GraspNet: An Efficient Convolutional Neural Network for Real-time Grasp Detection for Low-powered Devices

Umar Asif[1], **Jianbin Tang**[1], **Stefan Harrer**[1]
[1] IBM Research Australia
umarasif@au1.ibm.com, jbtang@au1.ibm.com, sharrer@au1.ibm.com

Abstract

Recent research on grasp detection has focused on improving accuracy through deep CNN models, but at the cost of large memory and computational resources. In this paper, we propose an efficient CNN architecture which produces high grasp detection accuracy in real-time while maintaining a compact model design. To achieve this, we introduce a CNN architecture termed *GraspNet* which has two main branches: **i**) An encoder branch which downsamples an input image using our novel *Dilated Dense Fire (DDF) modules* – squeeze and dilated convolutions with dense residual connections. **ii**) A decoder branch which upsamples the output of the encoder branch to the original image size using deconvolutions and fuse connections. We evaluated GraspNet for grasp detection using offline datasets and a real-world robotic grasping setup. In experiments, we show that GraspNet achieves competitive grasp detection accuracy compared to the state-of-the-art computation-efficient CNN models with real-time inference speed on embedded GPU hardware (Nvidia Jetson TX1), making it suitable for low-powered devices.

1 Introduction

Grasp detection is a visual recognition task in which the objective is to detect graspable regions in the images of objects in the environment [Lenz *et al.*, 2015]. The detected grasps are then used for high-level tasks (e.g., trajectory planning) in order to interact with the objects. Thus, grasp detection is a critical step as the subsequent steps of the grasping pipeline are dependent on the coordinates calculated in this step. However, grasp detection in real-world is challenging due to the factors such as: variations in the spatial layouts of the objects, changes in camera viewpoints, partial occlusions, background clutter, and the requirements of low computational complexity and fast runtime. With the recent advancements in deep learning, methods such as [Levine *et al.*, 2016; Redmon and Angelova, 2015] have demonstrated state-of-the-art grasp detection accuracies on the Cornell grasp dataset [Lenz *et al.*, 2015]. The success of these methods is largely attributed to the availability of a large amount of labelled training data, and typically large convolutional neural network (CNN) architectures (e.g., millions of trainable parameters of the model of [Lenz *et al.*, 2015]). However, large CNN models have high computational requirements which restrict their usage in applications with limited computational resources. On the other side, smaller CNN architectures (e.g., [Iandola *et al.*, 2016; Howard *et al.*, 2017]) focus on achieving real-time speed (by reducing network parameters), but at the expense of lower recognition accuracy. Real-time robotic applications demand both high accuracy and fast inference speed for reliable operations. Thus, it is critical to design CNN models which achieve the best accuracy within a certain computational allowance. In this paper, we propose a novel CNN model (termed *GraspNet*) which achieves competitive grasp detection accuracy and real-time inference speed on embedded GPU hardware compared to the state-of-the-art CNN models. In summary, the contributions of this work are as follows:

1) We formulate the task of grasp detection as pixel-wise labeling termed *grasp affordance segmentation* (Sec. 3) which segments graspable regions on the surfaces of objects. To achieve this, we propose a novel CNN architecture termed *GraspNet* (Sec. 4) which produces pixel-level labeling of graspable regions using RGB-D images.

2) The core building component of GraspNet is our novel convolution block termed *Dilated Dense Fire (DDF)* module (Sec. 4.1) which consists of squeeze and dilated convolutions interconnected through dense residual connections. The squeeze and dilated convolutions enable the network to aggregate multi-scale contextual information while maintaining a compact design. The proposed dense residual connectivity enables deep supervision throughout the network by reusing learned features across multiple layers, and thus reduces over-fitting.

3) We evaluated GraspNet for grasp detection using challenging RGB-D object datasets (Sec. 5.1) where we show that GraspNet is highly efficient in terms of fewer training parameters, faster inference speed, and competitive grasp detection accuracy compared to the state-of-the-art computation-efficient CNN models (Sec. 5). We also demonstrate GraspNet for grasp detection on a low-powered embedded GPU (Nvidia Jetson TX1) in a real-world robotic grasping setup (Sec. 6).

2 Related Work

We briefly review state-of-the-art related work on grasp detection and memory-efficient CNN models.

2.1 Grasp Detection

Recent research in robotic grasping has largely focused on performing grasp detection using deep learning [Kumra and Kanan, 2017]. In this context, methods such as [Redmon and Angelova, 2015; Asif *et al.*, 2017b] focused on predicting grasp rectangles (defined by its position, orientation, width and height) directly in images. One challenge with these methods is the availability of a large volume of labelled training data, which can be very expensive and time consuming to generate in real-world. To overcome this challenge, an alternative approach was presented in [Levine *et al.*, 2016], where the authors generated training data using a physics simulation engine [Levine *et al.*, 2016] to learn grasp scores. Recently, the work of [Johns *et al.*, 2016] presented another data generation approach, where reinforcement learning was used to learn a form of visual servoing by testing grasps on a real robot. However, their approach required several weeks of training using multiple robots working in parallel. Therefore, their approach lacks scalability and ease of deployment for low-powered embedded hardware. While most of the discussed approaches rely on models sufficiently trained on large amounts of training data to achieve high recognition accuracies, we propose an efficient CNN architecture which achieves high grasp detection accuracy in real-time and does not suffer over-fitting when trained from scratch using limited training data.

