EffiGrasp: Orchestrating Precision and Nimble Mastery in Grasp Detection

Shaochen Wang, Zhangli Zhou, Beihao Xia, Mingyu Cai, Ziyang Chen, Bin Li, and Zhen Kan

***Abstract*—Efficiency is a paramount concern in the field of grasp detection, particularly when applied to industrial manipulation scenarios. However, many existing neural network-based methods prioritize performance at the cost of large models and extensive computational resources. In this work, we propose a unified framework that achieves superior grasping detection accuracy while achieving *faster runtime*, *smaller model size*, and *lower computational budget* within a single system. Specifically, we introduce a model slimming technique tailored for robotic visual grasping, capable of accommodating various CNN architectures. This approach identifies and prunes insignificant feature channels during training, resulting in a lightweight model. Experimental results demonstrate that our approach reduces computational overhead by 63.9% with negligible impact on accuracy, and the model storage requirement is only 1/5th of the original model. Physical experiments on a Franka Emika Panda Robot System show outstanding performance in both single-object and cluttered environments. The code and pre-trained models will be available at https://github.com/WangShaoSUN/Prune-Grasp.**

I. Introduction

Over the past decade, the rise of deep learning has sparked a revolution across diverse domains. However, its widespread use has increased the demand for significant computational resources and storage capacity. In robotics, real-time grasp detection is critical. For instance, in industrial automation, robots must quickly detect and grasp moving objects on a production line, and any delays or inefficiencies can impact productivity.

In recent years, the integration of deep learning [1], [2] into robotics has gained prominence, with neural networks assuming pivotal roles in robot learning. As robotic manipulators become increasingly prevalent in industrial manufacturing, smart homes, and robotic assembly, addressing the challenges of low power consumption and compatibility with embedded devices has become imperative for deep learning-based robotic grasping applications. A significant hurdle in the adoption of deep learning technologies within robotics lies in the substantial demand for computational resources and adherence to real-time constraints. For instance, surgical manipulation and industrial assembly robots necessitate real-time performance, while domestic service robots must prioritize resource-efficient and low computational overhead solutions. Thus, the development of models that offer high performance with minimal computational footprint, real-time capabilities, and lightweight model sizes holds significant importance.

Convolutional neural networks (CNNs) have become the primary model in robotic visual grasp detection and manipulation. Lenz et al. [3] pioneered deep learning for grasp detection, achieving a 73.6% accuracy on the Cornell dataset, but with a lengthy 13.5s processing time per image. Recent efforts [4]–[7] have focused on enhancing grasping performance but often employ relatively large models, limiting real-time feasibility. While [7] leverages oriented anchor box mechanisms for grasp detection, the model parameters remain impractical for real-time deployment. In recent model architecture design, the shift from a few convolutional layers to hundreds of layers [8] has increased computational demands significantly. For instance, a 152-layer ResNet [8] for visual robotic manipulation entails over 60 million parameters and over 2×10^{10} floating-point operations (FLOPs), which is impractical for embedded platforms in robotics. There also exist works [9], [10], [11] focusing on lightweight model design and real-time performance for the robotic system. Morrison et al. [10] achieve a 7ms per image speed but at the cost of a lower 78.4% accuracy on the Cornell dataset. While these works contribute to lightweight and real-time robot learning, they often lack a comprehensive consideration of runtime, computational cost, and model size simultaneously. This work aims to delve deeper into these aspects.

Currently, there exist several challenges that hinder the deployment of deep neural networks in real-world robotic applications. i) Model size: The learning and representation capabilities of deep neural networks scale with the number of trainable parameters, often reaching millions. This demands substantial disk space for storage and significant memory for loading during inference. For instance, GPT-3 with 175 billion parameters, as exemplified by Brown et al. [12], poses practical difficulties when deploying it on an embedded human-robot interaction system. ii) Running-time: In practical robotic scenarios, the model's inference time directly impacts system performance. For instance, a neural network-based kinematic approach tailored for the Franka Emika Panda robot arm must exhibit rapid responsiveness, demanding stringent real-time performance. iii) Computation budget: Neural networks, particularly CNNs, are computationally intensive. In tasks involving visual inputs, CNNs may take several seconds

Shaochen Wang, Zhangli Zhou, Ziyang Chen, and Zhen Kan (corresponding author) are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China, e-mail: samwang@mail.ustc.edu.cn, zzl1215@mail.ustc.edu.cn, yy1220@mail.ustc.edu.cn , zkan@ustc.edu.cn.

Mingyu Cai is with the Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA, USA, e-mail: mingyucai@lehigh.edu.

Beihao Xia is with Huazhong University of Science and Technology, e-mail: xbh_hust@hust.edu.cn.

Bin Li is with the Department of Electronics Science and Technology, University of Science and Technology of China, Hefei 230026, China, e-mail: binli@ustc.edu.cn.

or even minutes to process a single image on a GPU-free machine. Broadly speaking, numerous notable research efforts [9], [11], [10] work towards the real-time aspects of robotics. However, a common limitation across these works is the lack of simultaneous consideration for performance, model parameters, computational overhead, and running time. Furthermore, many of these models are manually designed. Consequently, a central motivation driving our research is to establish a unified neural network-based real-time method for robotic tasks that effectively addresses these critical influencing factors.

In this paper, we introduce an efficient model slimming approach for real-time robotic grasping detection. Our method can be seamlessly integrated into existing robotic CNN architectures without requiring any hardware modifications for model inference. Beyond trainable parameters, the memory consumption of intermediate activation feature maps during model inference is also considered. We propose a channel-based slimming method that considers both activation feature maps and model parameters. For practical autonomous robotic manipulation systems, compared to the original model, our slimming version achieves a remarkable 77.6% reduction in model size and a 63.9% decrease in computational requirements, with nearly no performance degradation.

Our approach is highlighted in the following aspects: simplicity, modularity, and generalization. Simplicity: Only one forward and backward pass is required to prune and compress the model. Modularity: Our slimming module can be applied as a plugin suite to current CNNs-based grasp detection models. Generalization: The slimming model not only maintains accuracy and speed but also exhibits favorable generalization. In a nutshell, the contributions of this paper can be summarized in three folds:

- To the best of our knowledge, this is one of the first attempts to deliver a model slimming solution for real-world robotic applications, striking a balance between runtime, model size, and computational cost while preserving accuracy.
- We introduce a plug-and-play pruning strategy for current CNN-based grasping models, ensuring ease of deployment and committing to open-sourcing the system for the community.
- Detailed experimental results demonstrate that our slimming solution achieves a modest model size, efficient computational performance, and rapid inference speed.

II. Method

Problem Statement. Robotic visual grasping integrates grasp detection and manipulation. The grasp recognition module typically processes RGB or depth images as input and outputs viable grasp configurations. Both the accuracy and efficiency are critical for the entire grasping systems. However, empirical evidence from previous studies [13], [14] highlights the presence of numerous redundant parameters in neural networks, compromising real-time efficiency. Thus, our motivation is to identify and eliminate insignificant neurons in neural networks for robotic tasks, enabling the removal of low-contributing weights to create a lighter and faster model.

