applied sciences

MDPI

Article

RepNet: A Lightweight Human Pose Regression Network Based on Re-Parameterization

Xinjing Zhang and Qixun Zhou *

School of Electrical and Control Engineering, Xi'an University of Science and Technology, Xi'an 710054, China; zhangxinjing1105@163.com
* Correspondence: zhouqixun@xust.edu.cn

Abstract: Human pose estimation, as the basis of advanced computer vision, has a wide application perspective. In existing studies, the high-capacity model based on the heatmap method can achieve accurate recognition results, but it encounters many difficulties when used in real-world scenarios. To solve this problem, we propose a lightweight pose regression algorithm (RepNet) that introduces a multi-parameter network structure, fuses multi-level features, and combines the idea of residual likelihood estimation. A well-designed convolutional architecture is used for training. By reconstructing the parameters of each level, the network model is simplified, and the computation time and efficiency of the detection task are optimized. The prediction performance is also improved by the output of the maximum likelihood model and the reversible transformation of the underlying distribution learned by the flow generation model. RepNet achieves a recognition accuracy of 66.1 AP on the COCO dataset, at a computational speed of 15 ms on GPU and 40 ms on CPU. This resolves the contradiction between prediction accuracy and computational complexity and contributes to research in lightweight pose estimation.

Keywords: human pose regression; re-parameterized structure; multi-scale fusion; residual log-likelihood estimation

Citation: Zhang, X.; Zhou, Q. RepNet: A Lightweight Human Pose Regression Network Based on Re-Parameterization. *Appl. Sci.* **2023**, *13*, 9475. https://doi.org/10.3390/app13169475

Academic Editor: Krzysztof Koszela

Received: 31 July 2023
Revised: 14 August 2023
Accepted: 17 August 2023
Published: 21 August 2023

1. Introduction

Human pose estimation [1–3] refers to the locating of the keypoints of each part of the human body and the reconstructing the human joints and limb stems in images or videos, and it can also be regarded as searching for a specific pose in a space composed of all joint poses. With the development of deep learning, human pose estimation has become an important way [4–6] to solve the problems of image analysis and video understanding, and its research results have been utilized in a variety of fields, such as entertainment, sports, apparel, animation, and medicine.

Human pose estimation is categorized into 2D pose estimation [7–9] and 3D pose estimation [3], according to the space to be solved. Two-dimensional pose estimation deals with image or video frames containing multiple human bodies to determine the 2D coordinates of keypoints, and there are two ways of thinking to solve this problem. In the bottom-up approach [10–12], all keypoints are first detected and then assembled into actual poses using a matching algorithm. In contrast, in the top-down approach [13–15], the human body is first detected with a detector and cropped to an appropriate size, and the coordinates of the keypoints for each human body are predicted in a two-stage keypoint estimation network. In this work, we use a top-down approach that focuses on improving the detection of a two-stage single-person position estimation network.

The regression-based method [16–18] was first proposed as shown in Figure 1a, and it uses global mean pooling to reduce the dimensions of the feature graph and gives the predicted keypoint coordinates directly through the linear layer. Although this method is fast and efficient, the processing of the feature maps is too simple, and results in a direct loss

of spatial information. Moreover, the constraints between keypoints are not considered in the design of the networks, and these shortcomings limit the development of the regression method. Nevertheless, the regression-based method has great potential, and it is expected that its concept of end-to-end design will become popular in the future.

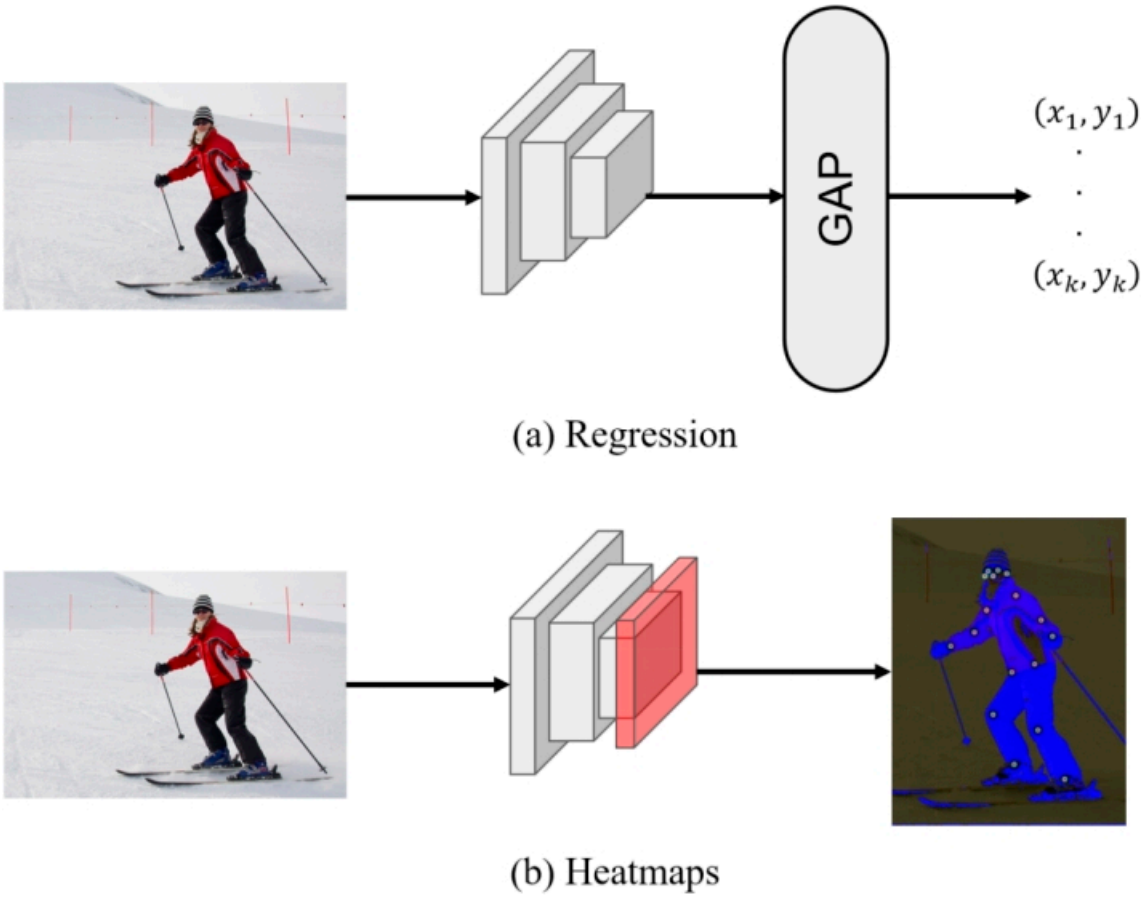


Figure 1. Two types of human pose estimation methods.

Currently, pose estimation tasks based on heatmap methods [14,15,19,20] have achieved the best results (SOTA). They use heatmap coding to supervise feature extraction and expand the size of the feature map via a deconvolution layer, as shown in Figure 1b. The heatmap method can obtain good spatial information, but it also introduces some disadvantages: the image scaling during heatmap coding will cause coordinate offsets and quantization errors [19,20]. In addition, to obtain the final prediction value, the non-maximum suppression algorithm is usually used, which makes end-to-end training impossible, and a huge consumption of computing power will be caused by the high-resolution heatmap required during the training.

Based on the above point of view, this paper is aimed at putting forward a lightweight human pose estimation model, RepNet, combined with a regression method and the idea of residual likelihood estimation [21]. It is based on re-parameterization [22] technology, and achieves the same reasoning efficiency as the simple model, while ensuring the accuracy of the complex model. The overall design block diagram is shown in Figure 2. In summary, the contributions of this paper are as follows:

1. In this paper, we propose a lightweight human pose regression framework that uses an end-to-end network design without a post-processing module, and can directly specify the coordinates of keypoints. The multi-parameterized module is adopted to reconstruct the main part of the convolution, while the well-designed training

structure is used to improve the training performance. Simplifying the structure of the network also improves the reasoning efficiency;

2. To solve the problem of the loss of precision in a simple model, the advanced idea of residual likelihood estimation is introduced in this paper. Good estimation performance can be realized by learning the underlying data distribution. RepNet achieves 63.4 AP with a parameter of 7.17 M on the COCO dataset, and the real-time performance can reach 70 FPS on RTX 3090.