2.2 Computation-efficient CNN Models

In the past few years, the research in deep neural networks has largely focused on the development of efficient CNN architectures for deployment on embedded devices. In this context, one stream of work focused on reducing model size after training. For instance, the methods in [Li *et al.*, 2016; Hubara *et al.*, 2016] proposed to prune redundant weights or quantize weights during or after training. Another stream of work focused on exploring computationally efficient variants of traditional CNN architectures. In this context, SqueezeNet [Iandola *et al.*, 2016] introduced an architecture which achieves classification accuracy equivalent to the well-known AlexNet [Krizhevsky *et al.*, 2012] with a significant reduction in training parameters (by almost 50X). ResNet [He *et al.*, 2016] proposed a bottleneck structure in terms of residual blocks (where a non-linear transformation of the input and its identity are combined by means of skip connections), and achieved superior classification accuracy. Recently, DenseNets [Huang *et al.*, 2017] introduced an architecture which iteratively concatenates outputs from previous layers, and achieved higher recognition accuracy compared to the conventional CNN models. However, the iterative growth of feature channels throughout the network in DenseNets demands high computational resources and produces low inference speeds (especially on embedded hardware). In this paper, we propose a CNN architecture which is optimized for fast inference and high accuracy. Our CNN model is built upon the ideas of squeeze [Iandola *et al.*, 2016] and dilated [Yu and Koltun, 2016] convolutions interconnected through dense [Huang *et al.*, 2017] and residual [He *et al.*, 2016] connections within an encoder-decoder architecture [Shelhamer *et al.*, 2017]. In experiments, we show that our CNN model outperforms the state-of-the-art computation-efficient CNN models in terms of grasp detection accuracy and inference speed.

3 Proposed Grasp Affordance Segmentation

Given an input image $\mathcal{I} \in \mathbb{R}^{C\times W\times H}$, where C, W, and H, represent the number of channels, width, and height of the input, we define its corresponding labeling by $\mathcal{Y} \in \mathbb{R}^{n_c\times W\times H}$, where $n_c = 2$ represents the number of classes (grasp affordance and background). We train our CNN model in a fully supervised manner by minimizing the cross-entropy loss [Shelhamer *et al.*, 2017]. Specifically, our loss function for N_s labeled training images is given by:

$$\mathcal{L} = \frac{1}{N_s}\sum_{i=1}^{N_s}\mathbf{S}_i, \tag{1}$$

where $\mathbf{S}_i \in \mathbb{R}^{(n_c\times W\times H)}$ represents the CNN normalized scores map for a sampled image i. It is computed by applying a SoftMax to the output of the final layer of the network:

$$\mathbf{S}_i = -\log\frac{e^{W_{y_i}^T f(\mathrm{x}_i)+b_{y_i}}}{\sum_{j=1}^{n_c} e^{W_j^T f(\mathrm{x}_i)+b_j}}, \tag{2}$$

where, $f(\mathrm{x}_i)$ represents the output of the CNN layer for a sample x_i. The terms y_i, W and b represent the class label, weights, and bias of the corresponding layer, respectively. To generate robotic grasps from the predicted scores ($\mathbf{S}$), we first generate a segmentation mask $\mathcal{M} = \mathrm{argmax}_y(\boldsymbol{S})$. Next, we compute the major and the minor axes of the largest blob in $\mathcal{M}$. Finally, we compute a 5-dimensional grasp $\boldsymbol{R}$, given by:

$$\boldsymbol{R} = [R_x, R_y, R_w, R_h, R_\theta], \tag{3}$$

where R_x and R_y represent the centroid of the grasp given by the centroid of the blob. The width of the grasp (R_w) is equal to the length of the blob along its minor axis plus 40 pixels. The height of the grasp (R_h) is set to 30 pixels. The orientation of the grasp (R_θ) is given by the angle between the x-axis and the major axis of the blob as shown in Fig. 2-C. Given the grasp representation $\boldsymbol{R}$ for a target object, the robotic arm (calibrated with the camera) uses the grasp position (R_x, R_y) and the grasp orientation R_θ to position and orient its gripper, respectively, to grasp the target object.

4 Proposed CNN Model - GraspNet (Fig. 1)

Fig. 1 shows the overall architecture of our CNN model which consists of two main branches: **i**) An encoder branch (Fig. 1-A) which takes an image as input (shown in Fig. 1-E), passes it through our novel *Dilated Dense Fire* (DDF) modules, and generates down-sampled feature maps. **ii**) A decoder branch (Fig. 1-B) which takes the down-sampled output of the encoder branch, passes it through deconvolution layers, and generates prediction maps of the size of the input image as shown in Fig. 1-F.

