A digital twin of intelligent robotic grasping based on single-loop-optimized differentiable architecture search and sim-real collaborative learning

Qing Jiao[1], **Weifei Hu**[1,2,3*], **Guangbo Hao**[4], **Jin Cheng**[1,2], **Xiang Peng**[5], **Zhenyu Liu**[1,2], **Jianrong Tan**[1,2]

[1] State Key Laboratory of Fluid Power and Mechatronic Systems, Zhejiang University, Hangzhou 310027, China

[2] School of Mechanical Engineering, Zhejiang University, Hangzhou 310027, China

[3] Innovation Center of Yangtze River Delta, Zhejiang University, Jiaxing 314100, China

[4] School of Engineering and Architecture-Electrical and Electronic Engineering, University College Cork, Cork, Ireland

[5] College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou 310023, China

*Corresponding author email: weifeihu@zju.edu.cn

First author: Qing Jiao

Tel.: +86 17857142639; **E-mail address:** jiaoqing@zju.edu.cn

Corresponding author: Weifei Hu

Tel.: +86 13819105649; **E-mail address:** weifeihu@zju.edu.cn

ORCID: 0000-0002-1571-583X

Author: Guangbo Hao

Tel.: +353 0214903793; **E-mail address:** g.hao@ucc.ie

Author: Jin Cheng

Tel.: +86 13606601663; **E-mail address:** jcheng@zju.edu.cn

Author: Xiang Peng

Tel.: +86 571 85290368; **E-mail address:** pengxiang@zjut.edu.cn

ORCID: 0000-0002-8244-2202

Author: Zhenyu Liu

Tel.: +86 13867470039; **E-mail address:** liuzy@zju.edu.cn

Author: Jianrong Tan

Tel.: +86 571 87951273; **E-mail address:** zjume@zju.edu.cn

Abstract

The effectiveness of deep learning models for vision-based intelligent robotic grasping (IRG) tasks typically hinges upon the deep neural network (DNN) architecture as well as the task-oriented annotated training samples. Nevertheless, current methods applied for designing DNN architectures depend on human expertise or discrete search by evolution and reinforcement learning algorithms, which leads to enormous computational cost. Moreover, DNNs trained solely on simulation-labeled data face challenges in direct real-world deployment. In response to these concerns, this paper proposes a new stable and fast differentiable architecture search method (DARTS) based on a single-loop optimization framework, named single-loop-optimized DARTS (SLO-DARTS). This method enables simultaneous updates to the weights and architecture parameters of neural networks by continuously relaxing the discrete search space. Additionally, a digital twin (DT) framework integrating the Grasp-CycleGAN method is developed to minimize the visual gap between simulated and real-world IRG scenarios, enhancing the transferability of DNNs trained in simulation. The DT framework can not only enhance the IRG accuracy but also save the costly expense of large-scale real labeled data collection. Experiments demonstrate that the proposed SLO-DARTS method achieves a time-efficient optimization process while delivering a DNN with improved prediction accuracy compared to the original dual-loop-optimized DARTS method. The developed DT framework produces IRG accuracies of 92.6%, 86.3%, and 83.7% for single household objects, single adversarial objects, and cluttered objects, respectively.

Keywords: Robotic grasping, Deep neural network, Architecture search, Collaborative learning, Digital twin, Automatic design

1. Introduction

Despite extensive research over several decades, intelligent robotic grasping (IRG) remains challenging, especially when robots are operated to grasp unknown objects in unknown and unstructured environments. To realize accurate grasp and reliable execution, the first step is to learn effective potential grasps according to the sensor information of the target object and its surroundings.

The two primary types of robotic grasp detection approach now in use are analytical and empirical methods. In terms of analytic methods, they calculate the stable grasp based on the geometry of target object as well as robotic kinematics and dynamics analysis (Bicchi, 1994; Hunt & Torfason, 1987; Ten Pas et al., 2017; Wöhlke, 1992). However, due to the reliance on the simplification of robotic constraints, such methods for IRG have poor generalization for different robotic environments. The latter empirical methods focus on learning the best grasp from a previous successful grasping experience. In early work on empirical methods, the similarities of the characteristics of the objects were used to perform pose estimation, and the grasp detection of unknown objects was unable to be generalized (Caldera et al., 2018). Convolutional neural networks (CNNs), which have emerged as a representative deep learning technique, serve to compensate for the above deficiencies and enhance the potential for planning grasps of unknown objects (de Souza et al., 2021; Ghazaei et al., 2019).

Due to significant advancements made by deep learning for computer vision (Chai et al., 2021), there is a rising trend in the utilization of deep learning approaches for IRGs. Generally, methods for robotic grasp detection using deep learning can be grouped into two categories: two-stage detector (TSD) and one-stage detector (OSD). Studies like (Ainetter & Fraundorfer, 2021; Chiu et al., 2022; Song et al., 2020) adopted TSD stacked with two neural networks, where the first is responsible for screening regions most likely to contain potential grasps in the initial stage, and then the second is dedicated to grasp localization and detection. Although the architecture of TSD is generally beneficial for improving accuracy, region generation process is often time-consuming. For the OSD used in (Park et al., 2020), the image is divided into multiple grids, each then subjected to detection rather than generating region proposals. Compared with TSD, OSD greatly improves the computational speed at the cost of lower prediction accuracy, which can be

compensated by the development of different CNN architectures. For example, Redmon et al. (2015) developed an AlexNet-based regressor (Krizhevsky et al., 2017) for detecting multiple grasps with fast computational time. Kumra et al. (2017) also introduced a real-time grasp detection approach based on regression but replaced the original AlexNet with the ResNet (He et al., 2016), which marginally improved the accuracy by 1.21% while ensuring a fast computational speed. It is understood that changes in the network architecture significantly result in a notable enhancement in the grasp prediction performance, demonstrating a great impact of CNN architecture design on the grasping performance. However, the majority of promising CNN architectures for robotic grasping in existing works are hand-crafted, where the selection of network hyperparameters relies not only on a great amount of human expertise but also on many trial and error.

In the past few years, neural architecture search (NAS), aimed at achieving CNN design through automated methods, has undergone extensive explorations and rapid advancements (Heuillet et al., 2024). In early works on NAS, the evolutionary algorithm or reinforcement learning (RL) is adopted to obtain network architectures superior to those designed by human experts. However, the above process requires significant computational costs due to the need for extensive sampling from the discrete architecture search space for independent training and evaluation, even if an optimized network architecture can be obtained. Aiming to improve computational efficiency, new NAS methods called weight-sharing search methods were proposed to define the search space as a super network/graph and share weights among all sub architectures (Cai et al., 2018; Xie et al., 2021). The recently widely studied differentiable architecture search (DARTS) method is one of the representative approaches (Liu et al., 2018). It continuously relaxes the discrete search space and introduces normalized and learnable operational variables, enabling the use of gradient-based bilevel optimization algorithms for CNN architecture design. Despite the search cost by several orders of magnitude reduced by DARTS based on the bilevel optimization framework, the phenomenon that DARTS-based methods are easily prone to aggregation of non-learnable operations in the selected architecture and cannot work stably to search high-performance architectures in image classification tasks have been noted by many scholars (Chu et al., 2020; Ye et al., 2022). In addition, the original dual-loop DARTS suffers from the inaccuracy

of gradient estimation and heavy computational burden, which motivates us to apply single-loop optimization in DARTS.

CNNs for IRGs often require numerous annotated data for the purpose of achieving desirable predictive performance. Even if several public open datasets are available, it is difficult to guarantee that these datasets fulfill the network's training requirements considering that different setups require different labels. However, manually collecting annotated datasets to train deep learning models under a specific grasping setting may involve hundreds of or thousands of real-world grasp attempts, which is remarkably expensive, time-consuming, and even impossible in some case. Kalashnikov et al. (2018) presented a scalable robotic RL framework called QT-Opt to address this challenging problem, which can autonomously collect a large amount of robot experience with minimal human intervention. However, the RL process lasted several weeks and involved more than 580k real-world grasps conducted by 7 robots, which is still time-consuming and costly. Hence, researchers turn to train deep learning models in high-precision simulations and then transfer them to real-world robots to save time and cost. Unfortunately, such models tend to perform poorly in real-world robotic grasping because of the nonignorable visual gap between simulated and real-world scenario. Bousmalis et al. (2018) used generative adversarial networks (GANs) to adjust synthetic images to resemble real-world ones, reducing the amount of real-world samples used in training high-performance grasp prediction networks. However, the training of GANs required abundant real-world data labeled by the pre-trained grasp network. Recently, the emergence and development of digital twin (DT) technology have pointed out a new direction to address the visual simulation-to-reality gap. In a DT framework, a virtual robot and its physical counterpart are synchronized in terms of their surrounding environments and behaviors to minimize the discrepancies between simulated and real-world robotic grasping (Hu et al., 2021). Hence, the similarity between simulated and real-world scenarios is maximized, thereby enhancing the transferability of the models learned in simulation. This paper develops a new DT of IRGs to efficiently improve the model accuracy for real grasping while avoiding expensive real-world robot training. The primary contributions of this work are threefold below:

(1) A novel multi-grasp one-stage detector based on oriented reference rectangles is designed to provide real-time and accurate robotic grasping detection.

(2) A new NAS method named single-loop-optimized DARTS (SLO-DARTS) using a single-loop optimization framework is developed to automatically optimize both the architecture and weights of CNNs. A more stable and precise architecture of a CNN for IRGs is then obtained.

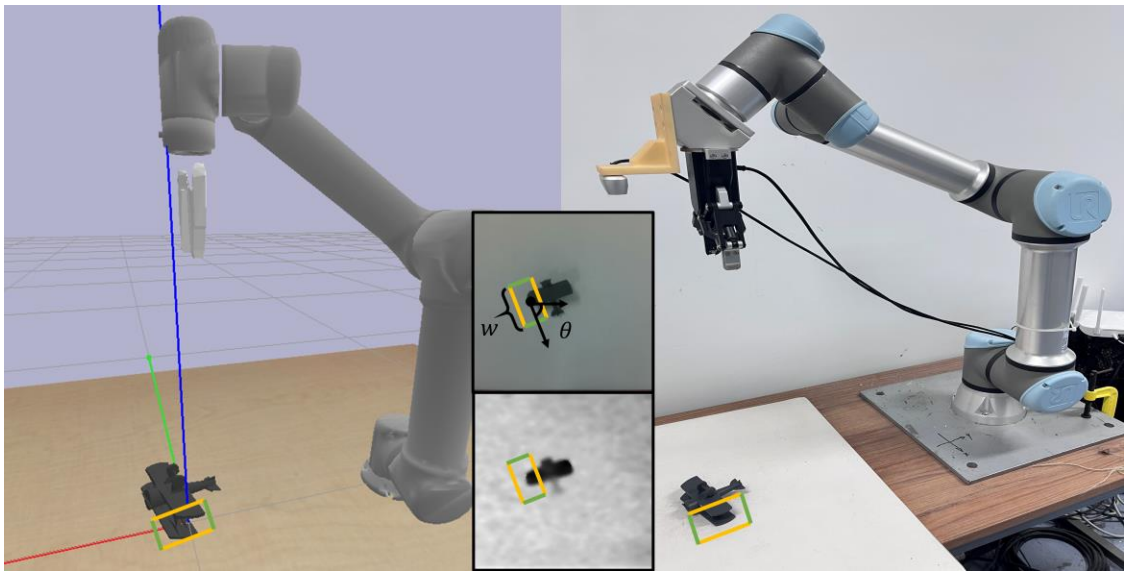
(3) A DT of IRG based on sim-real collaborative learning is developed to train CNN. The sim-real collaborative learning is realized by developing a Grasp-Cycle Consistent Generative Adversarial Network (Grasp-CycleGAN) which converts simulated images into pseudo-realistic ones for further training CNN.

The rest of this paper is in brief provided below. Section "Methodology" presents the multi-grasp model, details the principles of SLO-DARTS method, and provides the DT implementation process of IRGs. Section "Experiments" details the verification experiments of the SLO-DARTS method and the DT framework, with the subsequent "Results and discussion" section presenting and analyzing the findings. Conclusions and further research work are offered in "Conclusions and future work" section.

2. Methodology

2.1 Problem formulation of robotic grasping

The grasping scenario is proposed firstly much like (Hu et al., 2022). Given an n-channel image containing the target object and its surroundings, the IRG problem is simplified as executing antipodal grasps for novel objects by the two-finger parallel-jaw gripper perpendicular to a horizontal plane. To learn from the data, it is imperative to initially establish a clear definition of the fundamental elements for effective grasping.



(a) (b)

Fig. 1 Illustration of the virtual/physical robotic grasping scenarios. (a) Virtual robotic grasping scenario. (b) Physical robotic grasping scenario.

In the robot frame, a grasp can be set as a four-parameters representation $\mathrm{g} = (\boldsymbol{p}, \theta, w, c)$, which gives the real-world coordinates of the gripper's center $\boldsymbol{p} = (\mathrm{x}, \mathrm{y}, \mathrm{z})$, rotation angle θ, needed opening distance w and the graspable confidence (probabilities) c. Due to the parallel-jaw gripper's symmetry, the rotation angle θ is constrained between $-\pi/2$ and $\pi/2$.

We let $\mathbf{I} \in \mathbb{R}^{4\times H\times W}$ define an RGB-D image shot by the depth camera mounted at the gripper. Through the utilization of the known camera intrinsic and extrinsic parameters, it is easy to convert a grasp g into a grasp g_i in the pixel coordinate, represented by an oriented rectangle. The rectangle shown in Fig. 1 consists of a fixed edge (green edge) depicting the position of the gripper's jaws for grasping and an adjustable edge (yellow edge) representing both the direction and width of the gripper's opening. Formally, the 2D rectangle is uniquely defined by five parameters:

$$g_i = \{x_i, y_i, w_i, \theta_i, c\} \tag{1}$$

which gives the rectangle's center position in pixel coordinates (x_i, y_i), the grasp width w_i and the gripper rotation θ_i. Therefore, the problem of vision-based grasp detection can be transformed into detecting a valid grasp rectangle corresponding to the highest graspable probability in the image.