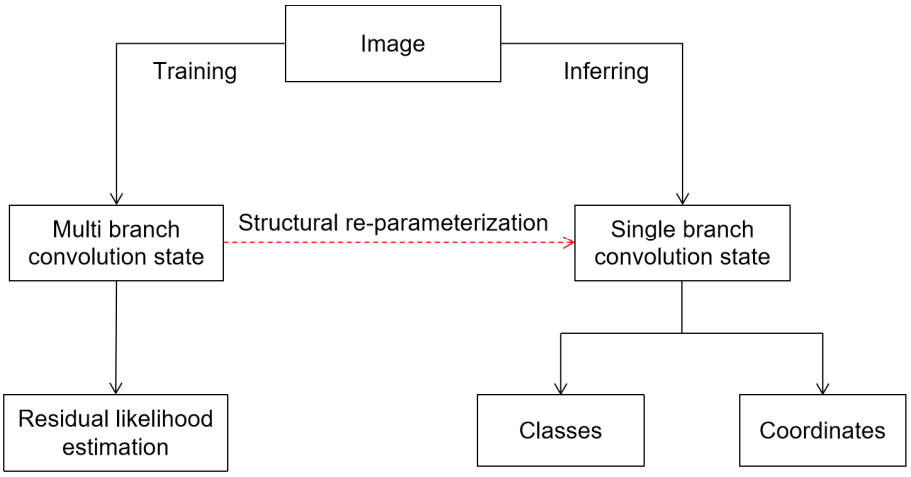


Figure 2. Overall flow diagram.

2. Related Work

Pose estimation methods based on s heatmap. At the beginning of the research, the partial constraint field (PAF) [11] is mainly used to model the spatial relationship among points in the pose estimation network, and the Hungarian algorithm is adopted for binary matching. Additionally, to group the points according to the coding distance [10], the relative position coding of each keypoint is learned. Since then, SimpleBaseline [14] uses a simple full-convolution network design to validate that high-resolution information representation is the key to pose estimation tasks. HRNet [15] has continued this design idea by adding repeated multi-scale information fusions to the network and proposing a new network backbone [23], and has achieved great success.

Pose estimation methods based on regression can directly predict the coordinates or offsets of keypoints of the human body. DeepPose [16] used a convolutional neural network for the first time to predict the coordinates of keypoints from local to global. IEF [24] continuously corrected the keypoint offsets predicted by the network through an iterative error feedback process. The Integral [25] unified heatmap representation using a simple integration operation and keypoint regression and solved the problem of non-trivial argmax in post-processing. Recently, Li et al. [21] revisited the pose estimation problem from a probabilistic statistical perspective, and proposed a new pose regression paradigm using residual log-likelihood estimation to learn the distribution changes in the underlying data, which outperformed heatmap-based methods in terms of accuracy for the first time and demonstrated the great potential of regression-based methods in the field of pose estimation.

Pose estimation methods based on Transformer. Transformer [26] is a novel sequence prediction network based on self-attention, which consists of an encoder and a decoder. In the field of natural language processing, Transformer achieves optimal results for multiple tasks, and is rapidly being extended to computer vision [27–29]. To solve problems such as pose estimation, Transpose [30] uses encoders instead of convolutional backbones to generate heatmaps, and it shows the spatial information captured by locating keypoints with the help of powerful global dependency and self-attention visualization features. PRTR [31] designed a two-level network architecture for human body recogni-

tion and keypoint coordinate regression. Inspired by the Token representation in NLP, Token coding is adopted to encode human keypoints in TokenPose [32], and a pure transformer pose estimation network is implemented. These methods show that Transformer has good capabilities for feature extractions and model constraints in both heatmap and regression methods.

3. Method

Lightweight network models can achieve good operational efficiency on resource-constrained devices, but the accuracy of their actual detection may be reduced relative to large models. Therefore, how to achieve a better balance between detection accuracy and computational complexity is the focus of this paper. In this paper, we propose a lightweight human pose estimation network with a good detection effect based on a balance between the two; we first explain its overall design scheme and then introduce the implementation details of the structural multi-parameterization technique and the functioning principle of the residual likelihood estimation, respectively, and, in the end, we show the loss function used in the training process of the model.

3.1. Networking Architecture

The RepNet network is based on a simple pose regression algorithm, developed by DeepPose [16], which was the first model to propose the idea of direct regression. In keypoint detection, DeepPose uses a convolutional network to extract image features, integrates global information using average pooling operation, and finally obtains prediction results in the fully connected layer whose network structure is shown in Figure 3a. This method provided a feasible direction for early research on regression ideas and, later, as more and more advanced algorithms were proposed, DeepPose was gradually abandoned by researchers due to its simple network structure and lower recognition performance. In this paper, the simple network structure of DeepPose is redesigned based on the structural multi-parameterization module, which effectively improves the efficiency of network inference. At the same time, a reference to the idea of residual probability estimation is added to the adaptive loss function, and the probability distribution of the input data is learned through training to achieve an improvement in the recognition accuracy, and the network structure is shown in Figure 3b.

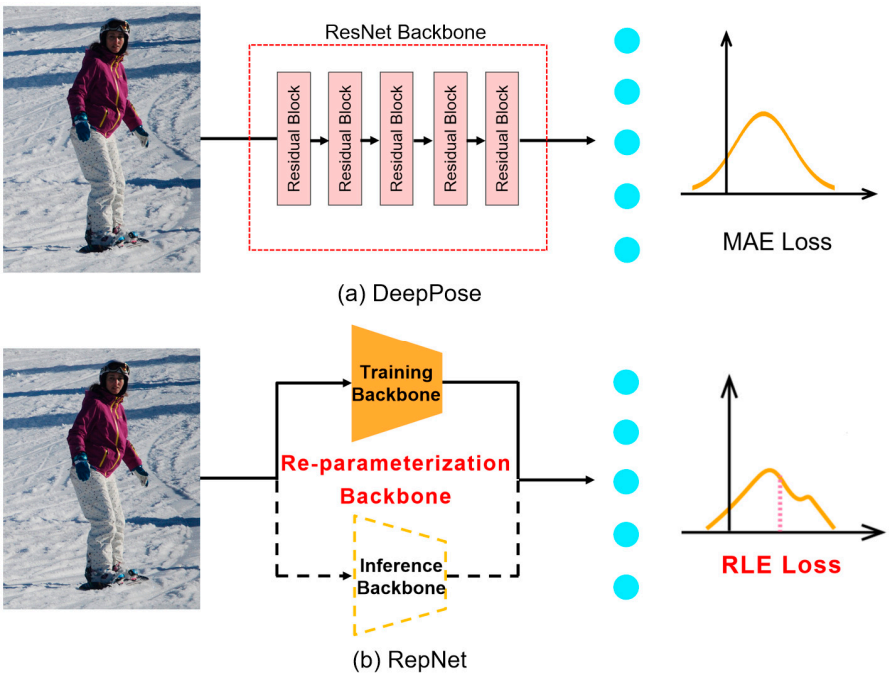


Figure 3. The overall network structure.