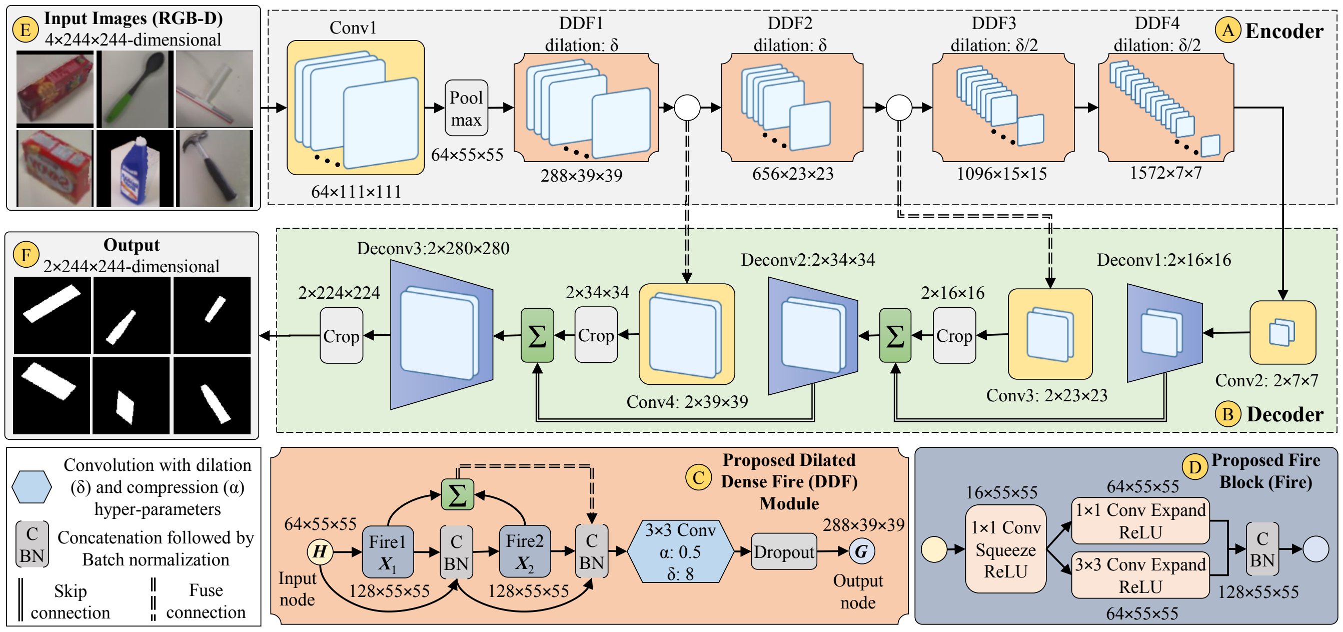


Figure 1: Overview of our GraspNet architecture for grasp affordance segmentation. It consists of two main branches: **i)** An encoder branch (A) which downsamples the input image using the proposed Dilated Dense Fire (DDF) modules (C). **ii)** A decoder branch (B) which upsamples the output of the encoder branch to the input image size using deconvolution layers and skip connections.

4.1 Proposed Dilated Dense Fire Module (Fig. 1-C)

Fig. 1-C shows the architecture of our DDF module which consists of two Fire blocks interconnected with dense residual connections, a convolution layer with an integrated channel-compression hyper-parameter α and a dilation hyper-parameter δ, and dropout [Srivastava *et al.*, 2014]. The Fire block of our DDF module is an enhanced version of the Fire block of [Iandola *et al.*, 2016] which consists of a squeeze convolution layer (with 1×1 filters) and an expand layer (with 1×1 and 3×3 filters). We enhance the effectiveness of the squeeze convolutions of [Iandola *et al.*, 2016] by incorporating Batch-Normalization (BN) [Ioffe and Szegedy, 2015] at the output of the Fire block as shown in Fig. 1-D. This enables the network to reduce the bias-shift effect during the training process, and produces highly discriminative feature maps without sacrificing the low-parameter design.

Proposed Dense Residual Connectivity

To increase information flow between layers in the network, we combine the outputs from the Fire blocks in a feed-forward fashion through element-wise summation and concatenation as shown in Fig. 1-C. Let $\boldsymbol{X}_l \in \mathbb{R}^{N_f \times W_f \times H_f}$ represents output of the l^{th} Fire block, where N_f, W_f, and H_f, represent the depth, width, and height of $\boldsymbol{X}_l$, respectively. For an input $\boldsymbol{H}$, the output of a DDF module $\boldsymbol{G} \in \mathbb{R}^{N_g \times W_g \times H_g}$ is given by:

$$\boldsymbol{G} = \mathcal{F}([\boldsymbol{H}, \boldsymbol{X}_1, \boldsymbol{X}_2, (\boldsymbol{X}_1 + \boldsymbol{X}_2)]), \tag{4}$$

where $[\cdot\ \cdot\ \cdot]$ represents concatenation and $+$ denotes an element-wise summation. The term $\mathcal{F}(\cdot)$ represents a non-linear transformation function composed of a 3×3 convolution followed by a rectified linear unit (ReLU). Our proposed dense residual connectivity has several advantages including: **i)** It increases variation in the input of subsequent layers leading to an implicit deep supervision. **ii)** The reuse of features learned by different layers enforces a regularizing effect on the objective function and the integrated dropout block reduces over-fitting.

