agriculture

Article

Real-Time Cattle Pose Estimation Based on Improved RTMPose

Xiaowu Li [1], Kun Sun [1], Hongbo Fan [2,*] and Zihan He [1]

[1] Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China; lxwlxw66@kust.edu.cn (X.L.); 20212204139@stu.kust.edu.cn (K.S.); 20212204100@stu.kust.edu.cn (Z.H.)
[2] Faculty of Modern Agricultural Engineering, Kunming University of Science and Technology, Kunming 650300, China
* Correspondence: 20110258@kust.edu.cn

Abstract: Accurate cattle pose estimation is essential for Precision Livestock Farming (PLF). Computer vision-based, non-contact cattle pose estimation technology can be applied for behaviour recognition and lameness detection. Existing methods still face challenges in achieving fast cattle pose estimation in complex scenarios. In this work, we introduce the FasterNest Block and Depth Block to enhance the performance of cattle pose estimation based on the RTMPose model. First, the accuracy of cattle pose estimation relies on the capture of high-level image features. The FasterNest Block, with its three-branch structure, effectively utilizes high-level feature map information, significantly improving accuracy without a significant decrease in inference speed. Second, large kernel convolutions can increase the computation cost of the model. Therefore, the Depth Block adopts a method based on depthwise separable convolutions to replace large kernel convolutions. This addresses the insensitivity to semantic information while reducing the model's parameter. Additionally, the SimAM module enhances the model's spatial learning capabilities without introducing extra parameters. We conducted tests on various datasets, including our collected complex scene dataset (cattle dataset) and the AP-10K public dataset. The results demonstrate that our model achieves the best average accuracy with the lowest model parameters and computational requirements, achieving 82.9% on the cattle test set and 72.0% on the AP-10K test set. Furthermore, in conjunction with the object detection model RTMDet-m, our model reaches a remarkable inference speed of 39FPS on an NVIDIA GTX 2080Ti GPU using the PyTorch framework, making it the fastest among all models. This work provides adequate technical support for fast and accurate cattle pose estimation in complex farm environments.

Keywords: cattle pose estimation; RTMPose; FasterNest Block; SimAM attention; Depth Block

check for updates

Citation: Li, X.; Sun, K.; Fan, H.; He, Z. Real-Time Cattle Pose Estimation Based on Improved RTMPose. *Agriculture* **2023**, *13*, 1938. https://doi.org/10.3390/agriculture13101938

Academic Editor: Hyeon Tae Kim

Received: 8 August 2023
Revised: 28 September 2023
Accepted: 1 October 2023
Published: 4 October 2023

1. Introduction

Intelligent agriculture technologies based on computer vision offer multiple advantages, such as automation, efficiency, and non-contact interaction, significantly enhancing animal welfare in livestock farming [1]. The deep learning methods are widely applied in the animal farming industry [2]. Pose estimation refers to accurately locating the keypoints of a target instance and connecting them according to a predefined logic to reconstruct a complete skeletal structure. In animal husbandry, the health of animals directly impacts production efficiency [3]. Accurate animal pose estimation is a crucial component of behavior recognition and lameness detection, such as poultry behavior recognition [4] and lameness detection in cattle [5]. Lameness typically occurs in older cows and adversely affects milk production [6]. Cattle pose estimation is essential in Precision Livestock Farming (PLF). Therefore, using computer vision technology to solve cattle pose estimations in complex farm environments is highly meaningful.

In recent years, the application of computer vision technology in animal pose estimation has garnered widespread attention among researchers. In 2019, Li et al. [7] applied the convolutional pose machine model, the stacked hourglass model, and the convolution heatmap regression model to estimate the poses of dairy and beef cattle. They conducted training and testing using 2134 cattle images and achieved an average PCKH value of 90.39%. In 2022, Liu et al. [8] introduced a high-resolution transformer model. They employed the efficient transformer architecture and designed a basic block with representative batch normalization. Additionally, they incorporated four PoolFormer blocks after the Multi-Branch Neural Network to balance the performance. Testing on the mammalian public dataset, AP-10K demonstrated superior performance compared to lightweight networks, such as Lite-HRNet and HRformer-Tiny. In the same year, Gong et al. [9] proposed a two-branch skeleton extraction network for multi-cattle pose estimation, conducting tests on single-target and dual-target images under varying lighting conditions, with single-cattle pose estimation achieving an accuracy of 85% during the day and 78.1% at night while dual-cattle pose estimation accuracy reached 74.3% and 71.6%. In 2023, Fan et al. [10] designed CoordConv and DO-Conv, which were applied to the bottleneck and basic block of the HRNet network to improve cattle pose estimation performance. The results showed that AP is 93.2% on 2432 complex environment images, parameters, and FLOPs, which were remarkably decreased. In conclusion, computer vision technology has found extensive applications in animal pose estimation. However, accurately and rapidly achieving cattle pose estimations in complex farm environments is challenging.

Pose estimation algorithms can be divided into two categories: top-down (e.g., [11]) and bottom-up (e.g., [12]). In the top-down method, the object detection algorithm is first used to detect object instances and generate bounding boxes. Next, the developed bounding box regions are resized to the same shape. In the end, keypoint extraction networks are used to create keypoints. Top-down methods are less sensitive to object instance size and offer higher accuracy. On the other hand, bottom-up approaches first detect keypoints for all objects and then group these keypoints into individual instances. Bottom-up methods have faster detection speeds. In the past, traditional top-down approaches were considered more accurate but slower, primarily because an additional detection process was introduced. However, with the improvement in the performance of object detection algorithms, the object detection component is no longer the inference speed bottleneck for the top-down paradigm [13]. Therefore, we adopted the top-down paradigm in this study because it can provide better pose estimation accuracy.

Recent technological advancements have significantly improved the performance of computer vision-based methods in pose estimation. In 2019, Sun et al. [11] introduced the HRNet network, which has found extensive applications, such as human pose estimation [14] and semantic segmentation [15]. HRNet maintains high-resolution representations throughout its architecture while incorporating low-resolution representations and multiple branches for information fusion, effectively capturing visual information within the target. In 2021, Yuan et al. [16] proposed HRFormer, which combines the Vision Transformer (ViT) and HRNet. Using HRNet's multi-resolution characteristics, basic modules composed of local window self-attention and FFNs significantly improve the performance of human pose estimation on the COCO dataset [17]. In 2022, Xu et al. [18] developed the VitPose model based on the Vision Transformer, which has a simple, scalable, and flexible structure and achieved state-of-the-art performance on the COCO dataset. While the above-mentioned methods have achieved excellent accuracy in human pose estimation, applying them to real-time environments for cattle pose estimation is challenging. Jiang et al. [13], addressing issues in the industry, explored the critical factors in pose estimation and proposed a high-performance real-time multi-person pose estimation framework: RTMPose. Previous pose estimation methods typically treated keypoint localization as a coordinate regression (e.g., [19,20]) or heatmap regression (e.g., [11,21]). Li et al. [22] addressed the computational complexity, complex post-processing, and quantization errors associated with 2D heatmap methods. They introduced a simple coordinate classification method for human pose estimation; the method treats keypoint localization as a subpixel classification of horizontal and vertical coordinates, eliminating the need for high-resolution intermediate representations and costly upsampling. Furthermore, sub-pixel scale coordinate classification effectively mitigates quantization errors, enabling efficient keypoint localization [13]. The high accuracy and speed of RTMPose result from its efficient framework. SimCC* [13] removed the costly upsampling layers from SimCC [22], reducing the complexity while maintaining good accuracy. Its backbone network adopts CSPNext-m [23], making the structure more efficient. Specifically, it requires simple consecutive downsampling, followed by sub-pixel classification using k (number of keypoints) keypoint representations for keypoint localization with SimCC*. However, RTMPose experiences performance degradation when dealing with real-world complex environments, such as multiple cattle, varying lighting conditions, and obstacles, such as rails and trough occlusions.