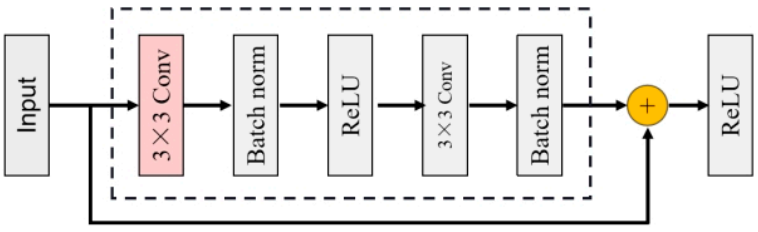
3.2. Multi-Parameter Trunk and Multi-Level Feature Fusion

In theoretical research, the computational resources required for training algorithms can usually be satisfied, while in actual engineering, the computational performance of the model inference is of more concern, and the machines deployed with models are often characterized by small memory and poor computational performance. Therefore, when the training network model is relatively large, the structure can be converted using the multi-parameterization technique, retaining the good characteristics required in the original model, and replacing the old network parameters with equivalent ones to achieve effective inference performance improvement.

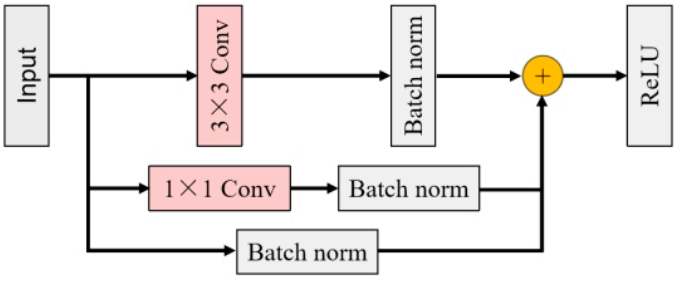
The RepNet network proposed in this paper is based on RepVGG and replaces the ResNet multi-branch convolutional module in the DeepPose model with a multi-parameterized structure. By constructing a new convolutional module with multiple branches, the training efficiency can be improved and, by decoupling the training structure and converting it into a single-branch convolution structure, memory can also be effectively saved, and fast reasoning speed can be realized. The network structures in two different states are described below.

3.2.1. Multi-Branch Structure during Training

The traditional single-branch convolution structure has the advantages of fast speed and small memory consumption, and it can achieve good flexibility and specificity. Unfortunately, due to the lack of skip connections to return information, the network is prone to gradient disappearance or gradient explosion in the training process, and it is difficult to achieve accurate detection results. On this basis, the ResNet network explicitly constructs a residual connection branch for the main part of convolution, whose output can be expressed as $y = x + f(x)$, as shown in Figure 4a. This multi-branch architecture can be regarded as a collection of many shallow models, and the information in training can be flowed to different branches by skip connections to achieve excellent performance. Inspired by the residuals module in the ResNet network, RepNet adopts a multi-branch topology for training. During training, 1×1 convolution branches are added to each 3×3 convolution layer to improve the efficiency of information transmission, and the output is $y = x + g(x) + f(x)$, where $g(x)$ is the output of 1×1 convolution. Figure 4b shows the design details of the structure.



(a) ResNet Multi-branch Structure



(b) RepNet Multi-branch Structure

Figure 4. ResNet and RepNet multi-branch structure comparison. (**a**) ResNet uses identically connected branches for training; (**b**) RepNet added a 1×1 convolution branch.

3.2.2. Re-Parameter Structure in Inferring

After obtaining the trained model, it is necessary to transform each multi-branch structure into a single 3×3 convolution for reasoning. For each original module, suppose the input is X, the output is Y, $W^{(3)}$ is a parameter of 3×3 convolution kernel, and $W^{(1)}$ is a parameter of 1×1 convolution kernel. $\mu^{(3)}, \sigma^{(3)}, \gamma^{(3)}, \beta^{(3)}, \mu^{(1)}, \sigma^{(1)}, \gamma^{(1)}, \beta^{(1)}$ are mean, variance, scaling coefficient, and migration coefficient of the normalized layer separately, and $\mu^{(0)}, \sigma^{(0)}, \gamma^{(0)}, \beta^{(0)}$ are the parameters of the normalized layer in the identical connection. The input and output of each module can then be expressed as:

$$\begin{aligned} Y = & \mathrm{bn}\left(X * W^{(3)}, \mu^{(3)}, \sigma^{(3)}, \gamma^{(3)}, \beta^{(3)}\right) \\ & + \mathrm{bn}\left(X * W^{(1)}, \mu^{(1)}, \sigma^{(1)}, \gamma^{(1)}, \beta^{(1)}\right) \\ & + \mathrm{bn}\left(X * W^{(0)}, \mu^{(0)}, \sigma^{(0)}, \gamma^{(0)}, \beta^{(0)}\right) \end{aligned} \tag{1}$$

where the BN function for each output channel can be written as:

$$\mathrm{bn}(X, \mu, \sigma, \gamma, \beta)_i = (X_i - \mu_i)\frac{\gamma_i}{\sigma_i} + \beta_i \tag{2}$$

According to the above formula, a 3×3 convolution with a BN layer can be transformed into a new 3×3 convolution layer. The weight $W\prime$ and offset vector $b\prime$ of this layer can be expressed as:

$$W_i' = \frac{\gamma_i}{\sigma_i} * W_i^{(3)} \tag{3}$$

$$b_i' = \frac{\mu_i \gamma_i}{\sigma_i} + \beta_i \tag{4}$$

This parametric transformation applies equally to the 1×1 convolution and the constant connection, where the constant connection can be viewed as a 1×1 convolution kernel with the unit matrix as the parameter weight. After applying the above transformations to all three branches, one 3×3 convolution kernel and two 1×1 convolution kernels with three bias vectors can be obtained equivalently. Then, each bias vector is summed up to obtain the final bias value and, at the same time, the 1×1 convolution kernel needs to be filled up and summed up with other convolution kernels to obtain the equivalent substituted 3×3 convolution kernel, and the whole structure's re-parametrization process is shown in Figure 5.

The feature fusion of the multi-parameterized backbone is shown in Figure 6 and, in this paper, we select the feature maps obtained from the last three stages, whose scales are $h_1 \times w_1$, $h_2 \times w_2$, and $h_3 \times w_3$, and the scale of each level of the feature maps is $1/2$ that of the previous level. We use 3×3 convolutional layers and dilation convolution [33,34] to transform the scales of the first and third level feature maps to $h_2 \times w_2$, respectively, and each level of the feature maps needs to be transformed to the same dimensions by 1×1 convolutional layers to transform the number of channels to the same dimension and, finally, fused as the output of the model backbone.

3.3. Residual Log-Likelihood Estimation

Designs based on regression ideas have shown great potential in the field of pose estimation compared to heatmap methods. However, regression networks are slightly less effective in terms of detection performance, while the lightweight model structure also entails some loss of accuracy. For this reason, we view the standard loss function as a specific assumption, model the output distribution with the maximum likelihood estimation, and construct the likelihood function based on the true latent distribution, which effectively improves the performance of human pose regression.