Proposed Network Compression

To improve model compactness and achieve a small memory footprint, we reduce the number of feature channels and the spatial resolution of the feature maps at each DDF module of the network. For this, we introduce a 3×3 convolution layer (at the end of a DDF module as shown in Fig. 1-C) with two integrated hyper-parameters. A channel-compression parameter α and a dilation parameter δ which control the depth and the spatial resolution of the output feature maps of the DDF module, respectively. For a given $\alpha \in (0.1, 0.5, 1.0)$ and $\delta \in (4, 8, 16)$, the output of a DDF module becomes $\boldsymbol{G} \in \mathbb{R}^{\alpha N_g \times (W_g - 2\delta) \times (H_g - 2\delta)}$. The default values for α and δ are set to 0.5 and 8, respectively. In experiments (Sec. 5.5), we show the trade off between memory size and accuracy of the proposed GraspNet for different values of α and δ.

4.2 Implementation

We train GraspNet in an end-to-end manner using $4 \times 224 \times 224-$dimensional RGB-D images and their corresponding $2 \times 224 \times 224-$dimensional grasp affordance masks. The encoder branch starts with a convolution layer $Conv1$, followed by max-pooling, and 4 DDF modules ($DDF1 - DDF4$) as shown in Fig. 1-A. Each DDF module contains two Fire blocks. The depths of the Fire block-pairs in $DDF1-DDF4$ are 128, 256, 384, and 512, respectively. The decoder branch uses a DAG topology [Shelhamer *et al.*, 2017] with 3 deconvolution layers which upsample the feature maps as shown in Fig. 1-B. Further fuse connections are defined from the encoder to the decoder branch to utilize fine-grained information from the early layers during the upsampling. The layers to be fused are first aligned through crop operations, and then their features are combined through element-wise sum as shown in Fig. 1-B. Our implementation is based on the Caffe library [Jia *et al.*, 2014]. Training was performed by Stochastic Gradient Descent (SGD) with a batch size of 16, a weight decay of 10e-4, and momentum of 0.9 for 300K iterations. The initial learning rate was set to 1e-3, and divided by 10 at 50% and 75% of the total number of iterations.

5 Experiments

5.1 Datasets

GraspSeg Dataset

To evaluate GraspNet for grasp affordance segmentation, we introduce a new object grasp dataset named *GraspSeg*. It provides 33,188 RGB-D images of 42 object instances categorized into 15 different classes as shown in Fig. 2-A. The dataset also provides 6896 RGB-D images of indoor scenes for testing. The test images contain multiple objects arranged in different layouts as shown in Fig. 2-B. The ground truth in GraspSeg is available in the form of pixel-wise annotations for grasp affordances and object segmentations as shown in Fig. 2-D. We annotated the dataset using an extended version of the scene labeling framework of [Asif *et al.*, 2017a] and [Asif *et al.*, 2016]. For evaluation, we used the *object-wise splitting* strategy of [Lenz *et al.*, 2015], which splits object instances randomly into train and validation subsets (i.e., the training set and the validation set do not share any images from the same object). This splitting strategy evaluates how well the model generalizes to objects which were not known by the model during training. GraspSeg dataset is challenging for grasp detection. This is because, the training images of the dataset contain single objects without background information as shown in Fig. 2-A. However, the test images contain multiple objects per image, where the objects appear in close proximities of each others, producing partial occlusions as shown in Fig. 2-B.

Cornell Grasp Dataset

We also evaluate GraspNet for grasp detection on the popular Cornell grasp dataset [Lenz *et al.*, 2015], which contains 885 RGB-D images of 240 objects. The ground-truth is available in terms of grasp-rectangles. For evaluation, we used the object-wise splitting and the image-wise splitting criteria used in previous works [Lenz *et al.*, 2015]. The object-wise splitting splits the object instances randomly into train and validation subsets. The image-wise splitting splits all the images of the dataset randomly into five folds. The image-wise splitting strategy evaluates how well the model generalizes to new positions and orientations of known objects.

5.2 Evaluation Metrics

Grasp Detection

We evaluate GraspNet for grasp detection using the "rectangle-metric" proposed in [Jiang *et al.*, 2011]. A grasp is considered to be correct if: **i)** the difference between the predicted grasp angle and the ground-truth is less than 30°, and **ii)** the Jaccard index of the predicted grasp and the ground-truth is higher than 25%. The Jaccard index for a predicted rectangle $\boldsymbol{R}^*$ and a ground-truth rectangle $\boldsymbol{R}^g$ is defined as:

$$J(\boldsymbol{R}^g, \boldsymbol{R}^*) = \frac{|\boldsymbol{R}^g \cap \boldsymbol{R}^*|}{|\boldsymbol{R}^g \cup \boldsymbol{R}^*|}. \tag{5}$$