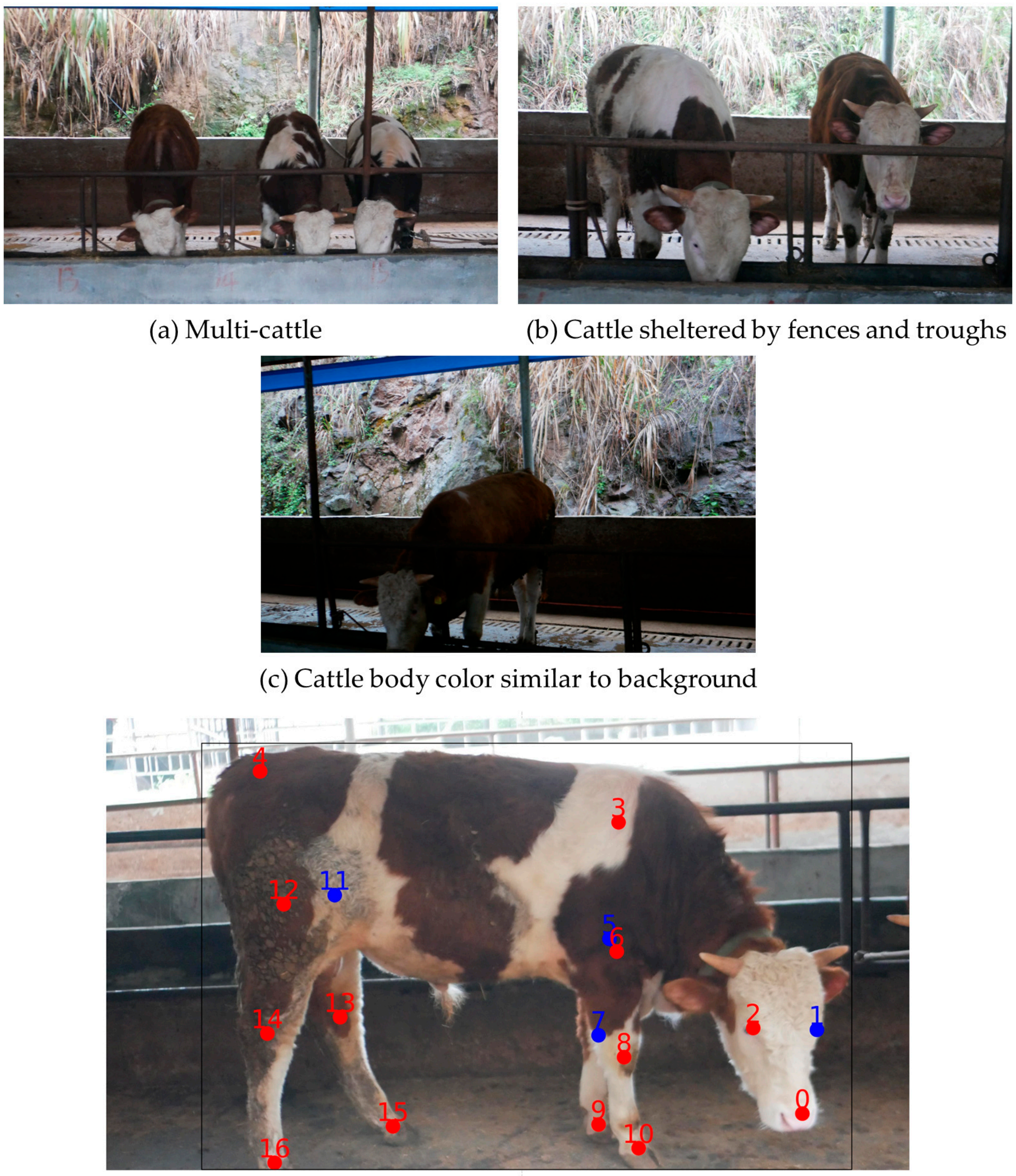
In this paper, we have built upon the efficient structure of RTMPose and optimized its overall architecture. First, we have incorporated the FasterNest Block into the Backbone, which contains three branches. This enhancement strengthens RTMPose's utilization of high-level image features, thereby improving the model's feature learning capabilities. Second, within the CSPLayer [13], channel attention has been replaced by the SimAM attention module. This substitution addresses RTMPose's lack of spatial learning capabilities. The SimAM attention module effectively leverages spatial and channel information within the feature maps, inferring three-dimensional attention weights and enhancing the model's spatial learning capabilities. Lastly, we have replaced the 7×7 large kernel convolutions in RTMPose with the Depth Block, addressing the model's insensitivity to semantic information while enhancing network accuracy and decreasing parameters. We have tested our method on the cattle and publicly available AP-10K datasets, and it exhibits a lower parameter and computation cost than existing state-of-the-art models and maintains higher model accuracy. In this study, our contributions are as follows:

- Efficient Keypoint Localization: This paper introduces an enhanced RTMPose method. First, it leverages the highest-level image features. Then, it locates keypoints by performing sub-pixel scale classification in the horizontal and vertical directions for only k (number of keypoints). Compared to the RTMPose algorithm, our approach significantly improves detection accuracy while maintaining inference speed.
- FasterNest Block with Three-Branch Structure: RTMPose utilizes the highest-level image features as the basis for keypoint localization. To more effectively use these high-level image features, we introduce the FasterNest Block. This three-branch structure is embedded into the fourth stage of the backbone, enhancing the model's feature learning capabilities.
- Integration of SimAM Attention: RTMPose exhibits limited spatial learning capabilities. The SimAM attention module can infer three-dimensional attention weights for feature maps. In the CSPLayer, channel attention is replaced with SimAM attention, augmenting RTMPose's spatial learning capabilities without additional parameters.
- Lightweight Depth Block: Since 7×7 convolutions add computation parameters and overhead to the RTMPose model, the Depth Block module based on depth separable convolutions, which is more suitable for lightweight models than traditional convolutions, has been designed in this paper. It replaces the conventional 7×7 convolution in the Head, addressing the insensitivity of the model to semantic information while reducing parameters.

2. Materials and Methods

2.1. Data Collection and Data Annotation

The experimental data for this study were collected from the Ximenta'er Cattle Farm, located in Huangjinbao Village, Huaihua City, Hunan Province, China. Raw experimental data were acquired using a monocular camera, capturing Ximenta'er cattle. There were 43 cattle and 3036 images, with a resolution of 3456×2304 pixels, saved in JPEG format. Additionally, 10 video segments, each approximately 15 min long, were selected to evaluate model inference speed. The experimental data were captured over seven days, with images taken during daylight. The brightness of the images also varied with changing lighting conditions. The dataset included images of individual cattle and groups of cattle from multiple angles, with group sizes ranging from 2 to 4 cattle. To enhance the model's robustness, complex scenarios, such as numerous cattle, and obstructions, such as fences and feed troughs, were introduced (e.g., Figure 1a–c). These diverse scenarios may impose limitations on the performance of deep models. A web-based tool called COCO-Annotator was employed for accurate keypoint annotation for manual labelling. The dataset creation process involved bounding box annotation and skeleton keypoint annotation. Bounding box annotation was initially performed to train the object detector. The goal was to minimize the area of the annotated bounding boxes, excluding irrelevant pixels that might negatively impact the experiment. Subsequently, 17 skeletal keypoint annotations were made, which included Nose, Left Eye, Right Eye, Shoulder, Tailbone, Front Left Leg, Front Right Leg, Front Left Knee, Front Right Knee, Front Left Hoof, Front Right Hoof, Back Left Leg, Back Right Leg, Back Left Knee, Back Right Knee, Back Left Hoof, and Back Right Hoof (labelled as points 0 to 16). These keypoint annotations enabled the depiction of the overall posture of the cattle from various angles. Keypoint information is illustrated in Figure 1d, with red dots representing visible keypoints and blue dots indicating keypoints that are not visible. After completing the dataset annotation, a random split was performed with a ratio of 7:2:1, resulting in a training set (2126 images), a validation set (607 images), and a test set (303 images).



(a) Multi-cattle (b) Cattle sheltered by fences and troughs

(c) Cattle body color similar to background

(d) Example of Cattle keypoints Labeling

Figure 1. Cattle Dataset Example.

2.2. Public Dataset AP-10K

In addition to the cattle dataset created in this paper, we further validated the model's generalization and transferability by incorporating the publicly available dataset, AP-10K [24]. The dataset contains 10,028 instances from 54 species of animals belonging to 23 different families, comprising 10,015 images. The number of keypoints and annotation format for each animal species in the AP-10K dataset is the same as that of the COCO dataset. The AP-10K dataset is split into a training set (7010 images), a validation set (1002 images), and test set (2003 images). By combining the AP-10K dataset with the cattle dataset, we can comprehensively evaluate the model's performance on different animal species, thereby validating its generalization and transferability.

2.3. Methods

This section explains the baseline model used in this study and the experimental improvement strategies. Section 2.3.1 introduces the real-time pose estimation network's fundamental components and working principles, RTMPose. Section 2.3.2 presents our final network architecture. Subsequently, Section 2.3.3, Section 2.3.4, and Section 2.3.5 describe the experimental enhancement approaches involving the FasterNest Block, SimAM Attention Mechanism, and Depth Block, respectively.

2.3.1. RTMPose

RTMPose is a simple and efficient real-time human pose estimation framework, as shown in Figure 2. To improve accuracy, previous networks (e.g., SimpleBaseline [25], Hourglass Network [21]) have used costly deconvolution or unsampling to recover low-resolution to high-resolution feature maps. restoring high resolution requires interpolation calculations and pixel post-processing; the cost of unsampling is very high. In contrast, RTMPose continuously downsamples the feature maps and uses k (number of keypoints) keypoint representations as the subpixel scale classification basis for the head. This achieves a streamlined feature extraction and coordinated classification process without unnecessary operations. Specifically, RTMPose-m [13] employs the CSPNext-m [23] framework as the image feature extraction component in the backbone. CSPNext-m utilizes 5×5 Depthwise Convolutions, striking a balance between speed and accuracy. SimCC*, on the other hand, removes the time-consuming upsampling layers present in SimCC while simultaneously reducing complexity and maintaining good accuracy [13]. SimCC* treats human pose estimation as two separate classification tasks in the vertical and horizontal coordinate directions. This results in a straightforward and efficient coordinate classification. SimCC* also employs a strategy that divides each pixel into several bins on average, thus achieving smaller quantization errors. Compared to traditional 2D heatmap methods, SimCC* reduces computation costs and improves keypoint localization accuracy. RTMPose-m consists of a backbone used for feature extraction and a classification head. The backbone consists of ConvModule, CSPLayer, and SPPFBottleneck. First, the ConvModule serves as the basic module for network downsampling and channel adjustment, composed of Conv2d, Batch Normalization, and the SiLU activation function. CSPLayer enhances the network's feature extraction capabilities through cascading. SPPFBottleneck pools the feature maps at different scales to effectively capture semantic information in the image, thus improving model performance. The head section comprises a 7×7 Convolution, FC (Fully Connected layer), GAU (Gated Attention Unit), and a coordinate classifier. The 7×7 Convolution layer reduces the 768 channels to k (number of keypoints) feature maps, which generate representations for k keypoints. The FC layer expands the keypoints' representations to the desired dimension of 256, controlled by hyperparameters. The GAU leverages self-attention to further incorporate global and local spatial information. Finally, the classifier performs coordinate classification in horizontal and vertical directions on the representations of k keypoints to achieve keypoint localization.