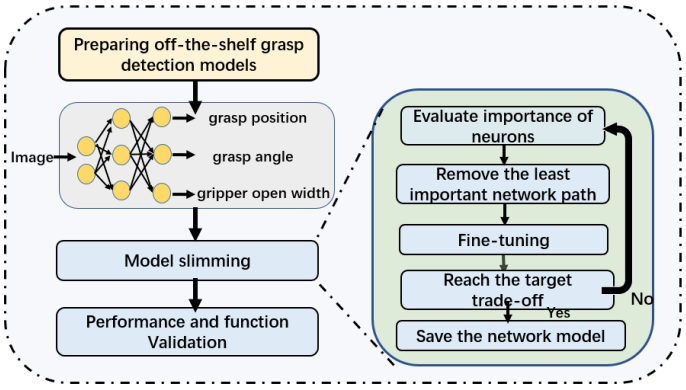


Fig. 1. The overview of the framework.

To investigate the impact of slimming on visual grasping performance, we consider a dataset $\mathcal{D}$ comprising image samples $\mathcal{I} = \{I_1, I_2, ..., I_n\}$, containing graspable objects and their corresponding grasp annotations $\mathcal{Y} = \{g_0, g_1, ..., g_n\}$. Each grasp annotation, denoted as g, is a 5-dimensional vector (x, y, θ, w, h), representing the grasping pose with center coordinates (x, y), orientation angle θ, and grasping bounding box dimensions (w, h).

The grasp detection model f is typically a neural network parameterized by weights $\mathcal{W}$, and $f(I_i)$ represents the predicted grasping configuration, including parameters like the grasping position, gripper's opening distance and rotation angle.

Given an input image I_i in the dataset $\mathcal{D}$ and its corresponding ground truth g_i, we define the total loss between grasp predictions and ground truth over the entire dataset as follows:

$$\mathcal{L}_{\mathcal{W}}(\mathcal{D}) = \sum_{i}^{n} \|f(I_i) - g_i\|^2. \tag{1}$$

In the slimming process, the objective is to obtain a function $\mathcal{F} : f_{\mathcal{W}} \rightarrow f'_{\mathcal{W}'}$, where f' has a similar model architecture and performance to f but with fewer parameters, denoted as $\mathcal{W}'$, which is a subset of $\mathcal{W}$. Let $\mathcal{L}_{\mathcal{W}'}(\mathcal{D})$ denote the loss over the parameter subset $\mathcal{W}'$ for all input images. The slimming process aims to identify a parameter subset that forms a sub-network while preserving the original model's performance as much as possible. To achieve this, we formulate the following constrained optimization problem:

$$\begin{aligned} &\arg \min_{\mathcal{W}' \in \mathcal{W}} \mathcal{L}_{\mathcal{W}'}(\mathcal{D}) \\ &\text{s.t.} \quad \|\mathcal{W}'\|_0 < B, \end{aligned} \tag{2}$$

where $\|\mathcal{W}'\|_0$ represents the number of non-zero parameters in the subset $\mathcal{W}'$, and B is a constrained hyper-parameter. Solving Eq. 2 allows us to find a pruned model with parameters $\mathcal{W}'$ that maintains similar performance while reducing model size and computational overhead. During this process, the parameters $\mathcal{W} = (w_1, b_1), (w_2, b_2), ..., (w_L, b_L)$ of the original model f are selectively chosen to minimize the objective in Eq. 2. Here, for generality, the parameter $(w_i, b_i) \in \mathcal{W}$

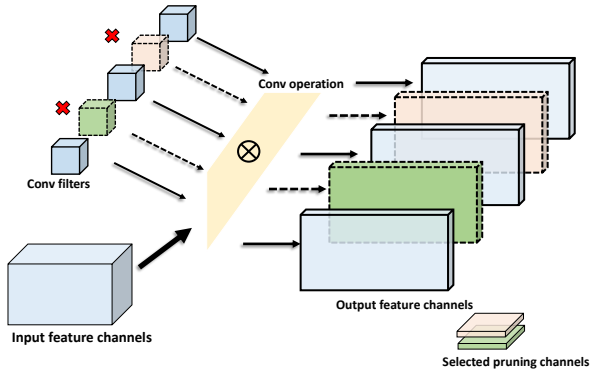


Fig. 2. The illustration of a channel-based pruning method, where the selected channel feature maps, along with the corresponding convolution kernels, are removed.

can represent an individual weight in a fully-connected layer, convolutional kernel, or even an entire convolutional filter.

To identify a suitable subset $\mathcal{W}'$ satisfying Eq. 2, one naive approach is to exhaustively prune each convolutional kernel and assess its impact on final performance. Intuitively, finding the optimal subset $\mathcal{W}'$ requires traversing all possible parameter combinations, necessitating $2^{|\mathcal{W}|}$ evaluations of the cost function. Unfortunately, even for a UNet-based grasping detection model with 3996 feature channels, obtaining an exact optimal pruning solution is computationally infeasible.

Slimming method. We frame slimming as an optimization problem, aiming to minimize the loss function $\mathcal{L}_{\mathcal{W}'}(\mathcal{D})$ while adhering to the constraint on the number of non-zero elements. However, this is a challenging non-convex optimization problem. In this work, our goal is to introduce a straightforward yet effective model slimming approach for robotic visual grasping. Specifically, we begin by assessing the impact on the loss function when pruning a specific convolutional filter from the original model, along with its immediate output.

For a specific convolution kernel c_i, let $\mathcal{L}_{\mathcal{W}}(\mathcal{D}, c_i = 0)$ denote the loss after pruning feature channel c_i. We define the difference in loss after removing c_i from the model as:

$$\Delta\mathcal{L}_{\mathcal{W}/c_i} = \left|\mathcal{L}_{\mathcal{W}}(\mathcal{D}) - \mathcal{L}_{\mathcal{W}}(\mathcal{D}, c_i = 0)\right|, \tag{3}$$

where quantifies the loss change on the entire dataset $\mathcal{D}$ when we eliminate feature channel c_i along with its corresponding convolutional filter. Since obtaining the exact loss difference is challenging, we apply a Taylor expansion to estimate it after removing a specific convolutional feature map c_i. Recall that for a function $f(x)$, its Taylor expansion at x_0 is given by $f(x) = f(x_0) + \frac{f'(x_0)}{1!} + \ldots + \frac{f^{(n)}(x_0)}{n!} + R_n(x)$ where R_n is a remainder. Analogously, we perform a first-order Taylor expansion to approximate $\mathcal{L}_{\mathcal{W}}(\mathcal{D}, c_i = 0)$. Assuming that the model parameters are independent of each other, the impact of pruning one convolutional filter on the loss function can be approximated as:

$$\mathcal{L}_{\mathcal{W}}(\mathcal{D}, c_i = 0) = \mathcal{L}_{\mathcal{W}}(\mathcal{D}) + \frac{\partial\mathcal{L}_{\mathcal{W}}}{\partial c_i}c_i + R_1(c_i), \tag{4}$$

where the remainder $R_1(c_i) = \frac{\partial^2\mathcal{L}_{\mathcal{W}}}{\partial({c_i}^2=\zeta)}\frac{{c_i}^2}{2}$, and ζ is a real number between 0 and c_i. Consequently, by substituting Eq. 4 into Eq. 3, we can estimate the ultimate impact of pruning a specific feature map on the final loss function:

$$\Delta\mathcal{L}_{\mathcal{W}/c_i} = \left|\frac{\partial\mathcal{L}_{\mathcal{W}}}{\partial c_i}c_i + R_1(c_i)\right|. \tag{5}$$