Affordance Segmentation

We also evaluate GraspNet for affordance segmentation on the GraspSeg dataset using the mean frequency weighted Intersection over Union ($f.w.IU$) metric. It is defined in [Shelhamer *et al.*, 2017] as:

$$f.w.IU = \frac{1}{\sum_k t_k} \sum_i \frac{t_i \cdot n_{ii}}{(t_i + \sum_j n_{ij} - n_{ii})}, \tag{6}$$

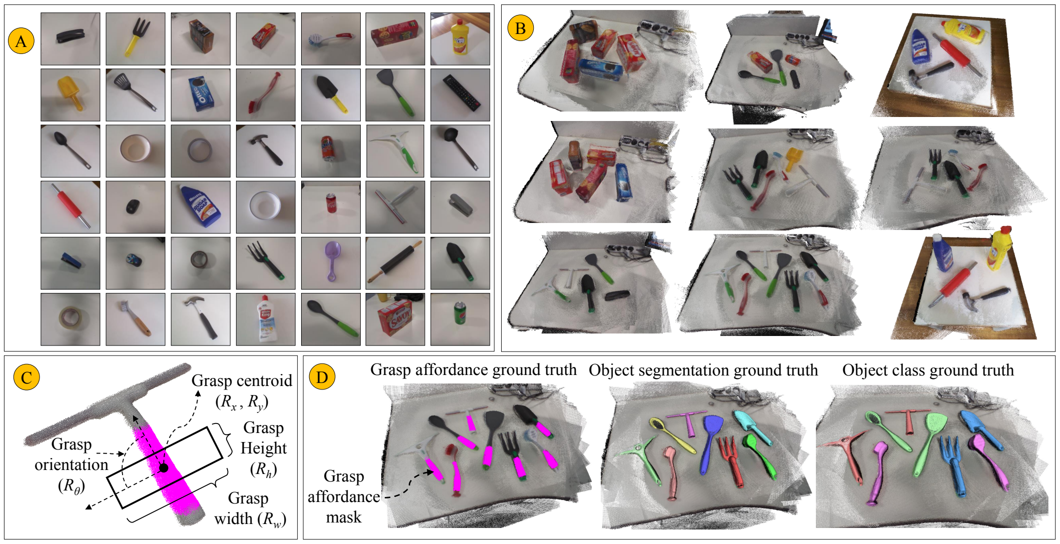
where, n_{ij} represents the number of pixels of class i predicted to belong to class j, and t_i represent the total number of pixels i in the ground truth segmentation.

5.3 Evaluation on the GraspSeg Dataset

First, we compare our model with three variants of SqueezeNet [Iandola *et al.*, 2016]: **i)** SqueezeNet (vanilla), **ii)** SqueezeNet (Residual) with residual connections, **iii)** SqueezeNet^1 with dilated convolutions, and **iv)** SqueezeNet^2 with residual connections and dilated convolutions. The results of these experiments are reported in Table 1 which shows that GraspNet clearly outperforms SqueezeNet with improvements of 17% and 16% compared to SqueezeNet^1 and SqueezeNet^2 in the grasp accuracy, respectively. We attribute these improvements to the proposed DDF modules (Sec. 4.1), where the concatenation of features through dense residual connections maximizes the variation in the information between the layers of the network. This enables GraspNet to learn more discriminative features compared to SqueezeNet [Iandola *et al.*, 2016], where the amount of information is limited between the layers of the network.

Next, we compare GraspNet with MobileNet [Howard *et al.*, 2017] and ENet [Paszke *et al.*, 2016]. Table 1 shows that our GraspNet achieved 20% and 7% higher grasp accuracy, and 4× and 6× faster inference speed compared to the ENet [Paszke *et al.*, 2016] and MobileNet [Howard *et al.*, 2017] models, respectively. **Finally**, we compare our model to the 50-layer ResNet [He *et al.*, 2016] and SegNet [Badrinarayanan *et al.*, 2017]. Table 1 shows that GraspNet improved the grasp accuracy by 6% and 9%, while being 12× and 15× smaller, 6× and 7× less compute-intensive, and 3× and 4× faster compared to the ResNet [He *et al.*, 2016] and the SegNet [Badrinarayanan *et al.*, 2017] models on Jetson TX1, respectively. These improvements are attributed to the proposed network compression (Sec. 4.1) which effectively increases the receptive field of the feature maps (and preserves the contextual information) across the network at the expense of a linear growth of the parameters.

Table 1 also shows that GraspNet consistently outperformed the compared models in terms of $f.w.IU$ metric. Fig. 3 shows qualitative results of affordance segmentation on the GraspSeg dataset. The results show accurate segmentations with respect to the ground truths. For example, grasp affordances for *spoon*, *garden tool*, and *food box* are accurately segmented. The proposed dense connectivity in the


A
B
C
Grasp centroid (R_x, R_y)
Grasp Height (R_h)
Grasp orientation (R_θ)
Grasp width (R_w)
D
Grasp affordance ground truth
Object segmentation ground truth
Object class ground truth
Grasp affordance mask

Class name	No. of images	Object Instances
Soda can	2280	3
Food box	6138	5
Stapler	2048	2
Cleaning brush	2100	3
Spoon	3618	5
Tape	1082	3
Viper	1082	3
Remote controller	1260	1
Hammer	300	2
Bowl	840	2
Mouse	2080	2
Garden tool	3040	4
bottle	4240	3
Rolling pin	2078	2
Beach tool	966	2
Total	**33188**	**42**

Figure 2: An overview of the proposed GraspSeg dataset. The ground truth in GraspSeg is available in terms of pixel-wise annotations for grasp-affordance segmentation and object segmentation as shown in D. The dataset is challenging because training images (shown in A) contain single objects, whereas test images contain multiple objects with partial occlusions and background clutter (shown in B).

