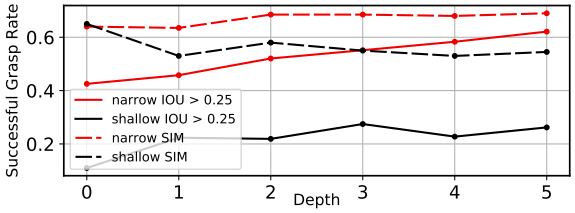

Fig. 3. Optimization experiments. *Narrow* starts with the same number of layers as the original model but less filters per convolution. *Shallow* is an extremely reduced version of the model with only 4 layers, shaped $5 \times 5 \times 4$ and $3 \times 3 \times 2$ respectively for both convolutional and transposed-convolutional layers.

GG-CNN	Accuracy	Parameters
Original	0.785	62420
Extended	**0.835**	67644
Optimized	0.715	**16104**

TABLE I

COMPARISON OF ACCURACY AND SIZE OF THE MODELS

rate of IOU > 0.25 between the predicted oriented box with highest robustness and the ground truth.

A. Extending the Architecture

To improve the overall performance of GG-CNN we took the published Keras model and grew it using beam search with a beam size of 3 and a depth of 5. The original beam search algorithm uses lookahead to increase the robustness of the exploration, however it adds a considerable overhead. We ran tests with and without lookahead. We also explored the impact of training epochs. In order to achieve a balance between exploration time and performance for the search with lookahead, we did one experiment where the offspring was trained for only 2 epochs but the selected candidate after each lookahead was retrained for an additional 8 epochs.

Figure 2 shows the results for these experiments. *IOU* success rates are directly optimized by the search and *SIM* performance reflects the generalization potential of the network in our simulation benchmark. In our experience, the best performance was obtained when training the offspring by only 2 epochs, with lookahead and extending only the feature extraction layers. Longer training times would overfit the model, which is reflected on the trial where the best candidate at each depth is retrained for 8 epochs. The convolutional layers of our final extended architecture are shaped $9 \times 9 \times 32, 5 \times 5 \times 16, 3 \times 3 \times 16, 3 \times 3 \times 16, 3 \times 3 \times 8, 3 \times 3 \times 8$ which is the result of consecutively applying the operations `deeper(3)`, `deeper(4)`, `wider(3)`, `deeper(3)` (where `deeper/wider` denotes the transfer operator and the digit is the index of the layer starting at 1).

B. Optimizing the Architecture

In order to find a smaller architecture with similar capabilities we constructed and trained two simplified networks: one *narrow* sized $C_{9\times9\times8}C_{5\times5\times4}C_{3\times3\times2}T_{3\times3\times2}T_{5\times5\times4}T_{9\times9\times8}$ and one *shallow* sized $C_{5\times5\times4}C_{3\times3\times2}T_{3\times3\times2}T_{5\times5\times4}$ where C and T denote convolutional and transposed-convolutional respectively. We used the same hyperparameters as for the previous experiments, but since these are smaller models we chose to expand both types of layers. Unsurprisingly, the *shallow* architecture is too underpowered to get anywhere close to the baseline performance (fig. 3) however the *narrow* model reaches an accuracy of 0.715 after 5 modifications. Something to be noted is the relatively high simulation performance of the initial networks. This can be explained by the fact that even if an end-effector is not perfectly aligned, when closing the fingers the object can get shifted towards a more robust grip.

IV. CONCLUSIONS

We were able to improve the accuracy of the state of the art GG-CNN architecture in 5% while only increasing its size by an 8% (see Table I). The shape of the filters in the extended architecture are a good example of why automated exploration is relevant, since this combination of layers is unlikely to be manually chosen. While we found an architecture $4\times$ smaller, it performs a 7% worse and we disregard it, considering that the original model is already lightweight enough to run in real time.

REFERENCES

[1] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

[2] X. Xu and S. Todorovic, "Beam search for learning a deep convolutional neural network of 3d shapes," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 3506–3511.

[3] D. Morrison, P. I. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *CoRR*, vol. abs/1804.05172, 2018.

[4] M. Savva, A. X. Chang, and P. Hanrahan, "Semantically-Enriched 3D Models for Common-sense Knowledge," *CVPR 2015 Workshop on Functionality, Physics, Intentionality and Causality*, 2015.

[5] "Cornell grasping dataset," http://pr.cs.cornell.edu/grasping/rect_data/data.php, (Accessed on 5/11/2019).