Convolutional Residual Network for Grasp Localization

Ludovic Trottier, Philippe Giguère, Brahim Chaib-draa
Computer Science and Software Engineering
Québec, Canada
Laval University
ludovic.trottier.1@ulaval.ca
{*philippe.giguere, brahim.chaib-draa*}*@ift.ulaval.ca*

***Abstract*—Object grasping is an important ability for carrying out complex manipulation tasks with autonomous robotic systems. The grasp localization module plays an essential role in the success of the grasp maneuver. Generally viewed as a vision perception problem, its goal is determining regions of high graspability by interpreting light and depth information. Over the past few years, several works in Deep Learning (DL) have shown the high potential of Convolutional Neural Networks (CNNs) for solving vision-related problems. Advances in residual networks have further facilitated neural network training by improving convergence time and generalization performances with identity skip connections and residual mappings. In this paper, we investigate the use of residual networks for grasp localization. A standard residual CNN for object recognition uses a global average pooling layer prior to the fully-connected layers. Our experiments have shown that this pooling layer removes the spatial correlation in the back-propagated error signal, and this prevents the network from correctly localizing good grasp regions. We propose an architecture modification that removes this limitation. Our experiments on the Cornell task have shown that our network obtained state-of-the-art performances of 10.85% and 11.86% rectangle metric error on image-wise and object-wise splits respectively. We did not use pre-training but rather opted for on-line data augmentation for managing overfitting. In comparison to previous approach that employed off-line data augmentation, our network used 15x fewer observations, which significantly reduced training time.**

***Keywords*-deep residual network; grasping; localization**

I. Introduction

Grasping objects with a robotic arm is an essential ability for performing complex manipulation tasks with robotic systems [1]–[5]. General purpose robots need to physically interact with the surrounding world in order to accomplish their goals. Actions such as opening doors, moving objects around or using tools, all require the fundamental skill of grasping objects. Integrating the necessary knowledge for succeeding at this task is one of the first steps towards the long-term goal of deploying intelligent robotic agents.

The ease with which humans perform object grasping is however not an accurate reflection of the true complexity of the task. Humans see objects and can identify without much difficulty the proper location where to put their hands for picking them up. They can also estimate other properties of the objects, such as the weight, by simply looking at them. This is due to years of jointly training the visual and motor cortices with sensory information from both the world and the body during interactions with objects. Robotic grasping still lags behind human grasping because replicating such learning conditions is a challenging problem.

Since visual perception plays an important role in object grasping [6], we focus in this paper on the perception system for robotic grasping. Its goal is localizing graspable regions by performing general scene understanding. The system interprets light and depth information, known as RGBD images, and outputs a list of candidate regions ordered by graspability. This involves elaborate and complex visual tasks, such as image segmentation, visual disambiguation and object detection. Due to the complexity of these tasks, learning an efficient perception system for extracting the most relevant information from the surrounding environment is essential. The performance of the overall grasping system highly depends on the graspability of the predicted region.

Convolutional Neural Networks (CNNs) have become the leading deep learning approach for visual recognition [7], [8]. Inspired by the biological visual cortex, CNNs aim at learning feature hierarchies from the data. High-level features are composed of low-level features using multiple layers of non-linear transformations [9]. Each layer in the hierarchy constitutes an increase in the level of abstraction from the layers below. This key principle allows CNNs to represent complex raw observations, such as RGBD images, into simple high-level concepts.

Originally proposed in the 80s, CNNs only became popular recently due to being difficult to train. Their hierarchical structure makes them vulnerable to the *vanishing gradient* problem. As the CNN gets deeper, the back-propagated error signal from the higher levels to the lower levels vanishes more easily. Learning representative features in the lowest layers can be more difficult due to the weak error signal. This may create a discrepancy between the learning speed of the higher and lower layer parameters, which can impede the network from obtaining high performances.

Recently, several advances in network regularization have been proposed for reducing vanishing gradients. Residual networks (ResNets) with identity shortcut connections are among the most recent ones and are growing in popularity.

The central idea in ResNets is learning additive residual functions instead of standard feed-forward functions as the main building blocks of the network. This is accomplished by connecting the output of low-level layers to the output of high-level layers with an identity skip connection, then by merging the outputs with an addition. Skip connections create information propagation paths through the entire network, which allows propagating signals directly from one layer to the next. Due to the increased information flow in both the forward and backward passes, deep residual networks are easier to train than their non-residual counterparts.