Model	Affordance $f.w.IU$ (%)	Grasp Accuracy (%)	Parameters (millions)	Model size (MB)	Memory (MB)	Tesla K80 Time (ms)	Jetson Time (ms)
[Iandola *et al.*, 2016] SqueezeNet (vanilla)	50.5	59.5	0.94	3.6	**260**	**18**	**119**
[Iandola *et al.*, 2016] SqueezeNet (Residual)	55.9	61.9	0.95	3.7	274	18	126
SqueezeNet[1] (Dilated)	58.3	69.4	1.08	4.2	274	20	140
SqueezeNet[2] (Dilated+Residual)	59.6	71.3	1.08	4.2	278	20	141
[Howard *et al.*, 2017] MobileNet	58.2	66.6	3.21	12.4	384	118	826
[Krizhevsky *et al.*, 2012] AlexNet	59.8	67.8	56.8	22.7	451	25	175
[Badrinarayanan *et al.*, 2017] SegNet	64.1	77.7	29.4	112.3	699	74	518
[He *et al.*, 2016] ResNet50	60.7	80.5	23.5	90.0	601	62	434
[Paszke *et al.*, 2016] ENet	68.3	81.7	**0.41**	**1.4**	523	97	680
GraspNet ($\alpha = 0.5, \delta = 8$)	**73.5**	**87.3**	3.71	7.2	425	19	133

Table 1: Comparison of GraspNet and other CNN models in terms of average grasp detection accuracy, average affordance segmentation accuracy, number of model parameters, model size, model memory consumption, and model inference time on the GraspSeg dataset.

encoder branch maximizes the variation in the information between the layers of the network. Consequently, GraspNet learns highly discriminative features and recovers fine structures (e.g., object contours) more accurately than the compared methods.

5.4 Evaluation on the Cornell Grasp Dataset

Table 2 shows a comparison of grasp accuracy produced by our model and other methods on the Cornell grasp dataset [Lenz *et al.*, 2015]. The results show that our model outperformed all the compared methods with an inference speed of only 24 ms per image. These improved results show that the proposed deep residual feature-fusion through DDF modules (Sec. 4.1) produces feature representations which are highly effective for the direct grasp regression using limited training images of the Cornell grasp dataset [Lenz *et al.*, 2015]. Among common failure cases, we observed that our model failed to predict the correct orientation of the grasp with respect to some objects as shown in Fig. 4.

Method	Accuracy (%) Object wise	Image wise	Time
[Jiang *et al.*, 2011] Fast search	58.3	60.5	50 sec
[Lenz *et al.*, 2015] Deep learning	75.6	73.9	13.5 sec
[Redmon *et al.*, 2015] MultiGrasp	87.1	88.0	76 ms
[Asif *et al.*, 2017] Random Forests	87.5	88.2	-
[Kumra *et al.*, 2017] Deep ResNets	88.9	89.2	103 ms
GraspNet	**90.2**	**90.6**	**24 ms**

Table 2: Grasp detection results on the Cornell grasp dataset.

5.5 GraspNet Design Space Exploration

To explore the design space of GraspNet, we performed experiments which highlight the impact of CNN architectural choices on model size and accuracy. GraspNet has 8 DDF modules, where each DDF module has two Fire blocks and two hyper-parameters α and δ which control the number of