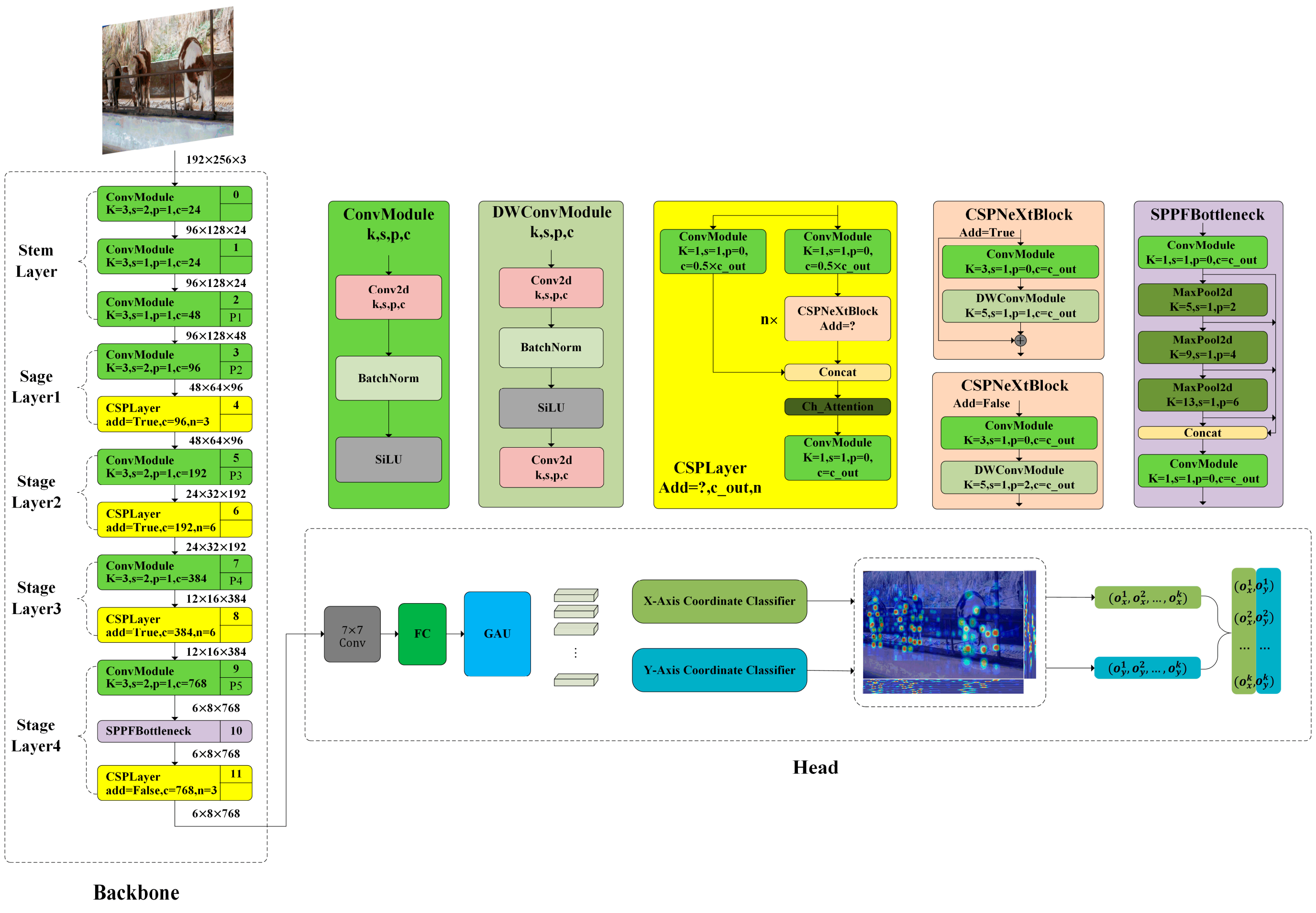

192×256×3
Stem Layer
ConvModule K=3,s=2,p=1,c=24 0
96×128×24
ConvModule K=3,s=1,p=1,c=24 1
96×128×24
ConvModule K=3,s=1,p=1,c=48 2 P1
96×128×48
Sage Layer1
ConvModule K=3,s=2,p=1,c=96 3 P2
48×64×96
CSPLayer add=True,c=96,n=3 4
48×64×96
Stage Layer2
ConvModule K=3,s=2,p=1,c=192 5 P3
24×32×192
CSPLayer add=True,c=192,n=6 6
24×32×192
Stage Layer3
ConvModule K=3,s=2,p=1,c=384 7 P4
12×16×384
CSPLayer add=True,c=384,n=6 8
12×16×384
Stage Layer4
ConvModule K=3,s=2,p=1,c=768 9 P5
6×8×768
SPPFBottleneck 10
6×8×768
CSPLayer add=False,c=768,n=3 11
6×8×768
Backbone
ConvModule k,s,p,c
Conv2d k,s,p,c
BatchNorm
SiLU
DWConvModule k,s,p,c
Conv2d k,s,p,c
BatchNorm
SiLU
Conv2d k,s,p,c
ConvModule K=1,s=1,p=0, c=0.5×c_out
ConvModule K=1,s=1,p=0, c=0.5×c_out
n×
CSPNeXtBlock Add=?
Concat
Ch_Attention
ConvModule K=1,s=1,p=0, c=c_out
CSPLayer Add=?,c_out,n
CSPNeXtBlock
Add=True
ConvModule K=3,s=1,p=0,c=c_out
DWConvModule K=5,s=1,p=1,c=c_out
CSPNeXtBlock
Add=False
ConvModule K=3,s=1,p=0,c=c_out
DWConvModule K=5,s=1,p=2,c=c_out
SPPFBottleneck
ConvModule K=1,s=1,p=0,c=c_out
MaxPool2d K=5,s=1,p=2
MaxPool2d K=9,s=1,p=4
MaxPool2d K=13,s=1,p=6
Concat
ConvModule K=1,s=1,p=0,c=c_out
7×7 Conv
FC
GAU
X-Axis Coordinate Classifier
Y-Axis Coordinate Classifier
$(o_x^1, o_x^2, ..., o_x^k)$
$(o_y^1, o_y^2, ..., o_y^k)$
(o_x^1,o_y^1)
(o_x^2,o_y^2)
...
(o_x^k,o_y^k)
Head

Figure 2. The illustration of the RTMPose-m.

2.3.2. RTMPose-FSD

The RTMPose-FSD structure is illustrated in Figure 3. First, this study introduced the FasterNest Block into the backbone. The FasterNest Block's three-branch design increased the RTMPose model's utilization of high-level image features, enhancing the model's ability to learn features. Next, in the CSPLayer [13], channel attention was replaced by the SimAM attention module to address the issue of insufficient spatial learning capability in RTMPose. The SimAM attention module leveraged feature maps' spatial and channel information to infer three-dimensional attention weights, enhancing spatial learning capability. Finally, the 7 × 7 large kernel convolution in RTMPose was replaced with the Depth Block to address the issue of RTMPose's insensitivity to semantic information. This achieved a lightweight quality while improving network accuracy.

2.3.3. FasterNest Block

Chen et al. [26] introduced a novel convolution technique called Partial Convolution (PConv). PConv reduces FLOPs while maintaining a high FLOPS, simultaneously reducing computational redundancy and memory access, all without sacrificing the convolution's feature extraction capability. As shown in Figure 4, PConv applies regular convolution only to a subset of consecutive channels to extract spatial features while the remaining channels remain unchanged. The FLOPs calculation for PConv is as follows:

$$\mathrm{FLOPs} = \mathrm{h} \times \omega \times \mathrm{k}^2 \times \mathrm{c_p}^2, \tag{1}$$

where h, ω, and c_p represent the input feature's height, width, and kernel, respectively; c_p denotes the continuous c_p channels that are taken as representatives for the entire feature map during computation.

