machines

MDPI

Article

Robotic Grasping Technology Integrating Large Kernel Convolution and Residual Connections

Liang Li, Nan Li, Rui Nan, Yangfei He, Chunlei Li *, Weiliang Zhang and Pan Fan

School of Mechanical Engineering, Baoji University of Arts and Sciences, Baoji 721016, China; liliang@bjwlxy.edu.cn (L.L.); linan@stu.bjwlxy.edu.cn (N.L.)
* Correspondence: 602lcl@bjwlxy.edu.cn; Tel.: +86-1589-130-1856

Abstract: To meet real-time grasping demands in complex environments, this paper proposes a lightweight yet high-performance robotic grasping model. The model integrates large kernel convolution and residual connections to generate grasping information for unknown objects from RGB and depth images, enabling real-time generation of stable grasping plans from the images. The proposed model achieved favorable accuracy on both the Cornell and Jacquard standard grasping datasets. Compared to other methods, the proposed model significantly reduces the number of parameters while achieving comparable performance, making it a lightweight model. Additionally, real-world experiments were conducted using a six-axis collaborative robot on a set of previously unseen household objects with diverse and adversarial shapes, achieving a comprehensive grasping success rate of 93.7%. Experimental results demonstrate that the proposed model not only improves grasping accuracy but also has strong potential for practical applications, particularly in resource-constrained robotic systems.

Keywords: large kernel convolution; residual connections; lightweight model; robotic grasping

Citation: Li, L.; Li, N.; Nan, R.; He, Y.; Li, C.; Zhang, W.; Fan, P. Robotic Grasping Technology Integrating Large Kernel Convolution and Residual Connections. *Machines* **2024**, *12*, 786. https://doi.org/10.3390/machines12110786

Academic Editor: Med Amine Laribi

Received: 20 October 2024
Revised: 3 November 2024
Accepted: 4 November 2024
Published: 7 November 2024

1. Introduction

Robotic grasping technology plays a critical role in modern industries and services. As manufacturing develops rapidly and labor costs rise, traditional robotic grasping technologies are becoming insufficient to meet current market demands. Consequently, robotic grasping techniques based on deep learning have become a mainstream research direction. Deep learning-based robotic grasping has gradually become common in industrial robotics, and numerous scholars have proposed related studies. However, due to limited generalization capabilities, although models perform well on specific training datasets, they often struggle when dealing with unseen objects or environments. Moreover, many deep learning models, particularly those with complex architectures, require substantial computational resources during inference, leading to insufficient real-time decision-making, limiting robot response speed and efficiency. At the same time, in unstructured environments and unknown object grasping tasks, robotic systems face more complex challenges. For example, the diversity of object shapes significantly increases the difficulty of model recognition and decision-making. The occlusion between objects requires the model to be more robust in order to deal with the problem of missing information. In addition, dynamically changing scenarios, such as logistics sorting on the move, automatic item handling in warehousing systems, and device grasping in medical scenarios, make it more difficult for the system to identify and accurately grasp in real time. These scenarios require models that can not only accurately locate and recognize diverse objects, but also have the ability to react quickly to adapt to changes in the environment and in complex operational requirements. Hence, it is necessary to use data augmentation techniques to generate diverse training data to enhance model adaptability to various objects and scenarios. Additionally, lightweight models should be developed to reduce parameters and computational loads, improving inference speed while maintaining accuracy. This paper focuses on robotic grasping technology that

integrates large kernel convolution and residual connections, covering two main research areas: grasp detection representation and grasp detection network architectures.

The core of grasp detection representation [1] lies in analyzing the target object image, combined with the calibration results of the vision system, to associate and transform the grasping information from a 2D image to the actual 3D grasping position and orientation. Currently, grasp representation technology [2] has been widely applied in the field of robotic grasping, yielding valuable research outcomes. For instance, Morrison et al. [3] introduced a method that predicts grasp quality for each pixel, providing grasp position and orientation by evaluating grasp quality at the pixel level, thereby improving the accuracy of grasping decisions. On this basis, Shi [4] proposed a pixel-level grabbing detection representation method based on RGB-D images, in which the adaptive grabbing width is used to adaptively represent the grabbing attributes, which effectively solves the grabber proximity conflict. Wang [5] et al. proposed a new deep learning method for symmetrical point gripping detection, which defines the end gripping actuator as a pair of fingertips, unlike the widely used 5-dimensional rectangle, so that unseen objects can be predicted more broadly and flexibly. These research outcomes provide valuable experience and technical support for grasp representation methods in robotic grasping tasks.

In addition to grasp detection representation, network architecture plays a crucial role in deep learning models, especially in applications like robotic grasping. It not only influences model performance, efficiency, and adaptability but also determines the practical application effectiveness of the model in specific tasks. Designing and selecting an appropriate network architecture is essential for achieving efficient and accurate robotic grasping technology. The advent of deep convolutional networks and the development of new designs for various CNN architectures [6–9] have made it possible to plan the capture of unknown objects. Current grasp detection network architectures are generally divided into two types: cascade methods and single-stage architectures. Cascade methods execute the entire grasp prediction process in stages, including target feature extraction, candidate region evaluation, and optimal grasping position evaluation. Lenz et al. [10] created the Cornell dataset and proposed a two-stage cascade detection model to learn five-dimensional grasping. In the first stage, a neural network extracts grasp prediction features and, in the second stage, grasp parameters are refined to output the optimal grasping position. Zhang et al. [11] introduced ROI-GD, which uses ROI features instead of the entire image for grasp detection. This architecture has certain advantages, guiding the network to learn different parameters at different stages, reducing the learning burden. However, it also has some drawbacks, such as increased inference time and repeated ROI region calculations.

In contrast, single-stage detection methods, with their simple and efficient structures, have become increasingly popular in object grasping. Single-stage methods train a grasp detection model to directly output the grasping position. In existing studies, Hosseini [12] proposed an improved pip line model that attempts to use grasping as a rectangular representation to detect different seen or unseen objects. Its main ideas include preprocessing, output normalization, and data augmentation to improve accuracy without slowing down the system. Kumra et al. [13] built a grasping network based on ResNet, extracting features from RGB and depth images to output classification and regression results for optimal grasping positions. Morrison et al. [3] used convolutional layers for encoding and decoding to predict grasps at the pixel level. Wu et al. [14] introduced an anchor-free method that completely uses convolutional networks, framing grasp detection as grasping rectangle regression and classification tasks. However, these methods primarily focus on local spatial information, which may limit the ability to capture global information associations, thereby hindering further improvements in detection accuracy. To address this issue, some researchers [15] have proposed the theory of effective receptive fields. The size of the effective receptive field is proportional to the convolution kernel size and scales with the square root of the number of convolution layers. Therefore, compared to simply increasing the number of

convolution layers, enlarging the kernel size can more efficiently enhance the effective receptive field. Large kernel convolution [16] has demonstrated excellent performance in applications, such as object detection and classification, compared to conventional CNN models. Xiao Ding [17] proposed a model called UniRepLKNet, which uses ultra-large kernel convolution to construct modern CNNs, overcoming the limitations of conventional CNNs in capturing global information. Although large kernel convolution can effectively capture global information, grasp detection also requires the extraction of detailed features, which may affect grasp detection performance. Additionally, large kernel convolution increases the computational load of the model, making it less conducive to deep model architectures.

In order to better improve the prediction performance of the model, it can be divided into three processes: increasing the number of layers, using a larger-size convolutional kernel, and using the self-attention mechanism. By increasing the number of layers to increase the performance of the model, Zhang et al. [18] emphasized increasing the number of layers in the deep network to improve model performance by combining multiple attention mechanisms. Chen et al. [19], based on the Efficient Net network, improved performance by increasing the depth and width of the network while maintaining training efficiency. By using a larger convolutional size, Wang et al. [20] used the feature capture enhancement module of large kernel convolution to improve the semantic feature capture ability and improve the receptive field, so as to highlight the key information. Luo et al. [21] enhance the performance of the network by increasing the size of the convolutional kernel, and at the same time integrate the deep large kernel volume into smaller deep convolution and deep void convolution to reduce a large amount of computational overhead and the parameters of the large kernel. In terms of using the self-attention mechanism, Guo et al. [22] emphasized the case of its performance improvement by various applications of the self-attention mechanism in the field of computer vision, including image classification, object detection and image generation, Yu et al. [23] added self-attention and cross-attention modules to the model to maintain saliency correlation and improve the consistency of intra-frame saliency detection. However, these methods also have disadvantages, such as increasing the depth of the model; in very deep networks, the gradient may disappear or explode, affecting the stability of training, increasing the computational complexity and memory requirements, and increasing the complexity of model training. The use of large-size convolutional kernels leads to power shortages, high computational cost, overfitting and slow processing speed; The computational complexity of using the self-attention mechanism in long sequences or high-resolution images is high, resulting in a long training time.

To overcome these challenges, this paper proposes a robotic grasp detection method based on large kernel convolution and residual connections. This paper is the first to fuse large kernel convolution with residual connection and apply it to the robot grasping detection scenario. Large-kernel convolution is used to capture the overall structure and context information of complex objects, combined with residual connections to alleviate the gradient vanishing problem in deep networks and enhance the stability of training. At the same time, the introduction of the self-attention mechanism enables the model to dynamically emphasize key features, thereby improving adaptability to complex scenes. Depth-separable convolution, on the other hand, maintains the characterization ability of the model by reducing the number of parameters and computational requirements. Through this multi-level technology integration, this paper aims to improve the performance and practicability of robotic gripping detection, especially for gripping tasks involving complex objects in unstructured and dynamic environments. First, the relationship between the definition of robotic grasping and the transformation from the image coordinate system to the robot coordinate system is studied to ensure accurate conversion. Then, the basic structures of existing residual networks and large kernel convolution networks are introduced, and the two are integrated, incorporating depth-wise separable convolutions [24] to optimize the network structure and reduce the number of model parameters, while maintaining

representation capacity. Efficient structures, such as SE Blocks, are employed to enhance model depth and general representational capacity. Subsequently, the proposed model is evaluated on publicly available Jacquard and Cornell datasets. Finally, real-world experiments are conducted using the trained network to grasp actual objects, validating the effectiveness of the proposed model.

2. Definition of Robotic Grasping

Grasping is a critical aspect of robotic operations, directly influencing the overall system performance and task success rate. With the widespread application of robotic technology in industrial automation and service sectors, selecting an appropriate grasping method has become a key research topic for improving operational efficiency and reducing error rates. An effective grasping strategy not only adapts to diverse operational environments but also significantly enhances the precision, stability, and safety of grasping. In this chapter, we discuss the definition and representation of grasping methods and introduce the transformation between image coordinate systems and robot coordinate systems. We also explore the principles and methods by which vision-guided robots achieve precise grasping.

2.1. Definition of Robotic Grasping Methods

The selection of a robotic grasping method is crucial, as it directly affects the system's overall performance and operational efficiency. Whether in industrial automation or service robots, the grasping method determines the types of objects the robot can handle and influences task precision, speed, and success rate. A suitable grasping strategy can enhance operational stability and safety, reducing failure rates and preventing object damage, thus maximizing the robot's capabilities in various complex environments.