2.2 SLO-DARTS of a multi-grasp detection model

Since a single image may contain multiple valid grasp, the proposed approach predicts all potential graspable candidates on the full image by applying neural networks once to an image. However, the neural network architecture can impact feature extraction capabilities, thereby affecting the precision of grasp prediction. Hence, designing an optimal neural network is also one focus of the proposed approach in this paper.

2.2.1 Grasp detection network based on oriented reference rectangles

The grasp detection network processes an RGB-D image containing an unknown object as well as its surrounding environment, producing a set of grasp rectangles with corresponding confidence scores. Inspired by the anchor box idea proposed in object detection (Ren et al., 2015), the reference rectangle is introduced to tackle the grasp detection issue. The only difference is the additional consideration of the grasping angle, that is, the original reference rectangle is pre-set with a default angle, which is called the oriented reference rectangle.

As illustrated in Fig. 2, we partitioned the captured image of a graspable object into $M \times M$ grid cells, with k oriented reference rectangles centered within each cell. Each oriented reference rectangle in a grid cell has a different angle but the same size. The difference between two adjacent oriented reference rectangles is π/k. For example, if k=6, the angles of the oriented reference rectangles in the grid cell could be $-5\pi/12$, $-\pi/4$, $-\pi/12$, $\pi/12$, $\pi/4$ and $5\pi/12$. With the provided prior information from the oriented reference rectangles, the relative offset between the grasp rectangle and the oriented reference rectangle in terms of the grasp width w_i and angle θ_i can be predicted. Due to the fixed width h_i of the gripper finger in the actual grasping scenario, predicting the width is not necessary. In terms of the center coordinates of the grasp rectangles, the proposed approach predicts relative location offsets with respect to the grid cell, which is much easier than directly predicting the coordinates. In summary, the grasp detection network predicts k grasp rectangles at each cell, where each grasp rectangle consists of 5 parameters and can be

reparametrized as $\{t_x, t_y, t_w, t_\theta, t_c\}$. Given a cell's offsets from the image's top-left corner (c_x, c_y), along with the predefined size p_w and angle p_θ of an oriented reference rectangle, the final predicted grasp rectangle then corresponds to the following:

$$\begin{aligned} x &= \sigma(t_x) + c_x \\ y &= \sigma(t_y) + c_y \\ w &= p_w e^{t_w} \\ \theta &= p_\theta + t_\theta \times \pi/2k \\ c &= \sigma(t_c) \end{aligned} \tag{2}$$

where $\sigma(\cdot)$ is a sigmoid function used to restrict the graspable probability to a range between 0 and 1 and ensure that the grasp rectangle's center falls into the cell.

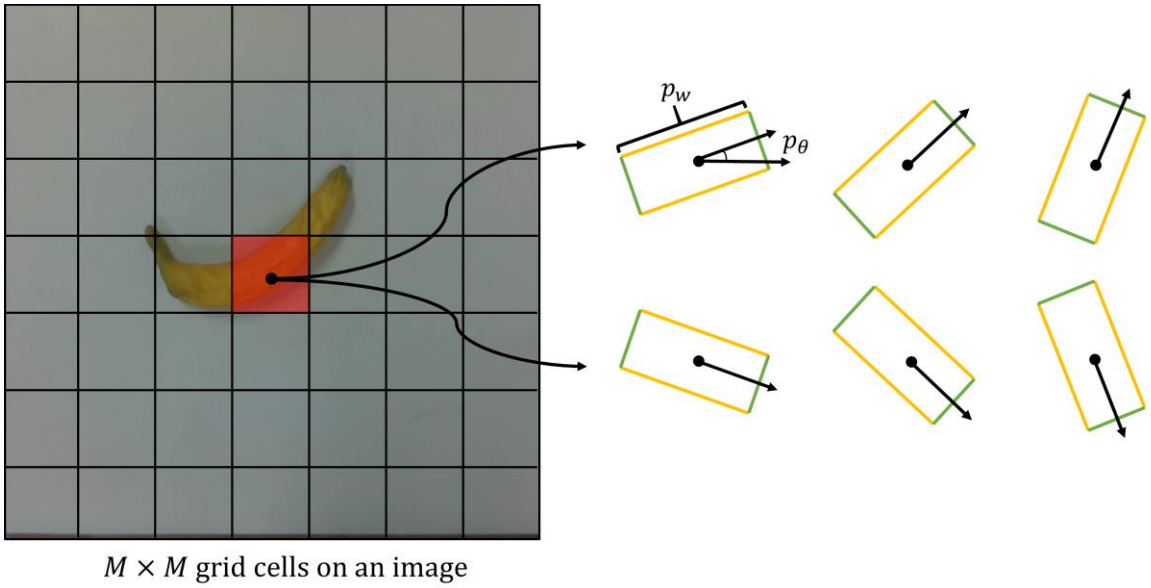


Fig. 2 The *k* oriented reference rectangles in a grid cell of the image.

The benchmarks corresponding to network predictions are established for network training, in the form of $\{\hat{t}_x, \hat{t}_y, \hat{t}_w, \hat{t}_\theta, \hat{t}_c\}$. $(\hat{t}_x, \hat{t}_y)$ are the location offsets of the benchmark grasp rectangle's center relative to the top-left corner of the grid cell containing it. $(\hat{t}_w, \hat{t}_\theta)$ are the offset values for the benchmark grasp rectangle's size and angle in relation to the positive oriented reference rectangle. In detail, an oriented reference rectangle meeting the two requirements listed below will

be assigned a positive label: (1) Both the benchmark and oriented reference rectangles should have their centers within the identical grid cell, and (2) the benchmark and oriented reference rectangles should have an angle difference within $\pi/2k$. The last parameter $\hat{t}_c$ is 1 if the reference rectangle is positive and 0 otherwise. With these definitions, the network loss function is calculated by combining classification and regression losses with appropriate weights, as follows:

$$\mathcal{L}\left(\{t\}\right)=\frac{1}{N}\left(\alpha_1\sum_i\mathcal{L}_{cls}\left(t_c^i,\hat{t}_c^i\right)+\alpha_2\sum_i\sum_{m\in\{x,y,w,\theta\}}\hat{t}_c^i\mathcal{L}_{reg}\left(t_m^i,\hat{t}_m^i\right)\right) \tag{3}$$

where i represents the index of an oriented reference rectangle within the entire set of such rectangles, t_m signifies the offset that the grasp detection network predicts, $\hat{t}_m$ is the corresponding benchmark offset value, and N denotes the total number of positive oriented reference rectangles. $\mathcal{L}_{cls}$ denotes the cross-entropy loss, whereas $\mathcal{L}_{reg}$ signifies the L2 loss. Traditional methods optimize network weights by minimizing the above loss without considering the network architecture design. To compensate for this, we propose an innovative optimization approach for IRGs that integrates both network architecture design and weight training, which will be explained in detail in "Single-loop-optimized differentiable architecture search for grasp detection" section.

2.2.2 Single-loop-optimized differentiable architecture search for grasp detection

For grasp detection using the presented differentiable architecture search method (SLO-DARTS), the initial step is to establish the search space encompassing all potential neural network designs and their associated weights. Given the complexity and diversity of CNN network architectures, the automated search for the best CNN using conventional discrete optimization methods becomes a time-consuming and laborious task. Therefore, an improved DARTS method referred to as SLO-DARTS is proposed for IRG in the next step, which relaxes the discrete architecture optimization to be continuous. Compared with the original dual-loop DARTS, the proposed SLO-DARTS adopts a single-loop optimization process to replace the bilevel optimization process, which optimizes the network weights and architecture parameters on the same data batch as well as at the same time.

Search space: Following the related literature, a computation cell composed of an ordered sequence of nodes is searched and then stacked a certain number of times to construct a target CNN according to the complexity of the task. Two categories of cells require learning: the reduction cell and the normal cell. Unlike the normal cell that maintains the size and channel of input, the reduction cell cuts the size in half while expanding the channel by a factor of two.

The nodes within each cell are topologically organized, which could expand the architecture search space and avoid skipping the most promising CNN for IRGs compared to a node-chain structure framework (Hu et al., 2023). As shown in Fig. 3, assuming that there are B nodes in each cell, each node represents a feature map that saves the sum of results calculated by the respective operation $o(\cdot)\epsilon O$ of all its predecessors:

$$x^n = \sum_{m<n} o^{(m,n)}\left(x^m\right) \tag{4}$$

Here, $o^{(m,n)}$ denotes the operation performed on node *m*, depicted by the directed edge in Fig. 3. *O* represents the operation space, which includes the following candidate operations: skip-connect; separable convolutions; dilated separable convolutions; and standard convolutions. Two kernel sizes are used for the above convolution operation: 3 × 3 and 5 × 5. Hence, the optimal network architecture search task can be transformed to identify the optimal operation across all cells and edges, as demonstrated by the highlighted solid arrows in Fig. 3.

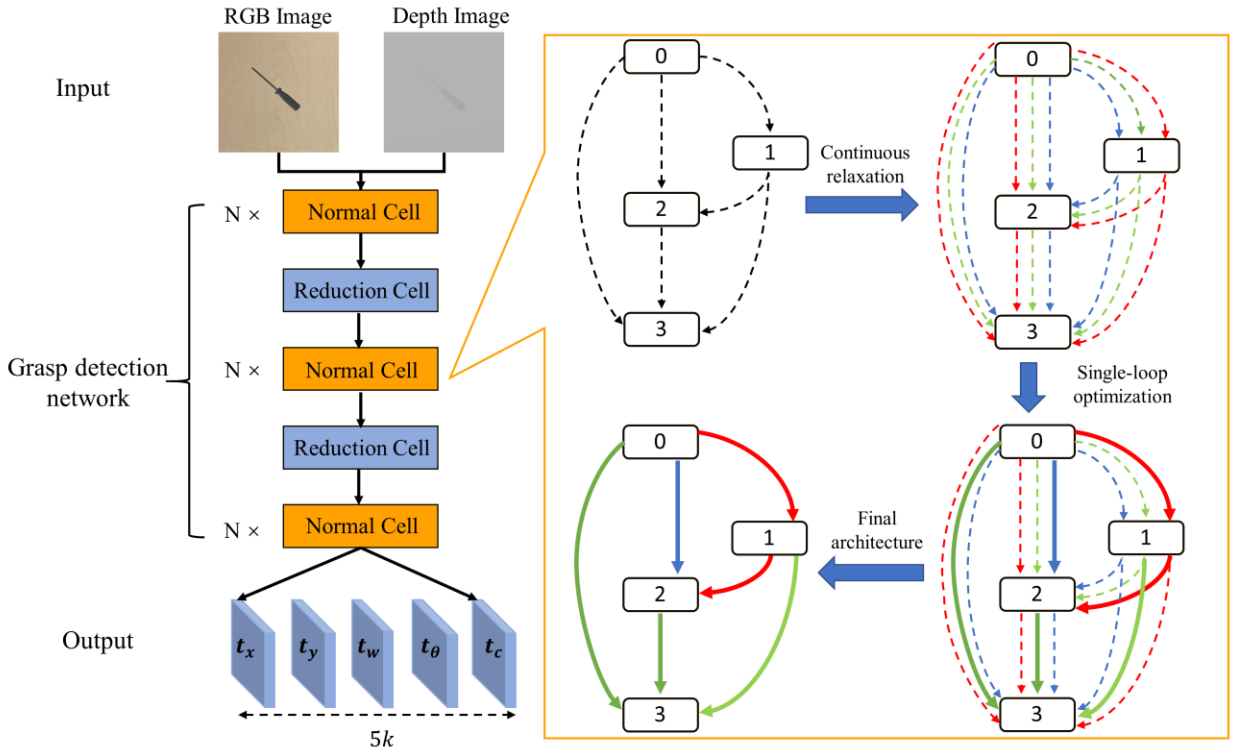


Fig. 3 Illustration of the procedure for implementing SLO-DARTS.

Optimization algorithm: Due to the ineffective and laborious process of evaluating network architectures individually inside the discrete search space, DARTS converts the categorical choice of operations on each edge into a weighted sum across all operations in *O* for the purpose of making the space continuous:

$$\bar{o}^{(m,n)}(x^m) = \sum_{o \in O} \frac{\exp\left(\alpha_o^{(m,n)}\right)}{\sum_{o' \in O} \exp\left(\alpha_{o'}^{(m,n)}\right)} o\left(x^m\right) \tag{5}$$

where $\alpha_o^{(m,n)}$ is a continuous variable of operation *o* performed on node *m* representing the architecture parameter. The weight of each operation is a softmax of the architecture parameter $\alpha_o^{(m,n)}$, signifying the operation's relative contribution among all operations in *O*. The discrete architecture search task is simplified into the process of learning a continuous operation vector $\boldsymbol{\alpha} = \{\alpha^{(m,n)}\}$ after following the relaxation process described by Eq. (5). The final searched

architecture can be derived by substituting the mixed operation $\bar{o}^{(m,n)}$ on each edge with the highest-contributing operation (i.e., $o^{(m,n)} = \text{argmax}_{o \in O}\, \alpha_o^{(m,n)}$) once the optimal operation vector $\boldsymbol{\alpha}$ has been learned.