In practice, we typically omit the term $R_1(c_i)$ for simplification. Eq. 5 provides a criterion for assessing the significance of a featured channel by quantifying its gradient information. Consequently, the importance of convolutional kernels in each layer becomes directly evident through the generated activation maps. To compute $\Delta\mathcal{L}_{\mathcal{W}/c_i}$,, it is convenient to accumulate the gradient of the loss function $\mathcal{L}_{\mathcal{W}}$ concerning each element in the activation channel map of that layer during the back-propagation process. This entire computation process is defined as:

$$\Delta\mathcal{L}_{\mathcal{W}/c_i} = \frac{1}{N}\sum_{j}^{N}\frac{\partial\mathcal{L}_{\mathcal{W}}}{\partial c_{i,j}}, \tag{6}$$

where N represents the length of the vectorized activation feature map. In this context, $\frac{\partial\mathcal{L}_{\mathcal{W}}}{\partial c_{i,j}}$ signifies the gradient information associated with each entry in the convolutional activation map.

The approach above provides a way to approximate the minimal change in the loss function by pruning the gradient flat feature map. Our aim is to develop a pruning method, for robotic visual tasks, that can reveal unimportant channels and removes them from the original network to eliminate their associated weights. Working from the entire set of parameters $\mathcal{W}$, we iteratively evaluate and gradually prune the less critical parameters.

Below, we analyze the computational savings achieved through channel slimming. Our discussion focuses on convolutional layers, but the pruning principle applies similarly to other layers. In the context of a 2D kernel operating on an input plane referred to as the input feature map, the resulting output is known as the activation map. Convolution filters applied to input feature maps produce relevant outputs, serving as inputs to subsequent convolutional layers. Each layer in the model contains individual feature maps denoted as x_l, where l ranges from 1 to L. In layer i, a convolution operation with parameterized kernels acts on the current feature map input, given by:

$$x_i = ReLU(x_{i-1} * W_i + b_i), \tag{7}$$

where $ReLU$ represents the Rectified Linear Unit activation function. Let $x_i \in \mathbb{R}^{c_i \times h_i \times w_i}$ denote the feature map generated at layer i, where c_i denotes the number of feature channels, and h_i and w_i represent the height and width of the input feature map.

The convolution parameters for each layer are represented as a 4-dimensional tuple: (c_i, c_{i+1}, k, k), where c_{i+1} denotes the number of filters, and k represents the kernel size. The convolutional layer transforms the current input plane $x_i \in \mathbb{R}^{c_i \times h_i \times w_i}$ into the output activation $x_{i+1} \in \mathbb{R}^{c_{i+1} \times h_{i+1} \times w_{i+1}}$ by applying a set of filters $\mathcal{F} \in \mathbb{R}^{c_{i+1} \times k \times k}$. When a featured channel is pruned, its associated convolutional filter in the previous layer is also removed, resulting in a savings of $c_i k^2 h_{i+1} w_{i+1}$ operations. By analogy, the corresponding

kernel in the next layer needs to be eliminated as well. For instance, in Fig. 2, two less important activation channels highlighted in green and pink are pruned, along with their corresponding convolutional kernel weights.

Following slimming, the resultant model demonstrates enhanced efficiency in terms of runtime memory, model size, and computational cost when compared to the original oversized model. As highlighted in [13], removing a small proportion of insignificant weights can enhance performance and lead to improved generalization. Nevertheless, it's worth noting that in certain instances, pruning may adversely affect model performance. However, such performance degradation can often be mitigated through re-training the pruned network.

In practice, for the purpose of achieving a streamlined representation conducive to high-precision and real-time grasp detection, our grasp model employs three grasp heads, collectively represented as $\mathbf{G} = Q, W, \Theta \in \mathbb{R}^{3\times W\times H}$. Here, Q, W, Θ are three heat maps of the same size as the input image. Q denotes the grasp score map, with each pixel indicating the success rate of grasping at that specific point. W represents the gripper opening width, and Θ signifies the corresponding rotation angle of grasping at each point. The optimal grasp can be determined by searching the grasp score map, where $g^* = \arg\max Q$. It is worth pointing out that the proposed channel-oriented pruning approach can be adaptable to current CNN models used in robotics.

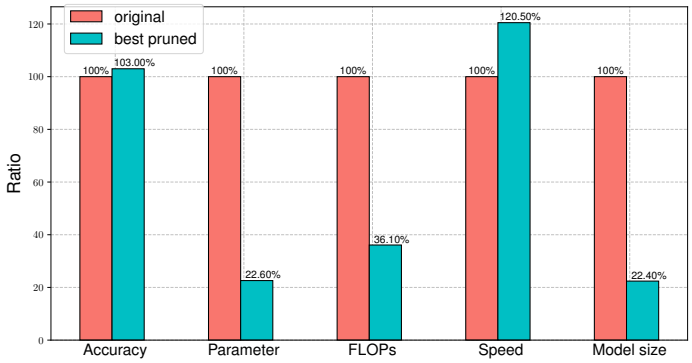
III. Experiment

In this section, we present comprehensive experimental results to validate the effectiveness of our model slimming approach for robotic manipulation tasks. Our validation consists of the following components: (i) General Pruning Solution for Robotic Tasks. We conduct experiments on current popular backbone e.g., FCN and UNet for the grasp detection model. (ii) Extensive ablation studies to assess the model's performance, parameter count, FLOPs, model size, and computational requirements. (iii) Real Robot Scenario Validation. We assess the performance of our pruned method on a physical Franka Panda robotic arm. Our lightweight model demonstrates strong performance in both single-object scenarios and cluttered environments, meeting real-time operational demands.

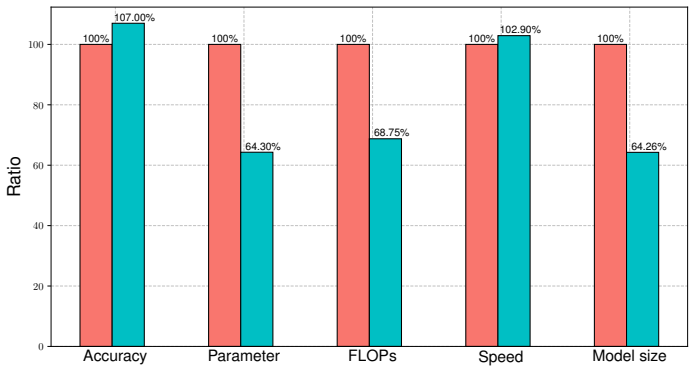
A. Datasets and Evaluation Criteria

Datasets: In the domain of grasp detection, two widely recognized benchmark datasets are commonly used: *Cornell* [3] and *Jacquard* [15] datasets. The Cornell dataset comprises 885 images featuring 244 distinct objects. Each object is captured from various viewpoints, with each image being meticulously annotated. On the other hand, the Jacquard dataset consists of more than 50,000 images containing 11,000 objects. This dataset is primarily generated in simulation, offering more comprehensive bounding box annotations for grasping.