Current research has provided reasonable descriptions of robotic grasping methods. In this paper, the end-effector of the robot is oriented so that its z-axis is perpendicular to the grasping plane. The end-effector chosen for this study is an electrically powered two-finger parallel gripper. During the grasping process, the actual grasping target is defined by Equation (1). As shown in Figure 1, the position p of the gripper represents the coordinates of the center point of the gripper, as expressed in Equation (2). W refers to the required opening width of the gripper during grasping, and the angle of rotation around the z-axis during grasping is denoted by θ. Since the end-effector is a two-finger parallel gripper, θ is constrained within a defined range $[-\pi/2,\pi/2]$. The grasp quality Q represents the probability of a successful grasp, where Q ranges between [0, 1]; the higher the value, the higher the likelihood of successful grasping.

$$G = (p, \theta, W, Q) \tag{1}$$

$$p = (x, y, z) \tag{2}$$

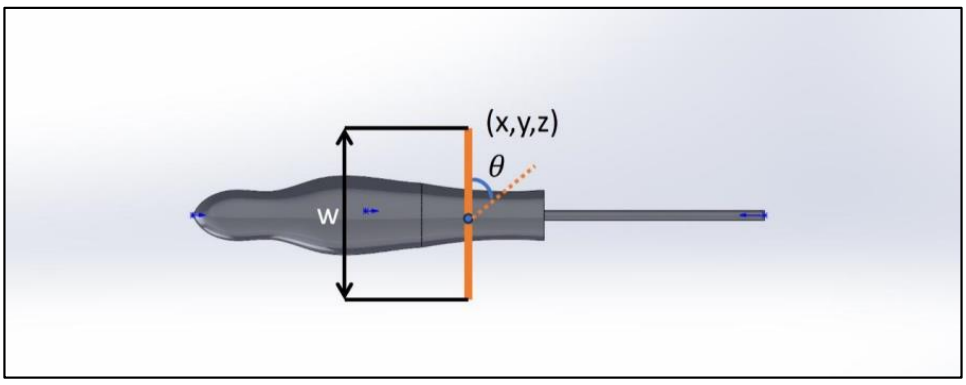


Figure 1. Grasp Representation.

2.2. Transformation Between Image and Robot Coordinate Systems

In robotic grasping and vision-guided systems, the transformation between the image coordinate system and the robot coordinate system is a critical step. The image coordinate system, captured by depth cameras, contains both 2D planar and depth information about objects, whereas the robot coordinate system describes the robot's position and orientation in 3D space. To enable a robot to accurately grasp a target object, the information from the image coordinate system must be correctly transformed into the robot coordinate system.

This transformation process involves observing the entire grasping state through RGB-Depth (RGB-D) images. RGB-D images are provided by RGB-D cameras and consist of a color image $P_{rgb} \in \mathbb{R}^{H \times W \times 3}$ and a depth image $P_{depth} \in \mathbb{R}^{H \times W}$. Using the known intrinsic parameters of the depth camera, the grasping position p in the 3D world can be converted into a pixel index $\hat{p} = (u, v)$ with a specific depth h. The grasping depth represents the z-axis position of the grasping point. Therefore, the grasp detection problem is transformed into identifying a specific grasp pose to successfully grasp the object. By detecting the grasping points output by the model, the pixel coordinates (u,v) in the depth map can be obtained. Using the depth value, the grasp point's coordinates in the camera coordinate system can be calculated. The pixel coordinates (u,v) and the depth value depth_z are converted into 3D coordinates (X,Y,Z) in the camera coordinate system using the camera's intrinsic parameters, as shown in Equation (3).

$$\begin{cases} X = (u - c_x) \cdot \dfrac{depth_z}{f_x} \\ Y = (v - c_y) \cdot \dfrac{depth_z}{f_y} \\ Z = depth_z \end{cases} \tag{3}$$

At this stage, the coordinate positions in the camera coordinate system cannot be directly applied to the robot. The information from the image coordinate system needs to be precisely transformed to map to the robot's coordinate system. This transformation is typically achieved through hand–eye calibration. Given the known camera coordinate system, hand–eye calibration is used to compute the transformation matrix between the camera coordinate system and the robot coordinate system, denoted as camera2robo. This matrix includes the rotation matrix camera2robot [0:3,0:3] and the translation vector camera2robot [0:3,3:0]. The target position in the camera coordinate system is transformed into the grasping position in the robot coordinate system using Equation (4).

$$target_{position} = camera2robot[0:3, 0:3] \cdot target + camera2robot[0:3, 3:0] \tag{4}$$

This step enables the robot to perform precise grasping actions based on image information, forming a crucial part of vision-guided systems.

3. Robotic Grasping Network Based on Improved Large Kernel Convolution and Residual Networks

In robotic grasping tasks, accurately and efficiently identifying grasping points is a key challenge for achieving automated grasping. To improve the accuracy and efficiency of grasp detection, convolutional neural networks (CNNs) have been widely applied in robotic grasping tasks in recent years. However, traditional convolutional networks still face certain limitations in capturing global information and enhancing model representation capabilities. To address these issues, this paper proposes a robotic grasp detection method based on improved large kernel convolution and residual networks.

In this chapter, we will explore the design ideas and optimization strategies of the improved large kernel convolution and residual network in detail. By incorporating large kernel convolution and residual connections, we aim to enhance the model's feature extraction capability for complex image tasks and to improve training stability. Furthermore,

we will validate the effectiveness and robustness of the proposed model in robotic grasping tasks using both public datasets and real-world experiments, which will be detailed in Chapter 4.

3.1. Introduction to Large Kernel Convolution and Residual Connection Models

Traditional convolutional networks exhibit limitations in capturing global information and enhancing model representation capabilities. Large kernel convolution networks, by utilizing larger convolution kernels, expand the receptive field of the network, allowing it to better capture global features from images and improve its understanding of complex scenes. Residual networks, on the other hand, introduce residual learning mechanisms through skip connections, which address issues such as vanishing gradients and degradation in deep networks, significantly improving the training efficiency and performance of deep networks. These two network structures have been widely applied in the field of deep learning, providing a solid theoretical foundation for subsequent network improvements and innovations.

The large kernel convolution network (LKCN) improves the performance and efficiency of convolutional neural networks (CNNs) by employing large-sized convolutional kernels. Traditional convolutional networks typically use smaller convolution kernels, such as 3×3 or 5×5, to extract local features. In contrast, LKCN uses larger kernels (e.g., 7×7, 11×11, or even larger) to increase the receptive field. The structural diagram is shown in Figure 2. This method allows the network to cover a wider input region in a single convolution operation, thus capturing more global information. The extraction of this global information enhances the network's ability to understand the overall features of the image, rather than being limited to local features. Moreover, large kernel convolutions can reduce the number of convolutional layers required, simplifying the network structure while improving the efficiency of feature extraction and the model's representation capacity.

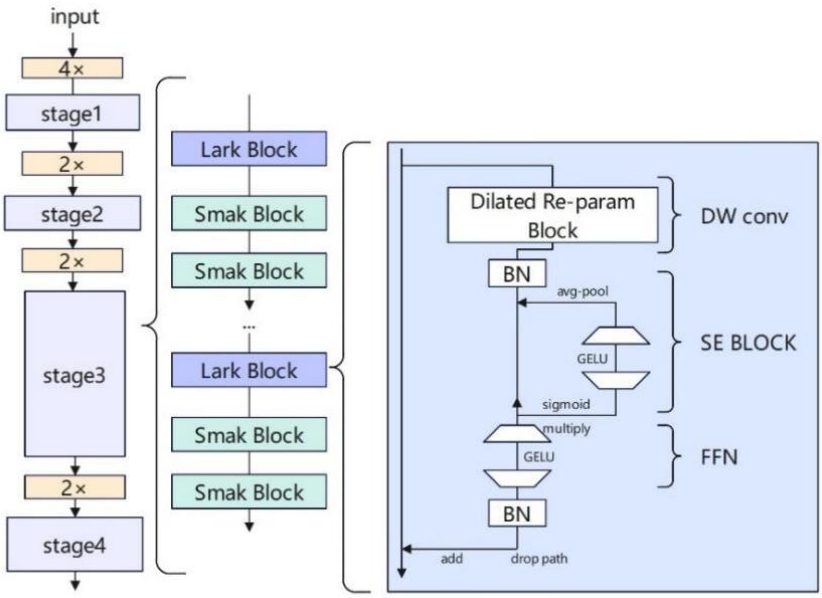


Figure 2. Schematic Diagram of the Large Kernel Convolution Network Structure.

The residual network (ResNet) is a deep convolutional neural network architecture designed to solve the problems of vanishing gradients and network degradation in deep network training. The core principle of ResNet lies in introducing residual learning, which simplifies the learning task of the network by using residual blocks. The structural diagram is shown in Figure 3. Each residual block includes a skip connection that directly adds the input to the output of the block. This structure allows the network to optimize the residual between the input and the target during training, i.e., the

difference between the input and the target, rather than directly learning the target function, as expressed in Equation (5). This approach alleviates the vanishing gradient problem, enabling the gradients to propagate more effectively to deeper layers of the network during back-propagation. Additionally, skip connections help mitigate network degradation, allowing for deeper network stacking without a significant increase in training error. As a result, ResNet improves the model's representation capacity and training efficiency. Through this innovative network structure, ResNet demonstrates significant performance improvements in handling complex visual tasks.

$$y = H(x, w_h) + x \tag{5}$$

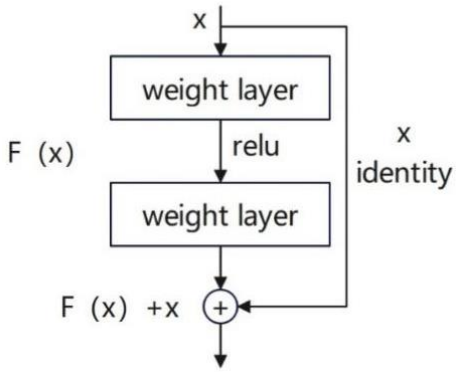


Figure 3. Schematic Diagram of the Residual Network Structure.

3.2. Model Construction Based on Improved Large Kernel Convolution and Residual Networks

This section introduces an improved model that combines large kernel convolutional networks with residual networks.

Large kernel convolutional networks utilize larger convolutional kernels to capture more extensive global information, while residual networks solve the issues of vanishing gradients and network degradation through the introduction of residual learning mechanisms. The integration of these two networks not only expands the model's receptive field, enabling better extraction of global features, but also maintains the stability and efficiency of training deep networks. By merging the advantages of these two network structures, the proposed improved model demonstrates superior accuracy and robustness in handling complex visual tasks and robotic grasping operations.

3.2.1. Overview of the Model Architecture

In deep learning, the overall architecture of a model is a critical factor that determines its performance, efficiency, and adaptability to various tasks. A well-designed architecture not only efficiently extracts image features but also enhances the network's expressive power through multiple layers of convolution, pooling, and activation operations. This section will provide a detailed overview of the model's structure, showcasing the entire computational process from input to output.