The original DARTS establishes a bilevel optimization framework to update the architecture parameters $\boldsymbol{\alpha}$ and the weights $\boldsymbol{\omega}$ based on gradient descent algorithm to obtain the optimal CNN:

$$\min_{\boldsymbol{\alpha}} \quad \mathcal{L}_{val}\left(\boldsymbol{\omega}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}\right) \tag{6}$$

$$\text{s.t. } \boldsymbol{\omega}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{\omega}}\left(\mathcal{L}_{tra}\left(\boldsymbol{\omega}, \boldsymbol{\alpha}\right)\right) \tag{7}$$

where $\mathcal{L}_{tra}$ and $\mathcal{L}_{val}$ represent the training loss and validation loss, respectively. In order to estimate the gradient of architecture parameters $\boldsymbol{\alpha}$ in a computationally inexpensive manner, the original DARTS proposed an approximate idea for the weight optimization (Eq. (7)) where a single training step is used to replace training to convergence for the purpose of reducing the computational cost:

$$\boldsymbol{\omega}^*(\boldsymbol{\alpha}) \approx \boldsymbol{\omega} - \xi \nabla_{\boldsymbol{\omega}} \mathcal{L}_{tra}\left(\boldsymbol{\omega}, \boldsymbol{\alpha}\right) \tag{8}$$

where ξ represents the learning rate for a single training step.

Although the original DARTS advocates the efficiency and superiority of bilevel optimization, it suffers from the following problems: (1) the inaccuracy of gradient estimation in the outer-level optimization leads to unstable convergence and performance drop. In fact, the non-learnable operations (i.e., skip-connect) likely dominate in the early stage of architecture searching, and this phenomenon may last until the end. Consequently, the final selected architecture's few trainable parameters and weak expressive power lead to reduced performance. (2) Only the training data are used to train $\boldsymbol{\omega}$, which could negatively affect the learning of the network weights. Since $\boldsymbol{\omega}$ is over-parameterized, it tends to align too closely with the training data but struggles to fit the validation data, leading to a wider gap between the training and validation errors. (3) Bilevel optimization is more time-consuming and memory-intensive than single-loop optimization. These difficulties drive us to employ single-loop optimization instead of bilevel optimization, where we combine

training data and validation data as a training set to update $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ on the same data batch simultaneously. The above single-loop optimization process can be formulated as:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\omega}} \quad \mathcal{L}_{tra}\left(\boldsymbol{\omega},\boldsymbol{\alpha}\right) \tag{9}$$

Here, the update rule of $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ is shown as:

$$\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t-1} - \eta_t \nabla_{\boldsymbol{\omega}} \mathcal{L}_{tra}\left(\boldsymbol{\omega}_{t-1}, \boldsymbol{\alpha}_{t-1}\right) \tag{10}$$

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} - \delta_t \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{tra}\left(\boldsymbol{\omega}_{t-1}, \boldsymbol{\alpha}_{t-1}\right) \tag{11}$$

where η_t and δ_t are the learning rates. The Adam optimizer and the SGD optimizer are adopted to update the operation variables $\boldsymbol{\alpha}$ and weights $\boldsymbol{\omega}$, respectively. After completing the above single-loop optimization, the value of $\boldsymbol{\alpha}$ is used to rank all candidate operations on each bridge. Next, the operation linked with the largest α on each edge is selected to derive the discrete optimal CNN. The last step consists of training the weights $\boldsymbol{\omega}$ of the formed CNN from scratch, so that the SLO-DARTS-optimized CNN is optimal in terms of architecture and weights. Algorithm 1 below displays the pseudocode of the SLO-DARTS method.

Algorithm 1 SLO-DARTS—Single-Loop-Optimized Differentiable Architecture Search

---

Require: The training dataset *tra*; the operation space *O*; total training iterations *epoch*; the learning rates η_t and δ_t.

Initialize the operation variables $\boldsymbol{\alpha} = \left\{\alpha^{(m,n)}\right\}$ and weights $\boldsymbol{\omega}$
Create a mixed operation $\bar{o}^{(m,n)}$ parametrized by $\alpha_o^{(m,n)}$ for each edge
for $t \leftarrow 0$ **to** *epoch* **do**
    Compute the loss $\mathcal{L}_{tra}(\boldsymbol{\omega}_{t-1}, \boldsymbol{\alpha}_{t-1})$
    Update weights $\boldsymbol{\omega}_t$ by descending $\eta_t \mathcal{L}_{tra}(\boldsymbol{\omega}_{t-1}, \boldsymbol{\alpha}_{t-1})$
    Update operation variables $\boldsymbol{\alpha}_t$ by descending $\delta_t \mathcal{L}_{tra}(\boldsymbol{\omega}_{t-1}, \boldsymbol{\alpha}_{t-1})$
end for
Select the optimal operation $o^{(m,n)} = \text{argmax}_{o \in O}\ \alpha_o^{(m,n)}$ on each edge
Derive the discrete optimal CNN architecture
Train the CNN weights from scratch

To assess the grasp detection accuracy of the optimized CNN, a rectangular metric similar to (Kumra et al., 2020) is used. If the rectangle that the optimized CNN predicted meets the requirements listed below, it is considered a successful prediction. They are: (i) the rotation angular difference of the benchmark grasp rectangle from the predicted one falls within π/6; and (ii) the Jaccard similarity coefficient between the benchmark and predicted grasp rectangle exceeds 25%. The formula of the Jaccard similarity coefficient can be represented as follows:

$$J(U,V)=\frac{U\cap V}{U\cup V} \tag{12}$$

Herein, U and V stand for the predicted and benchmark grasp rectangles, respectively. The numerator and denominator represent the intersection and union area of two rectangles.

2.3 Digital twins of IRGs based on sim-real collaborative learning

To achieve desirable predictive performance, the proposed SLO-DARTS of the grasp detection model requires large amounts of annotated training data. Although some public-open datasets (e.g., Cornell Grasp Dataset) for IRGs have been provided, there exist two problems: (1) the labels corresponding to the real-world images in some datasets are manually labeled rather than obtained by real robot grasping, leading to some nonnegligible error; and (2) the setups such as the grippers and objects in open datasets may not be exactly the same as ours. However, obtaining real-world annotated datasets to optimize CNNs is significantly tedious and costly according to previous studies (Levine et al., 2018). An appealing alternative is simulated training, but models trained in simulation environments cannot be directly applied to real-world scenarios because of the significant differences in texture, noise distribution, and light rendering between simulated and real-world images. In this section, an IRG digital twin (DT) shown in Fig. 4 is developed to avoid expensive real robot training and bridge the sim-to-real gap to further improve the accuracy of real-world IRG, which continues the learning of the SLO-DARTS-optimized CNN in both virtual and real space.

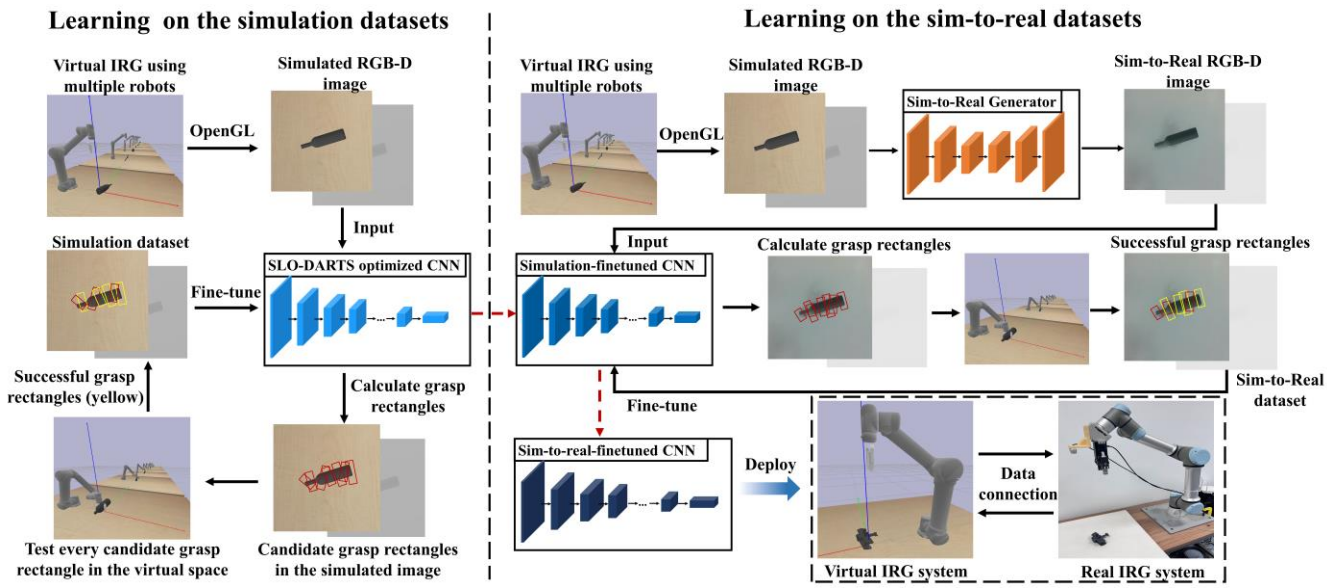


Fig. 4 Overview of DT pipeline developed for intelligent robotic grasping.

2.3.1 Digital twin system construction

The physical robotic grasping system and its corresponding virtual robotic grasping model need to be set up first. A robot arm, a two-finger gripper, a depth camera, and graspable objects comprise the physical hand-eye robot system, as depicted in Fig. 4. The virtual robotic grasping model simulated within the Bullet Physics Engine (Erez et al., 2015) includes two parts: (i) the geometry model including the robot arm, gripper and objects and (ii) the physics model for simulating the physical robotic action using robotic kinematics and dynamics theories. In addition, data communication between virtual and physical systems is carried out by means of the socket method based on TCP/IP protocol. In this way, the virtual robot can not only remotely display the real-time grasping process of its real counterpart but also directly control the real robot.

2.3.2 Sim-real collaborative learning strategies

Learning grasp detection model on the simulation datasets: After deploying the pretrained SLO-DARTS-optimized CNN in the virtual environment, we conduct the fine-tuning process of this network based on the simulation datasets generated by the multirobot grasping simulation. The only difference between the fine-tuning process and the training process shown in “Grasp

detection network based on oriented reference rectangles" section is that the benchmark rectangles are determined from the results of virtual grasping simulation. As depicted in Fig. 4, the specific steps for network fine-tuning are as follows:

1) Import CAD geometry models of the objects and robots into the virtual space built by the physics engine Bullet.
2) Place a randomly chosen graspable object in the virtual robot's area of view and then feed the SLO-DARTS-optimized CNN a simulated RGB-D image obtained by the OpenGL method to output a batch of grasp rectangles, each with a confidence score.
3) Select the predicted grasp rectangles with the top n_g confidence score rankings (n_g=10 in this paper) as the candidate grasp rectangles.
4) Plan and execute the grasp of the virtual robot based on the candidate grasp rectangles. If the virtual robot is capable of grasping and lifting the target object to a certain height (0.1 m) and keep it from falling for a period of time (0.5 seconds), this candidate grasp rectangle can be considered as a successful sample; otherwise, it is a failure sample. Here, each target object is assigned with a small mass, and gravity is simulated during the virtual grasping process.
5) Gather all the successful rectangles as the benchmarks and then regard them and the corresponding RGB-D image as a simulation dataset to fine-tune the SLO-DARTS-optimized CNN.
6) Deploy multiple virtual robots (8 in this study) at the same time in the virtual environment to conduct Steps 2) to 5) simultaneously to speed up the training process.

Learning grasp detection model on the Sim-to-Real datasets: Despite our efforts on producing similar hardware setup in the virtual space, there are still visual gaps in the domain between simulated and real-world IRG scenarios. However, obtaining real-world annotated datasets is laborious and time-consuming, as it is not only difficult to control multiple robots simultaneously but also impossible to place the object at the same position before performing every candidate grasp. To tackle the above issues, Grasp-CycleGAN, a task-aware domain adaptation technique that integrates CycleGAN (Zhu et al., 2017) and implements the adaptation of simulated images into real-world images while maintaining the semantic features related to the grasp detection task, is proposed. In this way, the visual appearance gaps between simulated and real-

world images can be effectively bridged, allowing the network to be further fine-tuned based on sim-to-real datasets.