Evaluation Criteria: In assessing grasps, we adhere to established evaluation metrics, consistent with prior research [15]–[17]. A grasp representation is considered valid if it meets the following criteria:



(a) UNet-based grasp detection



(b) FCN-based grasp detection

Fig. 3. The comparisons of the best-pruned model with the original model. The red bar indicates the original model and the blue bar indicates the savings in terms of the number of parameters, FLOPs, speed, and model size for the model with the highest accuracy.

i). The orientation difference between the predicted grasp and manual annotation does not exceed $30°$.

ii) The Jaccard index, as defined in Eq. (8), between the predicted grasp g and the ground truth g is greater than 0.25.

$$J\left(g^*, g\right) = \frac{|g^* \cap g|}{|g^* \cup g|}, \tag{8}$$

where $|g^* \cap g|$ is the intersection of the predicted grasp rectangle and the ground truth, and $|g^* \cup g|$ is the union of two regions. We assess the efficiency of the grasp detection system using performance metrics, including frames per second (FPS), single-image processing speed (ms), and computational burden measured in floating-point operations (FLOPs).

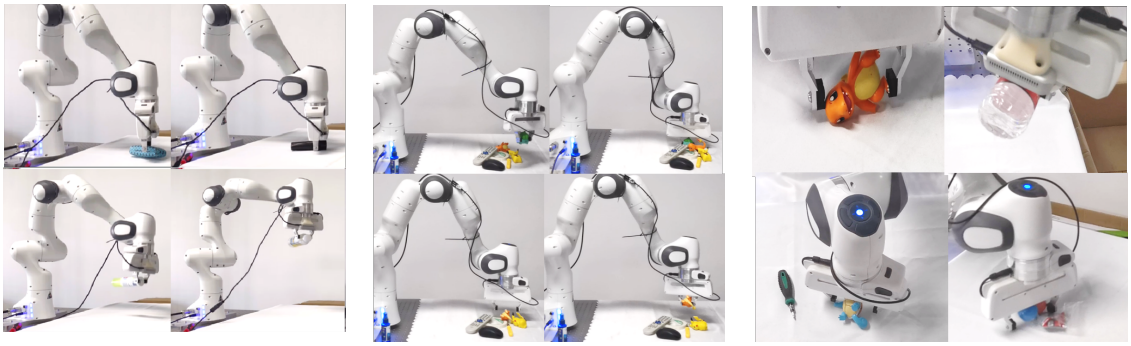
B. Implementation Details

We train all models from scratch using the AdamW optimizer [18] with a learning rate of $1e{-}4$, maintaining consistent settings throughout the training process. After completing the pruning phase, we construct a more lightweight model and subsequently fine-tune it to recover any lost accuracy. During the fine-tuning process on both the Cornell and Jacquard datasets, we keep the optimization settings identical to those used in regular training. Also, we assess wall-clock speed and memory consumption on a laptop.

TABLE I

THE PERFORMANCE OF DIFFERENT PRUNING RATIOS ON THE CORNELL GRASPING DATASET.

Model	Accuracy (%)	Parameters	Pruned	FLOPs	Computing reduction	FPS	Model Size
FCN-Grasp (Baseline)	98.4%	11.8 M	-	19.2 B	-	134	45.06MB
FCN-Grasp (20% Pruned)	**99.09%**	7.59 M	35.67%	13.2 B	31.2%	138	28.96MB
FCN-Grasp (50% Pruned)	98.64%	2.48 M	78.9%	7.68 B	60%	142	9.49MB
FCN-Grasp (70% Pruned)	96.32%	0.77 M	93.4%	4.67 B	75.6%	145	2.96MB
UNet-Grasp (Baseline)	98.5%	13.3 M	-	23.8 B	-	112	51.1MB
UNet-Grasp (20% Pruned)	98.5%	8.76 M	34.1%	15.6 B	34.4%	122	33.44MB
UNet-Grasp (50% Pruned)	**98.8%**	3.01 M	77.3%	8.60 B	63.8%	135	11.49MB
UNet-Grasp (70% Pruned)	95.8%	0.95 M	92.8%	5.07 B	78.6%	139	3.62MB



(a) Household object grasping (b) Cluttered environment grasping (c) Screenshots of challenging object grasps

Fig. 4. Physical grasping based on the pruned model using the Franka Panda robot.

C. Results and Discussion

Computation-budget. In the context of robotic tasks, the primary objective of model slimming is to reduce the computational budget and the number of parameters, facilitating deployment on real robotic embedded systems while meeting real-time requirements. Table I illustrates the impact of model slimming on various mainstream backbone networks during robotic visual manipulation tasks. The "Baseline" refers to an unpruned model trained under normal conditions. "50% Pruned" signifies a 50% reduction in channels. Additional metrics such as pruning ratio and FLOPs are detailed in Table I. Our results reveal substantial parameter redundancy in deep networks. Even with a mild 70% channel pruning, there is minimal degradation in grasp accuracy, and accuracy is consistently maintained. To emphasize the efficacy of model slimming in visual grasping, we depict the proportion of parameter and FLOPs savings for the best-pruned model in Figure 3. Notably, UNet-based grasping networks can undergo significant pruning without a substantial loss of accuracy, potentially due to the compensatory effect of skip connections in fusing information.

Running-rime. Table II displays the inference speed, measured in Frames Per Second (FPS), and running time for different baseline models evaluated on the Cornell dataset. Our prune-grasp model, tested on an NVIDIA GTX 3090, achieves an impressive 151.5 FPS. Notably, our slimming model outpaces the fastest model by 0.5 ms while delivering a 17.9% improvement in performance, showcasing a superior balance between accuracy and speed.

Model size. In the context of the FCN-based grasping model, pruning results in a model that closely matches the performance of the original model while achieving a remarkable tenfold reduction in model size. Moreover, the pruned model consistently generates confident grasping candidates for objects of various shapes. This pruning technique streamlines visual grasping models for deployment on embedded robot arms. Our tests affirm the pruned grasping model's robustness across diverse objects. In our experiments, the fine-tuned compact model consistently matches or surpasses the performance of the original model.

D. Ablation Studies

In this section, we conduct ablation studies to assess the robustness and efficiency of our slimming solution in the context of visual robotic manipulation tasks. To ensure equitable comparisons, we select well-known visual backbone networks, specifically FCN [25] and UNet [26], as feature extractors. We standardize the use of ReLU as the activation function and employ 3×3 convolutional kernels. We systematically evaluate each model at varying pruning ratios to analyze grasp accuracy, saved FLOPs, model size, and parameters. It is worth noting that excessive pruning, such as over 60% of channels, can significantly degrade model size, saved FLOPs, and parameters.Generally, wall-clock time and FLOPs savings exhibit similar magnitudes, while memory reduction shows a linear trend as the number of pruned channels increases. In Fig. 3, the red area highlights the original model, while the green area represents the model striking the optimal trade-off in accuracy, speed, and model size. Pruning clearly offers a favorable balance between speed and performance for the visual grasping model and can be readily applied to existing off-the-shelf grasping models. For models with parallel paths

TABLE II
THE ACCURACY ON CORNELL GRASPING DATASET.