To boost the model's performance, the proposed network is designed with an encoder–decoder structure. First, a $4 \times 300 \times 300$ image tensor is input into the model, undergoing standard convolution operations to capture low-level features. Next, batch normalization is applied to ensure smooth data distribution, reduce internal covariate shift, stabilize the training process, and speed up convergence. Following this, large kernel convolutions are used to extract higher-level features, allowing for better comprehension of global structures, object relations, and intricate details. The use of large kernel convolutions significantly increases computational cost, which is not conducive to fast inference. To mitigate this, depth-wise separable convolutions are employed, reducing parameters and computation while maintaining efficient feature extraction. The model then incorporates a Rectified Linear Unit (ReLU) activation function to enhance its ability to express complex features.

Residual connections are introduced to transmit information between layers, avoiding gradient vanishing and improving network depth. Dropout regularization is applied to enhance the model's generalization performance on both training and validation sets by randomly dropping layers. Finally, three feature maps are output, from which the necessary information for robotic grasping is extracted. The architecture of the model is illustrated in Figure 4.

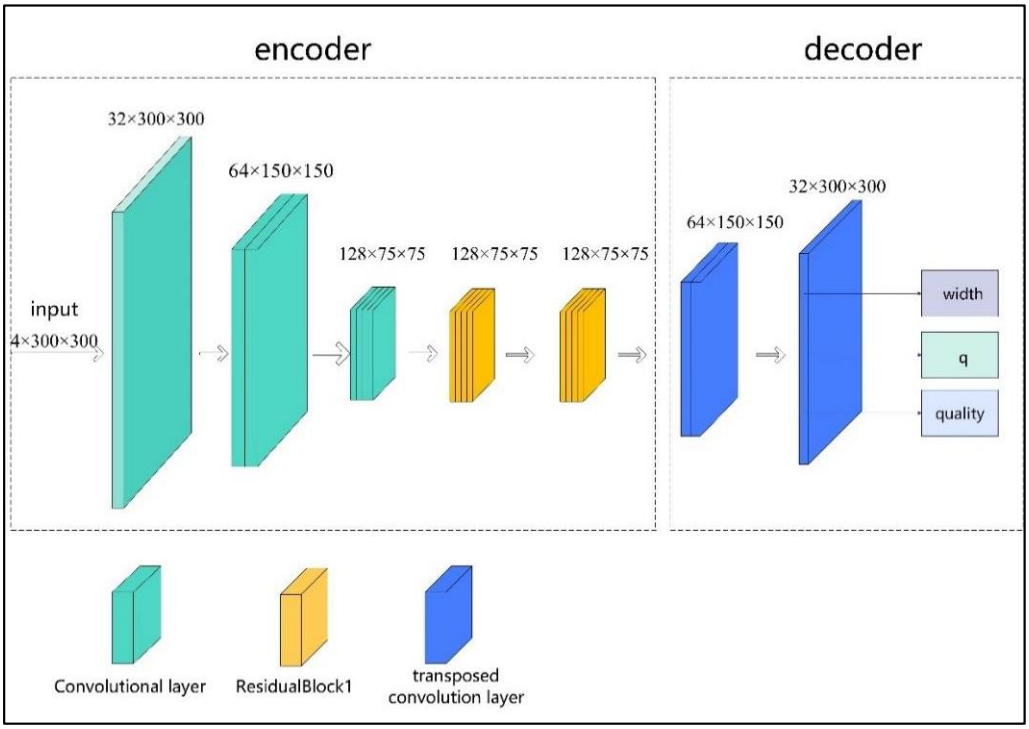


Figure 4. Model Architecture Diagram.

3.2.2. Encoder–Decoder Structure Design

In deep learning models, the encoder–decoder structure is a central design framework for tasks related to image processing and generation. The encoder's primary function is to convert the input image into compact feature representations through multiple layers of convolutional networks, extracting higher-level semantic information at each layer. As the depth of the network increases, the encoder gradually reduces the spatial dimensions of the feature map while increasing the number of channels, allowing for the capture of more global information. Correspondingly, the decoder is responsible for progressively restoring the high-dimensional features extracted by the encoder back to outputs that match the dimensions of the original input. Through deconvolution (or up-sampling) operations, the decoder enlarges the feature map, recovering spatial information. In tasks such as image segmentation, object detection, or robotic grasp prediction, the decoder's primary goal is to generate output results related to the input, i.e., the robotic grasping plan.

The principle of convolution operation is to open an active window of the same size as the template from the top left corner of the image; Multiply and add the window image and template pixels together to obtain a new pixel value; Move the activity window one column to the right and repeat the above steps; From left to right and top to bottom, a new image can be obtained.

As shown in Figure 4. The computations involved can be described as follows: for an input image of size 300 × 300 with n channels, the image is initially convolved with a 3 × 3 kernel, maintaining the dimensions at 300 × 300, and increasing the channel count to 32. Subsequently, the image is processed with a 4 × 4 kernel, reducing its size to 150 × 150 while the number of channels increases to 64. After another convolution with a

4 × 4 kernel, the dimensions decrease to 75 × 75, resulting in 128 channels. In the encoder section, we utilize a multi-layer Convolutional Neural Network (CNN) to extract features from the input image, as illustrated in Figure 5. Specifically, the encoder first processes the image using standard convolutional layers sequentially, where each convolutional layer is followed by Batch Normalization and nonlinear activation functions. This approach aids in accelerating convergence and enhancing the stability of the model. The 128-channel image of size 75 × 75 is then fed into the residual block, which integrates large kernel convolutions and residual connections for feature extraction, as depicted in Figure 6.

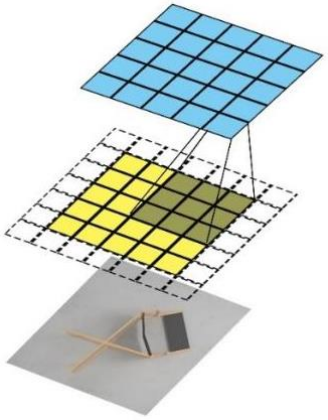
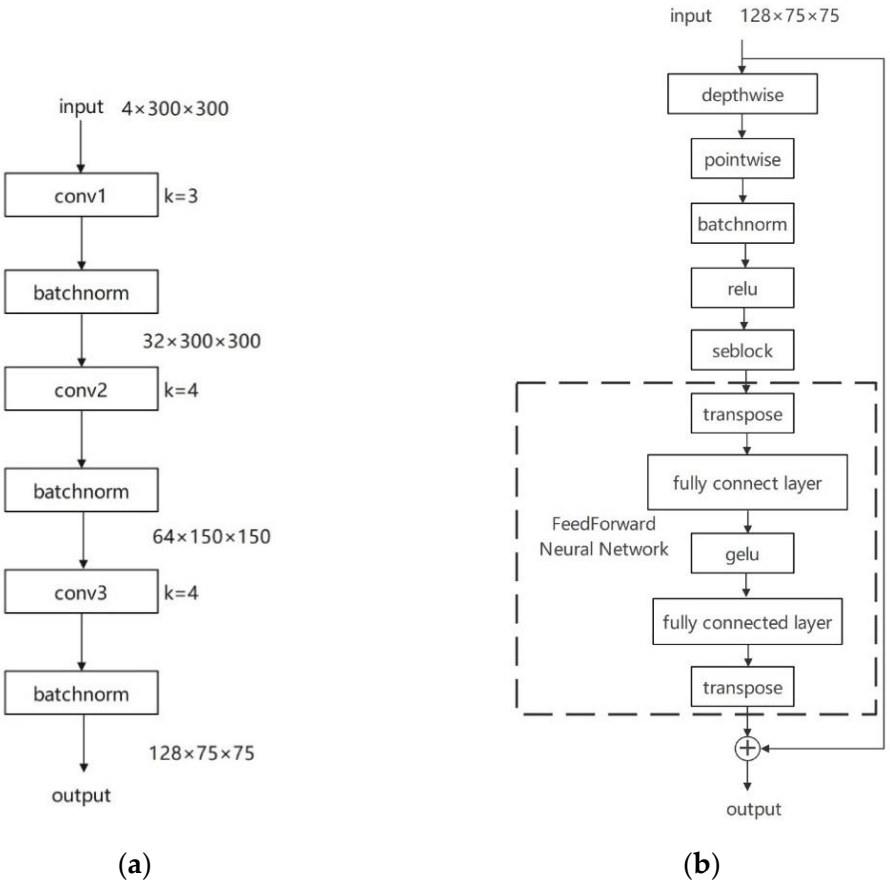


Figure 5. Convolution Operation Principle Diagram.



(**a**) (**b**)

Figure 6. (**a**) Standard Convolutional Layer Figure. (**b**) Convolutional Layer with Residual Connections and Large Kernel Convolution.

This process outputs a 128-channel image of size 75 × 75. The specific calculation is described by Equation (6):

$$\begin{cases} \text{width}_{\text{out}} = \dfrac{\text{W} - \text{k} + 2 \times \text{P}}{\text{S}} + 1 \\ \text{heigth}_{\text{out}} = \dfrac{\text{H} - \text{k} + 2 \times \text{P}}{\text{S}} + 1 \end{cases} \tag{6}$$

where W represents the width of the input feature map, H represents the height of the input feature map, K denotes the width and height of the convolutional kernel, P indicates the number of zeros to be padded to the feature map, S represents the stride, width_out is the width of the output feature map after convolution, and height_out is the height of the output feature map after convolution. After feature extraction from the 75 × 75 image, up-sampling is performed to restore the image size back to 300 × 300 using transposed convolution, with the calculation process outlined in Equation (7).

$$\begin{cases} \text{width\_out} = (\text{W} - 1) \times \text{S} - 2 \times \text{P} + \text{K} \\ \text{height\_out} = (\text{H} - 1) \times \text{S} - 2 \times \text{P} + \text{K} \end{cases} \tag{7}$$

In the decoder section, the decoder structure is symmetrical to the encoder structure, progressively enlarging the spatial dimensions of the feature maps through deconvolution or up-sampling operations. This process restores the compressed features to match the dimensions of the input image. The decoder combines multi-level semantic features for image reconstruction, enabling the model to extract fine-grained semantic information at different resolutions. Through this gradual enlargement process, the decoder can accurately locate small image features and output high-quality predictions, such as grasp points and target areas. For the input 128-channel image of size 75 × 75, transposed convolution is applied to restore the image size to 300 × 300, with the number of channels set to 4, ensuring consistency with the input image. The structural representation is shown in Figure 7.

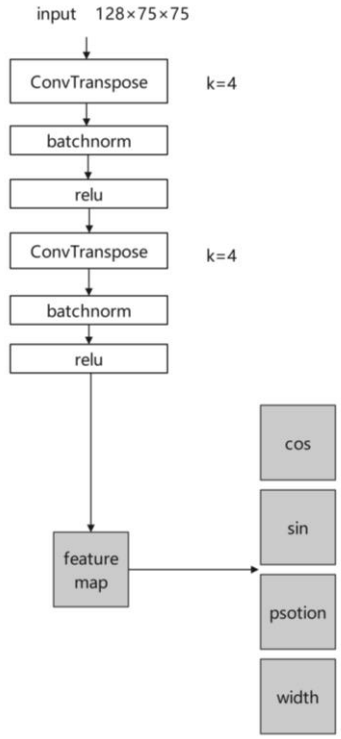


Figure 7. Schematic Diagram of the Decoder Structure.