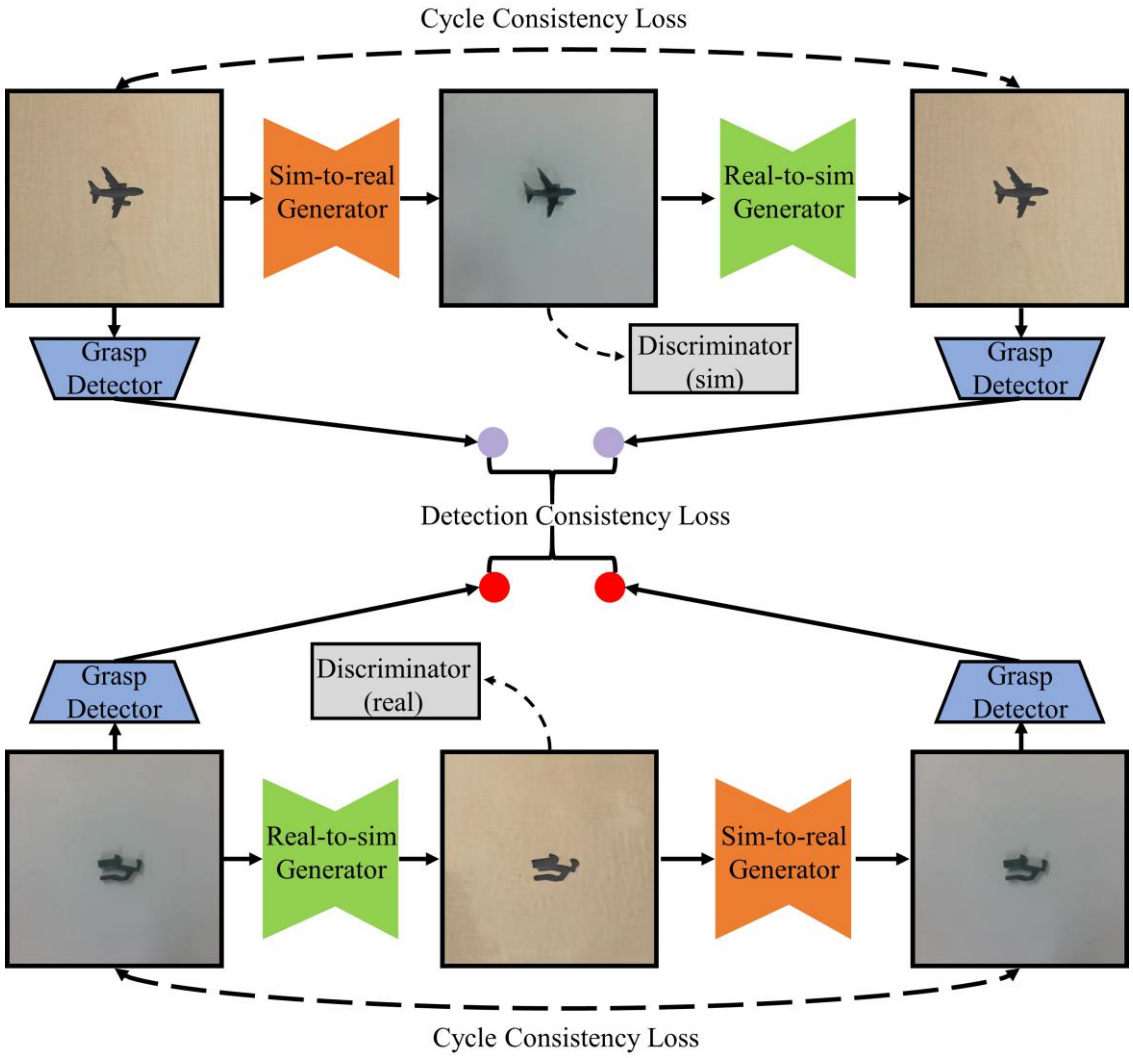


Fig. 5 Diagram of Grasp-CycleGAN stages. Top: a simulated image is adapted by a sim-to-real generator to be realistic, and then converted to a simulated image again by a real-to-sim generator. Bottom: the real image branch follows the same pipeline as the simulated image.

Similar to CycleGAN, the proposed Grasp-CycleGAN learns bidirectional mapping functions between two image domains (i.e., simulation and real in this paper), *S* and *R*, from unpaired examples $\{s_i\}_{i=1}^{N}$ and $\{r_i\}_{i=1}^{N}$ without labels. As shown in Fig. 5, the Grasp-CycleGAN comprises

two generators $G_{RS}: R \rightarrow S$ and $G_{SR}: S \rightarrow R$, where G_{RS} transforms images to the simulation style and G_{SR} transforms images to the real style. In addition, two discriminators, D_S and D_R, are introduced to be responsible for classifying images to the correct domain. For training Grasp-CycleGAN, the loss function of Grasp-CycleGAN can be expressed by the following formula:

$$\mathcal{L}_{Grasp-CycleGAN}\left(G_{RS}, G_{SR}, D_S, D_R\right) = \mathcal{L}_{CycleGAN}\left(G_{RS}, G_{SR}, D_S, D_R\right) + \lambda_{dec}\mathcal{L}_{dec}\left(s, r, G_{RS}, G_{SR}\right) \quad (13)$$

Here, the first term $\mathcal{L}_{CycleGAN}$ represents the loss of the original CycleGAN, which comprises three components (Yu, 2022) and can be written as:

$$\begin{aligned}\mathcal{L}_{CycleGAN}\left(G_{RS}, G_{SR}, D_S, D_R\right) = \mathcal{L}_{GAN}\left(G_{RS}, D_S, R, S\right) + \mathcal{L}_{GAN}\left(G_{SR}, D_R, S, R\right) \\ + \mathcal{L}_{cyc}\left(G_{RS}, G_{SR}\right) + \mathcal{L}_{\text{identity}}\left(G_{RS}, G_{SR}\right)\end{aligned} \quad (14)$$

The adversarial loss $\mathcal{L}_{GAN}$ is applied independently to each mapping function to align the style of the generated images with the target style, which can be formulated as:

$$\mathcal{L}_{GAN}\left(G_{RS}, D_S, R, S\right) = \mathbb{E}_{s\sim p_{\text{data}}(S)}\left[\log D_s\left(s\right)\right] + \mathbb{E}_{r\sim p_{\text{data}}(R)}\left[\log\left(1 - D_S\left(G_{RS}\left(r\right)\right)\right)\right] \quad (15)$$

$$\mathcal{L}_{GAN}\left(G_{SR}, D_R, S, R\right) = \mathbb{E}_{r\sim p_{\text{data}}(R)}\left[\log D_R\left(r\right)\right] + \mathbb{E}_{s\sim p_{\text{data}}(S)}\left[\log\left(1 - D_R\left(G_{SR}\left(s\right)\right)\right)\right] \quad (16)$$

The introduction of the cycle consistency loss $\mathcal{L}_{cyc}$ serves the purpose of preventing the input image from being mapped to an arbitrary permutation of target-style images. In other words, the mapping is reversible, ensuring consistency of the original input image to its reconstructed one after processing through two generators. Thus, the cycle consistency losses can be described by the following equation:

$$\mathcal{L}_{cyc}\left(G_{RS}, G_{SR}\right) = \mathbb{E}_{r\sim p_{\text{data}}(R)}\left[\left\|G_{SR}\left(G_{RS}\left(r\right)\right) - r\right\|_1\right] + \mathbb{E}_{s\sim p_{\text{data}}(S)}\left[\left\|G_{RS}\left(G_{SR}\left(s\right)\right) - s\right\|_1\right] \quad (17)$$

The identity mapping loss ensures that generators maintain the content of an input image when it is already in the target domain, expressed as:

$$\mathcal{L}_{\text{identity}}\left(G_{RS}, G_{SR}\right) = \mathbb{E}_{s\sim p_{\text{data}}(S)}\left[\left\|G_{RS}\left(s\right) - s\right\|_1\right] + \mathbb{E}_{r\sim p_{\text{data}}(R)}\left[\left\|G_{SR}\left(r\right) - r\right\|_1\right] \quad (18)$$

The second term of Eq. (13) is called the detection consistency loss, designed to penalize inconsistencies in grasp detection results (represented by the grasp rectangle corresponding to the

highest confidence score) when comparing the original input image with the generator's recovered one. This loss term can be viewed as a complement to the cycle consistency loss, enabling CycleGAN further to preserve more features relevant to the grasp detection task. As depicted in Fig. 5, we suppose a simulated image $s \in S$, and then the detection consistency loss can be calculated with the aid of a pre-trained SLO-DARTS-optimized CNN as:

$$\begin{aligned} rect_s &= CNN\left(s\right) \\ rect_{G_{RS}\left(G_{SR}\left(s\right)\right)} &= CNN\left(G_{RS}\left(G_{SR}\left(s\right)\right)\right) \\ \mathcal{L}_{\text{dec}}\left(s, G_{RS}\left(G_{SR}\left(s\right)\right)\right) &= \mathcal{L}_{\text{reg}}\left(rect_s, rect_{G_{RS}\left(G_{SR}\left(s\right)\right)}\right) \end{aligned} \tag{19}$$

where $\mathcal{L}_{reg}$ is the L1 loss and $rect = \{x, y, w, \theta\}$ represents the grasp rectangle predicted by the SLO-DARTS-optimized CNN whose weights are frozen and will not be updated with the training of the Grasp-CycleGAN. Similarly, for a real image $r \in R$, G_{RS} and G_{SR} should also satisfy the grasp detection consistency, so the entire detection consistency loss can be written as:

$$\begin{aligned} \mathcal{L}_{\text{dec}}\left(s, r, G_{RS}, G_{SR}\right) &= \mathcal{L}_{\text{dec}}\left(s, G_{RS}\left(G_{SR}\left(s\right)\right)\right) + \mathcal{L}_{\text{dec}}\left(r, G_{SR}\left(G_{RS}\left(r\right)\right)\right) \\ &= \mathbb{E}_{s\sim p_{\text{data}}(S)}\left[\left\| rect_{G_{RS}\left(G_{SR}\left(s\right)\right)} - rect_s \right\|_1\right] + \mathbb{E}_{r\sim p_{\text{data}}(R)}\left[\left\| rect_{G_{SR}\left(G_{RS}\left(r\right)\right)} - rect_r \right\|_1\right] \end{aligned} \tag{20}$$

Algorithm 2 provides a detailed description of the overall training process for the Grasp-CycleGAN. After the Grasp-CycleGAN is trained, the sim-to-real generator G_{SR} will be deployed in the virtual environment and used to transfer the simulated RGB-D images into very realistic RGB-D images, as depicted in Fig. 4. Here, the separate Grasp-CycleGANs are trained for RGB images and depth images respectively, that is, there are two sim-to-real generators being used in the process of converting simulated images into real-like images. Following the concatenation of converted real-like RGB and depth images along the channel dimension, realistic RGB-D images are formed. These images are then fed into the network fine-tuned on the simulation datasets to predict candidate grasping rectangles. Afterwards, the successful grasp rectangles are selected based on the same criteria as those in Step 4 to form a new sim-to-real dataset with the transferred realistic RGB-D images. Similarly, the above steps are performed simultaneously on multiple robots to speed up the new sim-to-real dataset generation. Finally, we perform further fine-tuning

of the network using the sim-to-real dataset, after which the final network is applied to the robotic grasping task in the real world.

Algorithm 2 Grasp-CycleGAN Training Algorithm

Require: Pre-trained SLO-DARTS-optimized *CNN*; randomly initial parameters $\theta_{G_{RS}}$, $\theta_{G_{SR}}$, θ_{D_S}, θ_{D_R} of G_{RS}, G_{SR}, D_S, D_R; real image domain R; simulation image domain S; the number of training iterations *epoch.*

Initialize empty buffer B_S, B_R to store the generated simulation and real images
for $t \leftarrow 0$ **to** *epoch* **do**
    Sample a mini-batch of s and r from image domains S and R
    Save the generated images $G_{SR}(s)$, $G_{RS}(r)$ in buffer B_R, B_S
    Compute the loss $\mathcal{L}_{Grasp-CycleGAN}(G_{RS}, G_{SR}, D_S, G_R)$
    Update parameters $\theta_{G_{RS}}$, $\theta_{G_{SR}}$ of two generators G_{RS}, G_{SR}
    Sample a mini-batch of s, r, $G_{SR}(s)$, $G_{RS}(r)$ from S, R, B_R, B_S
    Compute the loss $\mathcal{L}_{GAN}(G_{RS}, D_S, R, S)$
    Update parameters θ_{D_S} of the discriminator D_S
    Compute the loss $\mathcal{L}_{GAN}(G_{SR}, D_R, S, R)$
    Update parameters θ_{D_R} of the discriminator D_R
end for
Return G_{RS}, G_{SR}, D_S, D_R

3. Experiments

3.1 Setup and dataset

Hardware and Software: The hand-eye robot system in Fig. 1(b) used to perform real-world IRG experiments comprises a 6-DOF UR5E robot arm with an adaptive electric two-finger gripper model AG-160-95 mounted at its end. The robot arm's wrist is fitted with an Intel RealSense D-435i camera for capturing RGB-D images. Before each grasping attempt, the object to be grasped needs to be placed in a random position and posture within the operational space of the robot arm, which is an area of approximately 450 × 600 mm within the range of the camera. The initial height

of the camera needs to be set appropriately, ensuring full coverage of the object waiting to be grasped within its field of view. The SLO-DARTS experiments are carried out on a high-performance desktop running Ubuntu 22.04 operating system and installed NVIDIA GeForce RTX 4090 GPU. The DT training procedure and related tests are carried out on another high-performance desktop running Ubuntu 20.04 operating system and equipped with NVIDIA RTX 3090 GPU. The proposed grasp detection model and optimization algorithm are both implemented with Torch (Collobert et al., 2011) for its flexibility.

Dataset: The Cornell Grasp Dataset provided by Jiang et al. (2011) including 885 RGB-D images is used for CNN training and testing in the SLO-DARTS experiments. Each image is annotated with multiple successful and failure grasp rectangles.

For DT learning, 900 virtual 3D models chosen at random from 3DNet (Wohlkinger et al., 2012) are used. Out of these 900 objects, 600 are employed in the creation of simulation datasets and sim-to-real datasets for network fine-tuning, while the remaining 300 are used to test the accuracy of virtual robotic grasping.

The objects for physical robotic grasping are partly taken from the adversarial object set generated by the Dexterity Network (Dex-Net) (Mahler et al., 2017), which imposes higher demands for accurate grasping due to their intricate shapes. We print these objects with an adversarial geometry on a 3D printer according to their 3D models available online (Fig. 6(a)). In addition, some objects with distinct characteristics coming from both office and household scenes (Fig. 6(b)) are chosen to further demonstrate our network's grasping ability in the real world.

(a) (b)

Fig. 6 The objects employed in real-world grasping scenario. (a) adversarial objects; and (b) office & household objects.

