applied sciences

MDPI

*Article*

# CenterInst: Center-Based Real-Time Instance Segmentation

**Shu Tian *,† and Liang Ren †**

School of Computer & Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
* Correspondence: shutian@ustb.edu.cn
† These authors contributed equally to this work.

**Abstract:** Instance segmentation is a computer vision task that aims to give each pixel in an image an instance-specific label. Recently, researchers have shown growing interest in real-time instance segmentation. In this paper, we propose a novel center-based real-time instance segmentation method (CenterInst), which follows the FastInst meta-architecture. Key design aspects include a center-guided query selector, a center-guided sampling-based query decoder, and a lightweight dual-path decoder. The center-guided query selector selects queries via the per-pixel prediction of center point probabilities, avoiding excessive query proposals for single instances. The center-guided sampling-based query decoder adaptively generates local sampling points based on center positions, employing adaptive mixing to update queries without irrelevant sampling disturbances. The lightweight dual-path decoder enhances inference speed and maintains accuracy via pixel decoding on every layer during training but only utilizing the final layer's decoder during inference. The experimental results show CenterInst achieves superior accuracy and speed compared to state-of-the-art real-time instance segmentation methods.

**Keywords:** instance segmentation; real time; center-based



**Citation:** Tian, S.; Ren, L. CenterInst: Center-Based Real-Time Instance Segmentation. *Appl. Sci.* **2024**, *14*, 1999. https://doi.org/10.3390/app14051999

Academic Editor: Christos Bouras

Received: 18 January 2024
Revised: 21 February 2024
Accepted: 26 February 2024
Published: 28 February 2024



## 1. Introduction

Image segmentation addresses the problem of labeling pixels. There are different types of segmentation tasks, such as semantic, instance, and panoptic segmentation. Instance segmentation can be described as a combination of image segmentation and object detection. It aims to detect all objects and label all pixels that belong to the same object.

Instance segmentation is often time-consuming. For SOLO [1,2], researchers have started to pay attention to real-time instance segmentation. However, SOLO is based on dense detection. Therefore, nonmaximum suppression (NMS) is utilized for postprocessing, which limits the scale of the model. SparseInst [3] uses sparse detection to improve SOLO by eliminating NMS and completing the prediction of instance segmentation by generating an instance activation map. Instead of a convolution-based decoder, FastInst [4] uses Transformer to construct a double-channel decoder to complete real-time instance segmentation.

Although FastInst achieves better performance, there are still some problems in real-time instance segmentation: (1) The many-to-one problem in sparse detection occurs easily. FastInst takes advantage of a semantic segmentation network to extract proposal queries. However, semantic segmentation-based methods cannot easily extract a small number of effective proposal queries for one instance. (2) Feature sampling is not efficient. FastInst uses the cross-attention of Transformer to complete feature sampling from an E3 feature map. It relies on single-scale sampling in the E3 layer, which results in limited performance. To improve accuracy, the segmentation results of the previous stage are regarded as a mask to help the decoder focus less on irrelevant areas. However, it is not efficient enough, because a calculation of the similarity of all pixel features is still needed. Therefore, the multiscale feature cannot be used due to the low speed resulting from sampling the whole image. Additionally, the irrelevant features from the whole image lower the performance.

(3) The pixel features are updated too many times. FastInst uses the matching results of the first iteration as a limitation, and the training method introduced by GT mask-guided training is not effective enough. The pixel features need to update at every stage.

In this study, we solved the above problems by developing CenterInst, a novel center-based real-time instance segmentation model following the meta-architecture of FastInst. CenterInst boasts lower computational complexity, faster computation speed, and higher inference accuracy than FastInst. Our contributions are summarized below:

We developed an innovative center-based instance segmentation architecture that leverages the central information contained within instances. In this architecture, we generate initial queries guided by the center and perform local sampling based on center information. We then update query vectors using the sampled features. Finally, we generate mask kernels based on the updated query vectors to predict the instance segmentation results.

Inspired by CenterNet [5], we propose a center-guided query selector, which achieves more accurate initialization query selection by predicting the center point probability of each pixel on the feature map. Compared to segmentation-based methods, this approach alleviates the issue of generating too many queries for the same object.

We propose a center-guided sampling-based query decoder, which achieves local sampling based on instance centers. Compared to global sampling, it significantly reduces the number of sampled features and improves the correlation between sampled features and queries, thereby enhancing accuracy while maintaining inference speed.

To further improve the model's inference speed and optimize its real-time performance, we developed a lightweight inference structure and a corresponding training method. This structure enhances the inference speed by reducing the frequency of pixel feature updates. Additionally, during training, it retains the pixel feature update branches at each stage to assist in query update learning. This ensures both the enhancement in inference speed and the preservation of model accuracy.

We evaluated the performance of CenterInst on the MS COCO 2017 dataset [6] and compared it with that of state-of-the-art methods. The results demonstrate that our approach outperforms the current state-of-the-art method, FastInst, in terms of both speed and accuracy.

## 2. Related Work

The existing instance segmentation methods can be grouped into three categories, i.e., region-, center-pixel-, and sparse-based methods.

Region-based methods are traditional methods that first detect objects to determine the bounding box and then utilize fully convolutional networks [7] on candidate regions to perform segmentation. Mask R-CNN [8] is a classic region-based method that builds upon faster R-CNN [9] by adding a fully convolutional network for predicting segmentation results, enabling it to make predictions for classification, detection, and segmentation. Some methods [10,11] improve upon mask R-CNN to obtain a more precise bounding box and segmentation mask. As the computational burden is heavy, it is difficult to establish them as real-time methods.