Author	Method	Input Modality	Input size	Accuracy(%)		Time (ms)	FPS
				IW	OW		
Jiang [19]	Fast Search	RGB-D	227 × 227	60.5	58.3	5000	0.02
Morrison [10]	GG-CNN	D	300 × 300	78.4	84.3	7	142.8
Lenz [3]	SAE	RGB-D	227 × 227	73.9	75.6	1350	0.07
Wang [20]	Two-stage closed-loop	RGB-D	-	85.3	-	140	7.14
Redmon [9]	AlexNet, MultiGrasp	RGB-D	224 × 224	88.0	87.1	76	13.15
Asif [21]	STEM-CaRFs	RGB-D	224 × 224	88.2	87.5	-	-
Karaoguz [22]	GRPN	RGB	-	88.7	-	500	2
Kumra [6]	ResNet-50x2	RGB-D	224 × 224	89.2	88.9	103	9.70
Asif [11]	GraspNet	RGB-D	224 × 224	90.2	90.6	24	41.66
Guo [23]	ZF-net	D	-	93.2	89.1	-	-
Wang [17]	GPWRG	RGB	400 × 400	94.4	91.0	8	125
Zhang [24]	ROI-GD	RGB	-	93.6	93.5	40	25
Kumra [16]	GR-ConvNet	RGB-D	300 × 300	**97.7**	96.6	20	50
EffiGrasp (Ours)	Prune-Grasp (Speed and Size)	RGB	224 × 224	96.3	95.38	**6.5**	**151.5**
EffiGrasp (Ours)	Prune-Grasp (Accuracy)	RGB	224 × 224	**99.09**	98.57	7.24	138

TABLE III
THE ACCURACY ON JACQUARD GRASPING DATASET.

Authors	Algorithm	Accuracy (%)
Depierre [15]	Jacquard	74.2
Morrison [10]	GG-CNN2	84
Zhou [5]	FCGN, ResNet-101	91.8
Alexandre [27]	GQ-STN	70.8
Zhang [24]	ROI-GD	90.4
EffiGrasp (Ours)	Prune-Grasp	92.1

like UNet with skip-connections, channel pruning poses added complexity due to interdependencies among data streams. A deeper understanding of the model's data flow is essential to guide the pruning schedule design. Despite being initially designed for convolutional neural networks in robotic tasks, it is also applicable to fully-connected networks.

Table II presents accuracy results under both image-wise and object-wise splits on the Cornell dataset. We adhere to the training procedures outlined in [24], [28], and [29] for training on the Cornell dataset. Models are ranked by their release dates. Our slimming solution demonstrates competitive performance compared to state-of-the-art baselines. Notably, it achieves the lowest running time, nearly an order of magnitude faster than others. This outcome underscores that our slimming method effectively preserves model performance while significantly enhancing execution speed. In comparison to the image-wise split on the Cornell dataset, our prune-grasp is roughly half the size of the original model but three times faster than previous work. These results demonstrate that pruning facilitates a nuanced balance between accuracy and speed, making it advantageous for real robot deployment. Table VI lists the results on the Jacquard dataset. All models are ranked according to their release date. Note that the performance of our prune grasp is competitive when compared to the state-of-the-art baselines. For more investigation on ablation studies, the reader is referred to the supplementary material.

E. Grasping in Real World Scenarios

The physical experimental setup consists of a Franka Panda robotic arm with 7 degrees of freedom (7 DOF) and an Intel RealSense camera. The camera is mounted on the gripper of the end effector in an eye-in-hand configuration, as depicted in Fig. 7. In the grasping experiments, a variety of household objects with diverse sizes and shapes are employed. Some of these objects are novel and not present in the training dataset. Prior to each trial, the gripper is initially positioned approximately 40cm above the work area. In the manipulation experiments, we carry out the following tasks: 1) single object scenario, and 2) clutter environment. Real-time calculations are performed to determine the optimal grasping position and angle. For more details on physical experiments, the reader is referred to the supplementary material. Our grasping system has achieved outstanding performance in terms of both grasp success rate and real-time responsiveness, making it the ideal choice for a wide range of applications.

IV. CONCLUSION

This work presents a convolutional neural network (CNN) slimming approach tailored to address the challenges posed by deploying large, computationally intensive models in real-world robotic applications. The proposed pruning method aims to reduce model size, lower computational demands, and minimize memory requirements for general visual robotic systems. It identifies less significant channels based on activation feature map gradients and subsequently prunes them. Notably, this approach does not necessitate specialized hardware support and seamlessly integrates into standard robotic embedded platforms. Empirical evaluations on prominent robotic perception backbone networks, including FCN and UNet, yield promising results. Future research will extend these pruning techniques beyond parallel gripper jaws to encompass more intricate robotic systems, such as multi-finger dexterous hands.

REFERENCES

[1] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, "Visual manipulation relationship recognition in object-stacking scenes," *Pattern Recognit. Lett.*, vol. 140, pp. 34–42, 2020. 1

[2] J. Dong, Y. Cong, G. Sun, and T. Zhang, "Lifelong robotic visual-tactile perception learning," *Pattern Recognit.*, vol. 121, p. 108176, 2022. 1

[3] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robotics Res.*, vol. 34, no. 4-5, pp. 705–724, 2015. 1, 4, 6, 8

[4] Y. Song, L. Gao, X. Li, and W. Shen, "A novel robotic grasp detection method based on region proposal networks," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101963, 2020. 1

[5] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully convolutional grasp detection network with oriented anchor box," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 7223–7230. 1, 6, 8, 11

[6] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2017, pp. 769–776. 1, 6, 8

[7] F.-J. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018. 1, 8

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 1

[9] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1316–1322. 1, 2, 6, 8

[10] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *Int. J. Robotics Res.*, vol. 39, no. 2-3, pp. 183–201, 2020. 1, 2, 6, 8, 11

[11] U. Asif, J. Tang, and S. Harrer, "Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices." in *IJCAI*, vol. 7, 2018, pp. 4875–4882. 1, 2, 6, 8

[12] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020. 1

[13] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018. 2, 4, 9

[14] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2755–2763. 2, 8

[15] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3511–3516. 4, 6, 11

[16] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9626–9633. 4, 6

[17] S. Wang, X. Jiang, J. Zhao, X. Wang, W. Zhou, and Y. Liu, "Efficient fully convolution neural network for generating pixel wise robotic grasps with high resolution images," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 474–480. 4, 6, 8

[18] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017. 4, 8

[19] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3304–3311. 6, 8

[20] 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. 6

[21] U. Asif, M. Bennamoun, and F. A. Sohel, "Rgb-d object recognition and grasp detection using hierarchical cascaded forests," *IEEE Trans. on Robotics*, vol. 33, no. 3, pp. 547–564, 2017. 6