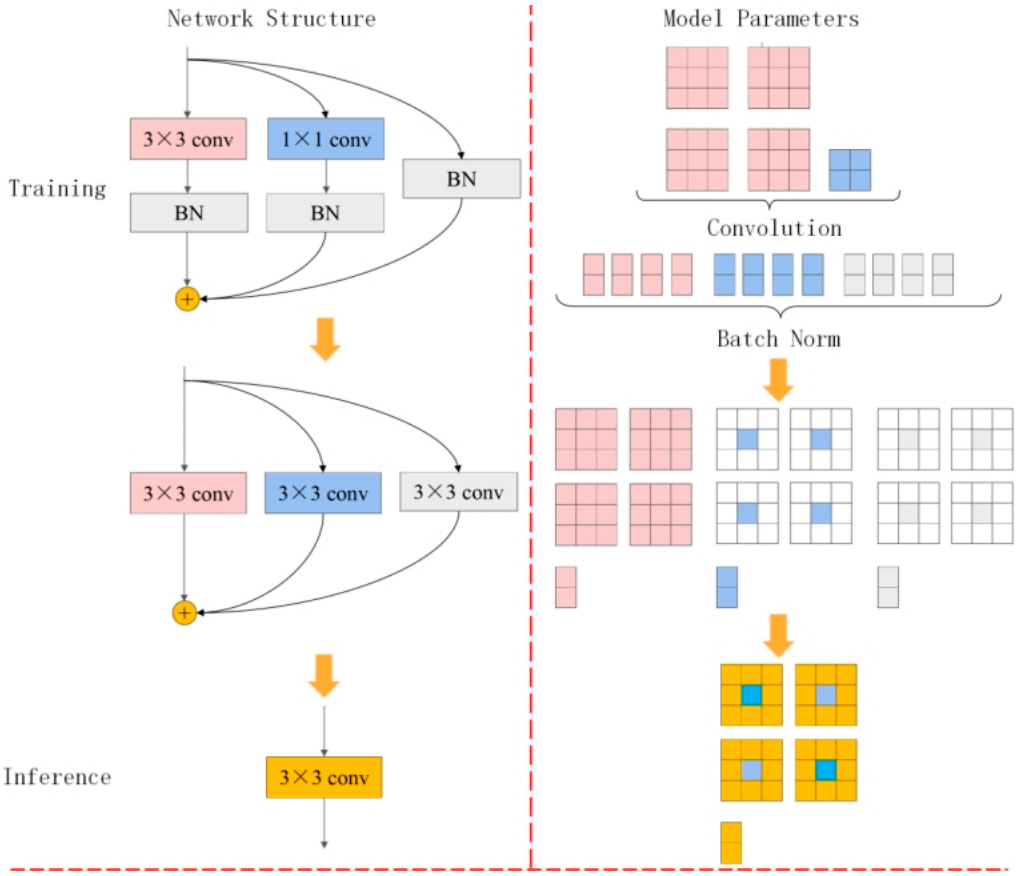


Figure 5. The structure reparametrizes a process in which a 1×1 convolution branch in training and an identically connected branch is equivalent to a 3×3 convolution in inference.

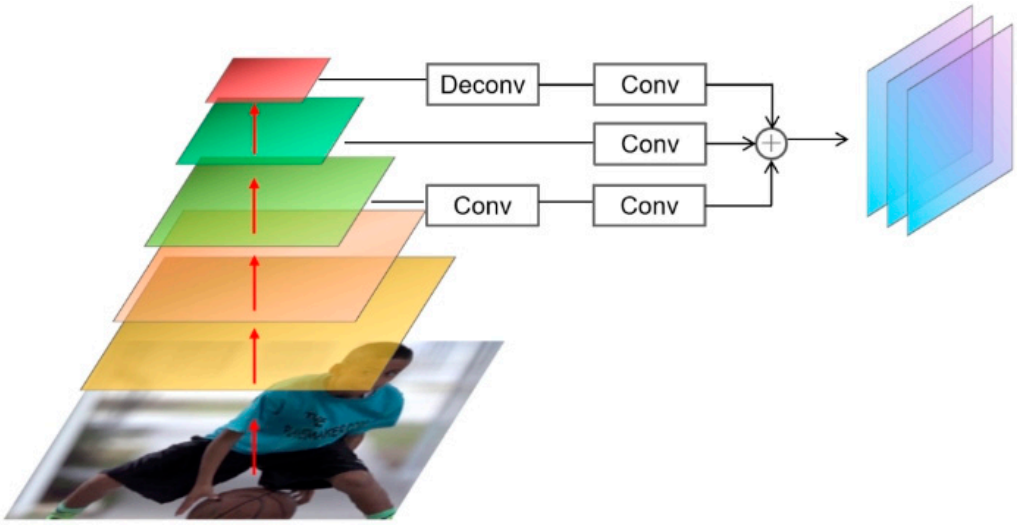


Figure 6. Feature fusion trunk.

In this paper, a novel and effective state-of-the-art idea, called residual log-likelihood estimation, is introduced in RepNet, which utilizes a normalized flow model to estimate the true data distribution and promote the development of pose regression. Specifically, the method first artificially gives a pre-existing data distribution model, and then gradually makes changes to the original distribution through learning, finally obtaining a loss function close to the true distribution. Compared with the original keypoint prediction, the RLE module does not need a complex network structure, and can use a ready-made flow

generation model to optimize the output results. In the implementation, the RLE module is trained by a maximum likelihood estimation process, and the regression network backbone and the RLE module can be updated simultaneously. In addition, the RLE module is not involved in the inference phase, and can bring significant improvements to the regression model without any additional time overhead.

3.3.1. Maximum Likelihood Estimation

In keypoint regression networks, maximum likelihood estimation is usually used to predict coordinates. From this point of view, when given an input image I, the probability distribution $P_\Theta(x|I)$ can be calculated to represent the probability of the predicted result shown at the position x, where Θ represents the learnable network parameters, and x is equivalent to the annotated real simple label μ_g. The whole training process can be viewed as an update of the parameter Θ to make the predicted result gradually close to the actual coordinates. The loss of the maximum likelihood estimation process can be expressed as:

$$L_{mle} = -\log P_\Theta(x|I)_{x=\mu_g} \tag{5}$$

Different regression losses can be realized in the above formulas by making specific assumptions about the output probability distribution. Taking the l_1 and l_2 losses commonly used in human pose estimation as an example, when the density function is set to be Laplace distribution, the l_1 loss can be expressed as:

$$l_1 = -\log P_\Theta(x|I)_{x=\mu_g} \propto \log \hat{\sigma} + \frac{\left|\mu_g - \hat{\mu}\right|}{\hat{\sigma}} \tag{6}$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the mean and variance of the prediction, respectively. When $\hat{\sigma}$ is a constant, the loss l_1 can be expressed as:

$$l_1 = \left|\mu_g - \hat{\mu}\right| \tag{7}$$

Similarly, when the default distribution is a Gaussian probability density function, the loss l_2 can be expressed as:

$$l_2 = \left(\mu_g - \hat{\mu}\right)^2 \tag{8}$$

It can be seen that the loss function depends on the representation of probability distribution $P_\Theta(x|I)$, and a more accurate density function can produce a better prediction effect. However, because the representation of the potential distribution is difficult to predict in advance, the RLE module uses a flow generation model for probability density function transformation.

3.3.2. Flow Generation Model

By training a generator G and sampling the input data, the depth-generating model transforms the sample z in simple distribution $\pi(z)$ into the sample x in complex distribution $p_G(x)$:

$$p_G(x) = \pi(z)|\det(J_{G^{-1}})| \tag{9}$$

In the flow-based generation model, $p_G(x)$ can be calculated directly by constructing a reversible transformation function. The key to the method is how to design the target distribution to be learned, and the details of the implementation of the invertible function are beyond the scope of this paper, so we will not repeat them here. For the pose estimation task, a straightforward idea is to first learn a simple Gaussian distribution $P_\Theta(z|I)$ through a regression network, and then transform it to a true distribution $P_{\Theta,\Phi}(x|I)$ of the data

with a flow model, as shown in Figure 7a; the relationship between the two is derived from Equation (10):

$$\log P_{\Theta,\Phi}(x|I) = \log P_{\Theta}(z|I) + \log\left|\det\frac{\partial f_{\Phi}^{-1}}{\partial x}\right| \tag{10}$$