Center-pixel-based methods assume that each pixel in the feature map can serve as the center of an instance and predict instance masks based on these centers. After mask prediction, NMS is applied for filtering. Typically, these methods are modifications of networks based on single-stage detectors like CenterNet, FCOS [12], etc. Among them, CondInst [13] decouples the task into mask kernel prediction and mask feature learning to generate object masks; YOLACT [14,15] improves mask quality by cropping segmentation regions based on single-stage detection results; and SOLO eliminates the detection head, directly predicting the segmentation results from the centers, and uses segmentation NMS to filter duplicate results. Since these methods are primarily single-stage approaches, they are fast.

These methods rely on many-to-one prediction and use NMS to eliminate redundancy. NMS has received much attention in real-time instance segmentation tasks. Based on

the bipartite matching sample assignment strategy, initially widely applied to end-to-end object detection and subsequently to instance segmentation, sparse-based detection methods using bipartite matching do not require postprocessing with NMS to eliminate redundant predictions. K-Net [16] is based on sparse instance convolution kernels, and segmentation is achieved using the kernel update head. QueryInst [17] was designed based on a sparse detector (sparse R-CNN [18]). The results are generated by a convolution kernel produced by the query and the proposal extracted by sparse R-CNN. SparseInst predicts sparse instance activation maps to represent objects. SOLQ [19] learns mask embeddings for instance segmentation. In methods based on Transformer [18,20] decoders, Mask2Former leverages the Transformer structure to enhance its instance segmentation performance through masked attention. Mask DINO [21] and MP-Former [22] enhance the decoding capability of the decoder by using noisy labeled inputs, improving their overall performance. While these methods significantly reduce the number of predicted objects, it is difficult to improve the inference speed due to the complex decoder structures of some of these methods.

The state-of-the-art real-time instance segmentation method, FastInst [4], is based on the Mask2Former structure. It introduces instance activation to guide query generation and employs a dual-path decoding structure for query decoding, achieving improved speed and accuracy. Our CenterInst model builds upon FastInst, further enhancing its inference efficiency and improving accuracy.

## 3. Methods

### *3.1. Overall Architecture*

As illustrated in Figure 1, CenterInst consists of three modules: the backbone, neck, and dual-path decoder. We began by inputting the image $Img \in \mathbb{R}^{H\times W\times 3}$ into the backbone network, which yielded features at three distinct scales: $B_3$, $B_4$, and $B_5$. These multiscale features were further processed using the PPM-FPN [3] module for extraction and enhancement, resulting in feature maps $E_3$, $E_4$, and $E_5$. These feature maps had resolutions of 1/8, 1/16, and 1/32 relative to the input image, respectively.

Next, the center-guided query selector estimated the likelihood of each pixel in feature map $E_4$ being an instance center and selected the top $N$ probabilities to propose instance center positions as follows: $C_0 \in \mathbb{R}^{N\times 4}$. Using these central position proposals, we performed bilinear interpolation sampling on the high-resolution feature map $E_3$ to generate $N$ instance query proposals, represented as $Q_0 \in \mathbb{R}^{N\times 256}$.

Following this, we fed these query proposals $Q_0$, central position proposals $C_0$, and multiscale feature maps into the dual-path decoder, which consists of two branches: the query decoder and pixel feature decoder. The center-guided sampling module of the query decoder branch calculates multiscale offsets for each instance query centered on its respective center point, enabling the determination of sampling point positions. Bilinear interpolation was used to sample the multiscale features, followed by a query update through adaptive mixing and self-attention. Subsequently, cross-attention was applied between the updated queries and the high-resolution pixel features $E3$ from the neck, thereby updating the pixel features used for assembling the segmentation results. The updated pixel features are denoted as $F_{mask} \in \mathbb{R}^{(\frac{H}{8}\times\frac{W}{8})\times 256}$. The dual-path decoder iteratively updated $I$ times, recursively utilizing the updated queries and center point positions from the previous stage. After each iteration, the updated queries $Q_I$ and updated pixel features $F_{mask}$ were employed for tasks related to detection, classification, and segmentation.