The objective of this paper is to make use of the recent advances in network regularization for training a deep CNN on the task of localizing good grasp regions. We contribute in the following ways:

1) A standard ResNet for object recognition uses global average pooling prior to the fully connected layers. We have observed in our experiments that global average pooling removes the spatial correlation of the error signal, which makes learning the spatial locations of the graspable regions impossible for the network. We propose a modification of the ResNet architecture that removes this limitation.
2) We show that we improve the state-of-the-art performance of previous methods without pre-training and with on-the-fly data augmentation. This results in the removal of the tedious pre-training stage, and a 15x reduction in training images for the fine-tuning stage.

The rest of this paper is divided as follows. We review the related works in Section II and elaborate on the problem description in Section III. We present the experiments in Section IV and discuss the results in Section V. We conclude our paper in Section VI.

II. Related Work

Several previous approaches used 3D modeling for localizing regions of high graspability [10]–[12]. Although these knowledge-based methods can perform well in controlled situations, they rely on building complex and accurate 3D representations. Requiring *a priori* the physical properties reduces their applicability in general grasping scenarios. The system rarely knows in advance the physical properties of objects it has never seen. With our approach, we seek to alleviate these limitations by performing grasp localization solely from the light and depth information.

Recent works have proposed viewing grasp localization as a vision detection problem. Approaches such as score function [13], extreme learning machine [14] or sparse dictionary learning [15] apply vision-related frameworks for processing RGBD images. They however suffer from a large computational burden due to incorporating an expensive sliding-window stage during their detection. With our approach, we seek to perform grasp localization in a single-shot manner. By only processing each image once, our goal is to accelerate detection and obtain faster speed performances at test time.

Other works in the field of neural networks and deep learning have recently been proposed. For instance, Lenz et al. [5] proposed a cascade of multi-layered perceptrons for predicting the graspability of an image sub-region. They however relied on different types of hand-designed regularization terms for reducing the difficulties of training their network. Other approaches such as two-stage closed-loop [16] and pre-trained AlexNet [4] instead opted for the more popular convolutional neural network model. They however needed a lengthy pre-training for initializing the weights. With our approach, we want to put aside hand-designed regularization terms and avoid the tedious pre-training stage. Our goal is to make use of the recent advances in residual networks and train a deep neural network on the small Cornell dataset achieving state-of-the-art performances without pre-training nor custom regularization terms.

Recent efforts have been devoted to training grasp localization system with a big data perspective [2], [17]. These methods train scalable approaches by exploiting hundreds of thousands of observations. While these initiatives are fundamental for achieving the long-term goal of autonomous robotic systems, large datasets are not common in constrained industrial contexts. They are usually too expensive to create. A grasp localization approach that can be trained on a small dataset will be more applicable in this case. As we show in our experiments, our residual network obtains state-of-the-art performances by only training on observations from the small Cornell dataset.

III. Problem Description

In this section, we elaborate on the problem of localizing candidate grasp regions. We introduce the grasp rectangle representation and describe the proposed residual CNN.

A. Grasp Localization

The goal of grasp localization is predicting a representation of the gripper configuration in a region of high graspability. For a standard gripper with two parallel plate-gripping fingers, a typical representation of the full 7-dimensional configuration is a low-dimensional projection of the gripper at the last stage of the grasp. Although several previous works have proposed different representations, such as a single 2D point [18] or a pair of points [19], these representations did not faithfully encompass the full dimensionality of the gripper. For instance, a single 2D point does not include the gripper orientation, while a pair of points does not include the height of the gripper's plates. Using incomplete representations leaves some aspects of the configuration to be estimated separately, which adds complexity and new sources of error.

The grasp rectangle is a faithful representation of the full 7-dimensional gripper configuration [13]. The 2D oriented

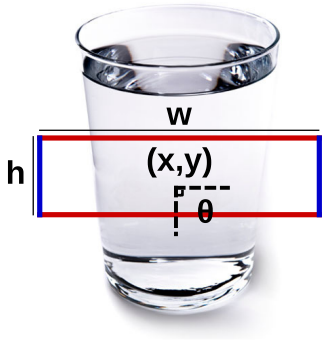

Figure 1. An example of a grasp rectangle R labeling a potential good grasp. This five-dimensional grasp representation is used with two-plated parallel grippers. The blue lines are the plates, the red lines show the gripper opening, (x, y) are the center coordinates and θ is the orientation.

rectangle takes into account the gripper's location, orientation and physical constraints. As shown in Figure 1, the rectangle specifies 5 parameters:

$$R = \{x, y, w, h, \theta\}\,, \tag{1}$$

where (x, y) corresponds to the center position of the gripper, (w, h) the opening and length of the plates, and θ the orientation with respect to the horizontal axis of the usual left-handed Cartesian coordinate system.