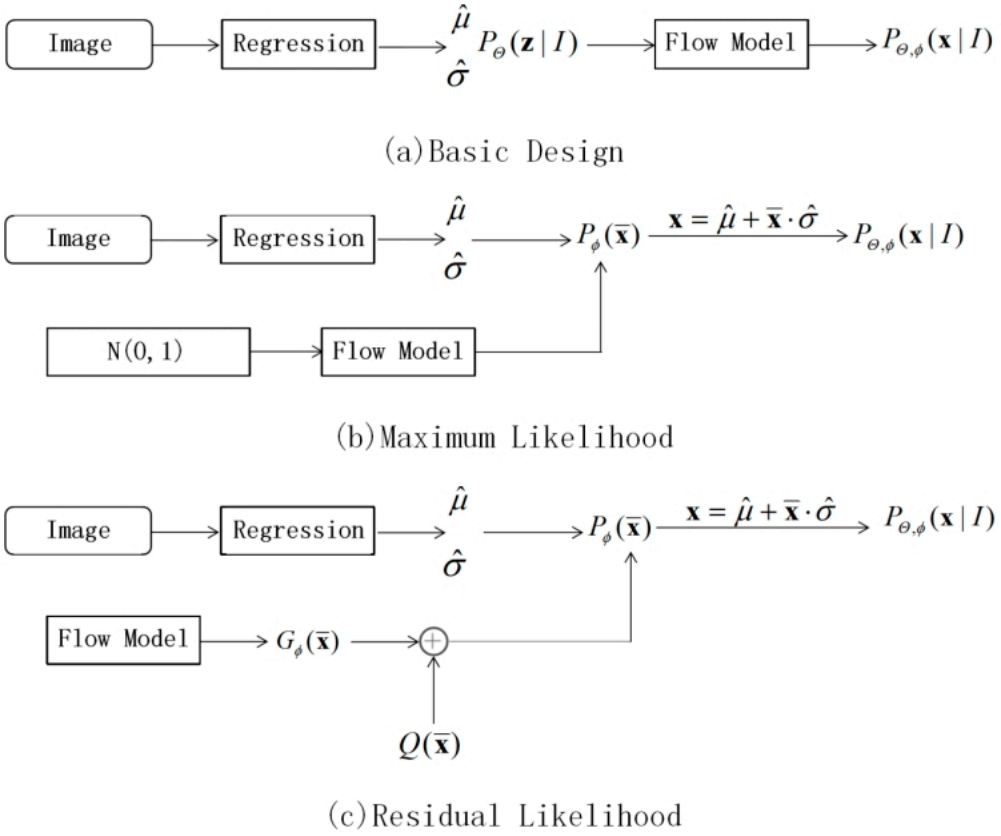


Figure 7. RLE design flow.

In Equation (10), Θ is the parameter of the regression network learning, and ϕ is the learning parameter of the flow model. As long as f_{Φ} is complex enough, it can fit any distribution. For a given arbitrary x, the likelihood probability can be obtained by calculating z in reverse. However, the keypoint of this method is the distribution of truth value in the data set, which does not meet the needs of this paper.

Based on the above research, we can further assume that the underlying distribution of all data shares the same density function cluster, but has different variance and mean on input I. The normal distribution can then be mapped to a primitive probability density function $P_{\Phi}(\overline{x})$ with a flow model, and the translation coefficient $\hat{\mu}$ and the scaling coefficient $\hat{\sigma}$ of the original distribution can be predicted by the regression network to obtain the final true distribution. At this point, the loss in Equation (5) can be re-expressed as:

$$Lmle = -\log P_{\Theta,\Phi}(x|I)\Big|_{x=\mu_g} = -\log P_{\Phi}\left(\overline{\mu}_g\right) + \log\hat{\sigma} \tag{11}$$

where $\overline{\mu}_g = \left(\mu_g - \hat{\mu}\right)/\hat{\sigma}$. Thus, the probability distribution based on the flow model allows for end-to-end training evaluation, as shown in Figure 7b. However, from Equation (11), it can be seen that the learning of the regression model depends entirely on the results of the transformation of the flow model, and the performance of the model is reduced. To avoid this problem, the idea of residual design can be followed, i.e., an initial distribution

function can be set artificially, and the training is on this basis. The logarithm of distribution $P_{\Phi}(\overline{x})$ can be expressed as:

$$\log P_{\Phi}(\overline{x}) = \log Q(\overline{x}) + \log G_{\Phi}(\overline{x}) \tag{12}$$

In Equation (12), $Q(\overline{x})$ is a known simple distribution, which can be set as a Gaussian distribution or a Laplace distribution probability density function, and $G_{\Phi}(\overline{x})$ is a distribution learned from the flow model, which is a residual estimate of the simple distribution to bring $P_{\Phi}(\overline{x})$ closer to the real data distribution. The design flow is shown in Figure 7c. The residual log-likelihood estimation can be calculated by Equation (11), which is expressed as:

$$Lrle = -\log P_{\Theta,\Phi}(x|I)\Big|_{x=\mu_g} = -\log Q\left(\overline{\mu}_g\right) - \log G_{\Phi}\left(\overline{\mu}_g\right) + \log \hat{\sigma} \tag{13}$$

The RepNet network uses Equation (13) to replace the original regression loss to train the model. In the concrete implementation of the RLE module, the lightweight flow model is selected to learn the residual distribution, and the mean $\hat{\mu}$ and deviation $\hat{\sigma}$ of the predicted data are taken as the actual results. In the test phase, the keypoint coordinates are obtained by outputting the mean value $\hat{\mu}$ directly, and the trained flow model will no longer participate in the reasoning process to avoid affecting the performance of the subsequent network operation.

4. Experiment

4.1. Implementation Details

Model settings. RepNet uses several multi-parameterized modules in Figure 4 to stack and implement the feature extraction backbone. Similar to ResNet, in the network, five stages are set up to perform downsampling operations on the feature maps to gradually realize the increase in the number of feature channels. Table 1 demonstrates the specifics of different stages when the input image is 256 × 256. In this paper, two models, RepNet-A and RepNet-B, are modelled, using the layer configuration of {1, 2, 4, 14, 1} for the five stages of the network, and different channel scaling factors are set for controlling the number of output channels, respectively. Following the setup for the RepVGG network, the channel scaling factors were set to 0.75 and 2.5 for model A, and 1 and 2.5 for model B. Model B was used to control the number of output channels.

Table 1. Backbone configuration.

Network Phase	Output Size	Channel Numbers of RepNet-A	Channel Numbers of RepNet-B
1	128 × 128	64 × 0.75	64
2	64 × 64	64 × 0.75	64
3	32 × 32	128 × 0.75	128
4	16 × 16	256 × 0.75	256
5	8 × 8	512 × 2.5	512 × 2.5

For the RLE module, the fixed distribution terms in Equation (13) are replaced with the Laplace transform as setup in the original paper. In the training, we use the off-the-shelf flow generation model RealNVP [35] for learning the residual likelihood estimation, and the reversible transformation function is obtained by multiple stacked fully connected layers, where the number of fully connected layers is set to 3, with each layer featuring 64 neutrons. The dimension of the fully connected layers of the final output result is set to $K \times 4$, and K is the number of predicted keypoints. In the testing phase, the average of the network predictions $\hat{\mu}$ will be used as the result of coordinate regression.

Dataset. We verified the effectiveness of the method on COCO [7] and MPII [8] datasets, respectively. The COCO dataset annotated 17 systemic keypoints for 250 K

individuals in more than 200 K images. Except for 57 K images that were used for training, it also included a validation set of 5 K images and a test set of 20 K for evaluating training results. The MPII dataset contains approximately 25 K images with over 40,000 individual annotations, highlighting a total of 16 keypoints in the human body. We divided the data into a 22 K training set and a 3 K test set, according to the official criteria.