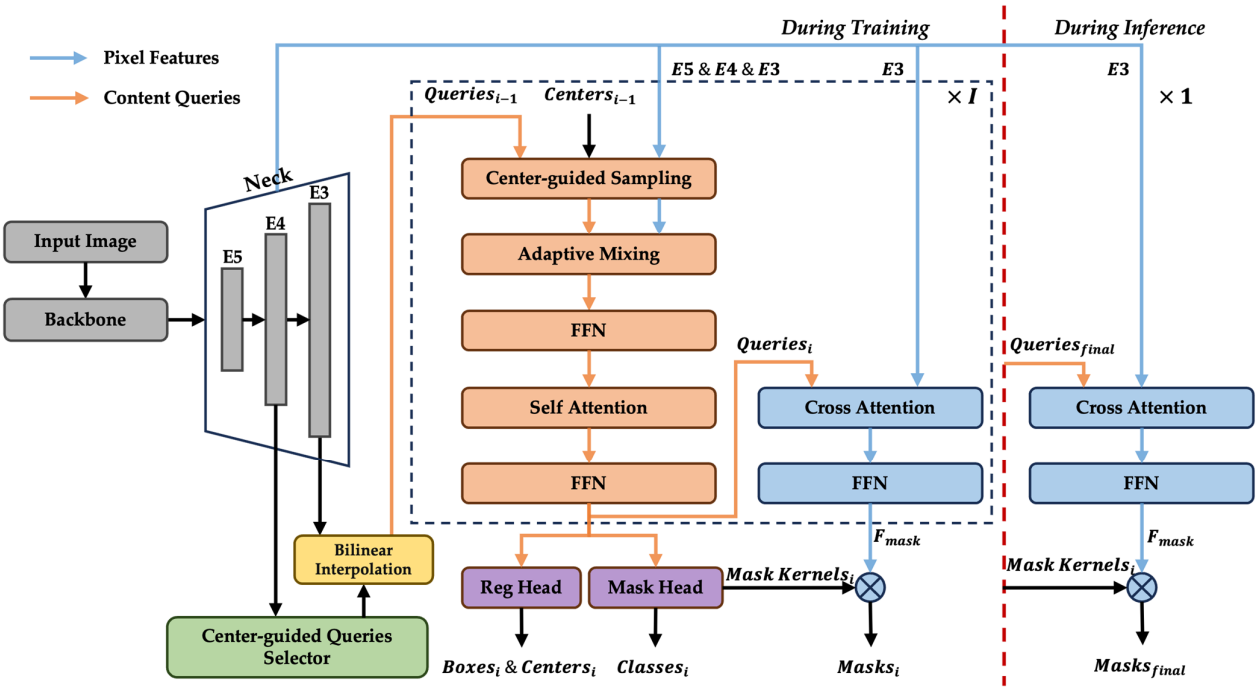


**Figure 1.** Model overview. CenterInst extracts and refines multiscale features through the backbone and neck. The center-guided query selector (Section 3.2) selects $N$ instance center position proposals from feature $E_4$ and obtains instance query proposals from feature $E_3$ based on these center positions. Subsequently, these queries and center positions are updated in a dual-path decoder (Section 3.3), which iteratively updates through $L$ stages, predicting classification, detection, and segmentation after each stage. Pruning the pixel feature decoding branch during inference further accelerates the processing speed (Section 3.3).

### *3.2. Center-Guided Query Selector*

Research [4,23] indicates that the method of initializing object queries can significantly impact the convergence speed and accuracy of a model. Sparse and precise query proposals can effectively enhance model performance. Inspired by Deformable DETR [23], FastInst adopts a semantic segmentation approach, filtering query proposals through the per-pixel prediction of class probabilities. However, in semantic segmentation tasks, it is necessary to perceive the entirety of an instance, which leads to high-probability classification predictions at multiple different locations of the same instance, resulting in redundant query proposals for the same instance. To ensure the sparsity and precision of queries, we were inspired by instance segmentation methods based on center point prediction models like SOLO and CondInst. Drawing on the CenterNet approach, we developed a center-guided query selector, which selects queries through the per-pixel prediction of center point probabilities. Since an instance has only one center point, using center points can effectively avoid generating too many query proposals for a single instance, and the introduction of center points also allows for more precise query proposals during sampling. The specific implementation is as follows:

In the center-guided query selector, we fed the input features $E4$ from the neck into the detection head to predict the center probabilities. The detection head consists of $3 \times 3$ convolution followed by $1 \times 1$ convolution:

$$[P_c,\ P_o, P_{wh}] = HEAD_{det}(E_4) \tag{1}$$

where $P_c \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times cls}$ is the per-pixel center point probability, $P_o \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times 2}$ is the relative center offset, and $P_{wh} \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times 2}$ is the target dimensions of $E_4$.

Subsequently, we selected the top $N$ pixel positions based on center probabilities $Pc$ as preliminary candidates for the instance locations:

$$\left[P_c^{topN},\ I^{topN}\right] = TopN(P_c) \tag{2}$$

where $P_c^{topN}$ represents the top $N$ center point probabilities, and $I^{topN}$ represents the indices of $P_c^{topN}$.

Subsequently, based on the selected pixel positions and the center point offset results $P_o$, we calculated the actual position of the center point $[X,\ Y]$, denoted as $C_0$.

$$C_0 = [X,\ Y] = Coord\left(I^{topN}\right) + P_o\left[I^{topN}\right] \tag{3}$$

Based on the center position proposals $C_0$, we obtained $N$ instance query proposals $Q_0$ from the high-resolution features $E_3$ using the bilinear interpolation method:

$$Q_0 = Bilinear(E3,\ C_0) \tag{4}$$

where $Bilinear(A,\ B)$ denotes the sampling of $A$ from location $B$. Extracting instance queries from high-resolution features provides higher initial query precision than when using low-resolution features, and utilizing bilinear interpolation for precise coordinate sampling can further enhance initial query precision.

### *3.3. Dual-Path Decoder*

In CenterInst, the dual-path decoder plays a crucial role in decoding query vectors and pixel features. It consists of two primary components: the query decoder branch and the pixel feature decoder branch. The task of the query decoder branch is sampling pixel features and utilizing these samples to update the query vector. Meanwhile, the pixel feature decoder branch, guided by the query vector, further refines and optimizes the pixel features. The dual-path decoder iteratively updates the query vector and pixel features and, after each iteration, predicts the results for detection, classification, and segmentation tasks.

#### 3.3.1. Center-Guided Sampling-Based Query Decoder

Numerous studies [21,24,25] have demonstrated that the introduction of multiscale features can significantly enhance the segmentation performance of instance segmentation models. However, in real-time instance segmentation tasks, for many methods, such as SparseInst and FastInst, to meet the requirements of real-time processing, the utilization of multiscale features in the decoder is often discarded due to constraints on inference time and computational costs. These methods employ a full-pixel attention mechanism in the decoding process. Their computational costs are heavy, as they require attention to all pixel features on the feature map for each instance query. Nevertheless, the use of full-pixel attention not only leads to an increase in computational complexity but may also compromise accuracy. For FastInst, this issue was identified, but, despite attempting to improve performance by reducing irrelevant receptive fields through the addition of masked attention [4], computational costs were not successfully reduced.

To further enhance model performance and alleviate computational burdens, we propose a center-guided sampling-based query decoder. By generating sampling points near the instance center, the number of sampling points is significantly reduced, which effectively reduces computational costs. This approach not only enables the generation of specific sparse sampling distributions for different queries, but also avoids the introduction of excessive irrelevant sampling points, thereby improving model accuracy.

The center-guided sampling-based query decoder is primarily composed of three parts: center-guided adaptive sampling, adaptive mixing, and self-attention. In the initial stage $S_0$, the input consists of the query proposal $Q_0$ and the center position proposal $C_0$

output by the query selector. In the subsequent iterative process, the content queries $Q_{l-1}$ and center positions $C_{l-1}$ updated in the previous stage are utilized.