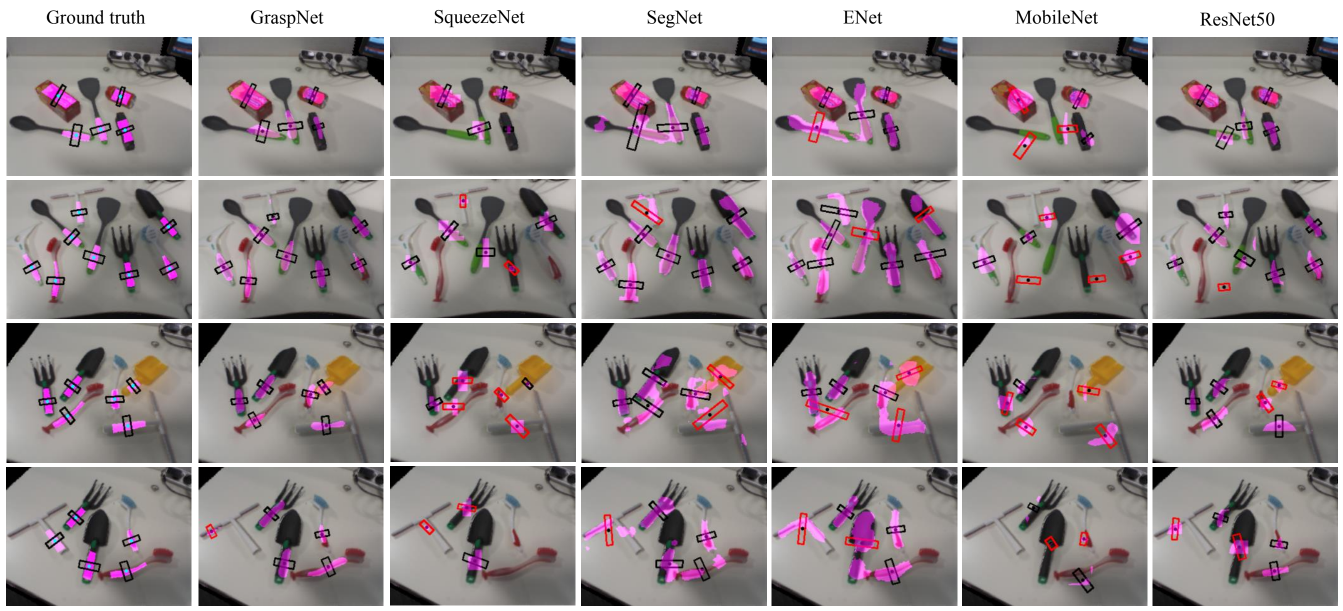


Figure 3: Qualitative results of grasp detection (rectangles shown in black) and grasp affordance segmentation (regions shown in magenta) on the GraspSeg dataset using the proposed GraspNet and state-of-the-art CNN models. Failure cases are highlighted in red rectangles.

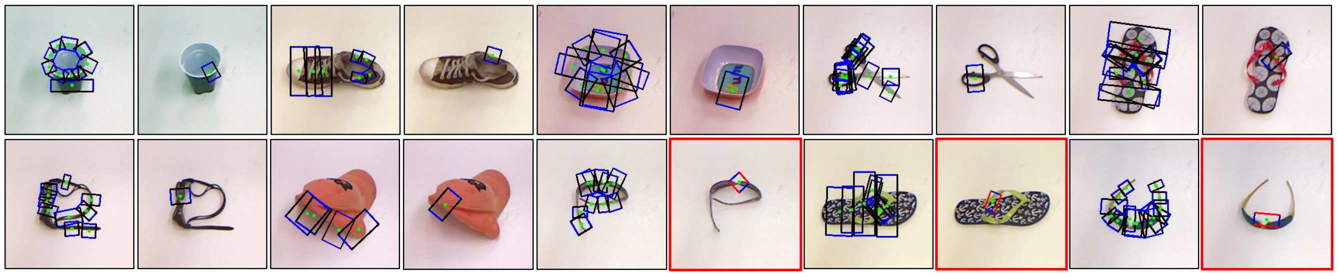


Figure 4: Grasps predicted by our model and the corresponding ground-truths of the Cornell grasp dataset. Failure cases are shown in red.

feature channels and the spatial resolution of the feature maps produced by the DDF modules, respectively. **First**, we investigate the effect of the channel-compression parameter α on model size and accuracy. In these experiments, we fixed the dilation parameter δ to 8. Table 3 shows that increasing α beyond 0.5 increases the accuracy from 87.3% with a 7.2MB model to 89.6% resulting in a 14.2MB model. **Next**, we investigate the *importance of spatial resolution in the network* by evaluating GraspNet with different values of δ. From Table 3, we see that reducing δ by half produced improvement in the accuracy from 87.3% to 90.5%. Table 3 also shows that when δ was increased to 16, GraspNet achieved a considerable boost in the inference speed at the expense of a small drop in the grasp accuracy. It is conceivable that a more extensive parameter search would give better insights into the design choices for the optimal trade-off between the accuracy, the model size and the inference speed.

Model	Grasp accuracy	Model size (MB)	Runtime (ms)
GraspNet ($\alpha = 0.1, \delta = 8$)	85.1	4.1	17
GraspNet ($\alpha = 0.5, \delta = 8$)	87.3	7.2	19
GraspNet ($\alpha = 1.0, \delta = 8$)	89.6	14.2	28
GraspNet ($\alpha = 0.5, \delta = 4$)	**90.5**	29.2	24
GraspNet ($\alpha = 0.5, \delta = 16$)	86.4	**3.8**	**13**

Table 3: Grasp accuracy, model size, and runtime of GraspNet for different values of its parameters α and δ on the GraspSeg dataset.