Finally, the joint design of the encoder and decoder not only achieves multi-level extraction and restoration of image features but also enhances the robustness and accuracy of the model in grasping tasks through the combination of large kernel convolutions and residual connections. The changes in image size are illustrated in Figure 8.

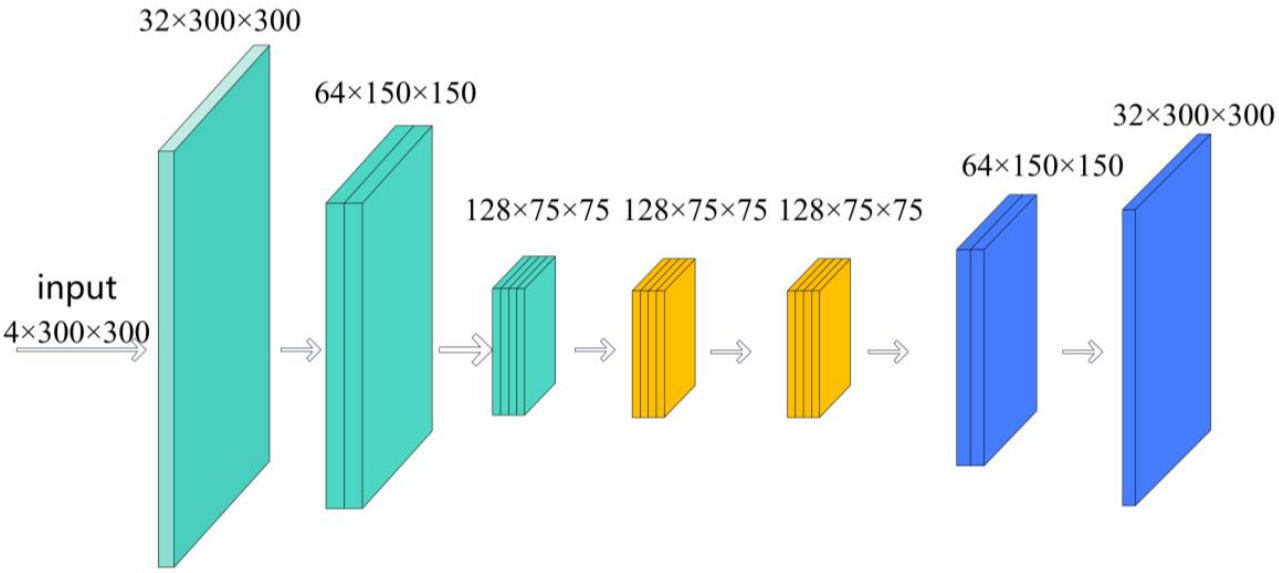


Figure 8. Schematic Diagram of Image Size Variation. Cyan represents the ordinary convolution part, yellow represents the large kernel convolution and residual connection part, and blue represents the deconvolution part.

3.2.3. Model Parameter Optimization

In deep learning, employing large kernel convolutions can significantly expand the receptive field of a model, enhancing its ability to capture global information, particularly in tasks such as object detection and complex image processing. However, the introduction of large kernel convolutions also leads to a substantial increase in the number of parameters, adversely affecting the model's computational efficiency and inference speed. To address this challenge, depth-wise separable convolutions are widely applied in network design to effectively mitigate the parameter explosion caused by large kernel convolutions. Depth-wise separable convolution decomposes the convolution operation into depth-wise convolution and point-wise convolution, significantly reducing computational load and parameter scale. Furthermore, a feed-forward neural network processes the features further after depth-wise separable convolutions, capturing complex relationships between features through dimensionality expansion and reduction, as well as nonlinear transformations, thereby enhancing the model's representational capacity. This combined design not only optimizes the performance of large kernel convolutions but also ensures higher efficiency and accuracy when handling complex image tasks.

In traditional large kernel convolution, the size of the convolutional kernel directly affects the number of parameters. When the kernel size is large (e.g., 5×5 or 7×7) and standard convolution is used, the computational expression can be described as follows in Equation (8):

$$N = K \times K \times W \times H \times N_i \times N_o \tag{8}$$

where $K \times K$ represents the spatial dimensions of the convolutional kernel, indicating the pixel area covered; $W \times H$ signifies the width and height of the output feature map, typically consistent with the input image size; N_i denotes the number of input channels, with each input channel convolving with a convolutional kernel; and N_o represents the number of output channels generated by combining multiple input channels and kernels. However, if depth-wise separable convolution is employed, the computation is divided

into two steps: first, using a large depth-wise convolution to operate independently on each channel, extracting internal information, calculated as follows in Equation (9):

$$N = K \times K \times W \times H \times N_i \tag{9}$$

Then, a point-wise convolution (1 × 1 convolution) merges across channels, calculated as Equation (10):

$$N_{point-wise} = 1 \times 1 \times W \times H \times N_i \times N_o \tag{10}$$

Thus, the total computational load for depth-wise separable convolution is the sum of the calculations Equation (11) from the depth-wise and point-wise convolutions:

$$\begin{aligned} \text{Total Computational Load} = &\frac{\text{Depth} - \text{wise Convolutions}}{\text{Point} - \text{wise Convolutions :}} = \\ &\frac{K \times K \times W \times H \times N_i + 1 \times 1 \times W \times H \times N_i \times N_o}{K \times K \times W \times H \times N_i \times N_o} = \frac{1}{N_o} + \frac{1}{K^2} \end{aligned} \tag{11}$$

From this, it is evident that when both N_o and K are large, the computational load of depth-wise separable convolution is significantly lower than that of standard convolution. This method effectively reduces redundant calculations across spatial and channel dimensions.

To further enhance the model's representational capacity, nonlinear transformations are applied to the input features to capture complex relationships across different levels of features, integrating global information from the model's input data. Building on the efficiency of depth-wise separable convolutions in reducing model parameters and computational load, a feedforward neural network (FNN) is employed, as illustrated in Figure 9. The FNN utilizes the ReLU activation function and a fully connected structure.

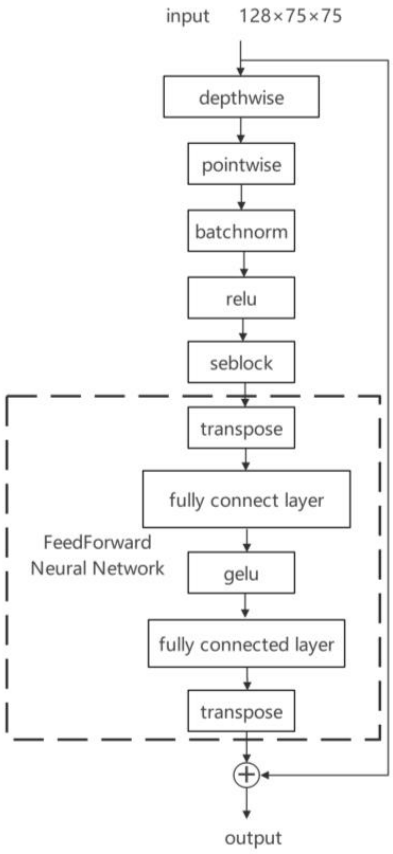


Figure 9. Module based on depth-wise separable convolution and feedforward neural network.

In this configuration, the ReLU activation function performs nonlinear transformations on the input features, capturing complex interrelationships among features at various levels. Specifically, it outputs positive values directly while suppressing negative values to zero, introducing nonlinearity that enables the neural network to learn complex patterns, as shown in Figure 10. The expression for the ReLU activation function can be represented as Equation (12):

$$\sigma(Z) = \mathrm{ReLU}(z) = \max(0, z) \tag{12}$$

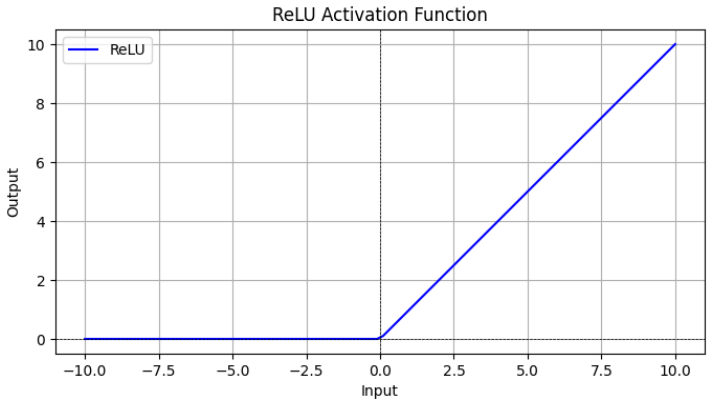


Figure 10. Illustration of the ReLU Activation Function.

The fully connected layer establishes complex nonlinear relationships by connecting every input node to each output node, thus addressing the issues of feature abstraction and information fusion.

With the optimized network structure, the model becomes more efficient in feature extraction and information transfer, laying a solid foundation for subsequent model training. As the training process iterates, we can further enhance the model's accuracy and robustness to adapt to various practical application scenarios. Consequently, the work outlined in this section not only provides theoretical support for model design but also offers effective solutions for performance improvement in practical applications.

3.3. Training Method

For the labeled datasets $D = \{(x_i, y_i)\}_{i=1}^{N}$, where the input consists of image data x_i and corresponding annotation information y_i, we trained the proposed LRCNN model to learn an end-to-end mapping function f: $D \rightarrow G_i$, where D represents the input images and G_i represents the grasping predictions generated by the network from the images.

We conducted experiments to evaluate the impact of various loss functions on the performance of our network. After several trials, we identified that the ReLU activation function was the most effective at addressing the problem of gradient explosion. The loss function is defined as shown in Equation (13):

$$L(y_i, \hat{y}_i) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{13}$$

The expression for the ReLU activation function is given in Equation (14):

$$\mathrm{ReLU}(x) = \max(x, 0) = \begin{cases} 0, & x < 0 \\ x, & x > 0 \end{cases} \tag{14}$$

In this formulation, N represents the number of samples, $\mathrm{pos\_output}_i$, $\mathrm{cos\_output}_i$, $\mathrm{sin\_output}_i$, and $\mathrm{width\_output}_i$ are the ground-truth values for the position, cosine, sine, and width for the i sample, and $\mathrm{ground\_truth\_pos}_i$, $\mathrm{ground\_truth\_cos}_i$, $\mathrm{ground\_truth\_sin}_i$ and $\mathrm{ground\_truth\_width}_i$ represent the corresponding predicted values generated by the model.