Center-guided adaptive sampling: We employed a linear layer to adaptively generate a set of sampling offsets $\{(\Delta x_{ij}, \Delta y_{ij})\} \in \mathbb{R}^{3\times N_{point}\times N_{head}\times 2}$ for each layer's feature map. Similar to multihead attention, $N_{point}$ and $N_{head}$ represent the number of sampling points per decoding head and the number of decoding heads, respectively. The sampling point coordinates were obtained by summing the offsets with the instance center position, denoted as $P_{ij} \in \mathbb{R}^{3\times N_{point}\times N_{head}\times 2}$, where $i$ and $j$, respectively, denote the indices of the sampling point and the sampling head, with $i \in [0,\ N_{point})$, $j \in [0, N_{head})$:

$$\{(\Delta x_{ij}, \Delta y_{ij})\} = Linear(Q_{i-1}) \tag{5}$$

$$P_{ij} = C_{l-1} + \{(\Delta x_{ij}, \Delta y_{ij})\} \tag{6}$$

Subsequently, through bilinear interpolation sampling, we sequentially completed the sampling of multiscale features layer by layer to obtain the sampled feature $f \in \mathbb{R}^{N_P\times N_C}$:

$$f = Bilinear(F,\ P_{ij}) \tag{7}$$

where $F$ represents the multiscale features, $N_P = 3 \times N_{point} \times N_{head}$ is the number of sampling points, and $N_C = \frac{256}{N_{head}}$ is the channel size.

AdaMixer [26] is an efficient method for the adaptive decoding of sampled features, termed adaptive mixing. This approach focuses solely on the sampled features, not requiring computation for all features. Following center-guided adaptive sampling, we utilized adaptive mixing to generate a query update based on the sampled features.

Furthermore, our decoder adopts a structure with cross-attention followed by self-attention. Specifically, the decoder first utilizes adaptive mixing to update queries through sampled features and then further integrates the relationships between queries through self-attention.

#### 3.3.2. Pixel Feature Decoder

The pixel feature decoder is the same as in FastInst, where we implemented cross-attention through multihead attention. In the pixel feature decoder, we generated queries using only the feature map E3 from the neck output, without utilizing multiscale features. Simultaneously, we generated keys and values using the updated query vectors. The goal was to further refine the instance information in pixel features using the updated queries, thereby enhancing the segmentation accuracy.

$$q = Linear(E3) \tag{8}$$

$$k,\ v = Linear(Q_l) \tag{9}$$

For the updated queries $Q_l$ and pixel features $E_3^*$, we used regression and segmentation heads to predict the segmentation results and center point positions. Specifically, our regression head predicts center point offsets and instance scales from the queries through a feedforward network. The segmentation head generates segmentation mask weights from the queries and predicts instance categories. Finally, the instance segmentation result was computed by applying the segmentation mask weights to the pixel features.

$$C_l = C_{l-1} + Head_{reg}(Q_l) \tag{10}$$

$$Cls, W_{mask} = Head_{mask}(Q_l) \tag{11}$$

$$Mask_l = W_{mask} * E_3^* \tag{12}$$

3.3.3. Lightweight Inference Architecture

To further improve the inference speed of the model, we introduce an innovative lightweight inference structure.

During training, we used the updated content queries and center positions from the previous stage as inputs. However, unlike FastInst, the pixel feature input only comes from the neck, not the previous stage. Subsequently, we updated the content queries and pixel features through a dual-path decoder. The detection, classification, and segmentation tasks were then completed through the regression and detection heads, with the detection head utilizing a structure comprising a 3-layer MLP. Predictions from each stage were supervised through respective losses. Regression loss employed GIOU loss and L1 loss; classification loss employed cross-entropy loss; and segmentation loss combined dice loss and cross-entropy loss calculations.

During inference, owing to the design of the dual-path decoder, only content queries and center positions are updated and used for the next stage at each step. Additionally, predictions for center points and updates to content queries do not depend on updates to pixel features. In other words, we only need to update pixel features in the final stage and utilize the updated pixel features to calculate the segmentation and classification results in inference. For the other stages, only updates to content queries and center point coordinates are necessary, without the need for computing segmentation and classification results.

Through this lightweight inference structure, we significantly reduced the computational load associated with decoding pixel features in multistage scenarios, thereby further shortening the inference time. Our experimental results demonstrated that this operation not only reduces inference time but also preserves model accuracy.

### *3.4. Contrastive Denoising Training*

Inspired by DINO [21,27], during the training process, we utilized real labels from the detection task and introduced artificially set noise with a magnitude of $\lambda$. Specifically, we designated the $[0,\ \lambda]$ interval as positive samples and the $[\lambda,\ 2\lambda]$ interval as negative samples. Multiple sets of positive and negative samples, generated from real labels, were input into a dual-path decoder. The goal was to enhance the decoding capability of the dual-path decoder by introducing noisy samples.

In this process, to prevent the possibility of label leakage, we implemented masks to shield the query vectors within the self-attention scope. It is crucial to note that due to the adoption of a dual-path decoder structure, during the feature query decoding process, to avoid label leakage, queries from the contrastive denoising group cannot be introduced. This ensures the accuracy and reliability of the model's training process.

## 4. Experimental Results

Next, we evaluated the performance of CenterInst on the MS COCO 2017 dataset [6] and compared it with that of state-of-the-art methods. We provide details of ablation experiments conducted to validate the effectiveness of each proposed component.