6 Robotic Grasping

We implemented GraspNet for grasp detection in real-world robotic grasping using Nvidia Jetson TX1 and our in-house robotic platform. In these experiments, the robot was tasked to pickup the objects placed within the workspace of the robot. To achieve this, we built an integrated framework for object recognition and grasp detection, where we deployed the Faster-RCNN model of [Ren *et al.*, 2015] to generate object proposals in the scene. A recognized object proposal is then fed to the proposed GraspNet to produce a grasp which is finally used by the robotic arm to grasp the target object. For these experiments, we performed more than 50 trials and evaluated the grasps based on force-feedback from the gripper. A grasp was considered successful if the robot raised and held the object in the air for 10 seconds (during which the

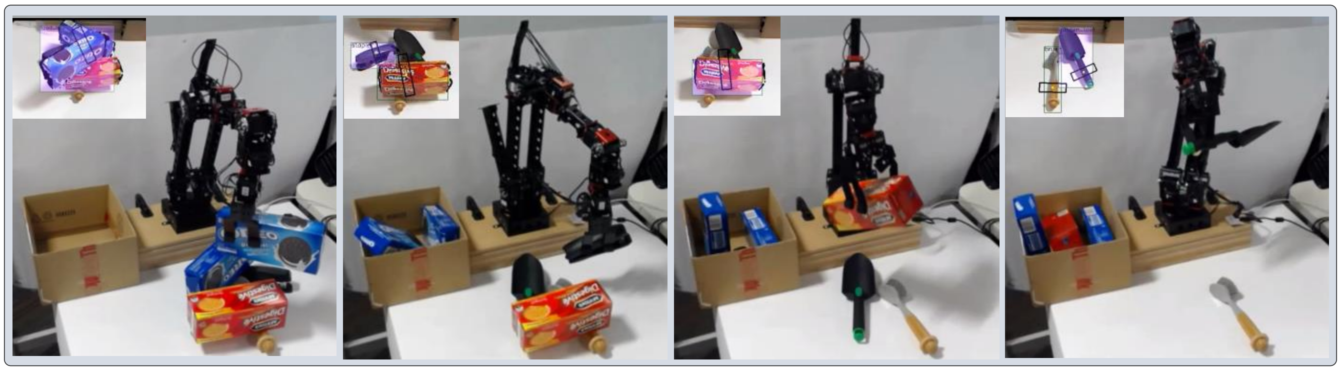


Figure 5: Real-world robotic grasping setup. An image acquired from Kinect is fed to Faster-RCNN [Ren *et al.*, 2015] to generate object proposals. These are fed to the proposed GraspNet to generate grasp inferences which are used by a robotic arm for robotic grasping. Both Faster-RCNN and GraspNet run on Nvidia Jetson X1 (see the video at: `https://youtu.be/bYEmAOImC90`).

grasp was confirmed through force feedback from the gripper). Fig. 5 shows some grasps from the real-world testing.

7 Conclusion and Future Work

We proposed a novel CNN architecture termed *GraspNet* which outperforms state-of-the-art computation-efficient CNN models in terms of grasp detection accuracy. To achieve this, we proposed a novel convolution block termed *Dilated Dense Fire (DDF) module*, which uses squeeze and dilated convolutions interconnected through dense residual connections and integrated compression hyper-parameters. In experiments, we show that GraspNet has a small memory footprint and achieves real-time inference speed on embedded GPU like Nvidia Jetson TX1. These attributes make GraspNet a highly suitable CNN model for embedded systems which can be deployed onto mobile robots. We also demonstrate the exploration of the design space of GraspNet to build smaller and faster models by trading off a reasonable amount of accuracy to reduce size and latency. For future work, we plan to extend the GraspNet architecture for simultaneous object recognition and grasp detection, to further reduce the overall latency of the recognition system, while maintaining a compact model design and real-time inference speed. We also plan to implement CRF-based post-processing steps to further improve the segmentation of grasp affordances.

References

[Asif *et al.*, 2016] Umar Asif, Mohammed Bennamoun, and Ferdous Sohel. Simultaneous dense scene reconstruction and object labeling. In *ICRA*, pages 2255–2262. IEEE, 2016.

[Asif *et al.*, 2017a] Umar Asif, Mohammed Bennamoun, and Ferdous Sohel. A multi-modal, discriminative and spatially invariant cnn for rgb-d object labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[Asif *et al.*, 2017b] Umar Asif, Mohammed Bennamoun, and Ferdous A Sohel. Rgb-d object recognition and grasp detection using hierarchical cascaded forests. *IEEE Transactions on Robotics*, 2017.

[Badrinarayanan *et al.*, 2017] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *PAMI*, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Howard *et al.*, 2017] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *CVPR*, 2017.

[Hubara *et al.*, 2016] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, pages 4107–4115, 2016.

[Iandola *et al.*, 2016] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[Jiang *et al.*, 2011] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *ICRA*, pages 3304–3311. IEEE, 2011.

[Johns *et al.*, 2016] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *IROS*, pages 4461–4468. IEEE, 2016.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[Kumra and Kanan, 2017] Sulabh Kumra and Christopher Kanan. Robotic grasp detection using deep convolutional neural networks. In *IROS*. IEEE, 2017.

[Lenz *et al.*, 2015] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *IJRR*, 34(4-5):705–724, 2015.

[Levine *et al.*, 2016] 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. *IJRR*, page 0278364917710318, 2016.

[Li *et al.*, 2016] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[Paszke *et al.*, 2016] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[Redmon and Angelova, 2015] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *ICRA*, pages 1316–1322. IEEE, 2015.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[Shelhamer *et al.*, 2017] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *PAMI*, 39(4):640–651, 2017.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Yu and Koltun, 2016] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.