Using the grasp rectangle representation makes grasp localization similar to object detection in vision. The task is formulated as a region-based regression problem, where the goal is predicting a grasp rectangle and a confidence score for each image sub-region [4]. The sub-region with the highest confidence score is selected as the candidate location and the associated rectangle as the candidate grasp configuration.

In this paper, we use a 7×7 grid, which amounts to predicting 49 grasp rectangles and 49 confidence scores for each image [4]. For the grasp rectangles, we take into account the rotational invariance of the gripper angle. Due to the symmetry of the rectangle, grasping at θ or $\theta + 180°$ results in the same gripper configuration. Managing this invariance is done by predicting the sin and cos of twice the angle [4]:

$$R' = \{x, y, w, h, \sin(2\theta), \cos(2\theta)\}\,. \tag{2}$$

As a result, the model's required output dimensionality for this regression problem is $7 \cdot 7 \cdot 6 = 294$ for the grasp rectangles and $7 \cdot 7 \cdot 1 = 49$ for the confidence score. Figure 5 shows examples of such predictions.

B. Residual Network

The aim of residual networks is to address the degradation problem, which is defined as the decrease in accuracy as depth becomes greater than a certain threshold [20]. While we may believe that the recorded performance reduction

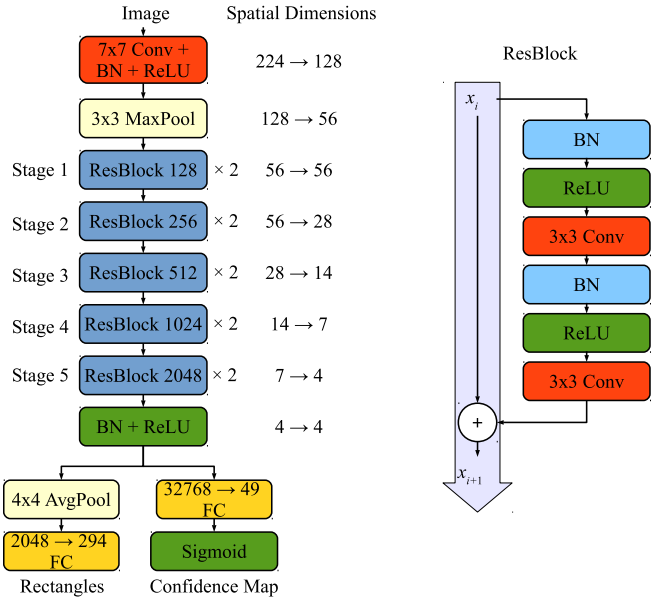

Figure 2. An illustration of our residual network (left) and the residual block (right) from which it is composed. A block has a residual mapping f_i and an identity skip connection. The output x_i of block i is processed twice by a series of batch normalization, ReLU and convolution layers, then is added to x_i. This gives the output x_{i+1}, which in forwarded to the next block. Our network does not use the standard global average pooling for predicting the confidence map, only for the rectangle configurations.

was caused by overfitting, experiments show otherwise [21]. Given a shallow well-trained network achieving good performance, it is always possible to artificially increase its depth by stacking identity layers without affecting accuracy. The existence of such solution indicates that a deep network should never produce worse performance than its shallower counterpart. However, learning identity layers with a fully-connected / convolutional layer followed by a non-linear activation function is a hard optimization task [21]. Even though a deep network optimal solution exists by construction, obtaining it may not be possible because the model cannot be adequately optimized.

Residual networks thus address the degradation problem by facilitating the learning of identity mappings. It is a type of architecture consisting of a stack of residual blocks. Each block contains a residual mapping f_i and a skip connection bypassing f_i. In this paper, we only consider identity skip connections [22]. Denoting x_i as the output of block i, a residual network is recursively defined as:

$$x_{i+1} = f_i(x_i) + x_i \tag{3}$$

With this formulation, the network can easily learn an identity mapping between x_i and x_{i+1} by pushing all parameters of f_i to zero.