[22] H. Karaoguz and P. Jensfelt, "Object detection approach for robot grasp detection," in *Proc. IEEE Int. Conf. Robot.Automat.*, 2019, pp. 4953–4959. 6

[23] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi, "A hybrid deep architecture for robotic grasp detection," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1609–1614. 6, 8

[24] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng, "Roi-based robotic grasp detection for object overlapping scenes," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4768–4775. 6, 11

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440. 5, 11

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241. 5, 11

[27] A. Gariépy, J.-C. Ruel, B. Chaib-Draa, and P. Giguere, "Gq-stn: Optimizing one-shot grasp detection based on robustness classifier," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3996–4003. 6, 11

[28] S. Wang, Z. Zhou, and Z. Kan, "When transformer meets robotic grasping: Exploits context for efficient grasp detection," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8170–8177, 2022. 6, 8

[29] Z. Zhou, S. Wang, Z. Chen, M. Cai, and Z. Kan, "A robotic visual grasping design: Rethinking convolution neural network with high-resolutions," *arXiv preprint arXiv:2209.07459*, 2022. 6, 8

[30] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol. 2, 1989. 8

[31] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. 8

[32] S. Nasrin, S. Ramakrishna, T. Tulabandhula, and A. R. Trivedi, "Supported-binarynet: Bitcell array-based weight supports for dynamic accuracy-energy trade-offs in sram-based binarized neural network," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5. 8

[33] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International conference on machine learning*. PMLR, 2015, pp. 2285–2294. 8

[34] R. Xu, F.-J. Chu, and P. A. Vela, "Gknet: grasp keypoint network for grasp candidates detection," *The International Journal of Robotics Research*, p. 02783649211069569, 2022. 8

APPENDIX

A. Related Work

1) Recent Advances for Robotic Grasping: Along with the fast growth of intelligent manufacturing, robotic grasping is increasingly being used in industrial applications. Fast acquisition of visual input from the sensors and the detection of the appropriate grasping pose are the initial steps for successful robotic grasping. Grasp detection [3] can be roughly divided into analytical and data-driven methods. The analytical methods generally construct the grasping configuration from the kinematic, geometric, and dynamic information of an object, and often require an accurate geometric model of the grasped object. Unfortunately, knowledge such as surface properties, friction coefficient, and mass information is not always available in practice.

Data-driven approaches can train a neural network from scratch without prior knowledge of the object. Convolutional neural networks have been gradually introduced to accomplish visual grasping tasks. Redmon et al. [9] propose an Alexnet-like architecture for grasping, where the blue channel is replaced with depth information. Kumar et al. [6] adopt two parallel ResNet to conduct grasping feature extraction, one using depth image as input and the other receiving RGB information. Chu et al. [7] introduce a two-phase framework to tackle the grasping mission, where the first phase detects the grasping contact point and the second phase decides the orientation of the grasping rectangle. Morrison et al. [10] propose a generative convolutional neural network to calculate the grasping quality score for each pixel in the image. A vision transformer [28] is introduced to build long-range dependencies on pixels for better clutter grasping. A sound grasp detection module needs to be robust to perceived noise, object motion, and imprecise kinematics. Meanwhile, the current closed-loop grasping system places higher real-time demands on the hardware.

2) Model Compression for Deep Learning: **Computation-budget**. On resource-constrained robot platforms and mobile devices, it is important to maintain a model with low computational cost. Recent researchers [11], [17], [19] employ models with compact representations in a hand-crafted manner. GraspNet [11] utilizes squeeze and dilated convolutions to reduce model parameters and mitigate calculation expenditure. Morrison et al. [10] design a lightweight fully convolutional network. Such specific models are manually designed by experts.

Running-rime. Motivated by neuroscience, optimal brain damage [30] is proposed to remove unimportant weights of the model to improve inference speed. Typical work is to prune insignificant weights to derive sparse weight matrices. This kind of weight pruning tends to require specialized software or hardware support. An alternative approach is channel pruning or filter pruning. Liu et al. [14] exploit the scale factor in the BN layer to eliminate inefficient channels. Molchanov et al. [31] perform channel selection using a greedy operation. Channel pruning is flexible and can contribute to reducing model parameters and improving inference speed.

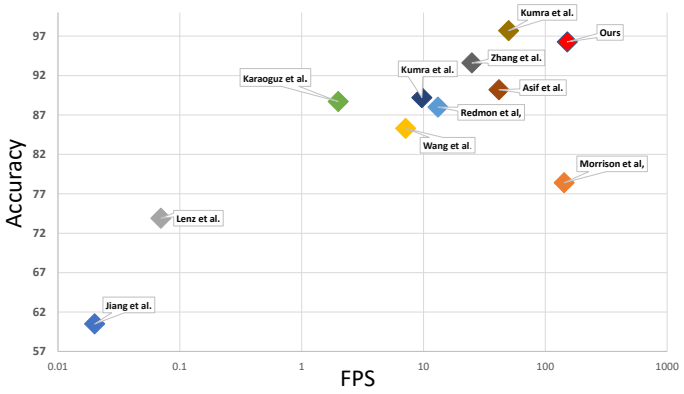


Fig. 5. The performance of SOTA methods. The nearer the top-right corner, the better the model's real-time and accuracy performance.

TABLE IV
THE COMPARISON OF BACKBONE NETWORK PARAMETERS.

Authors	Backbone network	Parameters
Kumra and Kanan [6]	ResNet-50	25.56M
Guo [23]	ZF-Net	5.3M
Zhou [5]	ResNet-101	44.55M
Xu [34]	DLA-34	15.74M
Wang [28]	Transformer	25.95M
Zhou [29]	HRG-Net	64.3M
Ours	FCN	**0.27M**
Ours	UNet	**0.38M**

Model-size. Slimming a neural network is not a new idea and can be traced back to 1990 [30]. The intuition is that neural networks contain a lot of redundant parameters, and many weights may not contribute much to the final output. Weight quantization [32] is widely used to model compression. HashSet [33] maps the weights into pre-defined groups before training. Therefore, it is possible to reduce the storage space by storing only the relevant hash index for each model parameter. However, most of the methods in this type can not save memory while the model is running, as the weights are often restored to their original precision during inference.

B. Detailed Implementation Details

On popular grasping datasets, e.g., Cornell and Jacquard, we evaluate our prune approach on two prominent grasping networks such as FCN-based and Unit-based networks. The RGB and depth images are normalized to the same range, with each channel subtracted from its mean and divided by the standard derivation. The original resolution of data is 640×480. A center crop is made to 224×224 to reduce the calculation workload.