The final total loss is calculated as a weighted sum of these individual loss components, as defined by the following Equation (15):

$$\text{total}_{\text{loss}} = \text{loss}_{\text{pos}} + \text{loss}_{\text{cos}} + \text{loss}_{\text{sin}} + \text{loss}_{\text{width}} \quad (15)$$

To handle the challenges of optimizing model training and scheduling, we used the Adam optimizer along with a learning rate scheduler, ReduceLROnPlateau. The Adam optimizer was configured using "optim.Adam", with the learning rate set to 1×10^{-5}. The ReduceLROnPlateau scheduler was applied to reduce the learning rate when the validation loss plateaued, allowing the model to continue learning more effectively even after initial training progress slowed. This approach ensured that the learning rate would be adjusted dynamically based on the validation performance, leading to better convergence during the training process.

4. Performance Testing and Evaluation of the Model Based on Public Datasets

In robotic grasping tasks, the performance of the model directly impacts its effectiveness and reliability in real-world applications. Evaluating the proposed model's validity and robustness through tests on public datasets is an essential step. By testing the model on standardized datasets, we can objectively and comprehensively assess its grasp detection capability across various scenarios and objects, as well as compare it with existing approaches. This chapter aims to test and analyze the performance of the improved model using publicly available robotic grasping datasets, including the Jacquard dataset [25] and the Cornell dataset [26]. First, we introduce the basic structure and characteristics of these datasets. Then, we elaborate on the model's testing process and evaluation metrics, showcasing the model's performance in grasp detection tasks through experimental results. Lastly, a comparison with other existing models is conducted to demonstrate the superiority of the proposed model, which integrates large kernel convolutions and residual networks.

4.1. Jacquard Dataset

Selecting an appropriate dataset is critical in the training of deep learning models. The quality, size, and diversity of the dataset directly influence the model's training effectiveness, generalization capability, and performance in practical applications. Poor dataset selection may lead to overfitting, poor generalization, or even an inability to meet the specific requirements of the task. Hence, we opted for the Jacquard dataset to train and evaluate the model. The Jacquard dataset is a publicly available dataset designed specifically for robotic grasping tasks, comprising over 54,000 RGB-D images and grasp annotations, covering a wide range of objects and grasping poses. Covering a wide range of categories from regular shapes (e.g., boxes, spherical objects) to irregular shapes (e.g., curved surfaces, curved and complex geometries), this diversity enables the model to learn the grasping characteristics of different object shapes, which is suitable for generalization in unstructured environments, the grasping of regular objects can help robots deal with stacked objects that are common in industrial manufacturing scenarios, and the grasping of irregular objects improves the adaptability of robots to heterogeneous objects in changing environments. The labels of the dataset are generated by 3D models, providing a large number of accurate grasping poses, and the grasping points of each object are labeled by a set of grasping attitude parameters (including grasping angle, grasping depth, etc.), and these label data meet the gripping requirements of the end effector of the robot. Through accurate annotation, it is ensured that more accurate grasping points are learned during training, and deviations caused by inconsistent manual annotation are avoided. This dataset, generated through simulation, supports the training and evaluation of models in complex environments for grasp detection tasks.

Additionally, during training, the choice of optimizer, learning rate scheduler, training parameters, and hardware configuration is crucial for the model's convergence, generalization, and training efficiency. Incorrect parameter settings can result in poor convergence,

overfitting, or underutilization of hardware resources, negatively affecting the model's performance in real-world tasks. Therefore, we utilized the Adam optimizer with the learning rate scheduler ReduceLROnPlateau. The model was trained for 100 epochs, with a batch size of 16, and a dropout rate of 0.1 to prevent overfitting. The experiments were conducted on a system equipped with an NVIDIA GTX 4060 GPU (16GB memory) and an Intel Core i5-13500HQ CPU. Running on Windows 11, CUDA 11.7, python version 3.11.5, and pytorch version 1.13 are used. GPU is from Nvidia in Taiwan, China, China. CPU from Intel in the United States

To evaluate grasping results, we adopted the rectangle metric proposed by Jiang et al. [17] to assess the model's performance. A grasp is considered successful if it meets the following criteria: (1) the Intersection over Union (IoU) between the predicted grasp rectangle and the corresponding ground truth exceeds 25%; (2) the difference between the predicted grasp angle and the true planar angle is less than 30 degrees. The dataset is divided into a training set and a testing set, where the former tests the model's generalization to new grasp poses and the latter validates its generalization to new objects. The calculation of the evaluation metric is shown in Equation (16).

$$\mathrm{IOU}(\mathrm{R}^*, \mathrm{R}) = \frac{|\mathrm{R}^* \bigcap \mathrm{R}|}{|\mathrm{R}^* \cup \mathrm{R}|} \tag{16}$$

Training and Evaluation on the Jacquard Dataset

Given the sufficient volume of the Jacquard dataset, there is no need for data augmentation. We used 90% of the Jacquard dataset to train the proposed network and the remaining 10% to validate the trained network. Grasp detection is evaluated based on the IoU metric, with an IoU threshold of 25%. Since the proposed model has only 600,000 parameters and each image takes an average of 15 ms to process, the model is well-suited for deployment in open-loop environments. The evaluation results are presented in Figure 11, and a comparison with existing methods is shown in Table 1.

Table 1. Comparison of LRCNN Model Performance with Existing Methods.

Method	Iput Size (Pixel × Pixel)	Number of Parameters	Accuracy (%)	Speed (ms)
GR-ConvNet v2	224 × 224	1,900,900	95.1	20 (GeForceGTX1080Ti, Nvidia, Taiwan, China)
GN	227 × 227	>11,000,000	96.0	120 (NVIDIATitan-X, Nvidia, Taiwan, China)
FCGN	227 × 227	>31,000,000	92.8	118 (NVIDIATITAN-X, Nvidia, Taiwan, China)
ROI-GD	—	>13,000,000	93.6	40 (GTX 1080TI GPU, Nvidia, Taiwan, China)
GR-ConvNet	224 × 224	1,900,900	94.6	20 (NVIDIA GeForceGTX1080Ti, Nvidia, Taiwan, China)
SE-ResUNet	224 × 224	—	95.7	25 (GTX 1080Ti, Nvidia, Taiwan, China)
LRCNN	300 × 300	600,000	95.0	15 (GTX 4060, Nvidia, Taiwan, China)

The ablation study is conducted to thoroughly analyze the contribution of each component within the model, determining which parts play a crucial role in improving overall performance. By systematically removing or modifying specific modules, we can assess the impact of each on the model's performance, validate the soundness of the design decisions, and uncover the model's robustness and optimization potential, providing a foundation for further improvements.

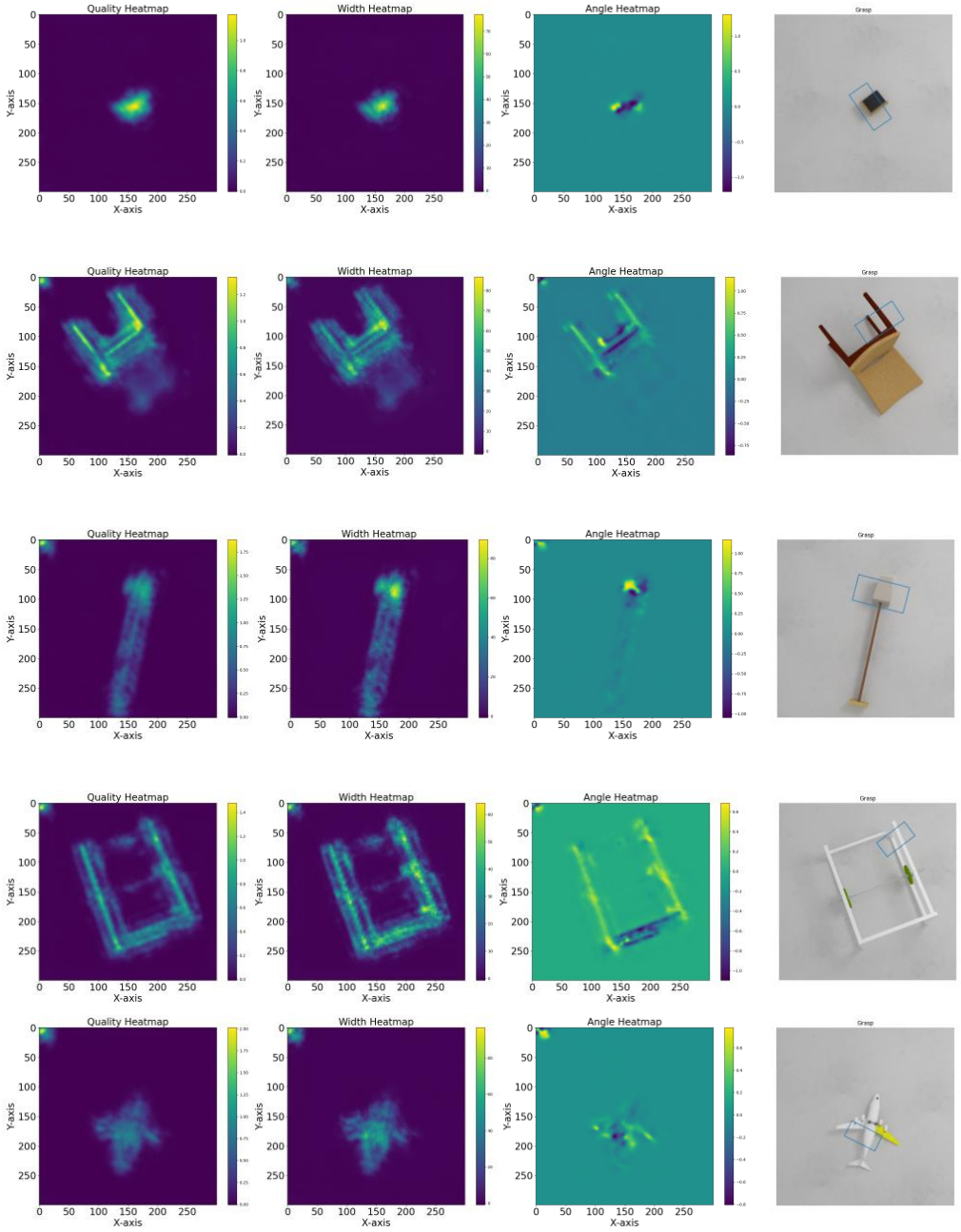


Figure 11. Results on the Jacquard Grasping Dataset: The First Three Columns Represent Model Outputs for Quality, Width, and Angle; The Fourth Column Shows the Grasping Rectangle Based on These Outputs. The blue box represents the grasping posture of the robot.

To verify the impact of large kernel convolutions and residual connections on the LRCNN model, we conducted experiments using the Jacquard dataset. The experimental setup includes the standard LRCNN model, a version without the large kernel convolution module (Version 1), a version without the residual connections (Version 2), and a version where both the large kernel convolutions and residual connections were removed (Version 3). All versions were trained under the same conditions, with their loss values and performance metrics recorded. By comparing the training process and final results of each version, we analyzed the specific contributions of large kernel convolutions and residual connections to the model's performance, providing insights for further optimization of the model.

The LRCNN model was compared with versions that lacked large kernel convolutions and residual connections. The training process for each version is shown in Figure 12, and the results of the ablation study are displayed in Figure 13.