3.2 SLO-DARTS experiments

As seen in "Grasp detection network based on oriented reference rectangles" section, the value of *k* representing the number of oriented reference rectangles is of significant importance to the network's prediction accuracy. The larger *k* is, the closer the benchmark rectangle and matched oriented reference rectangle are, which benefits the convergence of the regression loss function in Eq. (3). However, a larger *k* also results in the aggravation of the imbalance between positive and negative labels, thus negatively impacting classification loss. Hence, to identify an appropriate value of constant *k*, different values of *k* (3, 4, 6 and 9) are employed to train the CNNs. Each bridge in them selects the operation corresponding to the highest computational complexity (i.e., standard convolutions with filter sizes of 5 × 5), with floating-point operations (FLOPs) being used to evaluate the complexity. After determining the value of *k*, the automatic architecture search processes based on the proposed SLO-DARTS and the original DARTS can be carried out.

For the original DARTS using a bilevel optimization framework, the Cornell Grasp Dataset is divided into three parts: the training set (400 images), the validation set (400 images) and the

testing set (85 images), where the training set is employed for updating weights, the validation set for updating operation variables, and the testing set for testing model accuracy at each iteration during the CNN training process. For the SLO-DARTS using a single-loop optimization framework, the above training and validation set are combined together to form a new training set (800 images) for updating the weights and architecture parameters simultaneously. Once the optimal architecture has been found, randomly initialize the CNN weights, and training is conducted from scratch on the training set (800 images partitioned from the Cornell Grasp Dataset). Subsequently, the CNN performance is assessed using the testing set (85 images partitioned from the Cornell Grasp Dataset).

As a result of the graphic memory limitation, the original DARTS and SLO-DARTS are applied to optimize a CNN with 5 cells, where each cell consists of 3 nodes. The candidate operations in O for the bridge between every two nodes have been introduced in "Single-loop-optimized differentiable architecture search for grasp detection" section, where each operation is of one stride. Following the same setup as the relevant studies, the sequence of ReLU-Conv-BN is used for convolutional operations, with reduction cells located in one-third and two-thirds of the network. Batch size is 8 during the architecture optimization process. After obtaining the best architecture, we retrain the CNN weights from scratch using a batch size of 8.

3.3 Grasping experiments based on DT learning

Prior to the DT learning, the weights of the CNN with the optimal architecture formed based on SLO-DARTS are pretrained using the Cornell Grasp Datasets. To enhance the network's suitability for real robot grasping, further training of the weights using the proposed DT framework is necessary. 900 randomly selected CAD models from the 3DNet dataset are employed for the DT learning process, among which 600 objects are utilized as the training set to further update CNN weights, and the remaining 300 are utilized to test the virtual grasping accuracy of CNN. In the first stage of DT learning, we follow the steps outlined in "Sim-real collaborative learning strategies" section to generate labeled simulation datasets for the initial fine-tuning of the network, of which 1033 are employed for training and 109 for testing. In the second stage of DT learning, accounting for the visual gaps between simulation and reality, the proposed Grasp-CycleGAN is integrated into our DT framework to adapt the simulated images to realistic images to generate

labeled sim-to-real datasets for further network fine-tuning (1022 for training and 108 for testing). It is worth noting that the Grasp-CycleGAN needs to first be trained on the unpaired simulation and real datasets without labels, where the former are gathered from the virtual environment via the RGB-D rendering algorithm, while the latter are collected from the real world by the depth camera. There are 1035 RGB-D images in both the simulation and real datasets. For training, select at random 1000 RGB-D images from each dataset, while keeping the rest ones as the testing set. The network architecture of the generator in Grasp-CycleGAN refers to works from (Johnson et al., 2016), which contains two stride-2 convolutions, several residual blocks, and two upsample operations based on the nearest neighbor algorithm. For the network architecture of the discriminator, PatchGAN (Ledig et al., 2017) is used to partition the image into multiple overlapping areas, known as "patches", and then classify whether each patch is real or fake. Adam algorithm is employed in the training process of Grasp-CycleGAN with a batch size set to 1.

To accelerate the DT-training process, 8 virtual robot arms are deployed at the same time into the Bullet for virtual grasping, that is, the batch size is 8. It is worth mentioning that the DT learning process could adapt the CNN to better fit the real-world robotic grasping through mimicking the real-world scenario compared to CNN training based only on the Cornell Grasp Dataset. That's because there may be inevitable errors in the human-labelled grasp rectangles. After completing the DT-learning process, the fine-tuned CNN will be then transferred to perform real-world robot grasping tasks. As can be seen from Fig. 6(a) and (b), the real robotic grasping test involves the use of unknown objects, which include adversarial and household objects. The effectiveness of our method is assessed through two robot grasping task scenarios. The first involves the grasping of a static and single object, where any selected object is randomly positioned within the robot's field of view before every grasp. Grasping in clutter is the second, where a set of graspable objects (10 each time) are randomly put in the workplace before performing grasp. Even if only single object datasets are used for model training, the multiple-grasp prediction capability enables our grasp detection model to adapt to more complex scenarios as well, such as grasping in cluttered scenes. The grasping metrics for the above two scenarios are the same, that is, a grasp in the real world is considered successful when the gripper grabs and lifts the target object and then moves it to a specified area. Notably, there is no requirement for grasping a specific target in cluttered

scenes, so the order of grasping is not considered in this work. The robot will cease grasping only when all objects within the workspace have been grasped.

4. Results and discussion

4.1 SLO-DARTS experiments

The testing accuracy and the model size of CNNs under the different numbers of oriented reference rectangles k are listed in Table 1. Except for the output convolutional layer, all CNNs are built in the same way where each bridge selects the operation with the highest FLOPs. The model size increases with the value of k due to the positive correlation between the number of convolution filters in the output layer and the value of k. Different values of k also mean different angle differences between the two adjacent oriented reference rectangles. As Table 2 shows, the CNN exhibits the best testing accuracy when k is taken as 6 (i.e., the angle difference is $\pi/6$), surpassing those of CNNs with other values of k. The testing accuracy progressively improves as the value of k increases due to that the matched oriented reference rectangle is closer to the benchmark grasp rectangle. The smaller the offset to be predicted between the benchmark rectangle and matched oriented reference rectangle, the more beneficial it is for network regression learning. However, an excessively large value of k, for instance, k = 9, leads to a heightened imbalance between positive and negative labels, which makes classification difficult and lowers the accuracy of the CNN, as shown in Table 1. Hence, in the subsequent optimization experiments, k = 6 is selected.

TABLE 1 Comparison of the accuracy and model size of CNNs under different values of k.

The number of oriented reference rectangles k	3	4	6	9
Accuracy	0.8788	0.9293	0.9697	0.9596
Model size (MB)	0.388	0.389	0.390	0.391

Table 2 provides the comparison results including the testing accuracy and the overall optimization time among original DARTS, DARTS-F (Hu et al., 2023) and SLO-DARTS optimization methods. Here, the testing accuracy is obtained on the testing set (85 images) from the CNN with the best architecture and weights. Given the variance of model performance even under the same training setup, a total of 15 independent experiments for each method are carried out. It is obvious that the average testing accuracy of the optimized CNN generated by SLO-DARTS (98.45%) is higher than those obtained by original DARTS and DARTS-F. The above phenomena can be explained by the fact that a lot of skip-connect operations appear in the DARTS-optimized CNN, therefore the network is too shallow to learn visual features well, as shown in Fig. 9(b). Although DARTS-F method enhances performance by allowing inner-level optimization to converge completely, its essence remains based on a bilevel optimization framework, thus unable to fundamentally address the aforementioned issue. In addition, benefiting from the synchronous update of $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ instead of the sequential update, SLO-DARTS optimization takes 59% less time overall compared to the original DARTS. Due to adopting the new convergence criteria to replace a single training step scheme, DARTS-F does not have an advantage in terms of optimization time. To more intuitively compare three optimization methods, boxplots are used to depict the distribution of model performance, which can be seen in Fig. 7. In a statistical view, the disparity between the upper and lower quartiles of SLO-DARTS is smaller than that of the other methods, indicating a more concentrated performance distribution, and hence, better robustness.

Fig. 8 presents the performance comparison on the Cornell Grasp Dataset for CNNs optimized by DARTS, DARTS-F and SLO-DARTS, as well as four representative manually designed CNNs (AlexNet, VGG16, ResNet34 and ResNet101). It is easy to see that compared to other manually designed CNNs, the SLO-DARTS-optimized CNN shows its superiority in terms of both accuracy and model size in grasp detection tasks. Note that the model size of the SLO-DARTS-optimized CNN is reduced by about 2 orders of magnitude, which corresponds to a significant increase in computational efficiency.

TABLE 2 Comparison of results between DARTS, DARTS-F and SLO-DARTS optimization methods.

	DARTS	DART-F	SLO-DARTS
Accuracy of the optimized CNN (%)	97.10±1.27	97.78±1.12	98.45±0.73
Time of the optimization (hours)	1.30	9.25	0.82

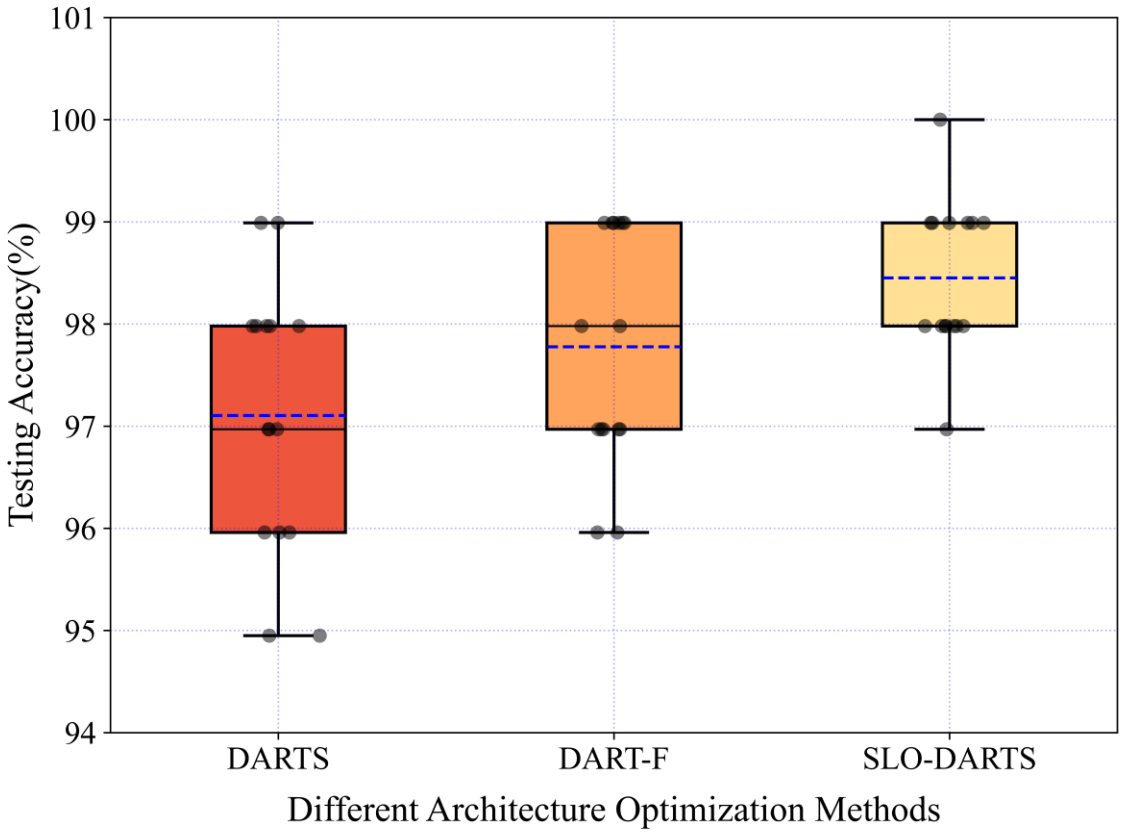


Fig. 7 Distributions of performance for DARTS, DARTS-F and SLO-DARTS optimization methods on the Cornell Grasp Dataset.