We train all models from scratch and adopt AdamW [18] as the optimizer with a weight decay of $1e-4$. All settings are consistent throughout the whole training process. Once the pruning is complete, we build a more lightweight model and then fine-tune it to regain dropped accuracy. On both Cornell and Jacquard datasets, the fine-tuning process maintains identical optimization settings to the regular training

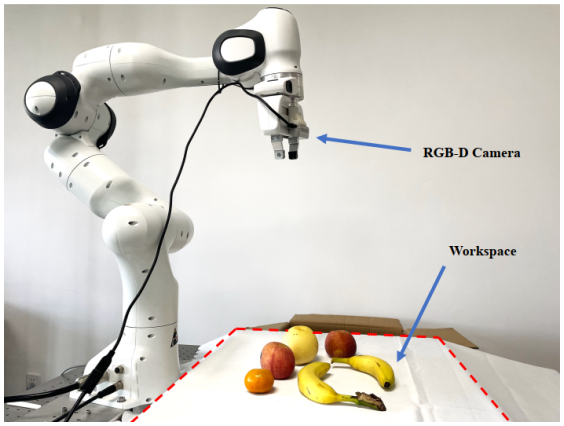


Fig. 6. The environmental workspace for robot grasping experiments.

process. The wall-clock speed and memory consumption are tested on a laptop. In practice, to produce a streamlined grasp representation for high-precision and real-time grasp detection, three grasp heads are attached to the output of the grasp model, denoted as

$$\mathbf{G} = \{Q, W, \Theta\} \in \mathbb{R}^{3\times W\times H}, \tag{9}$$

where Q, W, Θ are three heat maps of the same size as the input image. Q is the grasp score map where each pixel in Q represents the success rate of grasping at this point. Each pixel in W, Θ indicates the width of the gripper opening and the corresponding rotation angle of grasping at that point, respectively. Afterward, the best grasp can be obtained by searching the grasp score map, $g^* = \arg\max Q$. Eventually, the world coordinates can be transformed by the calibration parameters in the grasping system. In detail, the transformation operation is shown as follows:

$$g_r = T_{rc}\left(T_{ci}\left(g\right)\right), \tag{10}$$

where the real-world grasp g_r is translated through two successive matrix transformations. T_{ci} and T_{rc} refer to the conversion from 2D image space to camera frame space and from camera space to world coordinate system, respectively.

For grasp detection tasks, it is desired that the proposed lightweight approach can be used to slim down the off-the-shelf grasping model without compromising performance. Recent experiment results [13] have shown that in a fully-trained neural network, a large number of weights is very close to 0. This implies that such weights may not play a significant role to determine the outcome of neural networks. This makes it reasonable to prune neural networks designed for robots. Unfortunately, in practice, it is a challenging task to slim down the neural network so that it can be deployed on a robot system and performs well not only in simulation but also in the physical world. For example, for a pruned policy network with deep reinforcement learning, one has to ensure that the actions generated by the policy network would not lead to instability in response to the environment.

1) Progressively Pruning: The approach above provides a way to approximate the minimal change in the loss function by pruning the gradient-flat feature map. Our aim is to develop a pruning method, for robotic visual tasks, that can reveal unimportant channels and removes them from the original network to eliminate their associated weights. To meet the real-time implementation requirement for robotic tasks, we design an iterated approach that makes a trade-off between performance and speed. Working from the entire set of parameters $\mathcal{W}$, we iteratively evaluate and gradually prune the less critical parameters. In each pruning, one has to ensure that the newly obtained $\mathcal{W}'$ satisfies the l_0 bound. Note that merely greedily removing hidden layer feature maps with small gradients and their associated parameters may lead to sub-optimal results. To achieve a better balance between exploration and exploitation, we adopt an $\epsilon-greedy$ as the channel pruning strategy where the feature map to be removed is randomly chosen with a small probability ϵ and a greedy selection of the feature map with a small gradient using probability $1-\epsilon$.

$$\begin{cases} \arg\min_{c_i} \frac{1}{N}\sum_j^N \frac{\partial \mathcal{L}}{\partial c_{i,j}}, & probability \quad 1-\epsilon, \\ random \quad selection, & probability \quad \epsilon. \end{cases} \tag{11}$$

Also, to alleviate the randomness caused by ϵ, we additionally apply a restart strategy. After each round of iteration, ϵ is scaled smaller. The pruned model that performs best in the iterative process is then selected as the final model.

In practice, we observe that greedy strategies may lead to certain sub-optimal behaviors, while additional exploration is likely to yield long-term benefits. Another challenge is that, as the model goes deeper, the dimension and scale of feature maps vary from layer to layer. To perform a global and consistent pruning, we re-scale all the feature maps and normalized them.

TABLE V
THE PERFORMANCE OF ITERATIVE PRUNING SCHEME.

Iteration	Trained	Fine-tuned	Params	FLOPs
1	98%	98.8%	41.6MB	1.7×10^{10}
2	97%	98.5%	39.82MB	1.44×10^{10}
3	81%	98.3%	36.26MB	1.34×10^{10}
4	75%	97.9%	32.72MB	1.44×10^{10}
5	78%	97.5%	29.40MB	1.44×10^{10}

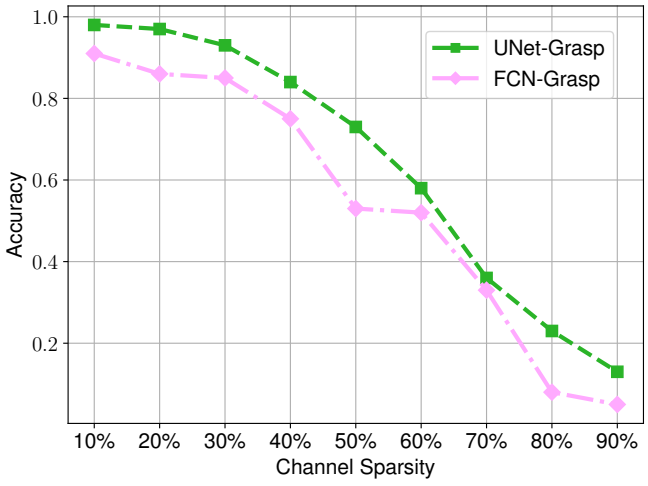


Fig. 9. The effect of pruning different channel proportions without retraining.

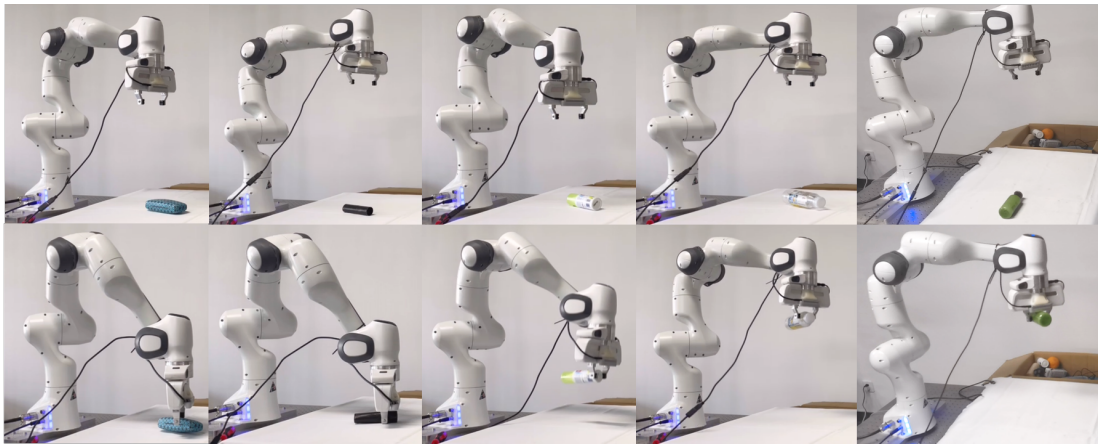


Fig. 7. Physical grasping based on the pruned model using the Franka Panda robot.