Previous work has shown that the performance of the network highly depends on the choice of the mappings f_i [22]. In this paper, we only consider full pre-activation

mappings, since they obtained the best performances in previous work [22]. As shown in Figure 2, they are defined as follows:

$$f_i(x_i) = W_{i2} \cdot \sigma(B(W_{i1} \cdot \sigma(B(x_i)))) \,, \tag{4}$$

where B corresponds to batch normalization [23], σ to the ReLU non-linearity [24], W_{i1} and W_{i2} to the weights. For CNNs, we replace the matrix multiplication $\cdot$ by a discrete 2D convolution $*$.

A standard residual CNN for classifying objects in RGB images uses a global average pooling layer at the end of the convolutional stage, i.e. prior to the fully-connected layers. For a convolutional stage that outputs a (n_f, n_h, n_w)-dimensional tensor, the global average pooling layer computes the average over dimensions n_h and n_w. This results in an n_f-dimensional vector that is fed to the subsequent fully-connected layers. A traditional CNN would instead flatten the (n_f, n_h, n_w)-dimensional tensor into a $(n_f \cdot n_h \cdot n_w)$-dimensional vector. Global average pooling helps to create correspondence between the feature maps and the object categories, which can improve classification performances.

We have observed in preliminary experiments that a residual CNN with global average pooling cannot solve the grasp localization regression problem of Section III-A. The network never converges to a suitable solution because it is unable to assign high confidence score to sub-regions of high graspability. In fact, it assigns the same confidence score to every sub-region which shows that it does not learn to localize grasp rectangles. This is due to the averaging nature of the pooling layer that causes spatial information loss in the error signal. The network is unable to spatially correlate its mistakes on the confidence scores with the gradient of the loss function because the spatial information is squashed by the average.

To solve this problem, we propose keeping the global average pooling only for predicting the grasp rectangles, and propose using a standard flattening layer for the confidence scores. While the global average pooling will not interfere with the spatial information of the confidence scores, it will create correspondence between the feature maps and the rectangle categories as it does for object categories in object recognition. Moreover, we also add a fifth residual stage to the standard four stages of a residual CNN. A residual stage is a series of residual blocks with either the same number of features or the same spatial dimensions. The purpose of this fifth residual stage is reducing the output dimensionality of the convolutional stage from $(1024, 7, 7)$ to $(2048, 4, 4)$. This reduces by approximately 850k the number of weights of the fully-connected layer for the confidence scores, which helps to manage overfitting. An illustration of our residual network is shown in Figure 2.

C. Managing Overfitting

Despite the recent advances in network regularization and architectures improving the ease with which CNNs learn, training a deep network remains a challenge. The biggest difficulty comes from learning feature representations generalizing to the complete data distribution. Representing complex observation is particularly difficult, especially when the amount of data is small. The network can easily overfit on the training dataset if no proper care is taken while training. Avoiding overfitting then becomes a central concern in this case.

One standard approach for preventing the network from overfitting is artificially increasing the amount of training data. The central concept in data augmentation is generating additional images from the existing one while keeping their class-specific characteristics. Each image is repetitively preprocessed by a series of class-preserving random transformations, each time generating a new image that resembles the original one. This preprocessing can be done either off-line or on-line. In the former case, the training set containing both the original images and the newly generated ones is fixed and used in a standard way for training the network. In the latter case, the images are generated on-the-fly during training, and the network sees new images every iteration. We opted for the on-line approach because we did not need to save images on disk.

The quality of the generated images highly depends on the choice of preprocessing. Using proper transformations can make a substantial difference to the performance of the network. We now elaborate on the ones that we used in our experiments. The transformations are presented according to their order in the preprocessing:

1) Standardization: From the original RGBD image, we estimate the depth normal N_x, N_y, N_z. We then subtract the mean and divide by the standard deviation (computed from the training set) of each 7 channels.
2) Rotation: The image is rotated by a random angle uniformly sampled from the range $[-15°, 15°]$.
3) Random center crop: A 320×320 crop is taken from the center of the original 640×480 image, translated by up to 50 pixels in both the horizontal and vertical directions.
4) Square scale: The 320×320 crop is resized to 224×224 using standard bilinear interpolation.
5) Color jitter: We randomly change the brightness, contrast and saturation with a blend uniformly sampled from $[0.6, 1.4]$.
6) Lighting: We perform Krizhevsky's Fancy PCA color augmentation on all RGB-D-$N_xN_yN_z$ channels [24].