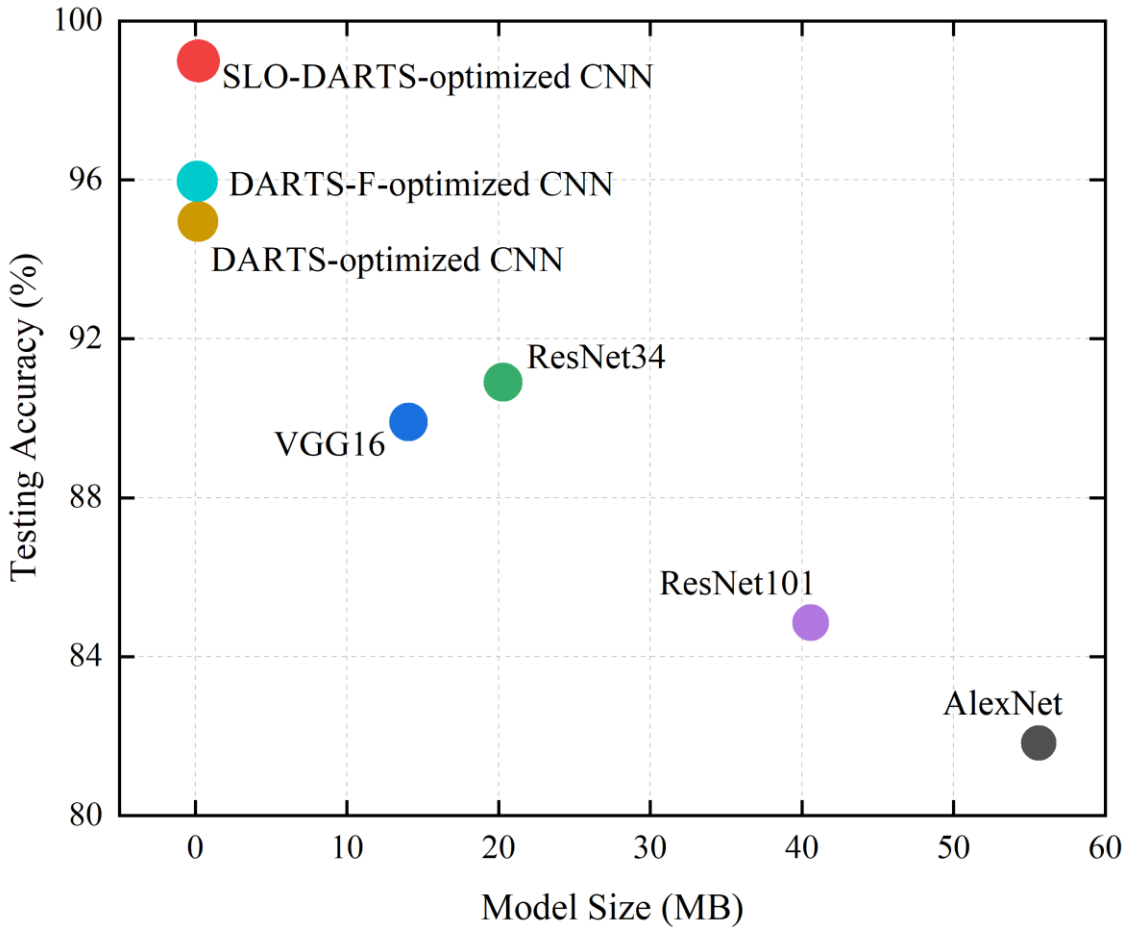
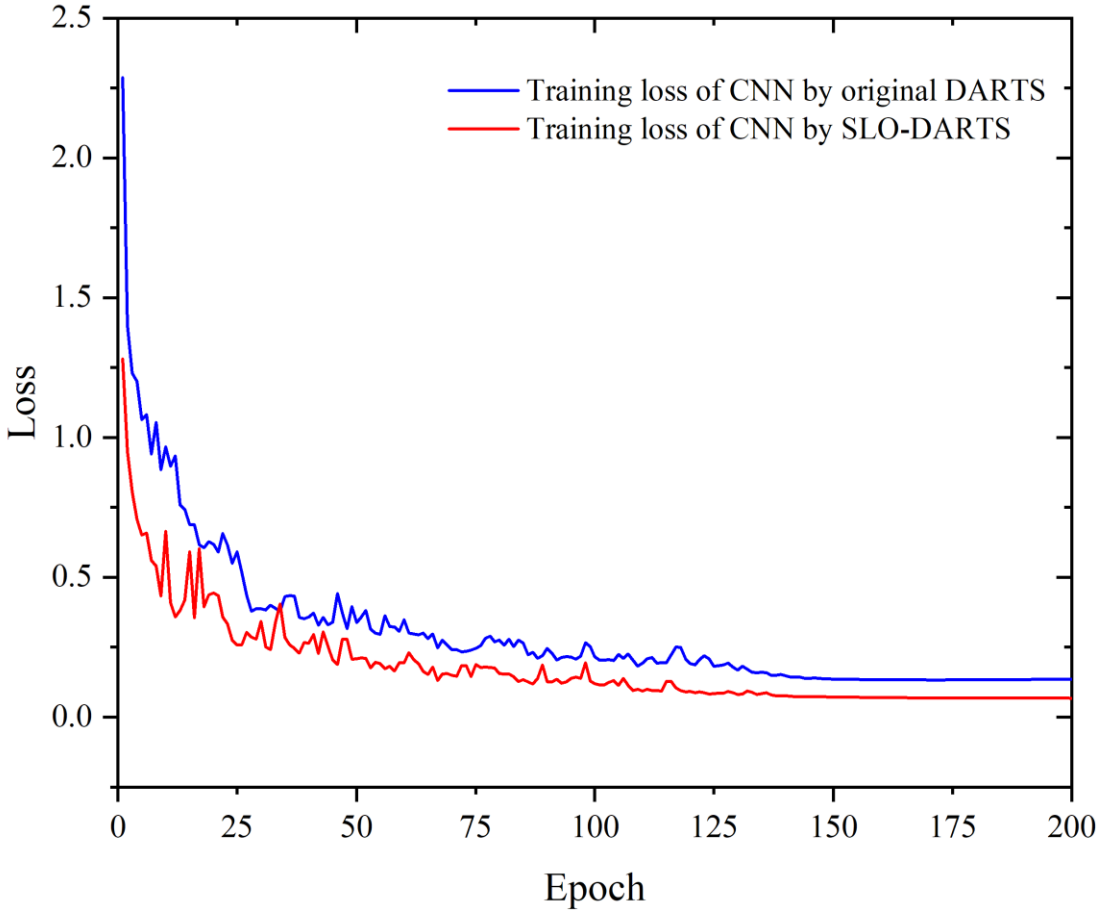
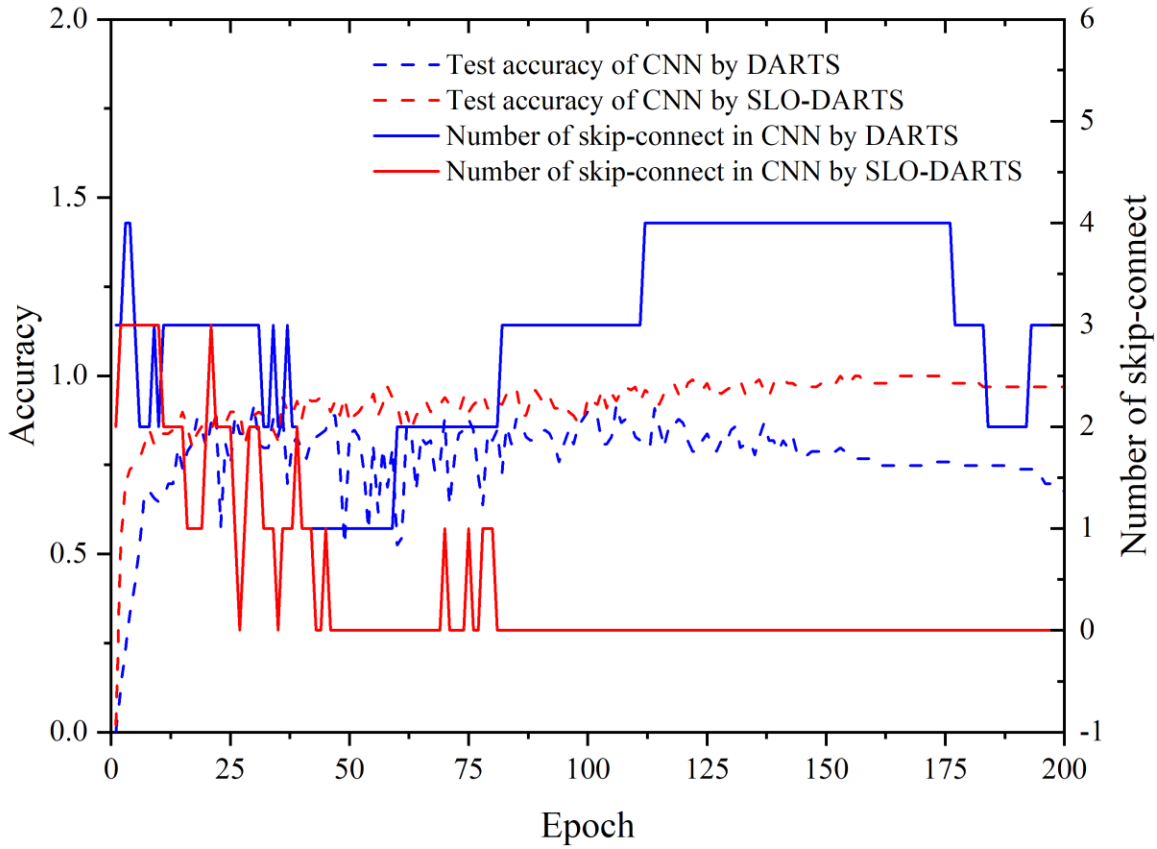


Fig. 8 Performance comparison results for CNNs optimized by DARTS, DARTS-F and SLO-DARTS, as well as four manually designed CNNs on the Cornell Grasp Dataset.


Training loss of CNN by original DARTS
Training loss of CNN by SLO-DARTS
Loss
Epoch
2.5
2.0
1.5
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200

(a)



(b)

Fig. 9 The optimization process of the original DARTS and the proposed SLO-DARTS. (a) The training loss at different epochs; and (b) the test accuracy and the number of skip-connect operations in both the normal and reduction cells of CNNs at different epochs.

Fig. 9 provides the trends in training loss value, test accuracy and number of skip-connect operations of CNNs during the optimization process by the DARTS (the blue curve) and the proposed SLO-DARTS (the red curve). Both training loss consistently decreases with the increase in epochs. It is also found that the proposed SLO-DARTS brings faster convergence and lower training loss to the optimization problems compared with the original DARTS. From Fig. 9(b), it is clear that many skip-connect operations (see the solid blue line in Fig. 9(b)) are involved in the architecture during the DARTS optimization process, leading to poor performance, indicated as the blue dashed line in Fig. 9(b). In contrast, the skip-connect operations gradually decrease to

zero in number within the selected architecture by SLO-DARTS, demonstrating the stability and superiority of a single-loop optimization framework. Notably, the number of skip-connect operations in the entire network architecture is several times that of the cell shown in Fig. 9(b). The selected cells and overall architecture by SLO-DARTS are depicted in Fig. 10.

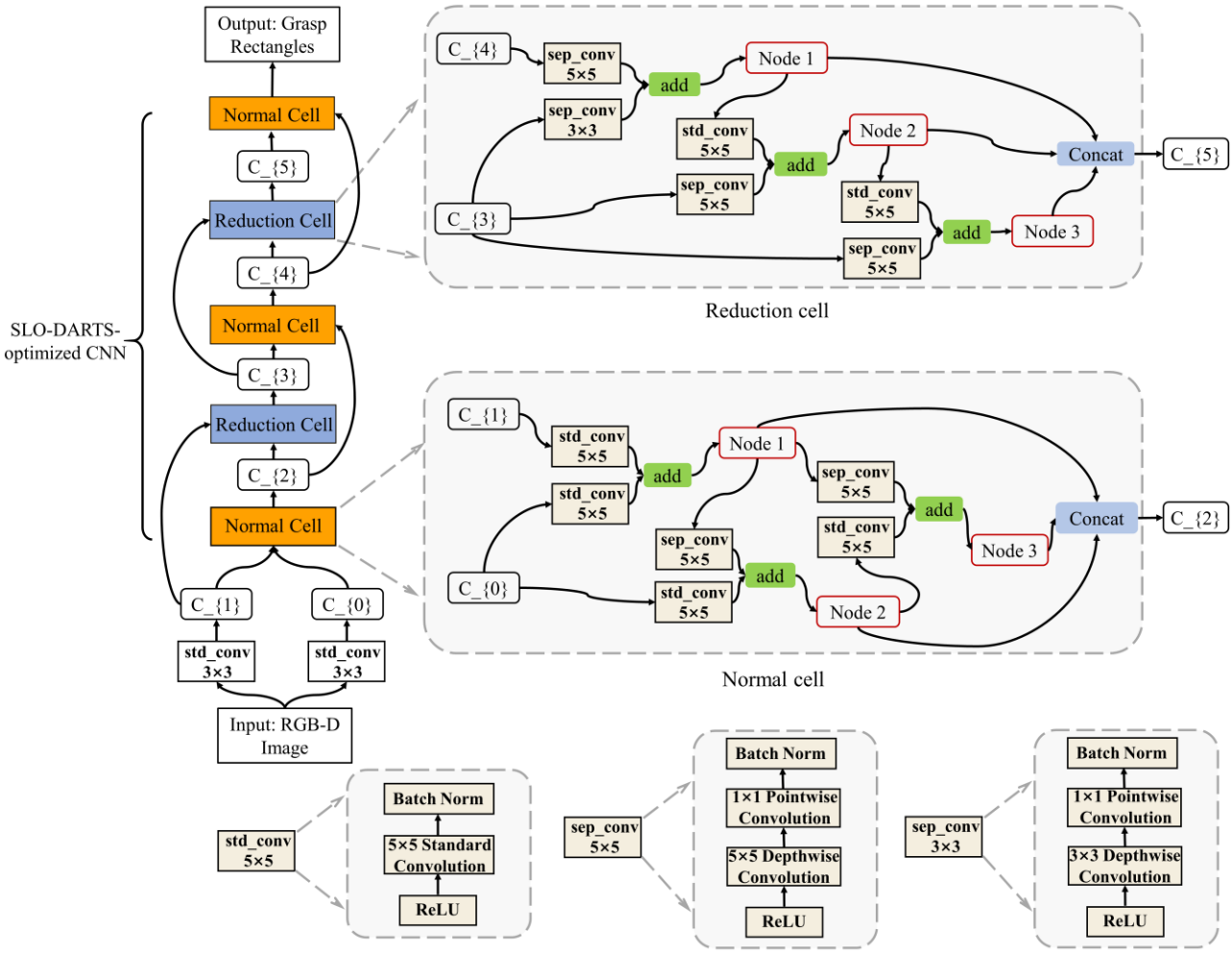


Fig. 10 The overall architecture and cells learned by SLO-DARTS.