Evaluation indicators. Average precision (AP) based on object keypoint similarity (Oks) is used as the evaluation index in COCO, and its calculation equivalent is shown in Equation (14). MPII calculates the average PCK index under the given threshold to evaluate the performance of the algorithm, as is shown in (15):

$$AP_M = \frac{\sum_m \sum_p \delta\left(oks_p > T\right)}{\sum_m \sum_p 1} \tag{14}$$

$$PCK^k_{mean} = \frac{\sum_p \sum_i \delta\left(\frac{d_{pi}}{d^{def}_p} \leq T_k\right)}{\sum_p \sum_i 1} \tag{15}$$

Train settings. All workflows of the RepNet network are based on the MMPose framework and are validated on the COCO and MPII datasets, respectively. All training images are pre-processed with random rotation of ± 30 degrees and random scale scaling of $[0.75, 1.25]$. During the training phase, the COCO dataset is processed by the network using an input size of 256×192. The learning rate used for the COCO dataset is 1×10^{-3}, and the learning rate trained for the MPII dataset is 5×10^{-4}. The optimizer is AdamW [36] with a multi-stage weight decay strategy, and 210 iterations were performed on an RTX-3090 GPU at a batch size of 64. The learning rate is reduced to 0.1 in the 170 s and 200 s, respectively.

Test settings. During the test, the same staff detector as DeepPose is adopted for evaluation, and the keypoints of the clipped single image are detected. The remaining parameters follow the setting of MMPose. The reasoning efficiency of the re-parameterized network model and other lightweight algorithms is tested under the environment of an RTX-3090 GPU and an Intel (R) Xeon (r) CPU E5-2620 V 3@2.40 GHz.

4.2. Experiment Results

COCO Dataset. Table 2 shows the experimental results of the proposed lightweight network on MS-COCO datasets. We tested the models with the configuration of RepNet-A and RepNet-B mentioned above. The results show that RepNet-A uses RepVGG-A0 as the model backbone and is focused on the edge device deployment, and achieves the detection accuracy of 63.4 AP on the COCO validation, while RepNet-B is based on RepVGG-A1, is on the focus on the service-side deployment, and achieves the detection accuracy of 66.1 AP on the COCO validation set. Compared to the other regression methods, the model proposed in this paper has achieved more competitive results. In addition, RepNet surpasses some earlier algorithms in the heatmap approaches [37], and has achieved results similar to those of the Lite-HRNet [38] model and PointSetNet [39].

The proposed network model can be used for lightweight pose estimation via the re-parameterization technique, and Table 3 shows the test results of the reasoning efficiency of RepNet and various lightweight pose estimation algorithms in the same experimental environment. Among them, YOLOv7-pose achieves the highest detection accuracy in a number of lightweight networks, but its parameter number and computation amount are much higher than other models, and its inference efficiency at the CPU side is poor, making it difficult to deploy to some edge devices with poor computing performance. Compared with the benchmark algorithm DeepPose, the RepNet network achieves a 12.1% improvement in detection performance with a reasoning efficiency of nearly five times, while compared with OpenPose and Integral algorithms, RepNet can improve detection accuracy and reasoning speed effectively. Both Lite-HRNet and RepNet are top-down approaches, while RepNet improved its performance by nearly ten times with an AP accuracy of less than 1.4%, far outperforming the Lite-HRNet test results. Finally,

we test two networks, MobileNet [40] and ShuffleNet [41], which are designed based on depth-separable convolutions. The experimental results show that RepNet achieves better detection performance and lighter reasoning results on the CPU side with similar computing performance when using residual likelihood estimation. Figure 8 shows the qualitative results of the COCO dataset.

Table 2. COCO validation experimental results.

Method	Backbone	Input Size	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR
Heatmap-based								
OpenPose	PAF	—	65.3	85.2	71.3	62.2	70.7	—
CPM	CPM	384 × 288	65.1	86.4	72.9	61.9	71.6	71.0
Lite-HRNet	Lite-HRNet	256 × 192	67.5	88.0	75.3	64.6	73.3	73.5
SimpleBaseline	ResNet-50	384 × 288	72.2	89.3	78.9	68.1	79.7	77.6
HRNet	HRNet	384 × 288	**75.8**	**90.6**	**82.7**	**71.9**	**82.8**	**81.0**
Regression-based								
DeepPose	ResNet-50	256 × 192	54.0	82.3	65.9	55.2	65.0	68.5
Integral	ResNet-50	256 × 256	63.3	85.9	70.3	59.3	71.5	**72.9**
PointSetNet	ResNeXt	—	65.7	85.4	71.8	—	—	—
CenterNet	Hourglass	—	64.0	—	—	—	—	—
RepNet-A	RepVGG	256 × 192	63.4	85.1	69.9	60.2	69.6	68.7
RepNet-B	RepVGG	256 × 192	**66.1**	**86.5**	**73.5**	**62.7**	**72.4**	71.1

Table 3. Reasoning speed test results.

Methods	Parameter Acquisition	GFLOPs	AP	AR	GPU Speed	CPU Speed	Model Size
OpenPose	49.8 M	136.0	65.3	—	656 ms	881 ms	199.5 M
Integral	34.0 M	7.28	63.3	72.9	28 ms	288 ms	136.7 M
DeepPose	23.58 M	4.04	54.0	68.5	24 ms	189 ms	97.0 M
Lite-HRNet	1.76 M	0.42	**67.5**	**73.5**	182 ms	347 ms	7.2 M
MobileNet	2.36 M	0.31	59.3	64.4	22 ms	359 ms	16.1 M
ShuffleNet	**1.02 M**	**0.14**	51.1	56.6	26 ms	128 ms	**4.1 M**
YOLOv7-pose	80.2 M	101.6	71.4	77.6	30 ms	561 ms	153 M
RepNet-A	7.17 M	1.49	63.4	68.7	**14 ms**	**32 ms**	27.4 M
RepNet-B	11.65 M	2.32	**66.1**	**71.1**	15 ms	40 ms	44.5 M

MPII datasets. Table 4 shows the test results of RepNet on the MPII dataset, and it proves that the method achieves the 85.8PCKh evaluation index with the input size of 256 × 256, outperforming other regression methods with the same configurations. Compared with the benchmark algorithm DeepPose, the detection accuracy of Model A and Model B is improved by 3.3% and 4.2%, respectively. And, Model B achieves similar results to the Lite-HRNet and SimpleBaseline networks based on the heatmap method with a lightweight network backbone, as shown in Figure 9.

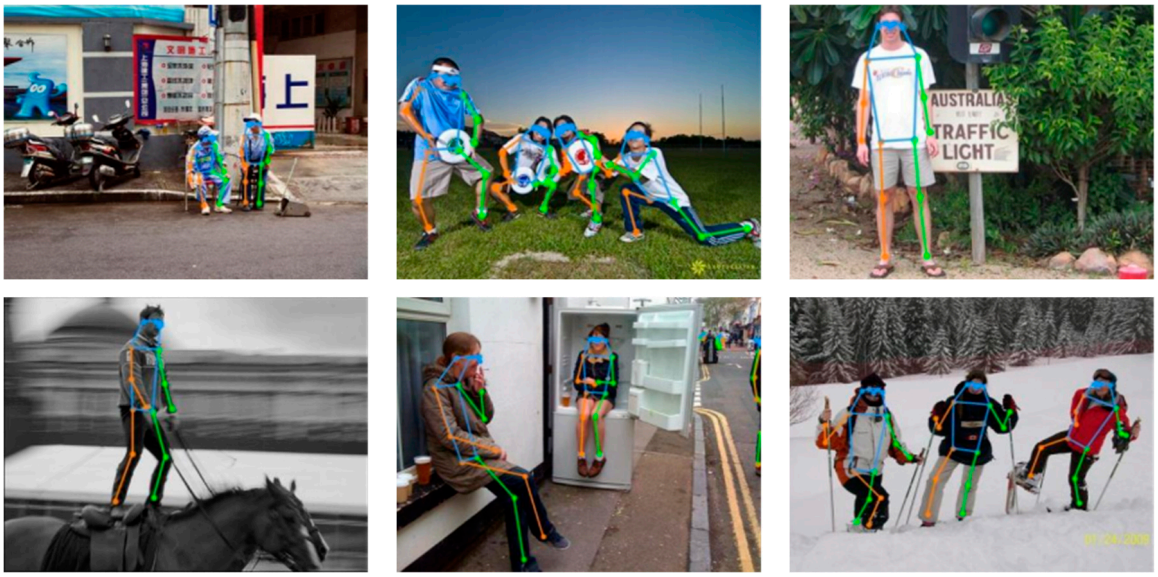


Figure 8. Results on COCO dataset.