Note that since the grasp rectangles are spatially localized within the images, we have to perform the same preprocessing on them for preserving the proper grasp region labels.

Another approach for preventing overfitting that is grow-

ing in popularity is pre-training on a large dataset. Instead of randomly initializing the weights by sampling them from a given distribution [25], the network initializes its weights by training on a large external dataset. This can be seen as a kind of expert-driven weight initialization, which can provide a better starting point for optimization than random weights.

Although pre-training can improve performance, it adds other constraints. A large external dataset with observations sharing the same input modalities and similar characteristics must be available for pre-training to take place. This is not always the case, especially in our context with RGBD images. Observations must be diversified enough so that the network can learn general features usable for the original task. A large number of observations does not guaranty diversity per se. Pre-training on a large external dataset also substantially increases the training time, which would best be avoided by directly training on the original task. For these reasons, we do not use pre-training in this paper and rather use random initialization [25]. We leave pre-training for future work.

D. Training Details

Training our residual network for solving the detection problem of Section III-A requires special considerations [4]. For each image, we select uniformly up to five ground truth rectangles in five different sub-regions. We construct a target confidence map that has confidence scores of 1 at the selected sub-regions and confidence scores of 0 at the other sub-regions. We then back-propagate the Euclidean distance loss function through the entire confidence map, but only through the selected grasp rectangles.

We trained our residual network with the following hyper-parameters. We used a batch size of 64, a weight decay of 0.0005, momentum of 0.9 and Nesterov accelerated updates. We train for 200 epochs at a starting learning rate of 0.1, which is divided by 5 at epoch 60, 120 and 160. All training is performed on a single NVIDIA K80 GPU.

IV. Experiments

In this section, we elaborate on our experiments and present our results on the Cornell dataset.

A. Dataset

The Cornell grasping dataset contains 885 Kinect RGBD images of 240 distinct everyday objects [13]. Each image is labeled with positive and negative grasp rectangles labeling both regions of good and bad graspability. Note that we only use the positive rectangles for training the network. As shown in Figure 3, the dataset contains a wide range of objects, each with rectangles of various dimensions, orientations and positions. The Cornell task is difficult because not all regions with high graspability are labeled, and the dataset has a small number of relatively complex input observations.

Figure 3. Cornell dataset example images. The dataset contains a good variety of everyday objects, where each object is labeled with grasp rectangles identifying regions of high graspability. The Cornell task is difficult because not all good grasp regions are correctly labeled.

B. Grasp Detection

The quality of the predicted grasp rectangle is computed using the rectangle metric [13]. Let $\bar{R}$ be the predicted rectangle and $\{R_1^*, \ldots, R_K^*\}$ the set of positive rectangles (ground truth). The localization is considered a success if the metric is positive for any ground truth rectangles R_k^* and a failure if it is zero for all of them. The metric computes two quantities:

1) Absolute angle difference: $|R(\theta) - R_k^*(\theta)|$
2) Jaccard index: $\frac{area(R \cap R_k^*)}{area(R \cup R_k^*)}$

The metric is positive if both the absolute angle difference is lower than 30° and the Jaccard index is greater than 25%.

We performed our experiments using 5-fold cross-validation. For further reducing variance, we report the best overall score achievable over five cross-validation tries, along with the median. We have two learning scenarios in our experiments, where each reflects a way of making the folds during cross-validation. In the first one, we split the observations according to the images (as in standard cross-validation). We refer to this experiment as image-wise splitting. In the second one, we split the observations according to the objects, gathering all images of each object in the same fold. We refer to this experiment as object-wise splitting. The first scenario examines the generalization performance for predicting new rectangle positions, orientations and sizes on already-seen objects. This is representative of a typical industrial context where the set of test objects is known beforehand. The second scenario studies the generalization performance for predicting rectangles on novel, unseen objects. This is representative of a typical everyday context for general purpose robots. Using both scenarios will give a

Table I
CROSS-VALIDATION DETECTION RESULTS ON THE CORNELL DATASET. THE VALUES REPRESENT THE AVERAGE RECTANGLE METRIC ERROR (IN %) OVER FIVE FOLDS.