4.2 Grasping experiments based on DT learning

To assess the image mapping capabilities of the proposed Grasp-CycleGAN, several standard metrics, such as the mean square error (MSE), peak signal-to-noise ratio (PSNR) (Winkler & Mohandas, 2008), structural similarity (SSIM) (Wang et al., 2004) and multi-scale structural

similarity (MS-SSIM) (Wang et al., 2003), are calculated between the simulated image s and the recovered one $G_{RS}\big(G_{SR}(s)\big)$. Table 3 lists the comparison results in the same datasets between the original CycleGAN, RetinaGAN (Ho et al., 2021) and our Grasp-CycleGAN. Here, MSE is the expected value for calculating the squared differences between corresponding pixels of two images, therefore a higher similarity between images can be indicated by a lower MSE. PSNR, SSIM, and MS-SSIM are responsible for quantifying image similarity, with larger values being better. It is clear that our Grasp-CycleGAN model obtains both higher PSNR, SSIM and MS-SSIM values along with lower MSE values, which indicates its better capability in image translation. To demonstrate the mapping effect more intuitively, some visual comparative results on mapping simulated images to real images among the three methods are given, as shown in Fig. 11. While RetinaGAN slightly improves transformations of the object and background compared to CycleGAN by introducing an object detection consistency loss, it relies on an additional object detector that needs to be trained on simulated and real images manually annotated with bounding box labels. In addition, RetinaGAN is only responsible for compensating object-level visual domain differences, which contributes little to the grasping task that focuses on local features of the graspable objects. It can be clearly observed that our Grasp-CycleGAN excels in both image transformation to the real style and the preservation of information about the features of graspable objects, especially those detailed features related to the grasping task.

TABLE 3 Comparison of the image mapping performances between CycleGAN, RetinaGAN and Grasp-CycleGAN under different metrics.

	MSE	PSNR	SSIM	MS-SSIM
CycleGAN	4.4378	41.8673	0.9914	0.9960
RetinaGAN	4.6084	41.6222	0.9910	0.9963
Grasp-CycleGAN	3.5517	42.7896	0.9926	0.9975

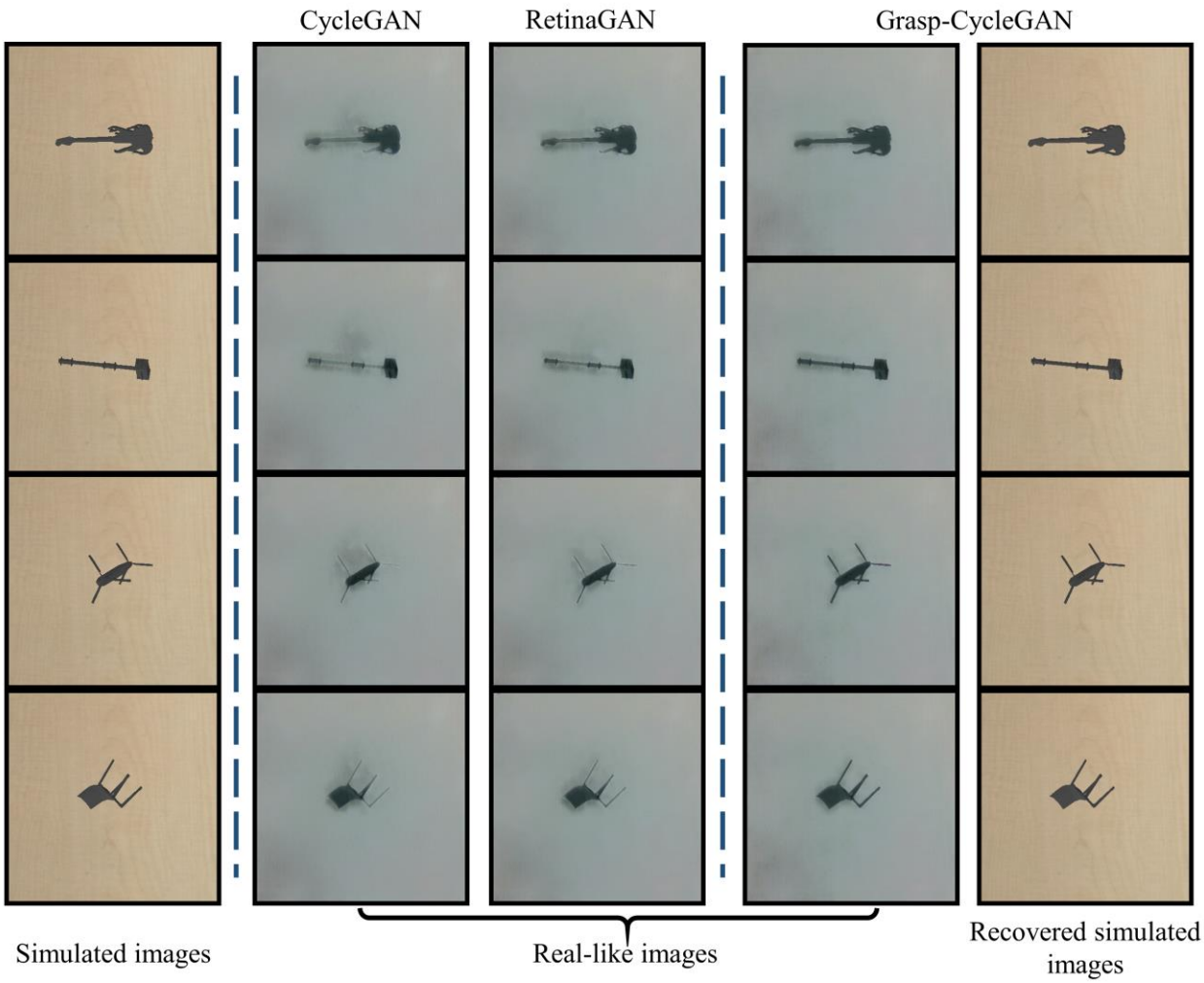


Fig. 11 Results of mapping simulated images to real-like images. The leftmost column: the original simulated images. The middle three columns: the real-like images generated by three different G_{SR}. The right most: the recovered simulated images generated by G_{RS} in the Grasp-CycleGAN.

The trends in CNN virtual grasping accuracy and training loss when fine-tuned by using the training set of the simulation datasets are shown in Fig. 12. After the SLO-DARTS-optimized CNN is fine-tuned based on the simulation datasets, there is a noticeable improvement in the virtual grasping accuracy, rising from 68% to 85%. This owes to the supplementary training data on one hand, and on the other hand, to the fact that the grasp rectangles labeled based on the results of virtual grasping are more accurate and effective than the human-labeled grasp rectangles in the

Cornell Grasp Dataset. Additionally, the virtual grasping accuracy on the testing set (300 objects from the 3DNet dataset) is also significantly improved from 65% to 82%, which indicates the strong generalization of the fine-tuned network for unknown objects.

Table 4 presents the robotic grasping results in the physical world using the SLO-DARTS-optimized CNN, the simulation-finetuned CNN and the sim-to-real-finetuned CNN. It can be seen that as the SLO-DARTS-optimized CNN is sequentially fine-tuned on the simulated and sim-to-real datasets, the real-world robotic grasping accuracies for single and cluttered objects are greatly improved on both the single and cluttered objects. The poorer performance of the simulation-finetuned CNN when compared to that of the sim-to-real-finetuned CNN could be attributed to a significant gap between simulation and reality, resulting in the network that is purely fine-tuned in the simulation environment not being able to be directly transferred to the real environment. Compared with the household objects, the adversarial objects manufactured by 3D printing have more complex geometry, leading to lower accuracies of real robotic grasping.

There are still many grasping attempts that have failed in real robotic grasping, as shown in Table 4. The possible reasons could be as follows: (i) The grasp height along the z-axis is not taken into account in our method, so even if the predicted rotation angle and opening width for grasping objects are correct, the grasp may fail. Particularly for adversarial objects with intricate shapes and some cylindrical objects, an incorrect grasp height can result in the grasp that does not meet the force-closure condition required for a successful grasp, as depicted in Fig. 13(a). (ii) Our method mainly focuses on the reduction of the visual gap, without much consideration of the physics-based differences between simulation and reality. This may result in labels being less than exhaustive or containing certain errors in the sim-to-real datasets. (iii) The overlaps and occlusions among graspable objects in clutter shown in Fig. 13(b) may cause collisions during the grasping process, leading to failure grasps.

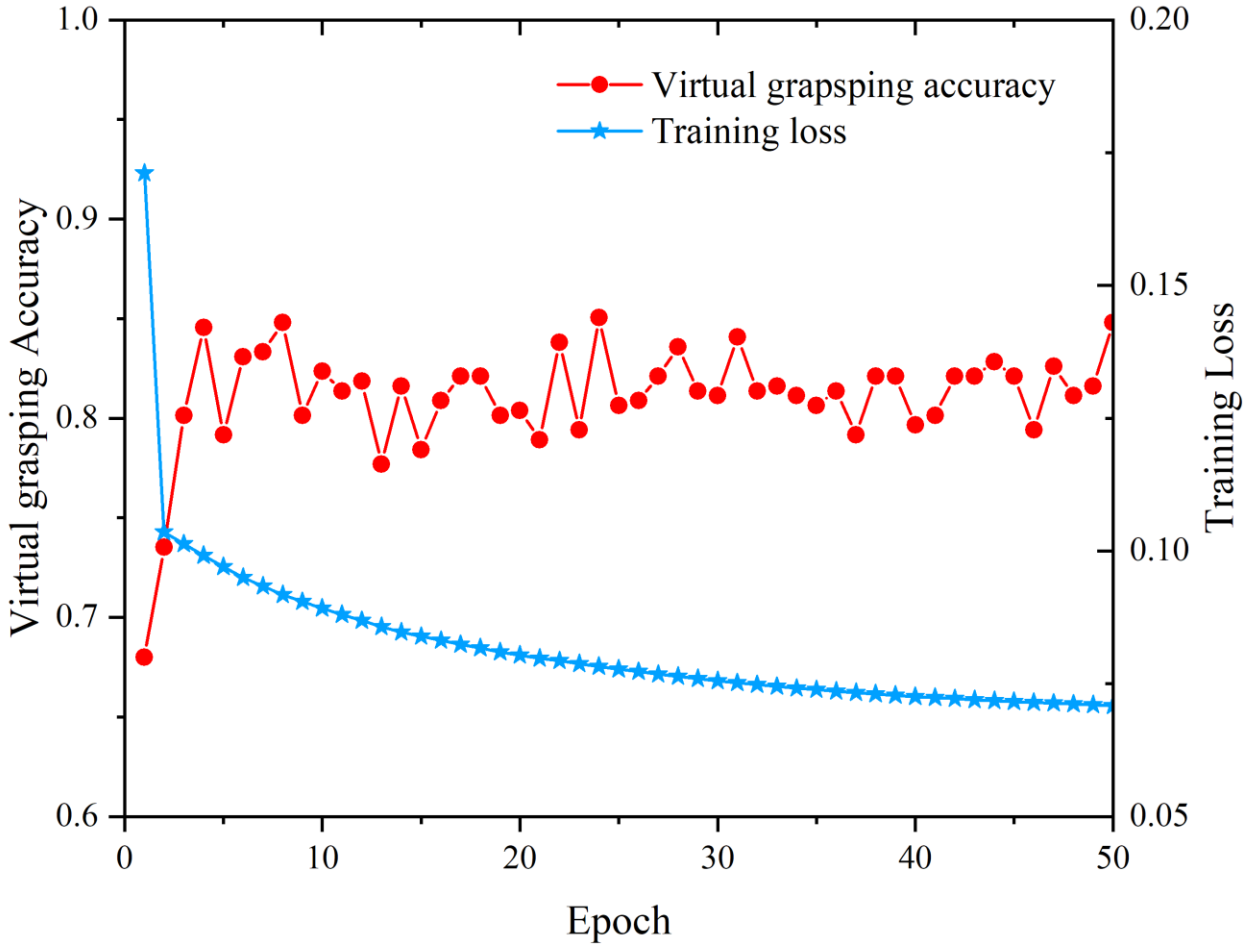


Fig. 12 The virtual grasping accuracy and training loss during the fine-tuning process in the virtual environment.

(a)

(b)

Fig. 13 Some failure cases in the real robotic grasping. (a) Grasping single object; and (b) grasping in cluttered scenes.

TABLE 4 Comparison of real-world robotic grasping accuracy using the SLO-DARTS-optimized CNN, the simulation-finetuned CNN and the sim-to-real-finetuned CNN.

Testing accuracy of real robotic grasping	SLO-DARTS-optimized CNN	Simulation-finetuned CNN	Sim-to-Real-finetuned CNN
Household objects	72.0%	79.3%	92.6%
Adversarial objects	66.2%	70.4%	86.3%
Cluttered objects	62.2%	68.6%	83.7%

5. Conclusions and future work

Advanced convolutional neural networks (CNNs) trained by sufficient task-oriented annotated samples can extract rich and efficient visual features for processing complex visual recognition tasks like grasp detection. This work first presents a novel stable and fast differentiable architecture search method based on a single-loop optimization framework, named single-loop-optimized DARTS (SLO-DARTS). This method performs continual relaxing of the discrete search space, followed by simultaneously updating both the weights and architecture parameters of the neural networks. To further finetune the SLO-DARTS-optimized CNN, a novel DT of IRG based on sim-real collaborative learning is developed, where the sim-real collaborative learning is realized by the developed Grasp-CycleGAN which can convert simulated images into pseudo-realistic ones. Based on the created pseudo-realistic images, CNN can be further trained, which could not only enhance the IRG accuracy but also reduce the costly expense of large-scale real labeled data collection, which could not only enhance the real IRG accuracy but also reduce the costly expense of large-scale real labeled data collection.