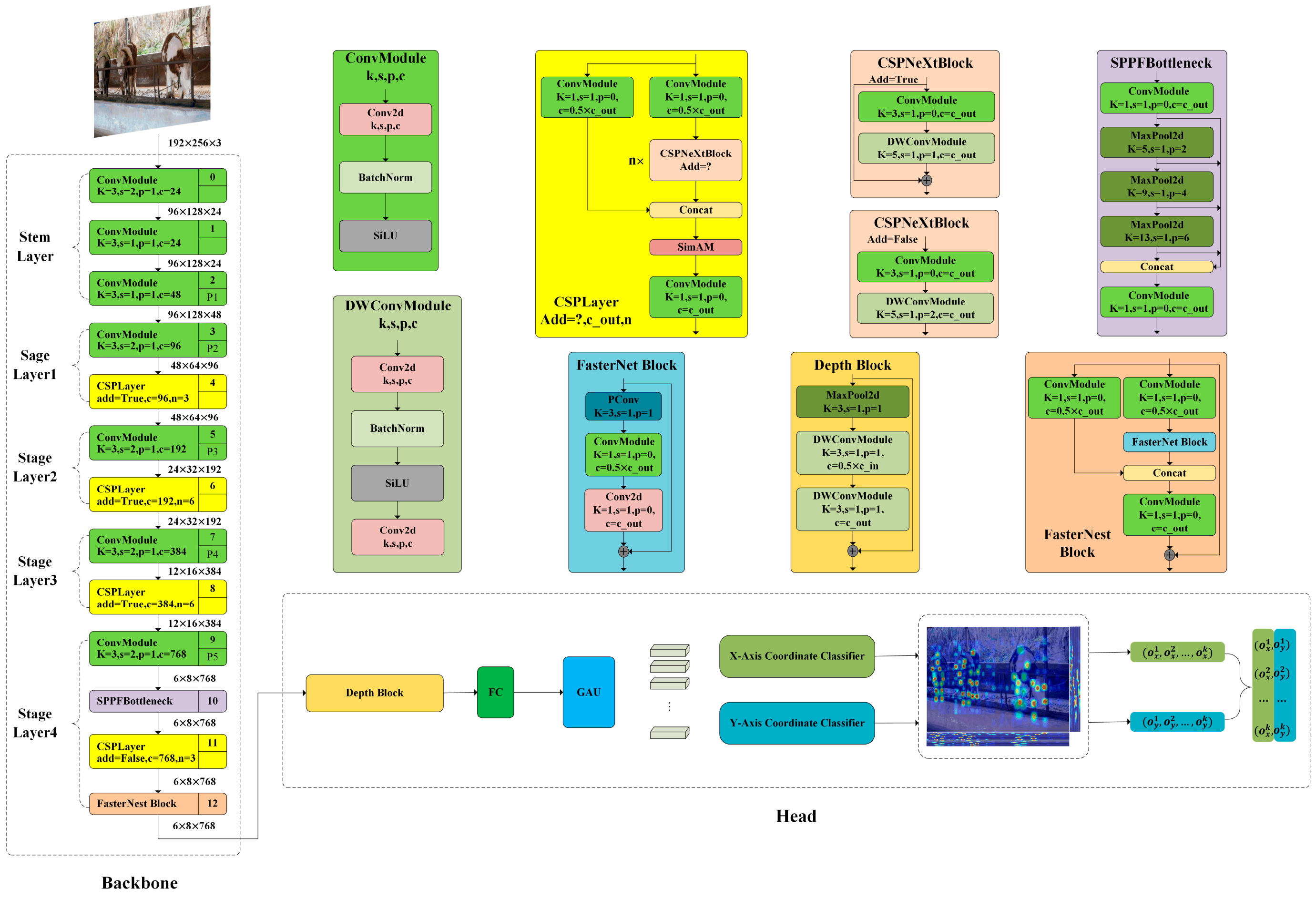

192×256×3
Stem Layer
ConvModule K=3,s=2,p=1,c=24 0
96×128×24
ConvModule K=3,s=1,p=1,c=24 1
96×128×24
ConvModule K=3,s=1,p=1,c=48 2 P1
96×128×48
Sage Layer1
ConvModule K=3,s=2,p=1,c=96 3 P2
48×64×96
CSPLayer add=True,c=96,n=3 4
48×64×96
Stage Layer2
ConvModule K=3,s=2,p=1,c=192 5 P3
24×32×192
CSPLayer add=True,c=192,n=6 6
24×32×192
Stage Layer3
ConvModule K=3,s=2,p=1,c=384 7 P4
12×16×384
CSPLayer add=True,c=384,n=6 8
12×16×384
Stage Layer4
ConvModule K=3,s=2,p=1,c=768 9 P5
6×8×768
SPPFBottleneck 10
6×8×768
CSPLayer add=False,c=768,n=3 11
6×8×768
FasterNest Block 12
6×8×768
Backbone
ConvModule k,s,p,c
Conv2d k,s,p,c
BatchNorm
SiLU
DWConvModule k,s,p,c
Conv2d k,s,p,c
BatchNorm
SiLU
Conv2d k,s,p,c
ConvModule K=1,s=1,p=0, c=0.5×c_out
ConvModule K=1,s=1,p=0, c=0.5×c_out
n×
CSPNeXtBlock Add=?
Concat
SimAM
ConvModule K=1,s=1,p=0, c=c_out
CSPLayer Add=?,c_out,n
CSPNeXtBlock
Add=True
ConvModule K=3,s=1,p=0,c=c_out
DWConvModule K=5,s=1,p=1,c=c_out
CSPNeXtBlock
Add=False
ConvModule K=3,s=1,p=0,c=c_out
DWConvModule K=5,s=1,p=2,c=c_out
SPPFBottleneck
ConvModule K=1,s=1,p=0,c=c_out
MaxPool2d K=5,s=1,p=2
MaxPool2d K=9,s=1,p=4
MaxPool2d K=13,s=1,p=6
Concat
ConvModule K=1,s=1,p=0,c=c_out
FasterNet Block
PConv K=3,s=1,p=1
ConvModule K=1,s=1,p=0, c=0.5×c_out
Conv2d K=1,s=1,p=0, c=c_out
Depth Block
MaxPool2d K=3,s=1,p=1
DWConvModule K=3,s=1,p=1, c=0.5×c_in
DWConvModule K=3,s=1,p=1, c=c_out
ConvModule K=1,s=1,p=0, c=0.5×c_out
ConvModule K=1,s=1,p=0, c=0.5×c_out
FasterNet Block
Concat
ConvModule K=1,s=1,p=0, c=c_out
FasterNest Block
Depth Block
FC
GAU
X-Axis Coordinate Classifier
Y-Axis Coordinate Classifier
$(o_x^1, o_x^2, \ldots, o_x^k)$
$(o_y^1, o_y^2, \ldots, o_y^k)$
(o_x^1, o_y^1)
(o_x^2, o_y^2)
(o_x^k, o_y^k)
Head

Figure 3. The illustration of the RTMPose-FSD.

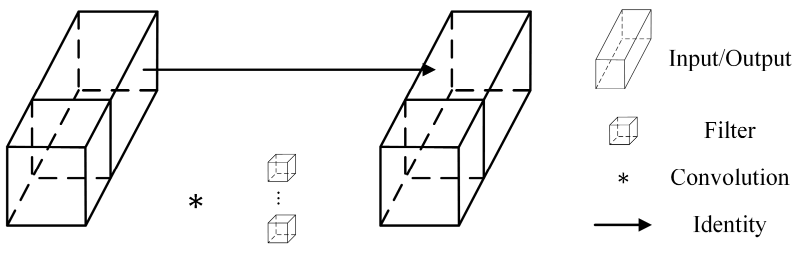


Figure 4. The illustration of the Partial Convolution.

For example, when $c_p = \frac{1}{2}c$, PConv only occupies 1/4 of the FLOPs of regular convolution, enabling better utilization of computational capabilities on the device. The design of the FasterNet Block is depicted in Figure 5. To fully and efficiently utilize information from all channels, PWConv (Pointwise Convolution) was added after PConv, where the structure indicated that the T-shaped receptive field could focus attention on the center position of the input feature map, as the center position often contained more critical feature information than adjacent positions [26]. By combining PConv with PWConv, not only was all channel information utilized, but also fewer computational resources were consumed.

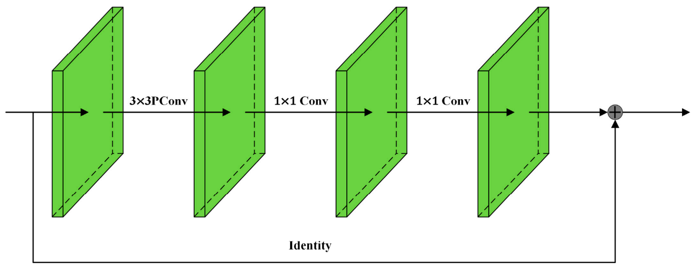


Figure 5. The illustration of the FasterNet Block.

The fourth stage of RTMPose's backbone contains the highest-level feature information, considered crucial for keypoint localization. Therefore, it is essential to use this high-level feature information more effectively. As shown in Figure 6, in this paper, we designed the FasterNest Block based on the FasterNet Block, specifically for the fourth stage of RTMPose, to enhance the model's ability to learn from features. Through the ablative study presented, we demonstrated that our model provided superior performance without significantly reducing inference speed.

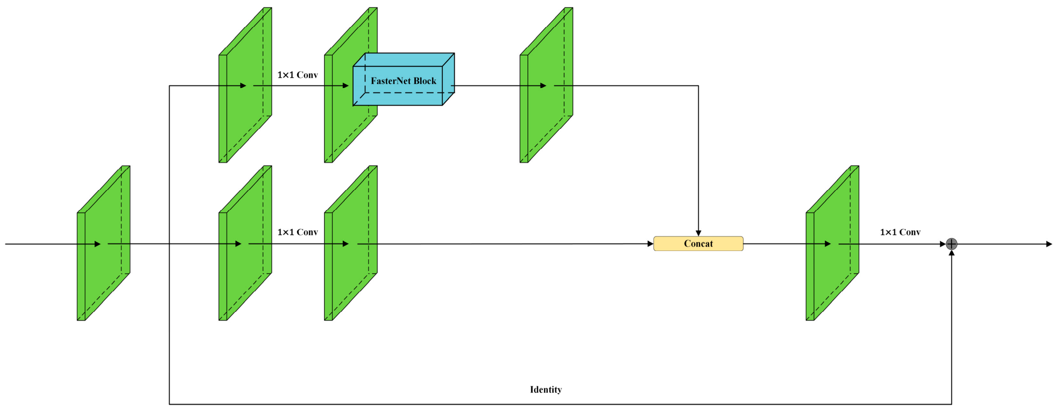


Figure 6. The illustration of the FasterNest Block architecture.