### *4.1. Implementation Details*

Our model was implemented using Detectron2 (v0.6) [28]. We used the AdamW [29] optimizer with a step learning rate schedule. The initial learning rate was 0.0001, and the weight decay was 0.05. We applied a learning rate multiplier of 0.1 to the backbone, which was ImageNet-pretrained, and decayed the learning rate by 10 at fractions 0.9 and 0.95 of the total number of training iterations. We trained our model for 36 epochs with a batch size of 16. For data augmentation, we used the same scale jittering and random cropping as in FastInst. For example, the shorter edge varied from 416 to 640 pixels, and the longer edge was no more than 864 pixels. We set the loss weights λcls, λce, λdice, λbox, and λgiou to 2.0, 5.0, 5.0, 2.0, and 1.0, respectively. λhm, λhw, and λloc were set to 1.0, 0.1, and 1.0, respectively. We set the query selector to generate $N = 100$ queries by default, and we predicted the 80 object categories in the COCO dataset. We report the AP performance

as well as the FLOPs and FPS. The FLOPs were averaged using 100 validation images. The FPS was measured on an RTX 3090Ti GPU with a batch size of one using the entire validation set. Unless specified, we used a shorter edge of 640 pixels with a longer edge not exceeding 864 pixels to test and benchmark the models.

### *4.2. Experimental Results*

In Table 1, we compare the performance of our method (CenterInst) with that of some state-of-the-art methods on the COCO test-dev dataset, focusing on accuracy and inference speed, as CenterInst is primarily designed for real-time instance segmentation tasks. To objectively measure the complexity of the proposed model, floating point operations (FLOPs) were computed in this study [30–32]. The evaluation was conducted on COCO test-dev. We provided two variations of CenterInst with different backbones and compared them with other state-of-the-art methods using the same backbone. The results indicated that CenterInst achieved higher accuracy while maintaining speed. Specifically, with the ResNet50 [33] backbone, our proposed CenterInst model achieved a 0.6 AP improvement over the current leading real-time instance segmentation model, FastInst, with fewer training epochs and less inference time. Additionally, we presented CenterInst-large, employing a dual-path inference structure like FastInst. It achieved a 0.9 AP improvement over FastInst but with a slight increase in inference time. Furthermore, using the ResNet-50-d-DCN [34,35] as its backbone, our model achieved a 49.2 FPS and a 41.0 AP, demonstrating higher efficiency and accuracy than FastInst, validating the effectiveness of our model.

**Table 1.** Performance of instance segmentation on COCO test-dev. We compared CenterInst with state-of-the-art methods, and it outperformed most previous real-time instance segmentation methods in both inference speed and accuracy. Notably, CenterInst surpassed the previous leading method, FastInst.

| Method | Backbone | Epoch | Size | $AP_{test}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $FLOP_S$ | FPS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SOLOv2 [2] | R50 | 36 | 448 | 34.0 | 54.0 | 36.1 | 12.9 | 34.7 | 48.7 | – | – |
| SparseInst [3] | R50 | 144 | 608 | 34.7 | 55.3 | 36.6 | 14.3 | 36.2 | 50.7 | – | – |
| YOLACT [14] | R50 | 54 | 550 | 28.2 | 46.4 | 29.2 | 9.2 | 29.3 | 44.8 | – | – |
| CondInst [13] | R50 | 36 | 800 | 37.8 | 59.1 | 40.5 | **21.0** | 40.3 | 48.7 | – | – |
| FastInst [4] | R50 | 50 | 640 | 38.6 | 60.2 | 40.6 | 10.8 | **56.2** | **75.2** | 75.5 G | 50.3 |
| CenterInst | R50 | 36 | 640 | 39.2 | 60.9 | 41.6 | 17.7 | 41.5 | 56.7 | 69.1 G | 54.1 |
| CenterInst-large | R50 | 36 | 640 | **39.5** | **61.3** | **41.8** | 18.0 | 41.9 | 57.3 | 84.9 G | 45.5 |
| SparseInst | R50-d-DCN | 144 | 608 | 37.9 | 59.2 | 40.2 | 15.7 | 39.4 | 56.9 | – | – |
| FastInst | R50-d-DCN | 50 | 640 | 40.5 | 62.6 | 42.9 | 10.0 | **54.9** | **74.5** | 77.9 G | 46.9 |
| CenterInst | R50-d-DCN | 36 | 640 | **40.9** | **63.1** | **43.4** | **18.9** | 43.4 | 59.8 | 71.5 G | 48.3 |

The "Epoch" column indicates the total number of epochs required during training, the "Size" column represents the dimensions of the input images during inference, and "FLOPs" denotes the floating point operations performed by the model during inference on input images. Bold data indicates that this method outperforms others in this metric. The same applies to bold data in subsequent tables.

### *4.3. Ablation Studies*

We analyzed the performance of CenterInst through a series of ablation experiments. Initially, we validated the effectiveness of the key components of the three proposals: the center-guided query selector, the center-based query decoder, and the lightweight structure. Subsequently, we explored the impact of other designs on CenterInst. Unless otherwise specified, we conducted the experiments using the ResNet-50 backbone for CenterInst with a lightweight design and without the contrastive denoising training strategy. All ablation results were evaluated on the COCO val2017 dataset.

Center-guided query selector: As shown in Table 2, we conducted comparative experiments between the segmentation-guided query selector in FastInst and our proposed center-guided query selector. The results indicated that the center-guided query selector exhibited superior performance while incurring almost no additional computational cost or inference time. This suggests that query proposals obtained based on the center position

are more accurate and representative, emphasizing the effectiveness of relying on the center position.

**Table 2.** Query selector. Our center-based query selector demonstrated superior performance with an almost negligible increase in inference time.

| Query Selector | $AP_{val}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $FLOP_S$ | FPS |
|---|---|---|---|---|---|---|---|---|
| Seg-guided | 37.3 | 59.0 | 38.9 | 16.5 | 40.5 | 59.1 | 69.0 G | 54.1 |
| Center-guided | **37.8** | **59.5** | **39.5** | **16.9** | **40.9** | **59.5** | 69.1 G | 54.1 |

"Seg-guided" denotes the initial query selection achieved through semantic segmentation, while "Center-guided" refers to our proposed method of selecting initial queries by predicting instance center probabilities. Bold data indicates that this method outperforms others in this metric. The same applies to bold data in subsequent tables.