Approaches	Image-Wise Split	Object-Wise Split
Jiang et al. [13]	39.5	41.7
HOG + ELM Kernel [14]	35.2	-
SAE, struct. reg. two-stage [5]	26.1	24.4
Two-stage closed-loop [16]	14.7	-
Pre-trained AlexNet [4]	12.0	12.9
NKM-N [15]	**10.60**	**11.83**
AlexNet (ours)	18.30 (19.21)	18.68 (19.53)
ResNet (ours)	10.85 (12.20)	11.86 (12.26)

Table II
SPEED COMPARISON AT TEST TIME ON THE CORNELL DATASET IMAGES.

Method	**Speed (fps)**
Jiang et al. [13]	-
NKM-N [15]	< 0.01
SAE, struct. reg. two-stage [5]	0.07
HOG + ELM Kernel [14]	0.09
Two-stage closed-loop [16]	7.11
Pre-trained AlexNet [4]	13.15
AlexNet (ours)	**13.72**
ResNet (ours)	11.5

more in-depth evaluation of the model.

C. Results

The detection performances on the Cornell dataset of our residual network, along with other state-of-the-art approaches, are presented in Table I. We report both the best score and the median obtained over five repeated five-fold cross-validation experiments (the median is the value inside the parenthesis in Table I). We also report the speed performance at test time for each approach in Table II. The performance test includes both the pre-processing step (starting when the image is loaded from disk) and the network prediction time (up until the rectangle metric is computed).

Our residual network obtained 10.85% and 11.86% rectangle metric errors for the image-wise and object-wise splits respectively, with a speed at test time of 11.5 frames per second (FPS). As comparison, Redmon et al. [4] who fine-tuned an Imagenet pre-trained AlexNet, obtained 12% and 12.9% rectangle metric error respectively, at a speed of 13.15 FPS. An important difference between Redmon et al.'s approach and ours is the number of training images. We trained our residual network with on-line data augmentation while Redmon et al. used off-line data augmentation. They generated 3000 images per training observations, which resulted in a training fold containing 2,124,000 unique images. They then fine-tuned their network for 25 epochs, which amounted to processing 53.1 M images. In our case, we trained for 200 epochs on the original fold size of 708 images, but generated new observations at each epoch. Our network only

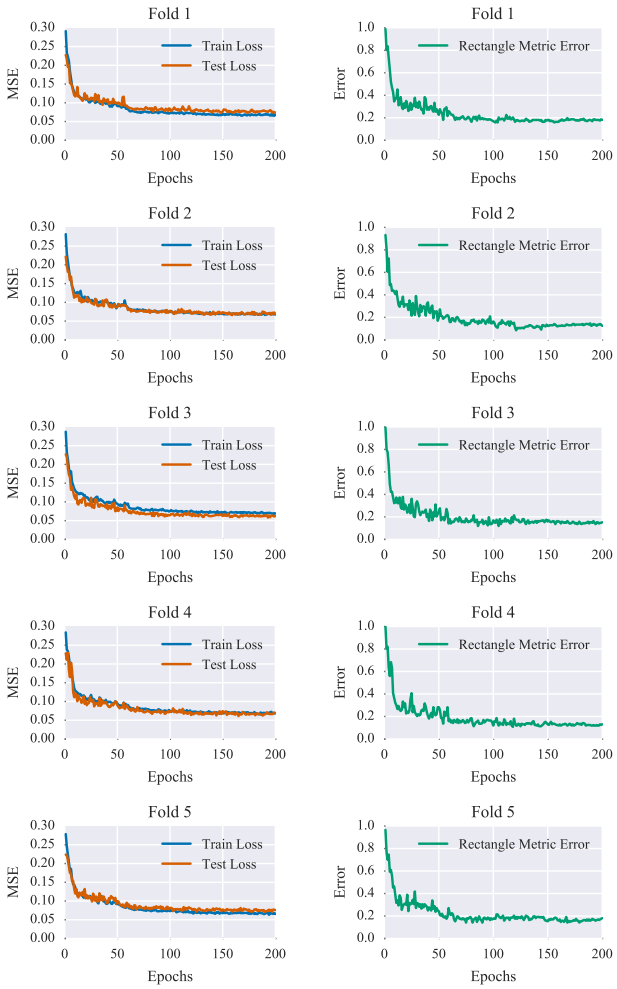

Figure 4. Learning behavior of our residual network during cross-validation training. The left column shows both the train and test loss convergence for each fold, while the right column shows the rectangle metric error convergence. The small difference between the train and test loss indicates no overfitting.

sees a total of 141,600 images, which corresponds to 15x fewer observations. Using a single NVIDIA K80 GPU, it only took around 6 hours to train our network on one fold. Our framework is more difficult, but also more convenient because it is less computationally demanding.