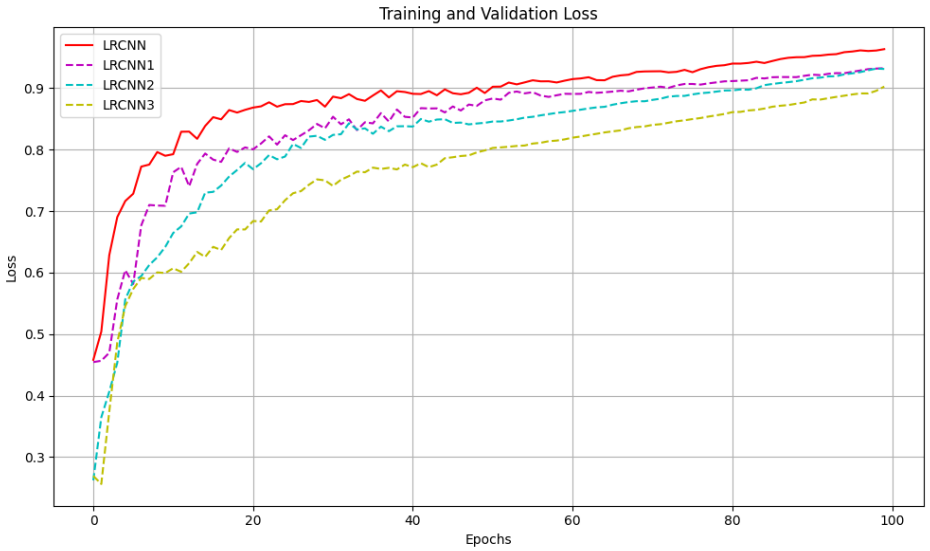


Figure 12. Training Process of Different Versions.

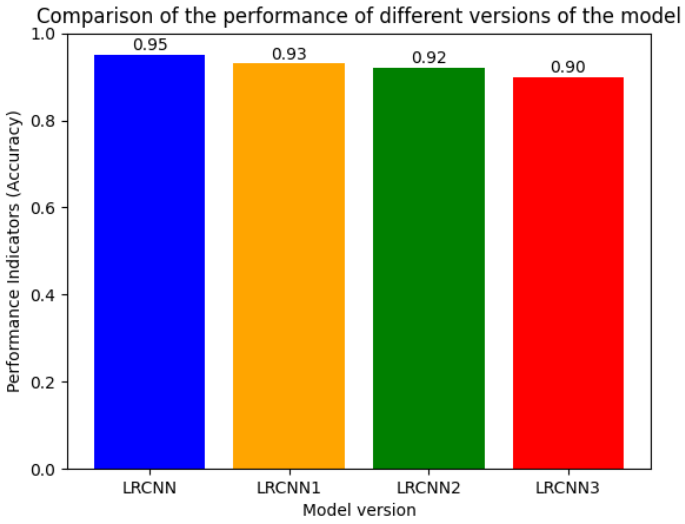


Figure 13. Comparison of Performance from Different Model Versions.

In order to better evaluate the gaps between different models, this paper selects a set of data from the dataset, as shown in Figure 14. The images were evaluated on four models.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}$$

where n is the total number of samples, LRCNN is used x_i, and LRCNN1, LRCNN2 and LRCNN3 are selected for $\hat{x}_i$. The results of the calculation using $RMSE$ are shown in Figure 15.

In this study, we quantitatively evaluated the performance of each model by root mean square error (RMSE). Specifically, LRCNN's RMSE is considered the best benchmark value, with RMSEs of 12.301 mm, 14.132 mm, and 47.144 mm in the X, Y, and Z directions, respectively. In comparison, the RMSE values of the other models are 11.709 mm, 13.826 mm, and 28.822 mm for LRCNN1, 11.634 mm, 13.484 mm, and 26.724 mm for LRCNN2, and

11.066 mm, 12.908 mm, and 25.233 mm for LRCNN3. These results show that, although the performance of other models is close to LRCNN in some dimensions, overall LRCNN still shows lower RMSE, indicating that it has a clear advantage in accuracy and stability in grasping tasks.

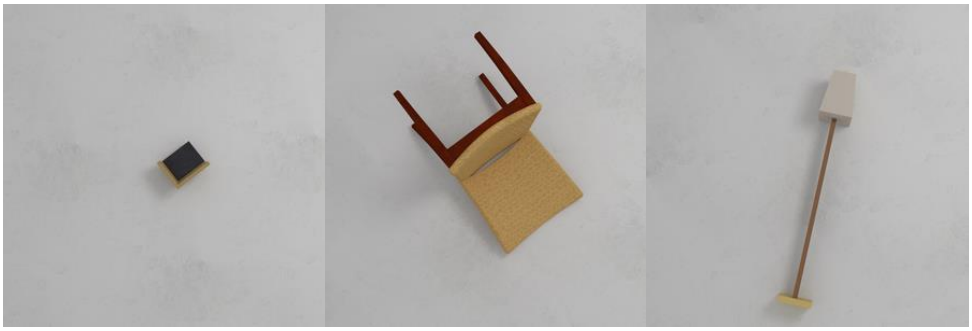


Figure 14. Schematic diagram of evaluating variant model data.

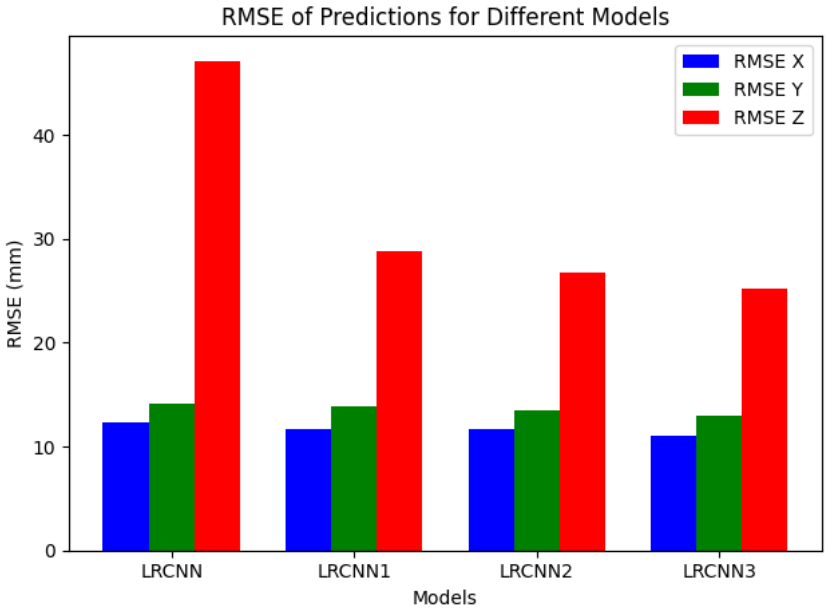


Figure 15. RMSE calculation results.

The results indicate that the model using only standard convolution structures performed the worst. The large kernel convolution module significantly expanded the model's receptive field, improving recognition accuracy, while the residual connection structure enhanced the model's ability to learn image features, supporting the recognition of unseen objects. When both large kernel convolutions and residual connections were combined, the model demonstrated its full potential across different views, further enhancing its capability to identify image feature information.

4.2. Model Training and Evaluation on the Cornell Dataset

The Cornell dataset is a widely used classic dataset in the field of robot grasping, which is mainly used for the training and evaluation of deep learning-based gripping detection models. The Cornell dataset contains about 50 different object types, covering objects commonly found in everyday life and industrial scenarios, including cups, boxes, tools, toys, and more. These objects have a wide range of shapes, including regular shapes, such as cubes and spheres, and irregular and complex geometries, such as tool shapes with bumps and depressions. Most of the objects in the Cornell dataset are placed on a desktop

in a laboratory environment, with a simple and unobstructed background, which allows the model to focus on the grasping characteristics of the objects. In some experiments, multiple objects are placed randomly to simulate a simple stacking situation. Although this design cannot fully simulate the unstructured environment, it still has high reference value for the basic training of the model. Although the Cornell dataset has limitations in scene complexity and sample size, it provides high-quality labels and reliable benchmarks for scraping tasks, which is suitable for verifying the performance of the model on basic scraping tasks and is one of the most widely used datasets in the field of robot scraping.

Following the cross-validation settings used in previous work, we employed both image-level and object-level data partitioning methods. In the image-level partitioning, the dataset is randomly split into training and validation sets while, in the object-level partitioning, objects present in the training set do not appear in the validation set. Table 2 compares the results of our proposed method with other existing methods for grasp prediction. Using the proposed LRCNN model, we achieved an accuracy of 97.0% with image-level partitioning and 96.0% with object-level partitioning. The results for unseen objects indicate that the proposed network is capable of predicting viable grasp positions for objects in the validation set. Furthermore, the recorded prediction speed of 15 milliseconds per image suggests that the proposed model is suitable for fast, real-time applications. Table 2 lists the results of various methods on the Cornell dataset, while Figure 16 shows the evaluation results on the Cornell dataset. Since the model underwent ablation testing on the Jacquard dataset, further ablation testing is not conducted in this section.

Table 2. Results of Different Methods on the Cornell Dataset.

Method	Iput Size (Pixel × Pixel)	Number of Parameters	Accuracy (%)	
			Image-Wise	Object-Wise
GR-ConvNet v2	224 × 224	1,900,900	98.8	97.7
GN	227 × 227	>11,000,000	96.0	96.1
GPN-GD	227 × 227	>31,000,000	97.2	97.1
DSGD	—	>13,000,000	97.5	—
LRCNN	300 × 300	600,000	97.0	96.5

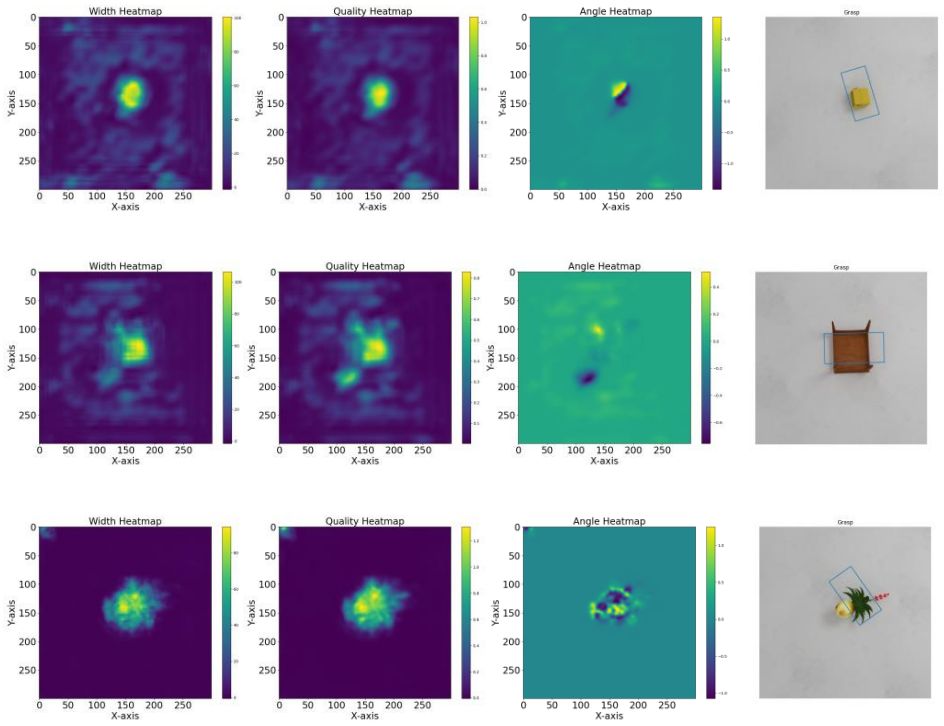


Figure 16. *Cont.*

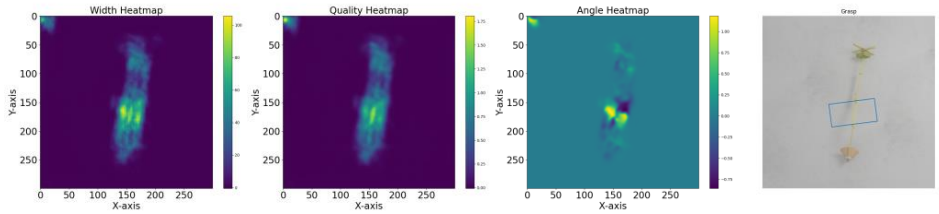


Figure 16. Results on the Cornell Grasping Dataset: The First Three Columns Represent Model Outputs for Quality, Width, and Angle; The Fourth Column Shows the Grasping Rectangle Based on These Outputs.