2.3.4. SimAM Attention Mechanism

In deep learning, the attention mechanism was initially proposed by Bahdanau et al. [27] in their machine translation research in natural language processing, laying the foundation for subsequent developments in attention mechanisms [28,29]. Inspired by neuroscience theories [30], Yang et al. [31] introduced an optimized energy function to discover the importance of each neuron. They proposed a simple and effective attention module called SimAM, which simultaneously considers spatial and channel attention. Unlike CBAM [28], it can refine features in both channel and spatial dimensions and apply the inferred three-dimensional attention weights to all feature maps without increasing the model's parameters. The definition of the energy function is as follows:

$$e_t(\omega_t, b_t, y, x_i) = \frac{1}{M-1}\sum_{i=1}^{M-1}(-1-(\omega_t x_i + b_t))^2 + (-1-(\omega_t x_i + b_t))^2 + \lambda\omega_t^2, \quad (2)$$

where i is the index along the spatial dimension,$M = H \times W$ represents the number of neurons in that channel, i.e., there are M energy functions in each channel, t, and x_i are the target neuron and other neurons within a single channel of the input feature $X \in R^{C\times H\times W}$, respectively; ω_t and b_t are the weighted sum and bias transformation.

An active neuron may suppress the activity of neighboring neurons, a phenomenon known as spatial inhibition [30]. In visual processing, neurons that exhibit significant spatial inhibition effects should be given higher priority (i.e., importance) [31]. The energy calculation for each neuron e_t^* is shown in Equation (3), where the target neuron is assigned a higher importance compared to surrounding neurons.

$$E_t^* = \frac{4\left(\hat{\sigma}^2+\lambda\right)}{(t-\hat{\mu})^2+2\hat{\sigma}^2+2\lambda}, \tag{3}$$

where $\hat{\mu}$ and $\hat{\sigma}^2$ are the mean and variance computed across all channel neurons, with $\hat{\mu} = \frac{1}{M}\sum_{i=1}^{M} x_I$, $\hat{\sigma}^2 = \frac{1}{M}\sum_{i=1}^{M}(x_i - \hat{\mu})^2$ being the minimum energy for a neuron. The importance of each neuron can be obtained through $1/e_t^*$. Equation (3) shows that the lower the energy e_t^*, the more distinct the target neuron t is from its surrounding neurons, indicating its higher importance for visual processing. The entire process can be represented as:

$$\tilde{X} = \text{sigmoid}\left(\frac{1}{E}\right) \odot X, \tag{4}$$

where E groups all e_t^* values along the channel and spatial dimensions. The Sigmoid function is used to limit excessively large E values, so it does not heavily impact the relative importance of each neuron. The symbol $\odot$ represents element-wise multiplication. To address the weak spatial learning capabilities in RTMPose and enhance the network's ability to learn features, as shown in Figure 7, we replaced channel attention in the CSPLayer module with the SimAM attention module while utilizing the feature map's three-dimensional weight information.

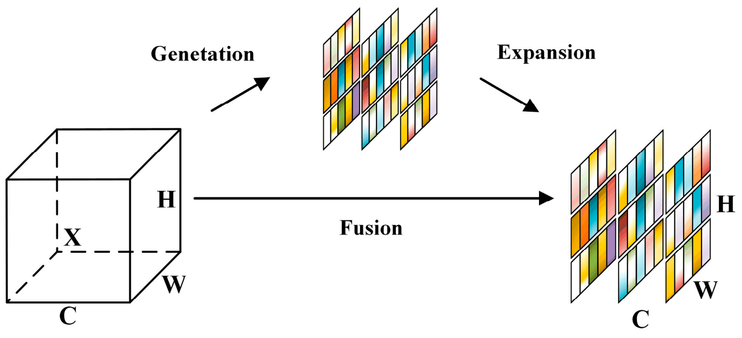


Figure 7. The illustration of the SimAM.

2.3.5. Depth Block

Depthwise separable convolution [32] is a method that decomposes the conventional convolution into Depthwise Convolution and Pointwise Convolution, achieving lightweight feature extraction and processing. Compared to traditional convolution operations, depthwise separable convolution reduces the computational and parameter requirements of the network, thereby reducing model complexity and storage demands. As large-kernel convolutions are less sensitive to semantic information, we designed a Depth Block based on depthwise separable convolution to replace the 7 × 7 conventional large-kernel convolution. The Depth Block is shown in Figure 8, where green represents feature maps, red components are MaxPool, and the time complexity is defined as follows:

$$O(\text{MaxPool}) = K^2 CWH. \tag{5}$$

where C, W, and H represent the input features' channel, width, and height, respectively. K represents the size of the MaxPool convolution kernel.

The Depth Block consists of two sets of 3 × 3 Depthwise Convolution and 1 × 1 Pointwise Convolution combined in the depth dimension, effectively utilizing the network's semantic information. Using feature map channels efficiently during the module instantiation process is essential. First, the first 1 × 1 Pointwise Convolution reduces the number of feature map channels to 384, and then, the second 1 × 1 Pointwise Convolution reduces it to 'k' (number of keypoints). The experiments demonstrate that this module reduced the network's computational and parameter requirements and improved network accuracy.

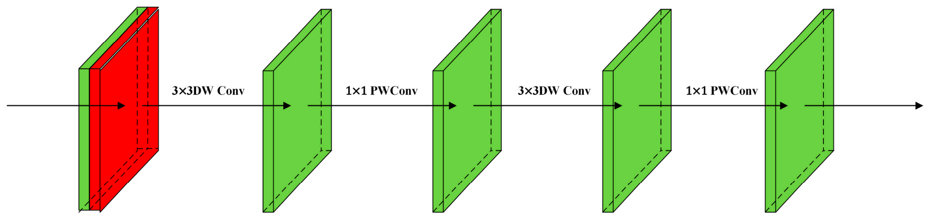


Figure 8. The illustration of the Depth Block.

2.4. Evaluation Indicators and Experimental Environment

2.4.1. Evaluation Indicators

In our work, we used Object Keypoint Similarity (OKS) as the evaluation metric:

$$\mathrm{OKS} = \frac{\sum_i e^{\frac{-d_i^2}{2s^2k_i^2}}\,\delta(v_i > 0)}{\sum_i \delta(v_i > 0)}. \tag{6}$$

where d_i represents the Euclidean distance between the predicted keypoint and its corresponding ground truth; v_i indicates whether the ground truth keypoint is visible; s is the object scale; and k_i is a per keypoint constant that controls falloff. We used the OKS-defined thresholds to calculate AP (Average Precision) and AR (Average Recall). The standard Average Precision (AP) and Recall (AR) scores include AP (the average of AP scores at 10 positions with OKS = 0.50, 0.55, . . ., 0.90, 0.95), AP^{50} (AP at OKS = 0.50), AP^{75}, and AR [17].

2.4.2. Experimental Setup

This study was conducted on an Ubuntu 20.04 operating system using hardware with an Intel Core i7-9700 processor, RTX 2080 Ti graphics card, and 32 GB of RAM. The programming language used was Python 3.8.0, and the deep learning framework was PyTorch 2.0.1. Below are the training details for the object detection and pose estimation algorithms:

Object Detection Algorithm Training: The training of the object detector employed horizontal flipping and Masic data augmentation. Each input image had a size of 640 × 640. The batch size was set to 8, and 200 epochs were conducted. We used the SGD optimizer with an initial learning rate of 4×10^{-4} and a weight decay of 0.001. The learning rate was then reduced to 1×10^{-4} after 130 epochs. The experiments were conducted from scratch.

Pose Estimation Algorithm Training: Data augmentation techniques included horizontal flipping, random retention of either the upper or lower body, scaling in the range of [0.6, 1.4], and rotation in the range of [−80, 80]. Each input image had a size of 192 × 256. We used the AdamW optimizer with a learning rate of 5×10^{-4} and weight decay of 0.05. The batch size was set to 32, and 250 epochs were conducted. A warm-up strategy was applied for the first 500 iterations, gradually increasing the learning rate from 5×10^{-5} to 5×10^{-4} as iterations increased and then reducing it to 2×10^{-4} over 180 epochs. All experiments were conducted from scratch.

3. Results

3.1. Validation Set Results

To validate the effectiveness of our model, we compared it with seven other models, namely Vgg-16 [33], Resnet-50 [34], Resnet-101, HRNet-w32, VitPose-S, HRFormer-B, and Swin-T [35]. We evaluated the performance of the models using five standard metrics: AP, AR, GFLOPs (Giga floating-point operations per second), Parameters, and FPS (Frames Per Second). GFLOPs and Parameters assess the model's computational complexity and storage requirements. FPS indicates the model's inference speed. Table 1 summarizes the performance of the cattle validation set. For all networks with an input size of 192 × 256, our improved RTMPose model has the fewest Params and GFLOPs, with values of 14.2 M and 2.0G, respectively. At the same time, it achieves an AP value as high as 88.6%, surpassing all other models.

Table 1. The validation results of different pose estimation models on the cattle validation set are presented. We only report the metrics for the pose estimation models and omit the object detection models. The best results are highlighted in bold.

Methods	Input Size	Params (M)	FLOPs (G)	AP	AP^{50}	AP^{75}	AR	FPS
Vgg-16	192 × 256	19.1	16.3	84.5	95.9	88.2	86.5	19
Resnet-50	192 × 256	34.1	5.5	84.8	92.3	88.3	87.0	33
Resnet-101	192 × 256	53.1	9.2	85.1	94.9	89.6	87.2	27
HRNet-w32	192 × 256	28.6	7.8	87.0	96.8	89.7	90.0	16
VitPose-S	192 × 256	24.3	5.4	86.6	96.2	89.5	88.4	32
HRFormer-B	192 × 256	43.3	13.3	86.6	96.5	88.9	88.5	16
Swin-T	192 × 256	32.8	6.2	79.8	95.7	85.4	82.0	30
Ours	192 × 256	**14.2**	**2.0**	**88.6**	**96.8**	**92.0**	**90.4**	**39**

To demonstrate the effectiveness of our model, we compared it with other top-down paradigm models. Taking the HRNet network as an example, it consists of different resolution branches, incorporating both high-resolution positional information and low-resolution semantic information, making it an efficient and competitive network. Compared to HRNet, our model requires only 49.7% of its GFLOPs and 25.6% of its parameters, while achieving a 1.6% higher AP.