To understand the difference between our training framework with on-line data augmentation and Redmon et al.'s training framework with off-line data augmentation, we fined-tuned an ImageNet pre-trained AlexNet on the Cornell dataset. Note that we used the same hyper-parameters and the same input modalities (RGD) as reported in [4] to train the network. We obtained 18.30% and 18.68% rectangle metric errors for the image-wise and object-wise splits, which represents absolute performance reductions of 6.30% and 5.78% respectively. Although our residual network performances are relatively similar to Redmon et al.'s AlexNet

performances, they are more different when compared to AlexNet performances on our training framework.

Another approach worth mentioning is NKM-N, which has the best performances in both the image-wise and object-wise splits. This approach is however not usable for real-time processing because of its speed performance at test time. As presented in Table II, NKM-N takes more than a minute to process an image while our approach processes around 11.5 images per second. This is due to the tedious sliding-windows technique used by the authors for detection, which is more computationally demanding than the optimized 2D convolutions in CNNs.

V. Discussion

One important aspect to consider when training CNNs is overfitting. Since deep neural networks usually have millions of parameters, they can easily overfit to the training data, especially when the number of observations is small. A straightforward test to perform for monitoring overfitting is looking at the distance between the train and test loss. Figure 4 presents the learning behavior of our residual network for each cross-validation fold. As shown in the left column, the train and test losses have a steady and similar decrease. Even though we do not use Dropout [26] to further regularize our network, these results show that the network is not in an overfitting regime.

Due to miss-labeling in the dataset, the rectangle metric performance can occasionally vary from one epoch to another. As shown in the right column of Figure 4, the score sometimes significantly changes even though the train loss varies slowly. To understand why this is the case, we show in Figure 5 an example of predicted grasp rectangles from our residual network. According to the rectangle metric, the top example is considered a success, while the middle and bottom ones are considered failures. It is however conceivable that grasping the scissors using the candidate grasp rectangle would be a success even though the rectangle metric labels the detection as a failure. The predicted grasp rectangles are indeed too wide to give a good Jaccard index, but a gripper with two parallel plate-gripping fingers would have succeeded by pinching the object. Although the rectangle metric is convenient due to its resemblance to bounding box in vision, it can add non-existing physical constraints that negatively bias the reported performances.

As we previously mentioned in Section III-C, we did not use pre-training in our residual network experiments. We instead randomly initialized the weights following He et al.'s initialization approach [25]. This leaves open the possibility of pre-training our network on a large external RGBD dataset prior to training it on the Cornell task. For instance, the recently published grasping dataset containing around 650,000 grasp attempts would be a candidate of choice [17]. The authors gathered Kinect RGBD images of grasp attempts from various camera placements and