5. Robotic Grasping Experiments in Real-World Environments

In this section, to thoroughly evaluate the model's performance in real-world environments, we demonstrate that the model also performs exceptionally well in robotic grasping tasks involving actual objects. The model is not only capable of generating the most robust grasp points for single-target scenarios, but also can produce multiple grasp points for cluttered, multi-target scenarios. To validate the model's grasping performance in real-world settings, this experiment selected 18 objects of varying shapes, sizes, and materials, covering typical household items and adversarial objects, These objects have different geometric characteristics, including regular shapes (such as cubes) and irregular shapes (tools, furniture), etc., for application scenarios in the industrial environment, including hammers, wrenches, screwdrivers, pliers, etc., which can simulate the combination of objects in the unstructured situation in the real environment to a certain extent; at the same time, considering the clamping force of the end effector, the size and weight of the selected object are within the working range of the electric gripper, as shown in Figure 17. The selection criteria aimed to represent a wide range of grasping situations that may occur in real environments, with significant geometric differences among the objects. Adversarial objects included some 3D-printed items with complex geometric shapes or surface characteristics, which posed challenges for perception and grasping.

Figure 17. Types of Grasping Target.

The outcome of each grasp attempt (success or failure) was recorded. By calculating the grasp success rate for each object, key factors influencing grasp performance, such as the object's shape, material, and approach angle, were analyzed. This data provided important insights for further algorithm optimization.

The real-world experiments were conducted using an AUBO 6-DOF collaborative robot. A two-finger parallel electric gripper was used to grasp the objects. The Intel RealSense Depth Camera D435 was employed to capture real-world images, equipped with a pair of stereo lenses for depth data and a monocular RGB lens for color images. The camera was mounted externally on the robot, positioned vertically downward, capturing RGB-D images with a resolution of 640 × 480 pixels.

5.1. Single-Object Grasping Experiment

All experiments were conducted in a controlled environment, with the test surface being a flat, obstacle-free table, ensuring that each object was isolated during each experiment, with no interference from other objects. This environment setup aimed to eliminate external disturbances, enabling an accurate assessment of the robot's grasping capability. Each object was tested in 20 different positions and orientations to ensure diversity in the grasping tasks. The experimental process consisted of three parts: grasp inference, grasp execution, and success determination. Grasp inference was based on the RGB-D images obtained by the camera, with the system determining the optimal grasp pose. Grasp execution involved the robot moving the gripper to the computed grasp position and performing the grasp operation. A grasp was considered successful if the robot successfully lifted the object from the table and moved it to a designated location. Figure 18 illustrates the grasping process, and Figure 19 shows the grasping experiment results.

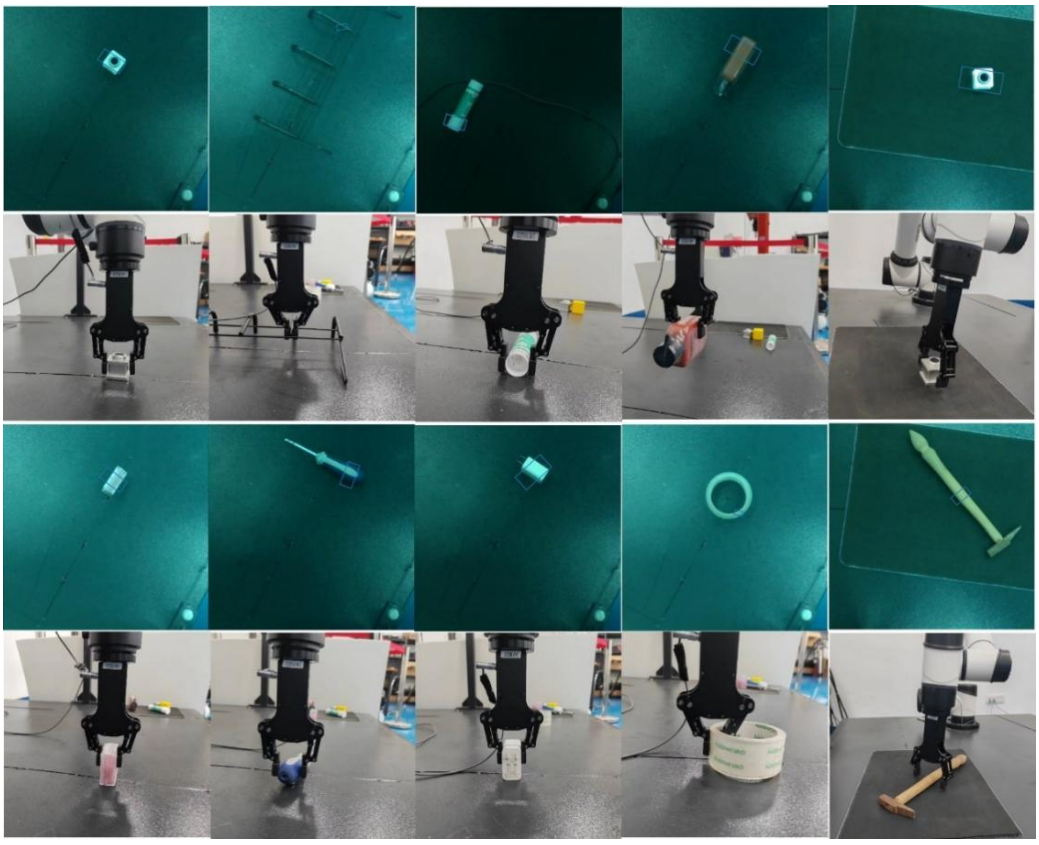


Figure 18. Single-Object Grasping Diagram.

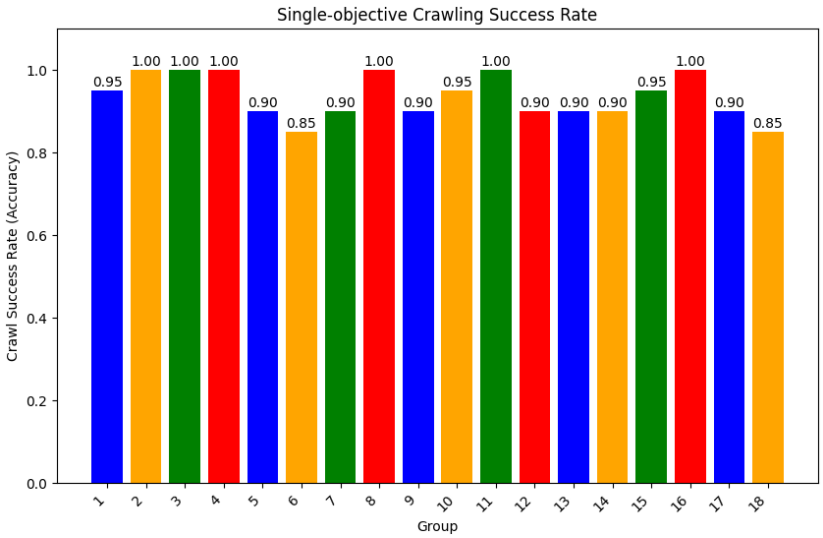


Figure 19. Robot grasping experiment results.

5.2. Multi-Object Grasping Experiment

In this section, to simulate cluttered multi-target scenarios, 7 to 10 different household items were randomly placed in the work area and randomly divided into four groups according to item numbers, with each group undergoing five rounds of grasping experiments. The group classifications are shown in Figure 20. In each experiment, the robot continuously attempted to grasp and remove objects from the scene. After each successful grasp, the robot placed the object in a designated location and continued to attempt to grasp the remaining objects. The experiment ended when all objects were removed from the scene or when no objects remained in the camera's field of view. Figure 21 illustrates the multi-object detection and grasping process.

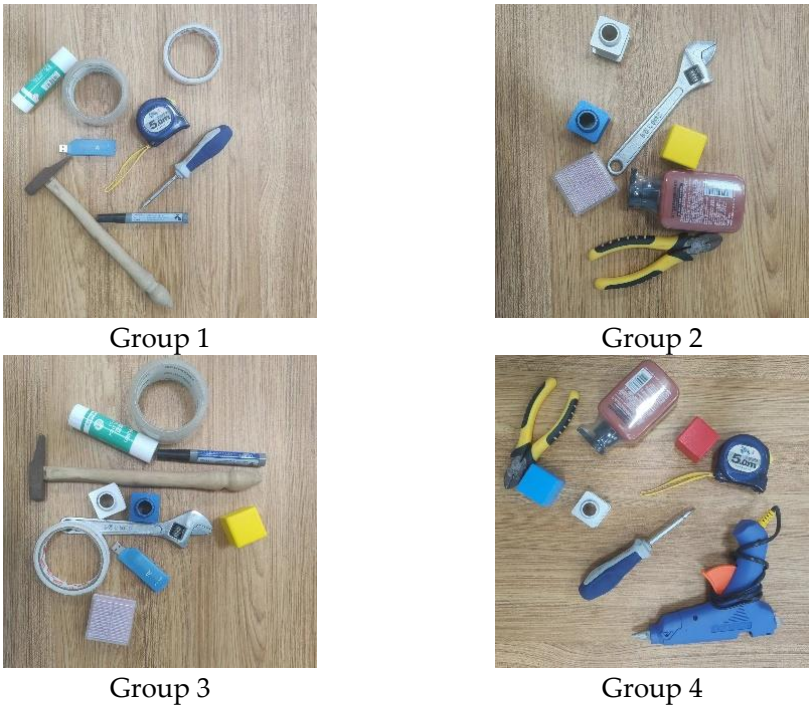


Figure 20. Group Classification Diagram.