Table 4. Experimental results on MPII dataset.

Method	Backbone	Hea	Sho	Elb	Wri	Hip	Kne	Ank	Mean	$\text{Mean}_{0.1}$
Heatmap-based										
Lite-HRNet	Lite-HRNet	96.2	94.6	86.9	80.6	87.1	82.0	76.9	86.9	31.2
Simple-Baseline	Mobile-Net	95.2	93.4	85.8	78.4	85.9	79.3	74.3	85.4	27.1
Simple-Baseline	ResNet-50	96.3	95.2	88.6	83.3	87.3	83.5	78.9	88.2	32.8
HRNet	HRNet-W32	97.1	95.9	90.3	86.4	89.1	87.1	83.3	90.3	—
TransPose	HRNet-W32	—	—	—	—	—	—	—	**90.3**	**41.6**
Regression-based										
DeepPose	ResNet-50	93.5	92.9	82.7	73.9	85.6	75.4	66.1	82.5	20.5
RLE	ResNet-50	—	—	—	—	—	—	—	85.5	26.7
PRTR	ResNet-50	94.6	93.1	83.1	74.1	84.7	74.7	69.4	82.8	22.6
PRTR	ResNet-152	96.1	94.4	86.1	78.5	87.6	81.8	74.6	86.3	26.8
RepNet-A	RepVGG	95.7	94.4	86.3	78.2	88.0	79.7	72.6	85.8	27.6
RepNet-B	RepVGG	**95.7**	**95.0**	**87.0**	**79.7**	**88.1**	**81.5**	**74.7**	**86.7**	**29.2**

4.3. Ablation Experiments

To compare the effects of different methods on the performance of the model, the RepNet-A network is used to perform the ablation experiments on the MPII dataset with the help of the MMPose framework, the following tables show the results of detection accuracy and reasoning speed with the improved modules.

The re-parameterization technique achieves lightweight reasoning performance by transforming the structure of the training model. Table 5 shows the test results of RepNet at different stages. It can be seen that the transformed reconfiguration model reduces both the number of parameters and the computational load, improves the reasoning speed on GPU and CPU significantly, and effectively reduces the size of the model capacity.

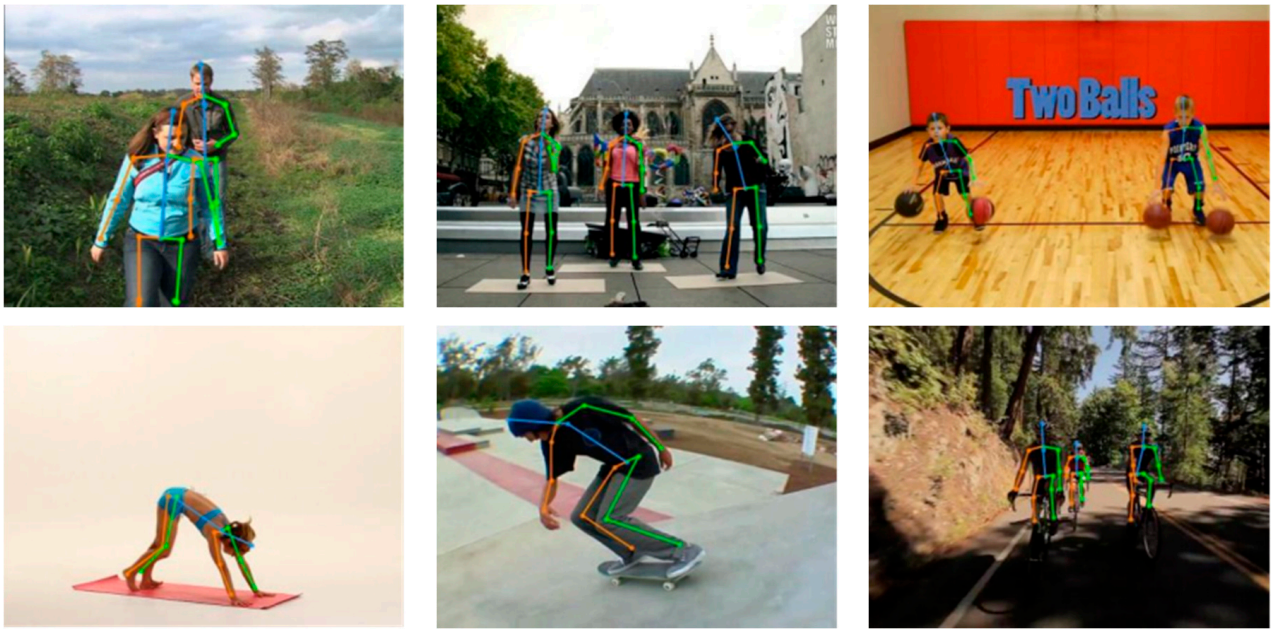


Figure 9. Results on MPII dataset.

Table 5. Results of model experiments at different stages.

Method	Stage	The Number of Parameters	GFLOPs	Reasoning Speed (GPU/CPU)	Size of Model
RepNet-A	Training	7.96 M	1.49	36 ms/81 ms	38.14 M
	Reasoning	**7.16 M**	**1.34**	**26 ms/50 ms**	**27.38 M**
RepNet-B	Training	12.95 M	2.59	40 ms/115 ms	57.52 M
	Reasoning	**11.64 M**	**2.32**	**28 ms/67 ms**	**44.47 M**

Table 6 demonstrates the impact of residual likelihood estimation and re-parameterized structure on model detection accuracy. A simple substitution of the direct regression header using the RLE module yields a 3.6% improvement in accuracy compared to the benchmark algorithm. In addition, when using the re-parameterized structure as the model backbone, the detection accuracy decreases by 0.3%, but the model parameters are reduced by 70% and the inference time per image is reduced to 26 ms. Overall, the improved algorithm proposed in this paper achieves a 3.3% enhancement in detection accuracy and a more than 80% increase in inference speed compared to the original network, and, in Figure 10, the qualitative results of the two approaches are compared.

Table 6. Ablation results of improved modules.

Method	The Number of Parameters	GFLOPs	Mean	Mean@0.1	Reasoning Speed (GPU/CPU)
ResNet + Direct Regression	23.57 M	4.04	82.5	20.5	33 ms/274 ms
ResNet + RLE	23.69 M	4.04	86.1	27.7	33 ms/277 ms
Multi-parameterized Structure + RLE	**7.16 M**	**1.34**	**85.8**	**27.6**	**26 ms/50 ms**

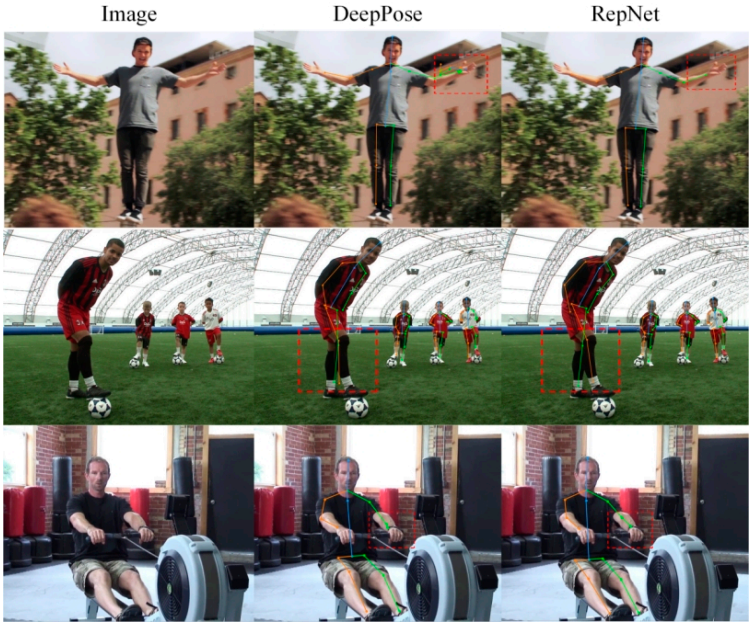


Figure 10. Comparison of qualitative results.