Furthermore, using the PyTorch framework, our model achieves a detection speed of 39 frames per second (FPS), making it the fastest detection speed. Figure 9 shows that with increased training epochs, the AP on the validation set gradually rises and eventually converges. In the end, our improved RTMPose achieved the highest AP value.

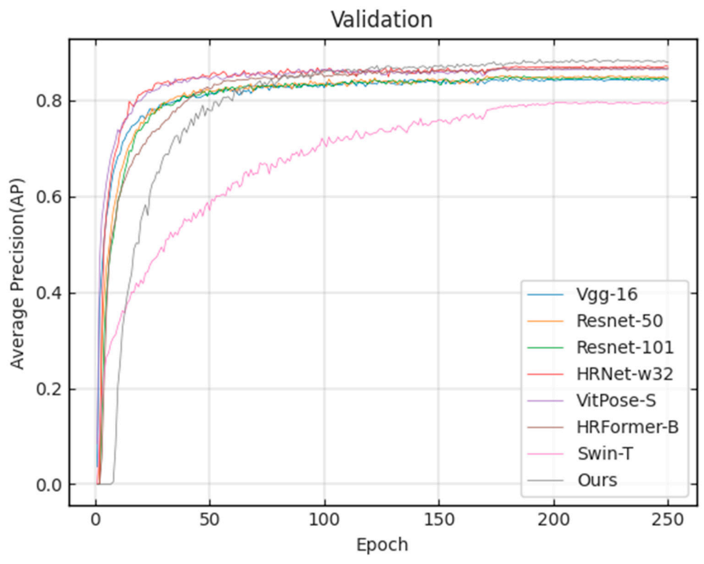


Figure 9. Pose estimation models validation set AP curves.

3.2. Test Set Results

Table 2 presents the evaluation results of the cattle test set. Similarly, our model outperforms all other models in all metrics. Specifically, our model outperforms the second-ranked model by 1.5 in AP, 0.2 in AP^{50}, 1.8 in AP^{75}, 1.9 in AR, and 6 in FPS while having the least number of model parameters and FLOPs. Table 3 presents the evaluation results of the AP-10K test set. Similarly, our model outperforms all other models in all metrics. Specifically, our model outperforms the second-ranked model by 1.5 in AP, 1.1 in AP^{50}, 2.1 in AP^{75}, and 1.0 in AR. In other words, RTMPose-FSD achieves better results with the fewest Params and FLOPs.

Table 2. The evaluation results of the cattle test set are provided. We only report the metrics for the pose estimation models and omit the object detection models. The best results are highlighted in bold.

Methods	Input Size	Params (M)	FLOPs (G)	AP	AP^{50}	AP^{75}	AR	FPS
Vgg-16	192 × 256	19.1	16.3	78.2	95.1	84.6	81.3	19
Resnet-50	192 × 256	34.1	5.5	78.8	96.1	84.3	82.3	33
Resnet-101	192 × 256	53.1	9.2	79.6	96.3	85.1	83.4	27
HRNet-32	192 × 256	28.6	7.8	81.4	96.5	88.4	83.9	16
VitPose-S	192 × 256	24.3	5.4	80.7	95.5	86.6	83.6	32
HRFormer-B	192 × 256	43.3	13.3	81.2	96.5	86.5	83.6	16
Swin-T	192 × 256	32.8	6.2	70.1	94.2	74.6	73.4	30
Ours	192 × 256	**14.2**	**2.0**	**82.9**	**96.7**	**90.2**	**85.8**	**39**

Table 3. The evaluation results of the AP-10K test set are presented. We only report the metrics for the pose estimation models and omit the object detection models. The best results are highlighted in bold.

Methods	Input Size	Params (M)	FLOPs (G)	AP	AP^{50}	AP^{75}	AR	FPS
Vgg-16	192 × 256	19.1	16.3	66.3	91.5	72.4	70.3	19
Resnet-50	192 × 256	34.1	5.5	66.5	91.7	72.7	70.4	33
Resnet-101	192 × 256	53.1	9.2	67.2	91.9	73.4	70.9	27
HRNet-32	192 × 256	28.6	7.8	70.5	93.0	77.6	74.2	16
VitPose-S	192 × 256	24.3	5.4	67.6	91.9	73.6	71.1	32
HRFormer-B	192 × 256	43.3	13.3	70.3	92.9	75.1	73.9	16
Swin-T	192 × 256	32.8	6.2	57.8	87.0	61.4	62.3	30
Ours	192 × 256	**14.2**	**2.0**	**72.0**	**94.1**	**79.7**	**75.2**	**39**

To better apply it to large-scale farms, we combine RTMDet-m with RTMPose-FSD to achieve multi-cow pose estimation. Figure 10 illustrates the prediction results of RTMPose-FSD on the cattle test set. Visual results demonstrate that even in complex scenarios, it accurately identifies the keypoints of each target. Additionally, Figure 11 shows the prediction results on the AP-10K public dataset, accurately estimating the poses of animals of various species at different scales. This suggests that our network architecture suits cattle pose estimation tasks and applies to other animals.

Figure 12 shows prediction results for state-of-the-art pose estimation algorithms, from top to bottom: HRNet-32, VitPose-S, Swin-T, HRFormer-B, and our method. Comparable results demonstrate that our approach exhibits the highest accuracy in detecting 17 keypoints, providing a more accurate depiction of the overall posture of cattle. Based on the experimental results, the improved RTMPose algorithm proposed in this paper is an excellent method for cattle pose estimation, demonstrating practical value. Our model achieves high accuracy and offers faster detection speeds, making it suitable for various applications, including commercial livestock farming.

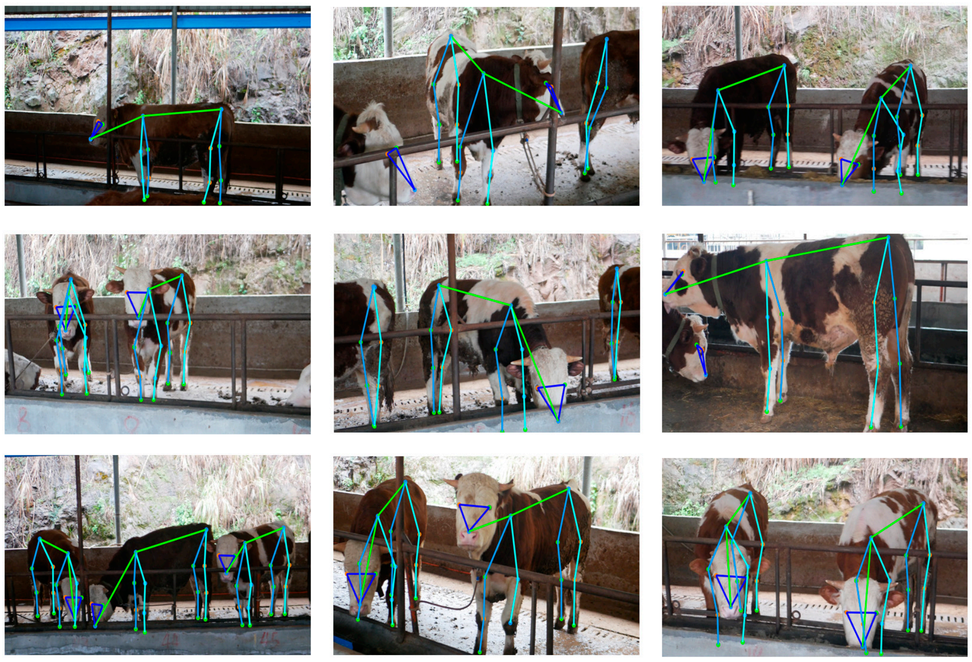


Figure 10. The detection results of RTMPose-FSD on the cattle test set. (Meaning of line colors: pure blue represents the head; green represents the neck and back; cyan represents the front left leg and back left leg; deep blue represents the front right leg and back right leg.)

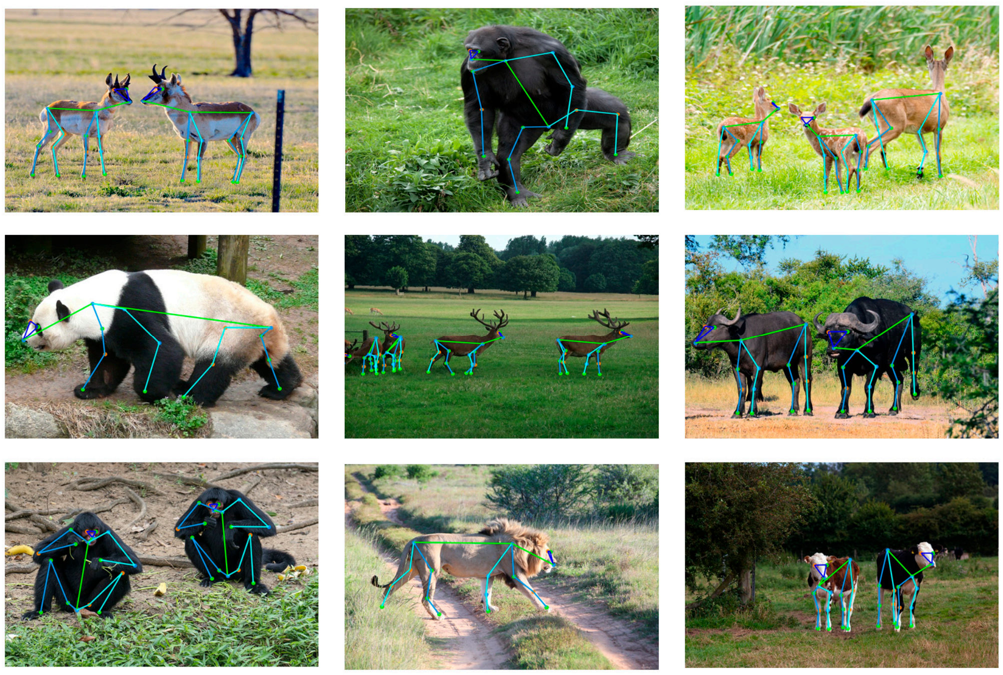


Figure 11. The detection results of RTMPose-FSD on the AP-10K test set. (Meaning of line colors: pure blue represents the head; green represents the neck and back; cyan represents the front left leg or left arm, back left leg or left leg; deep blue represents the front right leg or right arm, back right leg or right leg.)