Center-guided sampling query decoder: As shown in Table 3, our center-based query decoder with local sampling outperformed the global sampling query decoder. This indicates that local sampling based on the instance center position can effectively extract features from the feature map, reducing the introduction of irrelevant region features, thereby achieving a similar effect to the masked attention used in FastInst. The introduction of multiscale features further enhanced the model's performance.

**Table 3.** Query decoder. Our center-guided local sampling decoder performed significantly better than the global sampling decoder in FastInst.

| Query Decoder | $AP_{val}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| Global sampling | 36.3 | 57.5 | 38.1 | 14.1 | 38.9 | 58.2 |
| Local sampling | **37.8** | **59.5** | **39.5** | **16.9** | **40.9** | **59.5** |

"Global sampling" represents decoding queries achieved through global sampling, while "Local sampling" refers to our proposed method of query decoding achieved through center-guided local sampling. Bold data indicates that this method outperforms others in this metric. The same applies to bold data in subsequent tables.

Lightweight inference structure: Table 4 demonstrates the performance of CenterInst and FastInst, each with the same lightweight design. FastInst experienced a performance decrease of 1.3 AP when employing the lightweight structure, whereas CenterInst showed a modest decrease of only 0.3 AP. This further emphasizes that CenterInst effectively leverages information from multiscale feature maps using the center-guided local sampling query decoder module, without relying too heavily on more refined feature maps. As a result, CenterInst efficiently reduces model computation and inference time by minimizing pixel update frequency, thereby accelerating inference speed while maintaining accuracy.

**Table 4.** Lightweight inference structure. Comparison of the performance of FastInst and CenterInst when employing lightweight inference structures.

| Method | LIS | $AP_{val}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| FastInst | ✓ | 36.1 (−1.3) | 57.0 | 37.9 | 14.3 | 38.7 | 58.4 |
| FastInst | | 37.4 | 58.3 | 39.2 | 15.2 | 40.0 | 59.2 |
| CenterInst | ✓ | 37.8 (−0.3) | 59.5 | 39.5 | 16.9 | 40.9 | 59.5 |
| CenterInst-large | | 38.1 | 59.8 | 39.9 | 16.3 | 41.2 | 60.5 |

A checkmark (✓) in the "LIS" column indicates adoption of a lightweight inference structure and use of our proposed training method. Entries lacking a checkmark used the same structure without lightweight processing for both inference and training.

Query proposal: As shown in Table 5, for the center-guided query selector, we first replaced the high-resolution feature maps $E3$ with lower-resolution feature maps $E4$ for query acquisition, leading to a decrease in the model's predictive capabilities for small objects. Subsequently, we stopped obtaining queries using bilinear interpolation on the coordinates of the center point, resulting in an overall accuracy drop. This indicates that

high-resolution feature maps and precise coordinates effectively enhance the quality of query acquisition, thereby improving the model's accuracy.

**Table 5.** Query proposal. We ceased the use of high-resolution feature map sampling in the query selector and the use of center point coordinates for query sampling. After ceasing the use of center point coordinates for query sampling, we employed query sampling based on the pixel position of the center point in the feature map.

| | $AP_{val}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| CenterInst | **37.8** | **59.5** | **39.5** | **16.9** | **40.9** | **59.5** |
| - high resolution | 37.6 | 59.3 | 39.4 | 16.1 | **40.9** | 59.4 |
| - bilinear interpolation | 37.2 | 58.8 | 38.9 | 16.0 | 40.7 | 59.1 |

Note: "- high resolution" denotes not extracting queries from high-resolution pixel features; "- bilinear interpolation" indicates not using the bilinear interpolation method to extract queries. Bold data indicates that this method outperforms others in this metric. The same applies to bold data in subsequent tables.

Sampling point number: As shown in Table 6, we evaluated the impact of different numbers of sampling points and sampling layers on query decoder performance. It can be observed that, compared to single-scale sampling, multiscale sampling produced significant improvements, especially for small and large objects. In the case of multiscale sampling, selecting 12 sampling points per layer struck a balance between high accuracy and faster inference speed.

**Table 6.** Sampling point number. Employing multiscale sampling and an appropriate number of sampling points can lead to better model performance.

| *L* | *n* | *N* | $AP_{val}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 36 | 36 | 37.4 | 58.9 | 39.1 | 16.0 | **41.0** | 58.9 |
| 3 | 8 | 24 | 37.6 | 59.4 | 39.3 | 16.4 | 40.9 | 59.4 |
| 3 | 12 | 36 | **37.8** | **59.5** | **39.5** | **16.9** | 40.9 | 59.5 |
| 3 | 16 | 48 | 37.6 | 59.3 | 39.4 | 16.5 | 40.8 | **59.6** |

"*L*" represents the number of scale feature maps to be sampled, "*n*" represents the number of sampling points per scale feature map, and "*N*" represents the total number of sampling points. Bold data indicates that this method outperforms others in this metric. The same applies to bold data in subsequent tables.

Shared-parameter pixel feature decoder: As shown in Table 7, during the training process of the pixel feature decoder, we employed both parameter-sharing and non-parameter-sharing configurations. The experimental results indicated that using a pixel feature decoder with a non-parameter-sharing configuration yielded better performance during training.

**Table 7.** Shared-parameter pixel feature decoder. Experimenting with pixel feature decoders with and without parameter sharing.

| Shared Parameters | $AP_{val}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| ✓ | 37.5 | 59.3 | 39.2 | 16.5 | 40.8 | 59.2 |
| - | **37.8** | **59.5** | **39.5** | **16.9** | **40.9** | 59.5 |

For shared parameters, "✓" indicates the use of shared parameters, while "-" indicates the absence of shared parameters. Bold data indicates that this method outperforms others in this metric. The same applies to bold data in subsequent tables.