The regression network presented in this paper can achieve a more lightweight performance by reducing the input size. Table 7 shows the ablation comparison between RepNet and the SimpleBaseline method. As can be seen from the experimental results, when the heatmap-based pose estimation method reduces its input size to half of the original, the detection accuracy of keypoints will decrease sharply, which proves that the heatmap method is highly dependent on the input size. In contrast, the network proposed in this paper can still retain effective detection under the deterioration of the data environment and shows good robustness to be well deployed on some edge computing devices.

Table 7. Input Size Ablation Results.

Method	Input Size	Mean@0.1
SimpleBaseline	256 × 256	32.8
	128 × 128	15.7
RepNet	256 × 256	27.6
	128 × 128	17.1

5. Conclusions

In this paper, from the perspective of lightweight network design, a human pose estimation model with a re-parameterization structure is proposed. In order to design an easy-to-deploy network model, RepNet improves the regression algorithm by adopting the re-parameterization technique with the residual likelihood estimation idea, and implements a pose regression algorithm with a concise structure and good performance. Meanwhile, we validate the effectiveness of the proposed method through multiple sets of comparison experiments on COCO and MPII datasets. Overall, this paper achieves a good improvement to the early pose regression network based on the current advanced ideas in the field of deep learning, which provides a feasible direction for the research of lightweight models. The current RepNet still has some shortcomings when it comes to images with noise and low contrast. With the continuous development of deep learning technology in the future, more data enhancement methods will be used to innovate lightweight models, and they will further promote the application of human pose estimation in real scenarios.

Author Contributions: Conceptualization, X.Z.; Methodology, X.Z.; Software, X.Z.; Validation, Q.Z.; Data curation, X.Z.; Writing – original draft, X.Z.; Visualization, X.Z.; Supervision, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Shaanxi Provincial Department of Science and Technology, grant number 2023YBGY368.

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

References

1. Zheng, C.; Wu, W.; Chen, C.; Yang, T.; Zhu, S.; Shen, J.; Kehtarnavaz, N.; Shah, M. Deep learning-based human pose estimation: A survey. *arXiv* **2020**, arXiv:2012.13392. [CrossRef]
2. Munea, T.L.; Jembre, Y.Z.; Weldegebriel, H.T.; Chen, L.; Huang, C.; Yang, C. The progress of human pose estimation: A survey and taxonomy of models applied in 2D human pose estimation. *IEEE Access* **2020**, *8*, 133330–133348. [CrossRef]
3. Ionescu, C.; Papava, D.; Olaru, V.; Sminchisescu, C. Human3. 6M: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *36*, 1325–1339. [CrossRef] [PubMed]
4. Yan, S.; Xiong, Y.; Lin, D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, American, 2–7 February 2018; Volume 32.
5. Shi, L.; Zhang, Y.; Cheng, J.; Lu, H. Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 12026–12035.
6. Duan, H.; Zhao, Y.; Chen, K.; Lin, D.; Dai, B. Revisiting Skeleton-based Action Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 2969–2978.
7. 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 Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
8. Andriluka, M.; Pishchulin, L.; Gehler, P.; Schiele, B. 2D human pose estimation: New benchmark and state of the art analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–29 June 2014; pp. 3686–3693.
9. Li, J.; Wang, C.; Zhu, H.; Mao, Y.; Fang, H.S.; Lu, C. CrowdPose: Efficient Crowded Scenes Pose Estimation and A New Benchmark. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 19–20 June 2019; pp. 10863–10872.
10. Newell, A.; Huang, Z.; Deng, J. Associative Embedding: End-to-End Learning for Joint Detection and Grouping. *Adv. Neural Inf. Process. Syst.* **2016**.
11. Cao, Z.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 7291–7299.
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, 14–19 June 2020; pp. 5386–5395.
13. Newell, A.; Yang, K.; Jia, D. Stacked Hourglass Networks for Human Pose Estimation. In *Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam*; Amsterdam, The Netherlands, 11–14 October 2016, Springer International Publishing: Cham, Switzerland, 2016.
14. Xiao, B.; Wu, H.; Wei, Y. Simple Baselines for Human Pose Estimation and Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 466–481.
15. 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, 19–20 June 2019; pp. 5703–5963.
16. Toshev, A.; Szegedy, C. DeepPose: Human Pose Estimation via Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 1653–1660.
17. Zhou, X.; Wang, D.; Krhenbühl, P. Objects as Points. *arXiv* **2019**, arXiv:1904.07850.
18. Sirisha, M.; Sudha, S.V. Object Detection Using Deep Learning CenterNet Model with Multi-Head External Attention Mechanism. *Int. J. Image Graph.* **2022**, 2450021. [CrossRef]
19. Zhang, F.; Zhu, X.; Dai, H.; Ye, M.; Zhu, C. Distribution-Aware Coordinate Representation for Human Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 7093–7102.
20. Huang, J.; Zhu, Z.; Guo, F.; Huang, G. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 5700–5709.

21. 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, QC, Canada, 10–17 October 2021; pp. 11025–11034.
22. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. RepVGG: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual Conference, 19–25 June 2021; pp. 13733–13742.
23. Wang, J.; Sun, K.; Cheng, T.; Jiang, B.; Deng, C.; Zhao, Y.; Liu, D.; Mu, Y.; Tan, M.; Wang, X.; et al. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 3349–3364. [CrossRef] [PubMed]
24. Carreira, J.; Agrawal, P.; Fragkiadaki, K.; Malik, J. Human Pose Estimation with Iterative Error Feedback. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 4733–4742.
25. Sun, X.; Xiao, B.; Wei, F.; Liang, S.; Wei, Y. Integral Human Pose Regression. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 529–545.
26. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**.
27. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.
28. 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, 23 August 2020; Springer International Publishing: Cham, Switzerland, 2020; pp. 213–229.
29. Wang, Y.; Zhang, X.; Yang, T.; Sun, J. Anchor DETR: Query Design for Transformer-Based Object Detection. *arXiv* **2021**, arXiv:2109.07107.
30. Yang, S.; Quan, Z.; Nie, M.; Yang, W. TransPose: Keypoint Localization via Transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 11802–11812.
31. Li, K.; Wang, S.; Zhang, X.; Xu, Y.; Xu, W.; Tu, Z. Pose Recognition with Cascade Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual Conference, 19–25 June 2021; pp. 1914–1953.
32. Li, Y.; Zhang, S.; Wang, Z.; Yang, S.; Yang, W.; Xia, S.T.; Zhou, E. TokenPose: Learning Keypoint Tokens for Human Pose Estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 11313–11322.
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, 26 June–1 July 2016; pp. 770–778.
34. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–10 June 2015; pp. 3431–3440.
35. Dinh, L.; Sohl-Dickstein, J.; Bengio, S. Density estimation using real nvp. *arXiv* **2016**, arXiv:1605.08803.
36. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
37. Wei, S.E.; Ramakrishna, V.; Kanade, T.; Sheikh, Y. Convolutional pose machines. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 4724–4732.
38. Yu, C.; Xiao, B.; Gao, C.; Yuan, L.; Zhang, L.; Sang, N.; Wang, J. Lite-hrnet: A lightweight high-resolution network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual Conference, 19–25 June 2021; pp. 10440–10450.
39. Fangyun, W.; Xiao, S.; Hongyang, L.; Jingdong, W.; Stephen, L. Point-set anchors for object detection, instance segmentation and pose estimation. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Springer International Publishing: Cham, Switzerland, 2020.
40. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
41. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.