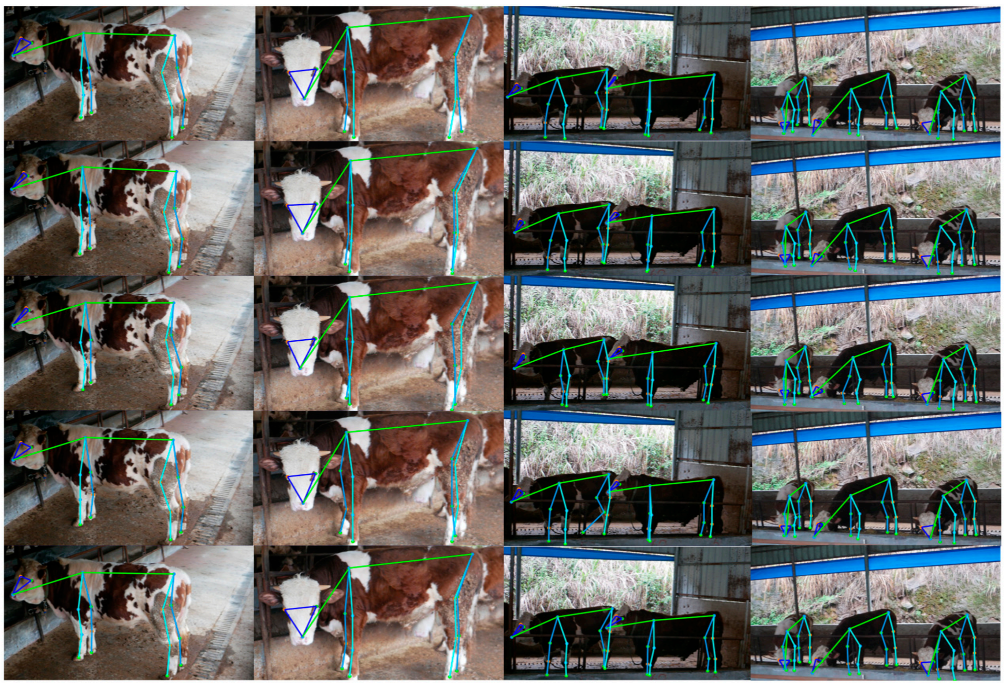


Figure 12. The prediction results of different state-of-the-art pose estimation algorithms. From top to bottom: HRNet-32 algorithm, VitPose-S algorithm, Swin-T algorithm, HRFormer-B algorithm, and our method. (Meaning of line colors: pure blue represents the head; green represents the neck and back; cyan represents the front left leg and back left leg; deep blue represents the front right leg and back right leg.)

3.3. Ablation Experiments

On the cattle test set, we conducted a series of ablation experiments to test the impact of each improvement on the model's performance. Table 4 shows the results of the ablation experiments with an input size of 192 $\times$ 256. Based on the baseline model, we divided our study into seven stages, individually testing each block and their combined effects to demonstrate their respective effectiveness. First, including the FasterNest module slightly increased model parameters and computations but led to a 0.5 AP improvement. The three-branch structure of the FasterNest Block enhances the utilization of high-level feature maps in the RTMPose model, improving its feature learning capacity. Next, by using the SimAM module to infer three-dimensional attention weights based on the spatial and channel information of feature maps, we gained a 0.3 AP improvement in spatial learning capacity without adding any parameters or computations. Finally, the Depth Block replaced the 7 $\times$ 7 large kernel convolution in the head for a lightweight design, reducing the model's parameters by 1.1M while increasing the AP by 0.4. We also tested the combined effect of using both methods, and the results showed that their combined action was better than using a single process (82.3 AP, 8.25 AP, and 82.2 AP compared to 82.0, 81.8, and 81.9). Ultimately, our improved network benefits from the combination of the FasterNest Block, SimAM, and Depth Block, achieving an overall improvement of 1.4AP without a significant decrease in inference speed. To demonstrate the effectiveness of the proposed methods, we visualized the results, as shown in Figure 13. From top to bottom, they represent the RTMPose algorithm and our method. The results show that when we added the FasterNest Block, SimAM, and Depth Block, the model's attention was more focused, and the deeper colours indicated higher confidence in detected keypoints.

Table 4. The impact of FasterNest Block, SimAM, and Depth Block on the model in terms of Parameters, GFLOPs, AP, and AR on the cattle test set is summarized. We only report the metrics for the pose estimation models and omit the object detection models.

No.	Baseline	FasterNest Block	SimAM	Depth Block	Params (M)	FLOPs (G)	AP	AR	FPS
1	√				13.6	1.9	81.5	84.4	40
2	√	√			14.6	2.0	82.0	85.3	39
3	√		√		13.6	1.9	81.8	84.9	39
4	√			√	12.5	1.9	81.9	85.1	42
5	√	√	√		14.6	2.0	82.3	85.6	39
6	√	√		√	14.3	2.0	82.5	85.7	40
7	√		√	√	12.5	1.9	82.2	85.6	40
8	√	√	√	√	14.2	2.0	82.9	86.0	39

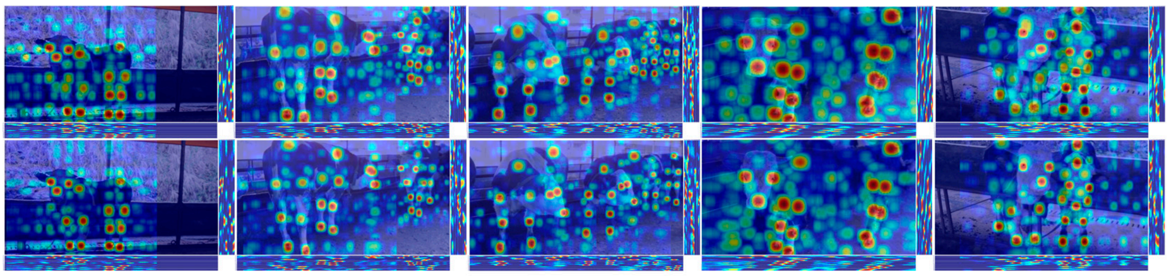


Figure 13. Visualization of our method's results. From top to bottom, they represent the RTMPose algorithm and our method.

4. Discussion

In this study, we proposed an improved RTMPose network for cattle pose estimation in complex environments, offering the advantages of speed and high accuracy. First, we designed the FasterNest Block with a three-branch structure, enhancing RTMPose's utilization of high-level feature information at different scales to improve the model's feature learning capabilities. Experimental results demonstrated that the FasterNest Block achieved excellent performance with minimal additional parameters while maintaining high inference speed. Next, the SimAM module was able to infer three-dimensional attention weights for the current layer, enhancing RTMPose's spatial learning abilities in complex real-world environments without the need for additional parameters. Finally, the Depth Block replaced the 7×7 large kernel convolution in RTMPose, improving the model's ability to learn semantic information and achieving a lightweight network while enhancing accuracy. Ablation experiments in groups 5, 6, and 7 demonstrated strong synergy between the FasterNest Block, SimAM, and Depth Block. Compared to our model, HRNet achieved the second-highest accuracy thanks to its efficient parallel multi-branch design and full-resolution feature representation throughout. Although HRNet achieved advanced performance, it suffers from redundancy and requires computationally costly intermediate high-resolution layers and upsampling. Our network optimally utilizes computational resources, maintaining high accuracy while achieving faster inference speeds. Therefore, this study can provide reliable technical support for efficient and rapid cattle pose estimation in complex farms.

In recent years, artificial intelligence technology has seen widespread successful applications in agriculture [36,37]. Deep learning-based pose estimation algorithms have been applied to various animal scenarios. In reference [10], a concise multi-branch network was constructed for cattle pose estimation. Improved bottlenecks and basic blocks enable precise localization of multiple cattle keypoints even in complex environments. Reference [38] employs the CenterNet algorithm and the proposed HRST algorithm for keypoint detection in standing pigs. The CenterNet algorithm is used for recognizing the pose of pigs, followed by applying the HRST algorithm for detecting the joint points of pigs.

Reference [39] introduced the RFB-HRNet algorithm and designed a framework consisting of three parts: target screening, animal pose estimation model, and extraction of animal gait parameters for quadruped animals. First, target screening is performed using object detection algorithms. Then, the RFB-HRNet algorithm is applied to extract keypoint data of the animals. Finally, the gait parameters of quadruped animals are derived using the keypoint data. HRNet-W48 and U-net algorithms are employed for detecting pecking injuries in turkey hens. The HRNet-W48 algorithm classifies injury locations, while the U-net algorithm precisely locates the affected areas [40]. Table 5 compares our proposed method with other animal pose estimation approaches.