Contrastive denoising training strategy: We enhanced the decoding capability of the decoder during the training of CenterInst using a denoising training strategy. As shown in Table 8, both the CenterInst and CenterInst-large models exhibit improved performance, with a significant enhancement in the decoder's decoding capability.

**Table 8.** Contrastive denoising training strategy. We applied the contrastive denoising training strategy to both CenterInst and CenterInst-large and observed an improvement in performance for both models.

| Method | CDT | $AP_{val}$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| CenterInst | - | 37.8 | 59.5 | 39.5 | **16.9** | 40.9 | 59.5 |
| CenterInst | ✓ | **38.4** | **60.0** | **40.4** | 16.3 | **42.0** | **60.8** |
| CenterInst-large | - | 38.1 | 59.8 | 39.9 | 16.3 | 41.2 | **60.5** |
| CenterInst-large | ✓ | **38.6** | **60.2** | **40.8** | **17.4** | **42.2** | **60.5** |

CDT represents the contrastive denoising training strategy. "✓" indicates the adoption of this strategy, while "-" indicates its absence. Bold data indicates that this method outperforms others in this metric. The same applies to bold data in subsequent tables.

### *4.4. Visualization Results*

To further demonstrate the prediction performance of CenterInst, we present some prediction results produced using CenterInst on the COCO validation set along with their ground truth, as shown in Figure 2. The proposed CenterInst model can produce accurate segmentation masks with detailed boundaries. In dense scenes, CenterInst is also capable of effectively distinguishing between different instances.

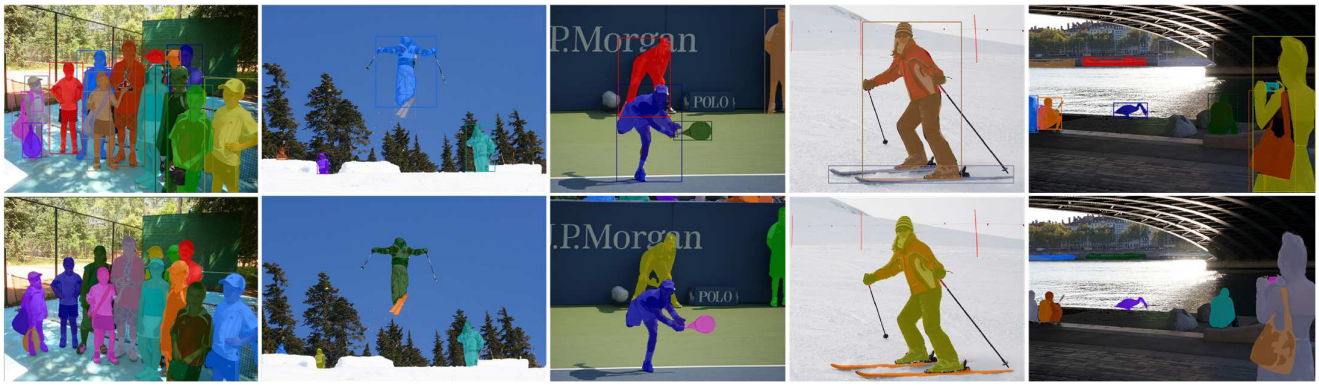

**Figure 2.** Visualization of some predictions on the COCO dataset. We used CenterInst with a ResNet50 backbone that achieved 38.4 AP on the validation set with a speed of 54.1 FPS on a single RTX3090Ti GPU. The first row shows the ground truth, while the second row shows the predictions. We set the confidence threshold to 0.5.

## 5. Conclusions

To address the challenge of real-time instance segmentation, this paper introduced a network, CenterInst, that achieves higher accuracy and faster speed than current state-of-the-art methods. Firstly, we introduced a center-guided query selection module, which utilizes the predicted probabilities of center points to enhance the quality of the selected queries. This lightweight module significantly improves query quality without substantially increasing the inference time. Secondly, we improved the query decoding branch by proposing a center-guided query decoding batch. This branch utilizes the coordinates of center points to achieve multiscale local sampling, effectively avoiding redundant sampling features. We employed the adaptive mixing method to decode queries using sampled features, which, compared to the multihead attention mechanism in Transformers, avoids the use of global attention, reducing the introduction of irrelevant features. This makes the query decoder more efficient and accurate. Simultaneously, we simplified the pixel feature decoding branch by proposing a lightweight inference structure. During training, we used a pixel feature decoding branch with the same number of layers as the query decoder, but, during inference, only the last layer of the pixel feature decoding branch is utilized. This structure significantly reduces the model's inference time, with only a slight impact on model accuracy. Finally, we introduced the contrastive denoising training

method, which further enhanced the decoding capabilities of the decoder and improved the model accuracy.

Our experimental results demonstrated that CenterInst achieves a significant improvement in performance and speed on the COCO test-dev dataset, with an AP value of 39.2 AP and an inference speed of 54.1 FPS on the ResNet50 backbone. Compared to FastInst, CenterInst achieves a 0.6 AP improvement in accuracy and a 3.8 FPS improvement in inference speed. This finding indicates that introducing center points effectively assists instance segmentation networks in sampling and query selection, thereby enhancing model accuracy and inference speed. This provides a new approach to addressing the challenge of real-time instance segmentation.