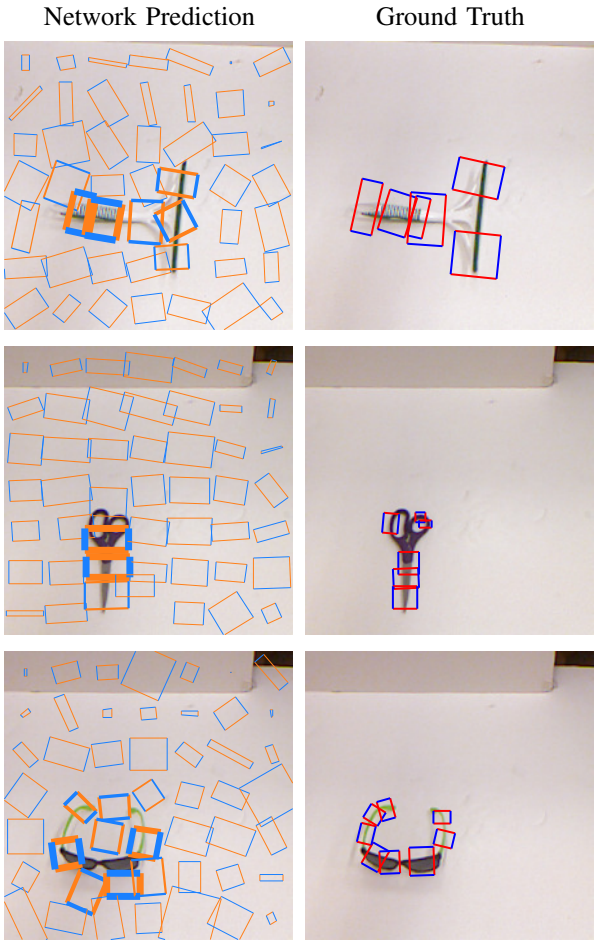

Figure 5. Examples of detection success (top example) and failures (middle and bottom). The left column shows the candidate rectangles predicted by our network, while the right column shows the ground truth. A rectangle with a large width indicates that the network is confident about its prediction. Although the middle example is considered a failure according to the rectangle metric, the grasping system would still have been able to grasp the scissors using the predicted rectangle.

hardware platforms using up to 14 robotic manipulators. Although the dataset is large by nowadays standards (around 886 GB) and training would require several GPUs, features learned on these highly diverse observations would be general and usable in several grasping contexts.

VI. Conclusion

The grasp localization module is an essential element for automating the grasping of ordinary, everyday objects. Generally viewed as a vision detection problem, its goal is localizing regions of high graspability by interpreting light and depth information. Recent developments in deep learning have made deep convolutional neural networks the model of choice for solving vision-related problems. In this paper, we investigated the use of residual networks for grasp localization. Our preliminary experiments have shown that the global average pooling layer at the end of the

convolutional stage in a standard residual network removes the spatial correlation of the back-propagated error signal. The network cannot spatially correlate its mistakes on the confidence map and learn to localize good grasp regions. We proposed applying the global average pooling layer only on the grasp rectangle predictions and use a flattening layer for the confidence scores. For reducing overfitting due to flattening high order tensors, we also proposed reducing the spatial size of the feature maps prior to flattening by increasing the number of residual stages. Our residual network obtained state-of-the-art performances on the Cornell dataset without pre-training and with on-line data augmentation. Our network obtained better performances with faster training time, convergence rate and a high prediction speed. Pre-training our residual network on a large RGBD dataset then fine-tuning it on the Cornell dataset would be an interesting avenue for future work.

ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation for providing the Tesla K80 GPUs for our experiments.

REFERENCES

[1] C. Eppner, S. Höfer, R. Jonschkowski, R. Martın-Martın, A. Sieverling, V. Wall, and O. Brock, "Lessons from the amazon picking challenge: Four aspects of building robotic systems," in *RSS*, 2016.

[2] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *ICRA*. IEEE, 2015, pp. 4304–4311.

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

[4] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *ICRA*, 2015, pp. 1316–1322.

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

[6] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *IJRR*, vol. 27, no. 2, pp. 157–173, 2008.

[7] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *PAMI*, vol. 38, no. 2, pp. 295–307, 2016.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015, pp. 91–99.

[9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[10] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner *et al.*, "Opengrasp: a toolkit for robot grasping simulation," in *SIMPAR*. Springer, 2010, pp. 109–120.

[11] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[12] R. Detry, C. H. Ek, M. Madry, and D. Kragic, "Learning a dictionary of prototypical grasp-predicting parts from grasping experience," in *ICRA*, 2013, pp. 601–608.

[13] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," in *ICRA*, 2011, pp. 3304–3311.

[14] C. Sun, Y. Yu, H. Liu, and J. Gu, "Robotic grasp detection using extreme learning machine," in *ROBIO*. IEEE, 2015, pp. 1115–1120.

[15] L. Trottier, P. Giguère, and B. Chaib-draa, "Sparse dictionary learning for identifying grasp locations," in *WACV*, 2017.

[16] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, p. 1687814016668077, 2016.

[17] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with largescale data collection," in *International Symposium on Experimental Robotics*, 2016.

[18] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *NIPS*. MIT Press, 2006, pp. 1209–1216.

[19] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *ICRA*. IEEE, 2010, pp. 5062–5069.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[21] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *arXiv preprint arXiv:1603.05027*, 2016.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015, pp. 1026–1034.

[26] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.