Fig. 8. The bin picking tasks using the pruned model.

C. Physical Experiment

The robotic manipulation platform comprises a 7 DOF Franka Panda robotic arm equipped with an Intel RealSense camera. The camera is affixed to the gripper of the end effector, operating in an eye-in-hand configuration. The complete experimental setup is depicted in Fig. 6. In our grasping experiments, we employ various household objects of diverse sizes and shapes, including novel objects absent from the training dataset. The system's grasp success rate is defined as the ratio of successful grasps to total grasp attempts. A successful grasp is achieved when the gripper lifts and holds an object in the air for at least 5 seconds. After developing a visual grasping model, a critical challenge remains in bridging the gap for real-world applications. The ideal grasp detection model should maintain a high grasp success rate and robustness when integrated into a physical robotic manipulation pipeline. In our manipulation experiments, we perform two tasks: 1) single object scenario, and 2) cluttered environment. Single-object grasping, devoid of distractors, provides an ideal evaluation scenario to assess the grasping model's accuracy. Isolated objects are randomly positioned on the table, as illustrated in Figure 7. Each object undergoes 10 grasping trials.

In a cluttered environment where objects are densely arranged and stacked, the grasping detection system must track grasped objects in real-time. This scenario serves as a combined evaluation of the grasping system's speed and accuracy.

Before each trial, the gripper is initially positioned approximately 40 cm above the working area. Subsequently, the system calculates the optimal grasping position and angle in real-

TABLE VI
THE ACCURACY ON JACQUARD GRASPING DATASET.

Authors	Algorithm	Accuracy (%)
Depierre [15]	Jacquard	74.2
Morrison [10]	GG-CNN2	84
Zhou [5]	FCGN, ResNet-101	91.8
Alexandre [27]	GQ-STN	70.8
Zhang [24]	ROI-GD	90.4
Ours	Prune-Grasp	92.1

Fig. 10. The visualization of grasp representation on Cornell dataset.

time. Once a pre-grasp pose is determined, the manipulator moves to the predefined position using a planner, and the gripper adjusts its wrist orientation to reach the desired angle.

D. Ablation Studies

Ablation studies are carried out in this section. First, a few visual backbone networks are chosen as feature extractors to evaluate the robustness and efficiency of our pruning method for visual robotic manipulation tasks. In Table II, FCN [25] and UNet [26], are selected for comparisons owing to their popularity in the convolution neural network community. For fair comparisons, we employ ReLu as an activation function and adopt 3×3 convolutional kernels. Second, each model is evaluated at different pruning ratios to test the grasp accuracy, saved FLOPs, model size, and parameters. We also notice that models with different architectures actually perform well due to the good learning ability of the neural network. However, the speed of execution and size of models vary considerably due to the complexity of different architectures.

a global greedy strategy is performed across all layers. The feature channels are sorted according to a certain criterion, and the channels lower than a pre-defined threshold are deleted. For example, to remove 70% channels of the model, the percentile threshold can be set as 70%. In this way, we can reach a more compressed model with fewer parameters, less running memory, and a lower computational budget than the original network.

In Figure 5, we present the inference speed, specifically Frames Per Second (FPS), of various baselines on the Cornell dataset. The x-axis represents the model inference speed (FPS) in logarithmic scale, while the y-axis denotes model accuracy in image-wise splits on the Cornell dataset. Our prune-grasp model, tested on an NVIDIA GTX 3090, achieves an impressive 151.5 FPS.

Notably, our prune-grasp model occupies a position close to the top-right corner in Figure 5, indicating a favorable balance between accuracy and speed.

TABLE VII

THE PERFORMANCE OF DIFFERENT PRUNING RATIOS ON THE CORNELL GRASPING DATASET.

Model	Accuracy (%)	Parameters	Pruned	FLOPs	Computing reduction	FPS	Model Size
FCN-Grasp (Baseline)	98.4%	1.18×10^7	-	1.92×10^{10}	-	134	45.06MB
FCN-Grasp (10% Pruned)	98.07%	9.81×10^6	16.8%	1.55×10^{10}	19.2%	137	37.43MB
FCN-Grasp (20% Pruned)	**99.09%**	7.59×10^6	35.67%	1.32×10^{10}	31.2%	138	28.96MB
FCN-Grasp (30% Pruned)	98.4%	5.56×10^6	52.8%	1.10×10^{10}	42.7%	139	21.24MB
FCN-Grasp (40% Pruned)	98.3%	3.84×10^6	67.4%	9.23×10^9	51.9%	141	14.66MB
FCN-Grasp (50% Pruned)	98.64%	2.48×10^6	78.9%	7.68×10^9	60%	142	9.49MB
FCN-Grasp (60% Pruned)	96.49%	1.46×10^6	87.6%	6.19×10^9	67.7%	143	5.58MB
FCN-Grasp (70% Pruned)	96.32%	7.75×10^5	93.4%	4.67×10^9	75.6%	145	2.96MB
FCN-Grasp (80% Pruned)	93.4%	3.33×10^5	97.17%	3.2×10^9	83.3%	151	1.27MB
FCN-Grasp (90% Pruned)	88.92%	8.57×10^4	99.27%	1.43×10^9	92.5%	152	0.27MB
UNet-Grasp (Baseline)	98.5%	1.33×10^7	-	2.38×10^{10}	-	112	51.1MB
UNet-Grasp (10% Pruned)	98.4%	1.12×10^7	15.7%	1.85×10^{10}	22.2%	118	42.85MB
UNet-Grasp (20% Pruned)	98.5%	8.76×10^6	34.1%	1.56×10^{10}	34.4%	122	33.44MB
UNet-Grasp (30% Pruned)	97.8%	6.52×10^6	50.9%	1.30×10^{10}	45.3%	127	24.91MB
UNet-Grasp (40% Pruned)	98.6%	4.63×10^6	65.1%	1.08×10^{10}	54.6%	131	17.67MB
UNet-Grasp (50% Pruned)	**98.8%**	3.01×10^6	77.3%	8.60×10^9	63.8%	135	11.49MB
UNet-Grasp (60% Pruned)	97.9%	1.81×10^6	86.3%	6.71×10^9	71.8%	138	6.91MB
UNet-Grasp (70% Pruned)	95.8%	9.50×10^5	92.8%	5.07×10^9	78.6%	139	3.62MB
UNet-Grasp (80% Pruned)	96.2%	3.99×10^5	97%	3.28×10^9	86.2%	145	1.52MB
UNet-Grasp (90% Pruned)	**96.3%**	$\mathbf{9.87 \times 10^4}$	99.25%	1.47×10^9	93.8%	151	0.38MB