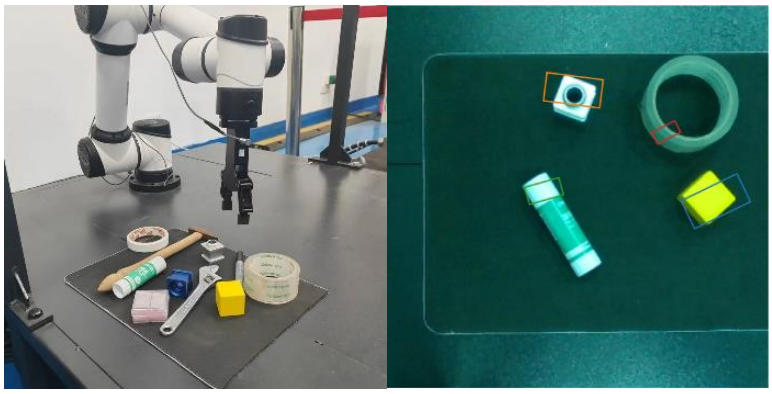


Figure 21. Multi-Object Detection Diagram.

As shown in Figure 22, the average grasp success rate across the experiments was 93.8%. The results of the multi-object grasping experiment demonstrated that the proposed model maintains a high success rate even when handling multiple objects, exhibiting strong robustness. Notably, when confronted with objects of complex shapes and with mutual interference, the model still managed to generate stable and accurate grasp plans. Additionally, the experiment showed that the model had significant advantages in terms of real-time performance, being able to respond quickly to grasping tasks, further validating its effectiveness and reliability in real-world applications.

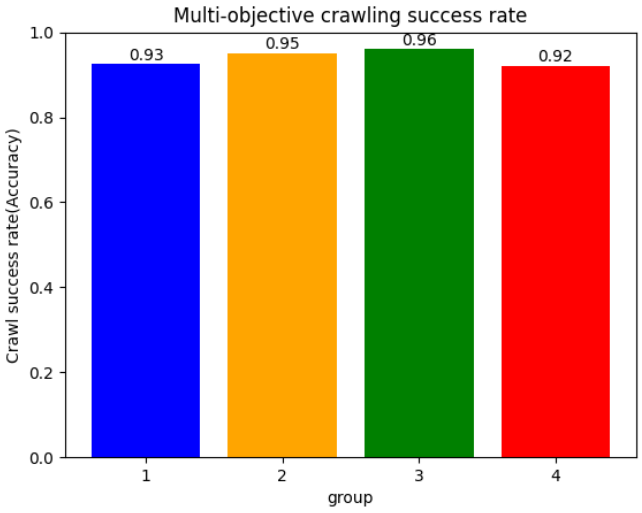


Figure 22. Multi-Object Grasping Success Rate.

6. Discussion and Conclusions

To address some of the existing challenges within current models, this paper presents a robot grasp detection method based on large kernel convolutions and residual connections. First, we analyzed the definitions of robot grasping approaches and the transformation relationships between image coordinate systems and robot coordinate systems to ensure the accuracy of coordinate conversion during grasping. Then, by integrating residual networks with large kernel convolutions, we proposed a hybrid method, incorporating structures such as depth-wise separable convolutions to optimize the network architecture, reducing model parameters while maintaining representational capacity. Efficient structural components, like SE Blocks, were also introduced to enhance model depth and improve its generalization capabilities. Subsequently, the proposed model was evaluated on publicly available datasets, including the Jacquard and Cornell datasets. Finally, real-world experiments were conducted using a robotic arm to grasp unseen objects, including common household items and objects in cluttered scenarios, verifying the method's effectiveness.

The results demonstrated that the system could reliably predict grasps for unseen objects and send these predictions to the robot for successful execution. Additionally, the model's fast prediction times make it suitable for closed-loop robotic grasping tasks. Although the training phase only used single-object datasets, the model performed robustly in predicting multiple objects in cluttered environments, indicating its potential adaptability to other industrial scenarios with promising results. This paper combines the large-kernel convolutional network with the residual network for the first time, applies it to the robot grasping scenario, and contributes a theoretical foundation for subsequent robot grasping in the unstructured environment.

This paper trains the model based on the public dataset, which can identify most common objects, has good generalization ability, and can successfully capture different kinds of objects but, when it is deployed in a specific factory, it is necessary to remake the dataset for the capture target and fine-tune the model to make it more in line with the target application scenario. Due to experimental limitations, the proposed model currently uses RGB-D images for 4-degree-of-freedom (DoF) grasp prediction, restricting its application to 6-DoF grasping. In addition, although RGB-D data provides basic spatial depth information, the existing models still lack the understanding of object characteristics in complex scenes, such as objects with significant differences in texture or physical properties. In addition to RGB-D data, incorporating other inputs, such as object surface textures or physical attributes, could enhance the model's understanding of object characteristics, improving grasp strategies. Moreover, utilizing self-supervised or reinforcement learning methods could enable the robot to gradually refine its grasp strategies through trial and feedback, helping it to adaptively learn the optimal grasping strategy in dynamic environments. This will help the robot to continuously adapt in the changing environment, thereby enhancing the generalization ability and robustness of the model in the unknown environment and providing a theoretical basis for applying this model to unstructured environments in the future.

Author Contributions: Overall scheme design, C.L.; Final draft review, L.L.; Writing, N.L.; Create a chart, R.N.; Literature search, Y.H.; Data analysis, W.Z.; Data collection, P.F. All authors have read and agreed to the published version of the manuscript.

Funding: Key Project of Shaanxi Provincial Department of Science and Technology (Program No. 2024QY2-GJHX-38), L.L.; Construction of the "Scientists + Engineers" Team in Qinchuangyuan, Shaanxi Province (Program No 2023KXJ-021), L.L.; Graduate Innovation Research Project Funding from Baoji University of Arts and Sciences (Program No.YJSCX23YB60), N.L.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chen, Y.L.; Cai, Y.R.; Cheng, M.Y. Vision-Based Robotic Object Grasping—A Deep Reinforcement Learning Approach. *Machines* **2023**, *11*, 275. [CrossRef]
2. Santoso, J.T.; Wibowo, M.C.; Raharjo, B. Predicting the robot's grip capacity on different objects using multi-object grasping. *Int. J. Intell. Robot. Appl.* **2024**, *8*, 546–559. [CrossRef]
3. Morrison, D.; Corke, P.; Leitner, J. Learning robust, real-time, reactive robotic grasping. *Int. J. Robot. Res.* **2020**, *39*, 183–201. [CrossRef]
4. Shi, C.; Miao, C.; Zhong, X.; Zhong, X.; Hu, H.; Liu, Q. Pixel-Reasoning-Based Robotics Fine Grasping for Novel Objects with Deep EDINet Structure. *Sensors* **2022**, *22*, 4283–4303. [CrossRef] [PubMed]
5. Wang, Y.; Zheng, Y.; Gao, B. Double-dot network for antipodal grasp detection. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 4654–4661.
6. Sun, H.; Cui, X.; Song, Z. Precise grabbing of overlapping objects system based on end-to-end deep neural network. *Comput. Commun.* **2021**, *176*, 138–145. [CrossRef]
7. Rasheed, M.; Jasim, W.M.; Farhan, R. Enhancing robotic Grasping with Attention Mechanism and Advanced Unet Architectures in Generative Grasping Convolutional Neural Networks. *Alex. Eng. J.* **2024**, *102*, 149–158. [CrossRef]
8. Zheng, T.; Wang, C.; Wan, Y.; Zhao, S.; Zhao, J.; Shan, D.; Zhu, Y. Grasping Pose Estimation for Robots Based on Convolutional Neural Networks. *Machines* **2023**, *11*, 974. [CrossRef]

9. Zhong, X.; Chen, Y.; Luo, J.; Shi, C.; Hu, H. A Novel Grasp Detection Algorithm with Multi-Target Semantic Segmentation for a Robot to Manipulate Cluttered Objects. *Machines* **2024**, *12*, 506. [CrossRef]
10. Lenz, I.; Lee, H.; Saxena, A. Deep learning for detecting robotic grasps. *Int. J. Robot. Res.* **2013**, *34*, 705–724. [CrossRef]
11. Zhang, H.; Lan, X.; Bai, S.; Zhou, X.; Tian, Z.; Zheng, N. Roi-based robotic grasp detection for object overlapping scenes. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 4768–4775.
12. Hosseini, H.; Masouleh, M.T.; Kalhor, A. Improving the Successful Robotic Grasp Detection Using Convolutional Neural Networks. In Proceedings of the 2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), Mashhad, Iran, 23–24 December 2020; pp. 1–6.
13. Kumra, S.; Kanan, C. Robotic grasp detection using deep convolutional neural networks. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 769–776.
14. Wu, Y.; Zhang, F.; Fu, Y. Real-Time Robotic Multigrasp Detection Using Anchor-Free Fully Convolutional Grasp Detector. *IEEE Trans. Ind. Electron.* **2022**, *69*, 13171–13181. [CrossRef]
15. Luo, W.; Li, Y.; Urtasun, R.; Zemel, R. Understanding the effective receptive field in deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 4905–4913.
16. Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P. Segformer: Simple and Efficient Design for Semantic Segmentation with Transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12077–12090.
17. Ding, X.; Zhang, Y.; Ge, Y.; Zhao, S.; Song, L.; Yue, X.; Shan, Y. UniRepLKNet: A Universal Perception Large-Kernel ConvNet for Audio Video Point Cloud Time-Series and Image Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 5513–5524.
18. Zhang, H.; Wu, C.; Zhang, Z. Resnest: Split-attention networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 2736–2746.
19. Tan, M.; Le, Q. Efficientnetv2: Smaller models and faster training. In Proceedings of the International Conference on Machine Learning, PMLR, Online, 18–24 July 2021; pp. 10096–10106.
20. Wang, W.; Li, S.; Shao, J. LKC-Net: Large kernel convolution object detection network. *Sci. Rep.* **2023**, *13*, 9535. [CrossRef] [PubMed]
21. Luo, P.; Xiao, G.; Gao, X. LKD-Net: Large kernel convolution network for single image dehazing. In Proceedings of the 2023 IEEE International Conference on Multimedia and Expo (ICME), Brisbane, Australia, 10–14 July 2023; pp. 1601–1606.
22. Guo, M.H.; Xu, T.X.; Liu, J.J. Attention mechanisms in computer vision: A survey. *Comput. Vis. Media* **2022**, *8*, 331–368. [CrossRef]
23. Ji, Y.; Zhang, H.; Jie, Z. CASNet: A cross-attention siamese network for video salient object detection. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 2676–2690. [CrossRef] [PubMed]
24. Howard, A.G. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
25. Depierre, A.; Dellandréa, E.; Chen, L. Jacquard: A large scale dataset for robotic grasp detection. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3511–3516.
26. Jiang, Y.; Moseson, S.; Saxena, A. Efficient grasping from rgbd images: Learning using a new rectangle representation. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3304–3311.