Table 5. We present a performance comparison of our method and relevant animal pose estimation methods under the same evaluation metrics.

References	AP	AP^{50}	AP^{75}	AR
[10]	93.2	94.9	93.2	96.4
[38]	77.4	95.9	90.4	82.8
[39]	75.0	95.8	81.8	78.1
[40]	32.2	73.5	24.6	38.3
Ours	82.9	96.7	90.2	85.8

The method proposed in this paper achieves, to some extent, fast and accurate cattle pose estimation. However, the congestion within cattle farms and variations in lighting conditions can make specific challenges to cattle pose estimation. Therefore, overcoming congestion among cattle to enable rapid and precise nocturnal pose estimation is important. Additionally, learning structural information about keypoints in cattle is crucial for pose estimation algorithms. Hence, future work needs further improvements and optimizations to enable pose estimation algorithms to understand the structural information of keypoints in cattle, enhancing keypoint localization capabilities, especially in congested situations.

5. Conclusions

To achieve cattle pose estimation in complex scenarios. based on RTMPose, this paper introduces the FasterNest Block and Depth Block, incorporating the SimAM attention module to achieve fast and accurate cattle pose estimation. Multiple test sets indicate that our model achieves the best results on the cattle test set with the lowest number of parameters and computational costs, surpassing the second-best (HRNet, Resnet-50) by 1.5 in the average accuracy and 6.0 in the inference speed. Ablation experiments demonstrate an improvement of 1.4 AP without significantly compromising the model's inference speed. Our model can be applied in complex farm environments, providing accurate and rapid inference of multiple cattle poses. In future work, we plan to enhance the representative capacity of the cattle further pose estimation model and construct a more extensive, comprehensive dataset for cattle pose estimation.

Author Contributions: Conceptualization, K.S. and H.F.; Data curation, K.S.; Funding acquisition, X.L.; Investigation, K.S.; Methodology, X.L. and K.S.; Software, K.S. and Z.H.; Supervision, X.L. and H.F.; Validation, X.L., K.S. and H.F.; Visualization, K.S.; Writing—original draft, K.S.; Writing—review & editing, X.L., H.F. and Z.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (NNSF) under Grant No.52061020 and No.62062046.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request.

Acknowledgments: We wish to provide thanks for the data provided by Changsha Golden cattle breeding livestock.

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

References

1. Nasirahmadi, A.; Edwards, S.A.; Sturm, B. Implementation of machine vision for detecting behaviour of cattle and pigs. *Livest. Sci.* **2017**, *202*, 25–38. [CrossRef]
2. Li, J.; Xu, W.; Deng, L.; Xiao, Y.; Han, Z.; Zheng, H. Deep learning for visual recognition and detection of aquatic animals: A review. *Rev. Aq-Uaculture* **2023**, *15*, 409–433. [CrossRef]
3. Hernandez-Patlan, D.; Tellez-Isaias, G.; Hernandez-Velasco, X.; Solis-Cruz, B. Technological strategies to improve animal health and production. *Front. Vet. Sci.* **2023**, *10*, 1206170. [CrossRef] [PubMed]
4. Fang, C.; Zhang, T.; Zheng, H.; Huang, J.; Cuan, K. Pose estimation and behavior classification of broiler chickens based on deep neural networks. *Comput. Electron. Agric.* **2021**, *180*, 105863. [CrossRef]
5. Barney, S.; Dlay, S.; Crowe, A.; Kyriazakis, I.; Leach, M. Deep learning pose estimation for multi-cattle lameness detection. *Sci. Rep.* **2023**, *13*, 4499. [CrossRef]
6. Warnick, L.; Janssen, D.; Guard, C.; Gröhn, Y. The effect of lameness on milk production in dairy cows. *J. Dairy Sci.* **2001**, *84*, 1988–1997. [CrossRef]
7. Li, X.; Cai, C.; Zhang, R.; Ju, L.; He, J. Deep cascaded convolutional models for cattle pose estimation. *Comput. Electron. Agric.* **2019**, *164*, 104885. [CrossRef]
8. Liu, S.; Fan, Q.; Liu, S.; Zhao, C. DepthFormer: A High-Resolution Depth-Wise Transformer for Animal Pose Estimation. *Agriculture* **2022**, *12*, 1280. [CrossRef]
9. Gong, C.; Zhang, Y.; Wei, Y.; Du, X.; Su, L.; Weng, Z. Multicow pose estimation based on keypoint extraction. *PLoS ONE* **2022**, *17*, e0269259. [CrossRef]
10. Fan, Q.; Liu, S.; Li, S.; Zhao, C. Bottom-up cattle pose estimation via concise multi-branch network. *Comput. Electron. Agric.* **2023**, *211*, 107945. [CrossRef]
11. Sun, K.; Xiao, B.; Liu, D.; Wang, J. Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5693–5703.
12. Cheng, B.; Xiao, B.; Wang, J.; Shi, H.; Huang, T.S.; Zhang, L. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5386–5395.
13. Jiang, T.; Lu, P.; Zhang, L.; Ma, N.; Han, R.; Lyu, C.; Li, Y.; Chen, K. RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose. *arXiv* **2023**, arXiv:2303.07399.
14. Nguyen, H.C.; Nguyen, T.H.; Nowak, J.; Byrski, A.; Siwocha, A.; Le, V.H. Combined yolov5 and hrnet for high accuracy 2d keypoint and human pose estimation. *J. Artif. Intell. Soft Comput. Res.* **2022**, *12*, 281–298. [CrossRef]
15. Seong, S.; Choi, J. Semantic segmentation of urban buildings using a high-resolution network (HRNet) with channel and spatial attention gates. *Remote Sens.* **2021**, *13*, 3087. [CrossRef]
16. Yuan, Y.; Fu, R.; Huang, L.; Lin, W.; Zhang, C.; Chen, X.; Wang, J. Hrformer: High-resolution transformer for dense prediction. *arXiv* **2021**, arXiv:2110.09408.
17. Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part V 13. Springer International Publishing: Zurich, Switzerland, 2014; pp. 740–755.
18. Xu, Y.; Zhang, J.; Zhang, Q.; Tao, D. Vitpose: Simple vision transformer baselines for human pose estimation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 38571–38584.
19. Toshev, A.; Szegedy, C. Deeppose: Human pose estimation via deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1653–1660.
20. Li, J.; Bian, S.; Zeng, A.; Wang, C.; Pang, B.; Liu, W.; Lu, C. Human pose regression with residual log-likelihood estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 11025–11034.
21. Newell, A.; Yang, K.; Deng, J. Stacked hourglass networks for human pose estimation. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part VIII 14. Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 483–499.
22. Li, Y.; Yang, S.; Liu, P.; Zhang, S.; Wang, Y.; Wang, Z.; Yang, W. Simcc: A simple coordinate classification perspective for human pose estimation. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer Nature: Cham, Switzerland, 2022; pp. 89–106.
23. Lyu, C.; Zhang, W.; Huang, H.; Zhou, Y.; Wang, Y.; Liu, Y.; Zhang, S.; Chen, K. Rtmdet: An empirical study of designing real-time object detectors. *arXiv* **2022**, arXiv:2212.07784.
24. Yu, H.; Xu, Y.; Zhang, J.; Zhao, W.; Guan, Z.; Tao, D. Ap-10k: A wild animal pose estimation benchmark. *arXiv* **2021**, arXiv:2108.12617.
25. Xiao, B.; Wu, H.; Wei, Y. Simple baselines for human pose estimation and tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Honolulu, HI, USA, 21–26 July 2018; pp. 466–481.

26. Chen, J.; Kao, S.-H.; He, H.; Zhuo, W.; Wen, S.; Lee, C.-H.; Chan, S.-H.G. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 12021–12031.
27. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
28. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 23–27 October 2018; pp. 3–19.
29. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing System, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
30. Webb, B.S.; Dhruv, N.T.; Solomon, S.G.; Tailby, C.; Lennie, P. Early and late mechanisms of surround suppression in striate cortex of macaque. *J. Neurosci.* **2005**, *25*, 11666–11675. [CrossRef]
31. Yang, L.; Zhang, R.Y.; Li, L.; Xie, X. Simam: A simple, parameter-free attention module for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2021; pp. 11863–11874.
32. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
33. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
34. 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, 26 June–1 July 2016; pp. 770–778.
35. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10012–10022.
36. Zhou, H.; Wang, X.; Au, W.; Kang, H.; Chen, C. Intelligent robots for fruit harvesting: Recent developments and future challenges. *Precis. Agric.* **2022**, *23*, 1856–1907. [CrossRef]
37. Eli-Chukwu N, C. Applications of artificial intelligence in agriculture: A review. *Eng. Technol. Appl. Sci. Res.* **2019**, *9*, 4377–4383. [CrossRef]
38. Wang, X.; Wang, W.; Lu, J.; Wang, H. HRST: An Improved HRNet for Detecting Joint Points of Pigs. *Sensors* **2022**, *22*, 7215. [CrossRef] [PubMed]
39. Gong, Z.; Zhang, Y.; Lu, D.; Wu, T. Vision-Based Quadruped Pose Estimation and Gait Parameter Extraction Method. *Electronics* **2022**, *11*, 3702. [CrossRef]
40. Volkmann, N.; Zelenka, C.; Devaraju, A.M.; Brünger, J.; Stracke, J.; Spindler, B.; Kemper, N.; Koch, R. Keypoint detection for injury identification during turkey husbandry using neural networks. *Sensors* **2022**, *22*, 5188. [CrossRef] [PubMed]