**Author Contributions:** Conceptualization, S.T. and L.R.; methodology, S.T. and L.R.; software, L.R.; validation, S.T.; formal analysis, S.T. and L.R.; investigation, S.T. and L.R.; resources, L.R.; data curation, L.R.; writing—original draft preparation, S.T. and L.R.; writing—review and editing, S.T.; visualization, L.R.; supervision, S.T.; project administration, S.T.; funding acquisition, S.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the National Key Research and Development Program of China (2020AAA0109700), the National Science Fund for Distinguished Young Scholars (62125601), and the National Natural Science Foundation of China (62076024).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The dataset used in this study was downloaded from: https://cocodataset.org/#download, accessed on 1 September 2023.

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

## References

1. Wang, X.; Kong, T.; Shen, C.; Jiang, Y.; Li, L. Solo: Segmenting Objects by Locations. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 649–665.
2. Wang, X.; Zhang, R.; Kong, T.; Li, L.; Shen, C. Solov2: Dynamic and Fast Instance Segmentation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17721–17732.
3. Cheng, T.; Wang, X.; Chen, S.; Zhang, W.; Zhang, Q.; Huang, C.; Zhang, Z.; Liu, W. Sparse Instance Activation for Real-Time Instance Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 4433–4442.
4. He, J.; Li, P.; Geng, Y.; Xie, X. FastInst: A Simple Query-Based Model for Real-Time Instance Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 23663–23672.
5. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as Points. *arXiv* **2019**, arXiv:1904.07850.
6. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft Coco: Common Objects in Context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
7. Li, Y.; Zhao, H.; Qi, X.; Wang, L.; Li, Z.; Sun, J.; Jia, J. Fully Convolutional Networks for Panoptic Segmentation. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; IEEE: Nashville, TN, USA, 2021; pp. 214–223.
8. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R.B. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
9. Ren, S.; He, K.; Girshick, R.B.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*, 1137–1149. [CrossRef] [PubMed]
10. Cai, Z.; Vasconcelos, N. Cascade R-CNN: High Quality Object Detection and Instance Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 1483–1498. [CrossRef] [PubMed]
11. Chen, K.; Ouyang, W.; Loy, C.C.; Lin, D.; Pang, J.; Wang, J.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; et al. Hybrid Task Cascade for Instance Segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; IEEE: Long Beach, CA, USA, 2019; pp. 4969–4978.
12. Tian, Z.; Shen, C.; Chen, H.; He, T. Fcos: Fully Convolutional One-Stage Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9627–9636.
13. Tian, Z.; Shen, C.; Chen, H. Conditional Convolutions for Instance Segmentation. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 282–298.

14. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. Yolact: Real-Time Instance Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9157–9166.
15. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT++ Better Real-Time Instance Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *44*, 1108–1121. [CrossRef] [PubMed]
16. Zhang, W.; Pang, J.; Chen, K.; Loy, C.C. K-Net: Towards Unified Image Segmentation. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 10326–10338.
17. Fang, Y.; Yang, S.; Wang, X.; Li, Y.; Fang, C.; Shan, Y.; Feng, B.; Liu, W. Instances as Queries. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 6890–6899.
18. Sun, P.; Zhang, R.; Jiang, Y.; Kong, T.; Xu, C.; Zhan, W.; Tomizuka, M.; Li, L.; Yuan, Z.; Wang, C.; et al. Sparse R-CNN: End-to-End Object Detection with Learnable Proposals. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; IEEE: Nashville, TN, USA, 2021; pp. 14449–14458.
19. Dong, B.; Zeng, F.; Wang, T.; Zhang, X.; Wei, Y. Solq: Segmenting Objects by Learning Queries. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 21898–21909.
20. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 213–229.
21. Li, F.; Zhang, H.; Xu, H.; Liu, S.; Zhang, L.; Ni, L.M.; Shum, H.-Y. Mask Dino: Towards a Unified Transformer-Based Framework for Object Detection and Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 3041–3050.
22. Zhang, H.; Li, F.; Xu, H.-S.; Huang, S.; Liu, S.; Ni, L.M.; Zhang, L. MP-Former: Mask-Piloted Transformer for Image Segmentation. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 18074–18083.
23. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020; pp. 1–16.
24. Cheng, B.; Misra, I.; Schwing, A.G.; Kirillov, A.; Girdhar, R. Masked-Attention Mask Transformer for Universal Image Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 1290–1299.
25. Cheng, B.; Schwing, A.; Kirillov, A. Per-Pixel Classification Is Not All You Need for Semantic Segmentation. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 17864–17875.
26. Gao, Z.; Wang, L.; Han, B.; Guo, S. Adamixer: A Fast-Converging Query-Based Object Detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5364–5373.
27. Zhang, H.; Li, F.; Liu, S.; Zhang, L.; Su, H.; Zhu, J.-J.; Ni, L.M.; Shum, H. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. In Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023; pp. 1–23.
28. Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; Girshick, R. Detectron2. 2019. Available online: https://github.com/facebookresearch/detectron2 (accessed on 1 September 2023).
29. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017; pp. 1–19.
30. Al Sameera, B.N.; Gaidhane, V.H.; Rajevenceltha, J. Image Focus Measure Based on Polynomial Coefficients and Reduced Gerschgorin Circle Approach. In *IETE Technical Review*; Taylor & Francis: Abingdon, UK, 2023; pp. 1–12.
31. Rajevenceltha, J.; Gaidhane, V.H.; Anjana, V. A Novel Approach for Drowsiness Detection Using Local Binary Patterns and Histogram of Gradients. In Proceedings of the 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 19 November 2019; pp. 1–6.
32. Yelampalli, P.K.R.; Nayak, J.; Gaidhane, V.H. A Novel Binary Feature Descriptor to Discriminate Normal and Abnormal Chest CT Images Using Dissimilarity Measures. *Pattern Anal. Appl.* **2019**, *22*, 1517–1526. [CrossRef]
33. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
34. He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; Li, M. Bag of Tricks for Image Classification with Convolutional Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 558–567.
35. Zhu, X.; Hu, H.; Lin, S.; Dai, J. Deformable Convnets v2: More Deformable, Better Results. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9308–9316.