A variety of SLO-DARTS experiments and grasping experiments were conducted to verify the proposed methods in this work. The constant k, representing the number of oriented reference rectangles, is set to 6 in this study to maintain the classification and regression loss balance during CNN training. The superiority of our method can be demonstrated by the fact that SLO-DARTS could derive an optimized CNN with higher grasp detection accuracy (98.99%) in a faster and more stable optimization process (0.82 h) compared to the original DARTS. Thanks to the proposed DT framework, the SLO-DARTS-optimized CNN fine-tuned in simulation obtains a grasping accuracy increase from 65% to 82% in the virtual world. Furthermore, the weights of CNNs are fine-tuned based on sim-to-real datasets so that the real IRG accuracies of 92.6%, 86.3% and 83.7% for single household objects, single adversarial objects and cluttered objects are obtained, respectively.

Although our method provides robust and precise real-world IRG results, some further research is urgently needed. For instance, stacking diverse cells could further expand the search space and enhance network performance. The Grasp-CycleGAN can be extended to not only address the visual gap but also any physics-based differences between simulation and reality,

thereby extending its applicability to other complex visual tasks such as robotic apple picking and robot navigation. Additionally, the current visual grasp detection model is limited to static unknown object grasping scenarios using the two-finger parallel gripper. It would be important and challenging to extend the capability of model to the dynamic unknown object grasping scenarios and different gripper configurations. Finally, thanks to the rapid development of large language models, they can be integrated with robots to assist in better understanding and executing the natural language commands from humans, thus enhancing human-human interactions.

Given the gradual advancement and intelligence of robotic systems, we cannot overlook the potential risks and challenges they may bring, as well as the ethical and societal impacts. For example, as the complexity of deep learning models employed to build intelligent robotic systems increases, so does the difficulty of comprehending and interpreting their decision-making processes. This black box nature makes the systems vulnerable to attack and, in the event of a security breach, it becomes challenging to accurately pinpoint the source of problem. Therefore, the trust between the users and the systems will become weaker as the decisions of the systems are perceived to be potentially biased or incorrect. Moreover, accountability is crucial for requiring providers of intelligent robotic systems to be responsible for decision outcomes and even unintended accidents. In addition, the development of intelligent robotic systems will increasingly displace human labor and exacerbate unemployment and socio-economic inequalities, thus requiring relevant policy intervention. In summary, there is an urgent need for some mitigation strategies such as the establishment of legal and regulatory frameworks, developing interpretable deep learning models, and retraining and skills upgrading programs for the unemployed.

Data availability

The Cornell Grasp Dataset is available at https://www.kaggle.com/oneoneliu/cornell-grasp. The 3DNet dataset is available online at https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/3dnet-dataset/. The adversarial object 3D mesh models are available at https://berkeleyautomation.github.io/dex-net/. Our codes may be available upon request.

References

Ainetter, S & Fraundorfer, F. (2021). End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb. In *Proceedings of 2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China. https://doi.org/10.1109/ICRA48506.2021.9561398

Bicchi, A. (1994). On the problem of decomposing grasp and manipulation forces in multiple whole-limb manipulation. *Robotics and Autonomous Systems*, 13(2), 127-147. https://doi.org/10.1016/0921-8890(94)90055-8

Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., & Konolige, K. (2018). Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *Proceedings of 2018 IEEE international conference on robotics and automation (ICRA)*, Brisbane, Australia. https://doi.org/10.1109/ICRA.2018.8460875

Cai, H., Chen, T., Zhang, W., Yu, Y., & Wang, J. (2018). Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA. https://doi.org/10.1609/aaai.v32i1.11709

Caldera, S., Rassau, A., & Chai, D. (2018). Review of deep learning methods in robotic grasp detection. *Multimodal Technologies and Interaction*, 2(3), 57. https://doi.org/10.3390/mti2030057

Chai, J., Zeng, H., Li, A., & Ngai, E W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6, 100134. https://doi.org/10.1016/j.mlwa.2021.100134

Chiu, M-C., Tsai, H-Y., & Chiu, J-E. (2022). A novel directional object detection method for piled objects using a hybrid region-based convolutional neural network. *Advanced Engineering Informatics*, 51, 101448. https://doi.org/10.1016/j.aei.2021.101448

Chu, X., Zhou, T., Zhang, B., & Li, J. (2020). Fair darts: Eliminating unfair advantages in differentiable architecture search. In *Proceedings of European conference on computer vision*, Glasgow, UK. https://doi.org/10.1007/978-3-030-58555-6_28

Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS workshop*.

de Souza, J P C., Rocha, L F., Oliveira, P M., Moreira, A P., & Boaventura-Cunha, J. (2021). Robotic grasping: from wrench space heuristics to deep learning policies. *Robotics and Computer-Integrated Manufacturing*, 71, 102176. https://doi.org/10.1016/j.rcim.2021.102176

Erez, T., Tassa, Y., & Todorov, E. (2015). Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *Proceedings of 2015 IEEE international conference on robotics and automation (ICRA)*, Seattle, Washington, USA. https://doi.org/10.1109/ICRA.2015.7139807

Ghazaei, G., Laina, I., Rupprecht, C., Tombari, F., Navab, N., & Nazarpour, K. (2019). Dealing with ambiguity in robotic grasping via multiple predictions. In *14th Asian Conference on Computer Vision*, Perth, Australia. https://doi.org/10.1007/978-3-030-20870-7_3

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of 2016 IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA. https://doi.org/10.1109/cvpr.2016.90

Heuillet, A., Nasser, A., Arioui, H., & Tabia, H. (2024). Efficient automation of neural network design: A survey on differentiable neural architecture search. *ACM Computing Surveys*, 56(11), 1-36. https://doi.org/10.1145/3665138

Ho, D., Rao, K., Xu, Z., Jang, E., Khansari, M., & Bai, Y. (2021). Retinagan: An object-aware approach to sim-to-real transfer. In *Proceedings of 2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China. https://doi.org/10.1109/ICRA48506.2021.9561157

Hu, W., Shao, J., Jiao, Q., Wang, C., Cheng, J., Liu, Z., & Tan, J. (2023). A new differentiable architecture search method for optimizing convolutional neural networks in the digital twin of intelligent robotic grasping. *Journal of Intelligent Manufacturing*, 34(7), 2943-2961. https://doi.org/10.1007/s10845-022-01971-8

Hu, W., Wang, C., Liu, F., Peng, X., Sun, P., & Tan, J. (2022). A grasps-generation-and-selection convolutional neural network for a digital twin of intelligent robotic grasping. *Robotics and Computer-Integrated Manufacturing*, 77, 102371. https://doi.org/10.1016/j.rcim.2022.102371

Hu, W., Zhang, T., Deng, X., Liu, Z., & Tan, J. (2021). Digital twin: A state-of-the-art review of its enabling technologies, applications and challenges. *Journal of Intelligent Manufacturing Special Equipment*, 2(1), 1-34. https://doi.org/10.1108/JIMSE-12-2020-010

Hunt, K & Torfason, L. (1987). A three-fingered pantograph manipulator—a kinematic study. *Journal of Mechanisms, Transmissions, and Automation in Design*, 109(2), 171-177. https://doi.org/10.1115/1.3267432

Jiang, Y., Moseson, S., & Saxena, A. (2011). Efficient grasping from rgbd images: Learning using a new rectangle representation. In *Proceedings of 2011 IEEE International conference on robotics and automation*, Shanghai, China. https://doi.org/10.1109/ICRA.2011.5980145

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of Computer Vision–ECCV 2016: 14th European Conference*, Amsterdam, The Netherlands. https://doi.org/10.48550/arXiv.1603.08155

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., & Vanhoucke, V. (2018). Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of Conference on Robot Learning (CoRL 2018)*, Zürich, Switzerland. https://doi.org/10.48550/arXiv.1806.10293

Krizhevsky, A., Sutskever, I., & Hinton, G E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90. https://doi.org/10.1145/3065386

Kumra, S., Joshi, S., & Sahin, F. (2020). Antipodal robotic grasping using generative residual convolutional neural network. In *Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, USA. https://doi.org/10.1109/IROS45743.2020.9340777

Kumra, S & Kanan, C. (2017). Robotic grasp detection using deep convolutional neural networks. In *Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada. https://doi.org/10.1109/IROS.2017.8202237

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., & Wang, Z. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Hawaii, USA. https://doi.org/10.48550/arXiv.1609.04802

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5), 421-436. https://doi.org/10.1177/0278364917710318

Liu, H., Simonyan, K., & Yang, Y. (2018). DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*, New Orleans, LA, USA. https://doi.org/10.48550/arXiv.1806.09055

Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Aparicio, J., & Goldberg, K. (2017). Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In *Robotics: Science Systems*, Cambridge, Massachusetts. https://doi.org/10.48550/arXiv.1703.09312

Park, D., Seo, Y., Shin, D., Choi, J., & Chun, S Y. (2020). A single multi-task deep neural network with post-processing for object detection with reasoning and robotic grasp detection. In *Proceedings of 2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France. https://doi.org/10.1109/ICRA40945.2020.9197179

Redmon, J & Angelova, A. (2015). Real-time grasp detection using convolutional neural networks. In *Proceedings of 2015 IEEE international conference on robotics and automation (ICRA)*, Seattle, WA, USA. https://doi.org/10.1109/ICRA.2015.7139361

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, Montreal, Canada. https://doi.org/10.48550/arXiv.1506.01497

Song, Y., Gao, L., Li, X., & Shen, W. (2020). A novel robotic grasp detection method based on region proposal networks. *Robotics and Computer-Integrated Manufacturing*, 65, 101963. https://doi.org/10.1016/j.rcim.2020.101963

Ten Pas, A., Gualtieri, M., Saenko, K., & Platt, R. (2017). Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14), 1455-1473. https://doi.org/10.1177/0278364917735594

Wang, Z., Bovik, A C., Sheikh, H R., & Simoncelli, E P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), 600-612. https://doi.org/10.1109/TIP.2003.819861

Wang, Z., Simoncelli, E P., & Bovik, A C. (2003). Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, Pacific Grove, CA, USA. https://doi.org/10.1109/ACSSC.2003.1292216

Winkler, S & Mohandas, P. (2008). The evolution of video quality measurement: From PSNR to hybrid metrics. *IEEE transactions on Broadcasting*, 54(3), 660-668. https://doi.org/10.1109/TBC.2008.2000733

Wöhlke, G. (1992). Automatic grasp planning for multifingered robot hands. *Journal of Intelligent Manufacturing*, 3, 297-316. https://doi.org/10.1007/BF01577271

Wohlkinger, W., Aldoma, A., Rusu, R B., & Vincze, M. (2012). 3dnet: Large-scale object class recognition from cad models. In *Proceedings of 2012 IEEE international conference on robotics and automation (ICRA)*, St Paul, Minnesota, USA. https://doi.org/10.1109/ICRA.2012.6225116

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Wang, L., Chen, Z., Xiao, A., Chang, J., & Zhang, X. (2021). Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Computing Surveys*, 54(9), 1-37. https://doi.org/10.1145/3473330

Ye, P., Li, B., Li, Y., Chen, T., Fan, J., & Ouyang, W. (2022). b-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of 2022 IEEE/CVF conference on computer vision and pattern recognition*, New Orleans, USA. https://doi.org/10.48550/arXiv.2203.01665

Yu, Y. (2022). Few Shot POP Chinese Font Style Transfer using CycleGAN. *Journal of Physics: Conference Series*, 2171(1), 012031. https://doi.org/10.1088/1742-6596/2171/1/012031

Zhu, J-Y., Park, T., Isola, P., & Efros, A A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of 2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy. https://doi.org/10.48550/arXiv.1703.10593v6

Acknowledgements

The authors are very grateful to the support from the National Natural Science Foundation of China, the Zhejiang Provincial Natural Science Foundation of China, and the "Pioneer" and "Leading Goose" R&D Program of Zhejiang Province. The authors also express gratitude to the reviewers and the editors for their valuable comments that have contributed to enhancing the quality of this paper.

Funding

The work was supported by the National Natural Science Foundation of China (grant numbers 52275275, U22A6001), the Zhejiang Provincial Natural Science Foundation of China (grant number LZ22E050006), and the "Pioneer" and "Leading Goose" R&D Program of Zhejiang Province (grant number 2023C01008).

Declarations

There are no conflicts of interest.

Nomenclature

B	The number of nodes within each cell
c	The graspable confidence
c_x	The cell's x-axis offset from the image's top-left corner
c_y	The cell's y-axis offset from the image's top-left corner
D_S	The discriminator responsible for classifying images to the simulation domain
D_R	The discriminator responsible for classifying images to the reality domain
g	The real-world representation of a grasp
g_i	The pixel representation of a grasp
G_{RS}	The generator responsible for transforming images to the simulation style
G_{SR}	The generator responsible for transforming images to the real style
H	The height of the image
I	An RGB-depth (RGB-D) image

k	The number of oriented reference rectangles
$\mathcal{L}$	The loss function
n_g	The number of candidate grasp rectangles
N	The number of positive oriented reference rectangles
$o^{(m,n)}$	Operations performed on node *m*
$\bar{o}^{(m,n)}$	A mixed operation on the edge connecting node *m* and node *n*
O	The operation space
$\boldsymbol{p}$	The real-world coordinates of the gripper's center position
p_w	The predefined width of an oriented reference rectangle
p_θ	The predefined angle of an oriented reference rectangle
R	The real image domain
S	The simulation image domain
t_m	The offset values predicted by the detection network
$\hat{t}_m$	The benchmark offset values
t_c	The graspable confidence predicted by the detection network
$\hat{t}_c$	The benchmark graspable confidence
U	The predicted grasp rectangle
V	The benchmark grasp rectangle
w	The gripper opening distance
W	The image width
$\alpha_o^{(m,n)}$	A continuous operation variable of operation *o*
$\boldsymbol{\alpha}$	The continuous operation vector
θ	The gripper rotation angle during grasping

ω	The weight coefficients of the neural network
ω*	The optimal weight coefficients of the neural network